Non Primitive Recursive Complexity Classes

Simon Halfon, ENS Cachan

August 22, 2014

supervised by Philippe Schnoebelen and Sylvain Schmitz
Team INFINI, LSV, ENS Cachan

1 Introduction / Summary File

The general context

The introduction of Well Structured Transition Systems (WSTS) in 1987 [8], i.e. transition systems that satisfies a monotony property with respect to some well-quasi-ordering (wqo), has led to an important number of decidability results of verification problems for several natural models: Petri Nets and VASS and a large number of their extensions, lossy channel systems, string rewrite systems, process algebra, communicating automaton, and so on. Surveys of results and applications obtained with this theory can be found in [9, 3, 1, 2]. The main idea behind these decidability results is a generic algorithm that explores a tree that must be finite by the wqo property: every infinite sequence of configurations of the system $(x_i)_{i\in\mathbb{N}}$ has an increasing pair, that is a pair $i < j$ such that $x_i \leq x_j$. Moreover, wqo theory has provided upper bound to these algorithms by bounding the length of so called bad sequences, (finite) sequences that do not have an increasing pair. The bounds obtained are non-primitive-recursive, which is unusual in verification. In addition, matching lower bounds has been proved for several models [19, 6].

The research problem

Considering the growing number of new problems with a non-primitive-recursive complexity, it is natural to ask for a finer classification, and therefore the formalization of non-elementary complexity classes. This has been achieved by the introduction of a hierarchy of ordinal-indexed fast-growing complexity classes $(\mathbf{F}_\alpha)_\alpha$ [18]. These classes are defined thanks to a family of fast-growing functions $(F_\alpha)_\alpha$ that are at the heart of the extended Grzegorczyk hierarchy, introduced by Löb and Wainer [15]. These functions from $\mathbb{N}$ to $\mathbb{N}$ are defined as follows: $\forall n \in \mathbb{N}$

- $F_0(n) = n + 1$,
- Given α an ordinal, $F_{\alpha+1}(n) = F_\alpha^n(n)$ (composition iterated n times)
- Given λ an ordinal limit of the sequence $(\lambda_n)_{n\in\mathbb{N}}$, $F_\lambda(n) = F_{\lambda_n}(n)$

$\mathbf{F}_\alpha$ is then simply defined as the class of problems that can be solved in deterministic time bounded by F_α^i for some fixed $i \in \mathbb{N}$. Note that $\mathbf{F}_3$ already corresponds to the class of problems solvable in elementary recursive time. In this report, we will pay a particular attention to the $\mathbf{F}_\omega$ and $\mathbf{F}_{\omega^\omega}$ classes, respectively

the class of Ackermannian and HyperAckermannian problems, *i.e.* problems that require Ackermannian and HyperAckermannian time to solve. Unfortunately, these classes are hard to manipulate, too few complete problems are known (see [18] for a recent survey). One of the current goal of the INFINI Team at LSV is to build a toolbox of complete problems for these two classes in order to

1. make the notation and manipulation of these classes universal, as in the case of the class NP, today widely used, which has not always been the case.

2. make easier the manipulations of complete problems for these two classes. As in the case of NP with the *Guide to the theory of NP-completeness* [10] by Garey and Johnson, gathering problems coming from different area, verification, logic, language theory, linear algebra and so on, of computer science allows one to choose the appropriate Ackermann-complete problem to prove a lower bound, *e.g.* the one with the simplest reduction.

For instance, one classical $\mathbf{F}_{\omega^\omega}$-complete problem is the problem of reachability for Lossy Channel Systems. Reductions from this problem can be tedious, it requires to encode the notion of run, in a problem that may not contain any dynamism. Then proving the correction of the reductions can be difficult, even when the intuition is simple. A more elegant problem has been introduced and proved $\mathbf{F}_{\omega^\omega}$-complete in [5]: the *Post Embedding Problem*. It is a variant of the well-known Post Correspondence Problem where equality on words is replaced by the subword relation. This problem is presented in section 2.

Our contribution

In order to find an "elegant" problem for the $\mathbf{F}_\omega$ class, the first step of this internship was to study variants of the Post Embedding Problem, that we have introduced and called *Regular Integer Problems* (see section 3). In Section 4, we prove the precise complexity of all the problems introduced in section 3. We finally conclude with a discussion that briefly summarize our results, but also presents all the problems left open along the way, all the ideas that could not be explored because of time constraints.

This work on Regular Integer Problems led to the submission of an article *Integer Vector Addition Systems with States* [12] by Christoph Haase and myself to the 2014 edition of the conference on *Reachability Problems* (RP). This article, which gather the main results obtained during my internship, has been accepted.

Arguments supporting its validity

In the end, no completely new $\mathbf{F}_\omega$ problem has been found, but a reformulation of the coverability problem for lossy counter machines has highlighted new variants of this problem to explore. Among others, this has led to the submission and acceptance of an article [12], which contains interesting and surprising results, even if these results are not directly linked to the non primitive recursive complexity classes.

Summary and future work

Many problems worth considering have been left open along the way. Even if they do not all deal with non primitive recursive complexity, these problems are interesting in different areas. A precise list can be found in conclusion of this report. In particular, I have considered a completely different problem during the last month of my internship, for which I could not determine its precise complexity, and that will constitute my immediate next work. If my intuition is correct, this problem is singular since the basic wqo analysis gives an upper bound that is too large.

Contents

2 Preliminaries

In this section, we provide most of the definitions that we rely on in this paper. We first introduce some general notations, then we remind the fundamental notions of the domain, such as Well Structured Transition Systems or Counter Machines; and finally introduce more specific definitions on which this work is based, and state associated results.

2.1 General Notations

Vectors. In the following, $\mathbb{Z}$ and $\mathbb{N}$ are the sets of integers and natural numbers, respectively, and $\mathbb{N}^k$ and $\mathbb{Z}^k$ are the set of dimension k vectors in $\mathbb{N}$ and $\mathbb{Z}$, respectively. We denote by $[k]$ the set of positive integers up to k, *i.e.* $[k] = \{1, \ldots, k\}$. By $\mathbb{N}^{k\times k}$ and $\mathbb{Z}^{k\times k}$ we denote the set of $k \times k$ square matrices over $\mathbb{N}$ and $\mathbb{Z}$, respectively. The identity matrix in dimension k is denoted by I_k and $\vec{e}_i$ denotes the i-th unit vector in any dimension k provided $i \in [k]$. For any k and $i, j \in [k]$, E_{ij} denotes the $k \times k$-matrix whose i-th row and j-th column intersection is equal to one and all of its other components are zero, and we use E_i to abbreviate E_{ii}. For $\vec{v} \in \mathbb{Z}^k$ we write $\vec{v}(i)$ for the i-th component of $\vec{v}$ for $i \in [k]$. Given two vectors $\vec{v_1}, \vec{v_2} \in \mathbb{Z}^k$, we write $\vec{v_1} \geq \vec{v_2}$ iff for all $i \in [d]$, $\vec{v_1}(i) \geq \vec{v_2}(i)$: it is the product ordering.

Words. An *alphabet* Σ is a finite set of symbols (or letters). Words over an alphabet Σ are finite sequences of symbols, the set of all words is denoted Σ^*. The empty word, denoted ϵ, is the empty sequence. Given two words $u, v \in \Sigma^*$, the subword relation $u \sqsubseteq v$ holds if u can be obtained from v by erasing some letters, *i.e.* if $u = u_1 \cdots u_n$ and $u \sqsubseteq v$ then there exist some words $v_0, \ldots, v_n \in \Sigma^*$ such that $v = v_0 u_1 v_1 \cdots u_n v_n$. A word $u \in \Sigma^*$ is said to be a prefix of $v = v_1 \cdots v_n$ if there is $i \in [n]$ such that $u = v_1 \cdots v_i$ or $u = \epsilon$, and u is said to be a suffix if there is a $i \in [n]$ such that $u = v_i \cdots v_n$ or $u = \epsilon$. Given a word $v \in \Sigma^*$, the set of all prefixes (resp. suffixes) of v is denoted $\mathrm{Pref}(v)$ (resp. $\mathrm{Suff}(v)$). The Parikh image of a word u is the vector $\pi(u) \in \mathbb{N}^{|\Sigma|}$ such that for $i \in [|\Sigma|]$, $\pi(u)(i)$ is the number of occurrences of the i-th symbol of Σ in u (we assume that letters in Σ have been ordered). For instance, two words have the same Parikh image if and only if one is a permutation of the other. Finally, given a word $u = u_1 \cdots u_n$, we will denote by $\tilde{u}$ its mirror word $\tilde{u} = u_n \cdots u_1$. The same notation will be used for languages. Note that $\mathrm{Pref}(u) = \widetilde{\mathrm{Suff}(\tilde{u})}$.

Morphisms. Given two monoids $(\mathcal{A}, +, e_{\mathcal{A}}), (\mathcal{B}, \times, e_{\mathcal{B}})$ with one law, $h : \mathcal{A} \to \mathcal{B}$ is a morphism if for all $a, b \in \mathcal{A}$, $h(a+b) = h(a) \times h(b)$ and $h(e_{\mathcal{A}}) = e_{\mathcal{B}}$. Morphisms used in this report use the following structures:

- $(\Sigma^*, \cdot)$, where Σ is an alphabet and $\cdot$ denotes the concatenation of words
- $(\mathbb{N}^k, +)$, where $+$ is the usual component-by-component addition
- $(\mathcal{A} \to \mathcal{A}, o)$: functions from $\mathcal{A}$ to $\mathcal{A}$, with the composition of functions.

Since there is no possible confusion, the structures will be given by their underlying set, the law being the corresponding one.

Presburger Arithmetic. Recall that *Presburger arithmetic (PA)* is the first-order theory of the structure $\langle \mathbb{N}, 0, 1, +, \geq \rangle$, *i.e.*, quantified linear arithmetic over natural numbers. It is well-known that the validity of an existential Presburger formula is an NP-complete problem, see *e.g.* [4].

2.2 Well Structured Transition Systems

A quasi-ordering is a reflexive and transitive relation. Well-quasi-orderings (wqo) are quasi-orderings that are well-founded and that do not have infinite antichains, *i.e.* infinite sequences with pairwise incomparable elements. The general class of transition systems presenting some monotony with respect to a wqo, called *Well Structured Transition Systems* (WSTS), is interesting for mainly two reasons: there are generic algorithms for solving reachability, coverability, termination or boundedness for models of this class; and many common models used in computer science, and more particularly in verification, belong to this class.

To defined Well Structured Transition Systems, we must first define transition systems.

Definition 1 (Transition System)**.** *A* Transition System *is a couple* $(\mathcal{S}, \mathcal{T})$*, where* $\mathcal{S}$ *is a possibly infinite set of* configurations *and* $\mathcal{T} \subseteq S \times \mathcal{S}$ *is a* transition *relation on* $\mathcal{S}$*.*

Given two configurations $s, t \in \mathcal{S}$ we write $s \to_{\mathcal{T}} t$, or simply $s \to t$ if $\mathcal{T}$ is clear from the context, when $(s,t) \in \mathcal{T}$. The reflexive and transitive closure of $\to_{\mathcal{T}}$ is denoted by $\to^*_{\mathcal{T}}$. A sequence $s = s_0, s_1, \ldots, s_n = t$ such that for all $i \in [n]$, $s_{i-1} \to s_i$ is called a run from s to t. We say that t is *reachable from* s when there exists a run from s to t, *i.e.* $s \to^* t$. Well Structured Transition Systems are ordered transition systems for which some monotonicity holds.

Definition 2 (WSTS)**.** *An ordered transition system* $(\mathcal{S}, \mathcal{T}, \preccurlyeq)$ *is a* Well Structured Transition System *if*

1. $\preccurlyeq \subseteq \mathcal{S} \times \mathcal{S}$ *is a wqo*
2. $\forall s, s', t \in \mathcal{S}.\ \ (s \preccurlyeq t \text{ and } s \to_{\mathcal{T}} s') \ \Rightarrow\ \exists t' \in \mathcal{S}.\ \ s' \preccurlyeq t' \text{ and } t \to^*_{\mathcal{T}} t'$

Given $\mathcal{W}$ a class of WSTS, we are mostly interested in this report by the two fundamental problems:

$\mathcal{W}$ Reachability/Coverability

INPUT: A WSTS $(\mathcal{S}, \mathcal{T}, \preccurlyeq) \in \mathcal{W}$, configurations $s_0, s_f \in \mathcal{S}$
QUESTION: *Reachability:* Is there a run $s_0 \to^*_{\mathcal{T}} s_f$?
Coverability: Is there a configuration $t \in \mathcal{S}$ s.t. $s_0 \to^*_{\mathcal{T}} t$ and $s_f \preccurlyeq t$?

The next section introduces two class of WSTS that make the reachability and coverability problems defined above respectively of Ackermannian and HyperAckermannian complexity.

2.3 Lossy machines

A *Lossy Channel System* is an automaton that writes to and reads from a FIFO channel that may lose messages.

Definition 3. *A* Lossy Channel System *(LCS) is a triple* $\mathcal{L} = (Q, \Gamma, \Delta)$*.* Q *a finite set of states,* Γ *a finite set of messages and* Δ *is a finite set of transitions* $\Delta \subseteq Q \times \{!, ?\} \times \Gamma \times Q$*.*

The associated semantic is an infinite transition system $(Q \times \Gamma^*, T)$ such that for any two states $p, q \in Q$, $w \in \Gamma^*$ and $a \in \Gamma$

- $((p,w),(q,wa)) \in T$ if $(p, !a, q) \in \Delta$
- $((p,aw),(q,w)) \in T$ if $(p, ?a, q) \in \Delta$
- $((p,w),(p,v)) \in T$ if $v \sqsubseteq w$

The third type of transition formalizes the idea of losses in the channel: at any time, some messages can be erased from the channel. Without this third type of transition, one obtain the usual notion of Channel Systems, said to be reliable, for which classical verification problems (termination, reachability, coverability, ...) are undecidable. Adding these lossy transition makes the transition system monotone for the well-quasi-ordering: $(p, v) \preccurlyeq (q, w)$ iff $p = q$ and $v \sqsubseteq w$. Generic wqo analysis to bound bad sequences give a $\mathbf{F}_{\omega^\omega}$ upper-bound for the aforementioned classical verification problems in the case of LCS. A matching lower bound has been proved in [6].

A *Lossy Counter Machine* is a counter machine whose counters may spontaneously have their value decreased.

Definition 4. *A* Lossy Counter Machine *(LCM) is a triple $\mathcal{M} = (Q, C, \Delta)$. Q is a finite set of states, $C = \{c_1, \ldots, c_d\}$ a finite set of counters, and Δ a finite set of transitions $\Delta \subseteq Q \times C \times Op \times Q$, where $Op = \{\texttt{++}, \texttt{--}, \stackrel{?}{=} 0\}$*

The associated semantic is an infinite transition system $(Q \times \mathbb{N}^d, T)$ such that for any two states $p, q \in Q$, $\vec{u}, \vec{v} \in \mathbb{N}^d$ and $i \in [d]$:

- $((p, \vec{u}), (q, \vec{v})) \in T$ if $(p, c_i\texttt{++}, q) \in \Delta$ and for any $j \in [d]$, $\vec{v}(j) = \vec{u}(j)$ iff $j \neq i$, and $\vec{v}(i) = \vec{u}(i) + 1$
- $((p, \vec{u}), (q, \vec{v})) \in T$ if $(p, c_i\texttt{--}, q) \in \Delta$ and for any $j \in C$, $\vec{v}(j) = \vec{u}(j)$ iff $j \neq i$, and $\vec{v}(i) = \vec{u}(i) - 1 \geq 0$
- $((p, \vec{u}), (q, \vec{u})) \in T$ if $(p, c_i \stackrel{?}{=} 0, q) \in \Delta$ and $\vec{u}(c) = 0$
- $((p, \vec{u}), (q, \vec{v})) \in T$ if $v \leq u$ for the product ordering on $\mathbb{N}^d$

Once again, the last rule is the one formalizing the losses of the counters. Without this last rule, one obtain Minsky machines, known to make undecidable the problems we consider. With this lossy rule, the transition system given by the semantic is well-structured with respect to the well-quasi-ordering: $(p, \vec{u}) \preccurlyeq (q, \vec{v})$ iff $p = q$ and $u \leq v$ for the product ordering. Generic wqo analysis give in this case a $\mathbf{F}_\omega$ upper bound, and a proof of a matching lower bound can be found in [19].
The two following well-known restriction of counter machines are also used in this report:

- A counter machine with a reliable behavior and no zero-test transition is called a *VASS* (Vector Addition Systems With States). It is a fundamental model in verification, whose reachability problem is known to be decidable [16], but no upper bound has been established. Its coverability problem is known to be EXPSPACE-complete [17]
- A VASS with only one state ($|Q| = 1$) is called a *VAS* (Vector Addition System)

2.4 Post Embedding Problems

This subsection recalls the problems introduced in [5] and the associated results, as a motivation to the problems defined in the next section. The *Post Embedding Problem*, and its most interesting extensions, are decidable variants of the well-known *Post Correspondence Problem* (*PCP*):

Post Correspondence Problem
INPUT: Two morphisms $u, v : \Sigma^* \to \Gamma^*$
QUESTION: Is there a word $\sigma \in \Sigma^+$ such that $u(\sigma) = v(\sigma)$?

Such a small change as asking for $u(\sigma)$ to be only a subword of $v(\sigma)$ ($u(\sigma) \sqsubseteq v(\sigma)$) makes the problem decidable, and even "trivial" (solvable in NL). This problem is known as *Post Embedding Problem*. Now, a

little modification gives a still decidable variant, but with non-primitive recursive complexity: the *Regular Post Embedding Problem* (*PEP*)

Regular Post Embedding Problem (*PEP*)
INPUT: Two morphisms $u, v : \Sigma^* \to \Gamma^*$, a regular language $R \subseteq \Sigma^*$
QUESTION: Is there a word $\sigma \in R$ such that $u(\sigma) \sqsubseteq v(\sigma)$?

This problem has been proved $\mathbf{F}_{\omega^\omega}$-complete in [5], by equivalence with the reachability problem for Lossy Channel Systems. More precisely, the two problems are equivalent with an intermediate problem: a solution σ to a Regular Post Embedding instance (u, v, R) is said to be *directed* (resp. *codirected*) if for any prefix (resp. suffix) τ of σ, $u(\tau) \sqsubseteq v(\tau)$. The two following problems are $\mathbf{F}_{\omega^\omega}$-complete as well.

Directed Regular Post Embedding Problem (*PEP*$_{dir}$)
INPUT: Two morphisms $u, v : \Sigma^* \to \Gamma^*$, a regular language $R \subseteq \Sigma^*$
QUESTION: $\exists \sigma \in R.\ \forall \tau \in \mathrm{Pref}(\sigma).\ u(\tau) \sqsubseteq v(\tau)$?

Codirected Regular Post Embedding Problem (*PEP*$_{codir}$)
INPUT: Two morphisms $u, v : \Sigma^* \to \Gamma^*$, a regular language $R \subseteq \Sigma^*$
QUESTION: $\exists \sigma \in R.\ \forall \tau \in \mathrm{Suff}(\sigma).\ u(\tau) \sqsubseteq v(\tau)$?

The intuition behind the equivalence between *PEP*$_{dir}$ and reachability for LCS is that the regular language R encodes the control state structure of the Lossy Channel System, while the two morphisms give a labeling for the transitions: v encodes the writing and u the reading. The directed restriction is essential to ensure that what is read has been written before. The first goal of my internship was to find an $\mathbf{F}_\omega$-complete problem of the same shape, it is then natural to follow the same intuition: R will encode the control state structure of a Lossy Counter Machine, and we need to replace Γ^* by a structure adapted to encode counters ranging over the integers (Γ^* was suited to encode a channel containing words). The next section presents the different choices of new structure we tried and the different results we obtained.

3 Regular Integer Problems

3.1 Definitions

A natural first attempt of structure to encode k counters is simply $\mathbb{N}^k$. This defines a problem we called the *Regular Integer Problem* (*RIP*)

Regular Integer Problem (*RIP*)
INPUT: An integer k, two morphisms $u, v : \Sigma^* \to \mathbb{N}^k$, a regular language $R \subseteq \Sigma^*$
QUESTION: $\exists \sigma \in R.\ u(\sigma) \leq v(\sigma)$?

where $\leq$ denotes the product ordering over $\mathbb{N}^k$ (*i.e.* the order that makes Lossy Counter Machines a WSTS), and u, v are morphisms for the addition in $\mathbb{N}^k$. Moreover, by taking $h = v - u$, the problem can be stated in a more compact way:

Regular Integer Problem (*RIP*)
INPUT: An integer k, a morphism $h : \Sigma^* \to \mathbb{Z}^k$, a regular language $R \subseteq \Sigma^*$

QUESTION: $\exists \sigma \in R.\ \vec{0} \leq h(\sigma)$?

Directed and codirected variants are defined as in the case of *PEP*.

As intended, the regular language R can encode a control state structure, and the morphism h can encode the additive actions, but not the zero-tests, and indeed, RIP_{dir} can be shown equivalent to coverability for VASS (see 4.5). In order to encode Lossy Counter Machines, and hence obtain a $\mathbf{F}_\omega$-complete problem, one need to consider a more general structure: affine transformations from $\mathbb{Z}^k$ to $\mathbb{Z}^k$. Given $\mathcal{A} \subseteq \mathbb{Z}^{k\times k}$, we define the *Regular Affine Integer Problem over* $\mathcal{A}$:

Regular Affine Integer Problem ($RAIP(\mathcal{A})$)

INPUT: An integer k, a morphism $h : \Sigma^* \to (\mathbb{Z}^k \to \mathbb{Z}^k)$, a regular language $R \subseteq \Sigma^*$ where $\forall a \in \Sigma\,, \exists A \in \mathcal{A}\,, \vec{b} \in \mathbb{Z}^d\,, \forall \vec{v} \in \mathbb{Z}^k.\ h(a)(\vec{v}) = A\vec{v} + \vec{b}$

QUESTION: $\exists \sigma \in R.\ \vec{0} \leq h(\sigma)(\vec{0})$?

Here, h is a morphism for the composition: for any $a \in \Sigma, \sigma \in \Sigma^*$, $h(a\sigma)(\vec{v}) = h(\sigma)(h(a)(\vec{v}))$. We will note $RAIP$ for $RAIP(\mathbb{Z}^{k\times k})$ the full fragment of affine transformations, and $RIPR$ for $RAIP(\{I_k\} \cup \{I_k - E_i \mid i \in [k]\})$ the fragment of reset transformations, that is diagonal matrices with coefficients among $\{0, 1\}$. Moreover, note that $RIP = RAIP(\{I_k\})$. Directed and codirected versions are defined as in the previous cases.

Finally, we introduce the restrictions of the problems RIP, $RIPR$ and $RAIP$ where the word solution σ is asked to belong to Σ^+ instead of a language R part of the input:

Existential Affine Integer Problem ($EAIP(\mathcal{A})$)

INPUT: An integer k, two vectors $\vec{v_0}, \vec{v_f} \in \mathbb{Z}^k$, a morphism $h : \Sigma^* \to (\mathbb{Z}^k \to \mathbb{Z}^k)$ where $\forall a \in \Sigma\,, \exists A \in \mathcal{A}\,, \vec{b} \in \mathbb{Z}^d\,, \forall \vec{v} \in \mathbb{Z}^d.\ h(a)(\vec{v}) = A\vec{v} + \vec{b}$

QUESTION: $\exists \sigma \in \Sigma^+.\ \vec{v_f} \leq h(\sigma)(\vec{v_i})$?

These *existential* problems are named EIP, $EIPR$ and $EAIP$ for the respective restrictions on the set of matrices $\mathcal{A}$. Unlike in the case of *PEP*, these problems are not trivial, instead they correspond to coverability for counter machines without states, namely VAS, possibly extended with resets or affine updates. Note that there is a little variation: the input now has two extra vectors that replace $\vec{0}$ in the question. All the problems defined before are invariant when adding these extra initial and final vectors: they can be encoded in the regular language. This variation only matters in the existential versions where there is no regular language.

3.2 Results

The following figure summarizes the complexity results obtained for all the problems introduced in the previous section. The arrows represents all the trivial reductions from one problem to another, since clearly $RAIP(\mathcal{A})$ reduces to $RAIP(\mathcal{B})$ when $\mathcal{A} \subset \mathcal{B}$. Note that all the results (upper and lower bounds) below hold for both unary and binary encoding of integers, and independently of the representation of a regular language (deterministic or non deterministic automaton, regular language, ...). Proofs of these statements are postponed to the next section.

As previously mentioned, the directed versions naturally correspond to the coverability problem for counter machines whose counters values are bounded to stay in $\mathbb{N}$, *i.e.* VASS, and extensions thereof. Removing this directed constraint can be seen as allowing counters to take negative values, and the existential versions simply are the stateless versions, *i.e.* VAS and extensions, with unbounded counters. Surprisingly, this class of $\mathbb{Z}$-counter machines (with or without states) has been much less studied than the natural number versions. Hence, we have compacted the results from the two first columns in an article *Integer Vector*

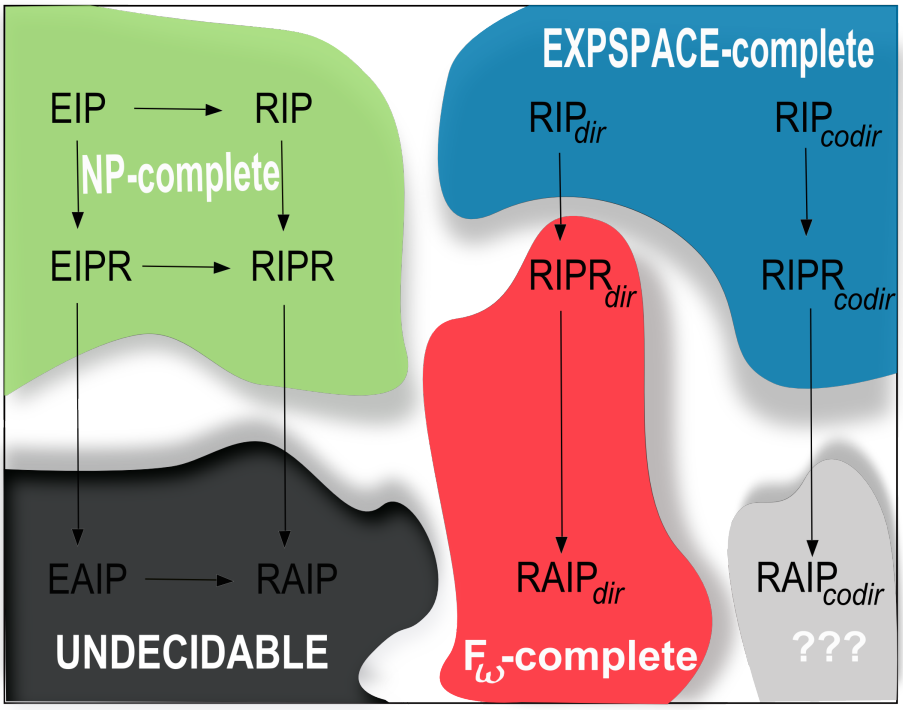

Figure 1: Complexities of the Regular Integer Problems. Arrows represents trivial reductions.

Addition Systems with States [12] that we submitted to the 2014 edition of *Reachability Problems* (RP). The paper has been accepted, and is joint to this report. Problems in this article are presented as reachability and coverability problems for different models of $\mathbb{Z}$-counter machines, with or without states. Finally, it seems that the codirected versions of the problems do not correspond to a natural verification question over counter machines, hence the associated complexity results where not presented in the article, but proofs of these can be found in the next section, except for $RAIP_{codir}$ which has been left open.

The next section contains a proof for each of the results gathered in the figure above. The two first column constitutes the content of our RP paper, hence some of the proofs are only sketched, details can be found in the article itself, which is joint to this report. The NP membership of *RIPR* constitutes the main result of the article, along with the complexity of the inclusion problem for $\mathbb{Z}$-counter machines, problem that is not mentioned in this report.

4 Proofs

Before all, observe that a morphism ranging into $\mathbb{Z}^k$ has the following property:

Lemma 1. *Given $h : \Sigma^* \to \mathbb{Z}^k$, σ and σ' two words with the same Parikh image, $h(\sigma) = h(\sigma')$.*

Proof. Addition on $\mathbb{N}^K$ is associative and commutative. □

Unfortunately, this property is lost for morphism ranging into affine transformations over $\mathbb{Z}^k$, since composition of affine transformations is not commutative. However, similar (but weaker) properties will be proved within the proofs, as it is needed. In the following, we will treat morphisms $h : \Sigma^* \to \mathbb{Z}^k$ as morphisms from Σ^* to translations over $\mathbb{Z}^k$, that is affine transformations using only the identity matrix.

4.1 EIP is NP-complete

In this subsection, it is highlighted that *EIP* simply is a system of linear equations over the integers. Indeed, by lemma 1, only the number of letters in the word solution σ is important.

Therefore, the *EIP* instance $(k, \vec{v}_0, \vec{v}_f, \Sigma, h)$ has a solution if and only if there exists a vector $\vec{x} \in \mathbb{N}^{|\Sigma^*|}$ such that $\vec{v}_0 + \sum_{a\in\Sigma} \vec{x}(a) \cdot h(a) \geq \vec{v}_f$. It is well known that solving an affine system of equations over $\mathbb{N}$ is NP-complete, and any linear system can be represented as a *EIP* instance.

Note that if we remove the two vectors $\vec{v}_0$ and $\vec{v}_f$ from the input, and ask for $h(\sigma)(\vec{0}) \geq \vec{0}$, then an instance can be solved in P. Indeed, the system becomes $\sum_{a\in\Sigma} \vec{x}(a) \cdot h(a) \geq \vec{0}$, which can be solved in $\mathbb{Q}$ in polynomial time using linear programming, then one obtain solutions in $\mathbb{N}$ by multiplying every equation by its common denominator.

4.2 EIPR is NP-hard

NP-completeness of *EIPR* can be deduced from the previous section and section 4.4. However, we prove in the following that the weaker version of *EIPR* where the vectors $\vec{v}_0$ and $\vec{v}_f$ are removed from the input is already NP-hard, by reduction from Subset Sum.

Subset Sum is the following problem:

Subset Sum

INPUT: A set $S = \{s_1, \ldots, s_k\} \subset \mathbb{N}$ and an integer $A \in \mathbb{N}$

QUESTION: Is there a subset $X \subseteq [k]$ s.t. $\sum_{i\in X} s_i = A$?

Given an instance of Subset Sum, consider the following *EIPR* instance:

- $\Sigma = S \cup a$
- $h : \Sigma^* \to (\mathbb{Z}^{k+2} \to \mathbb{Z}^{k+2})$
- $h(a)(X) = Z \times X + (A, -A, 1, \ldots, 1)$ where Z is the zero matrix
- for any $i \in [k]$, $h(s_i)(X) = X + (-s_i, s_i, 0, \ldots, 0) - E_{i+2}$

Correction of the reduction: there is $X \subseteq [k]$ such that $\sum_{i \in X} s_i = A$ if and only if there is a $\sigma \in \Sigma^+$ such that $h(\sigma)(\vec{0}) \geq \vec{0}$

($\Rightarrow$) Let $X = \{s_{i_1}, \ldots, s_{i_p}\}$ be a solution of the Subset Sum instance, then $\sigma = a s_{i_1} \ldots s_{i_p}$ is a solution to the *EIPR* instance:

$$h(\sigma)(0) = \begin{pmatrix} A - \sum_{i \in X} s_i \\ -A + \sum_{i \in X} s_i \\ 1 - (s_1 \in X) \\ \vdots \\ 1 - (s_p \in X) \end{pmatrix}$$

where $s_i \in X$ is the characteristic function of X applied to s_i. Since X is a solution to the subset sum instance, the two first component are null, and since X is a set, $1 - (s_i \in X) \geq \vec{0}$, which means that $h(\sigma)(\vec{0}) \geq \vec{0}$.

($\Leftarrow$) Let $\sigma \in \Sigma^+$ be a solution: $h(\sigma)(\vec{0}) \geq \vec{0}$. First observe that the letter a must occur in σ, if not, one of the k last component of $h(\sigma)(\vec{0})$ will be negative. Indeed, there must be at least one letter in σ, be it s_i, since only the letter a can add to component $j+2$, $h(\sigma)(\vec{0})(j+2) \leq -1$.

Now we can assume that the letter a only occurs once in σ as its first letter, since given any solution $\sigma = \sigma' a \sigma''$, we have:

$$\vec{0} = h(\sigma)(\vec{0}) = h(a\sigma'')(h(\sigma')(\vec{0})) = h(\sigma'')(h(a)(h(\sigma')(\vec{0}))) = h(\sigma'')(Z \times h(\sigma')(\vec{0}) + h(a)(\vec{0})) = h(a\sigma'')(\vec{0})$$

Thus, $a\sigma''$ is a solution as well. Finally, given a solution in which a occurs only once at first position, each letter s_i occurs at most once because of the $(i+2)$-th component. Consider X the set of indexes i such that s_i occurs in the solution, it is a solution to the Subset Sum instance, indeed both $A - \sum_{i \in X} s_i$ and $\sum_{i \in X} s_i - A$ are greater than 0, which means that $A = \sum_{i \in X} s_i$.

4.3 RIP is in NP

As in the case of *EIPR*, NP-completeness of *RIP* can be deduced from NP-completeness for *EIPR* and *RIPR*. However, it is important to note that the proof of NP-membership of *RIPR* generalizes the commonly known proof of NP-membership. It consists of showing that:

- It is enough to know the Parikh image of the solution
- Parikh images of a regular language is a semi-linear set that can be computed as an existential Presburger formula
- Testing the satisfiability of an existential Presburger formulas is a NP-complete problem

To show that *RIPR* can be solved in NP, one essentially needs to adapt these three steps to handle the reset operation: generalize the definition of the Parikh image and show that the set of generalized Parikh image of a regular language is still expressible as an existential Presburger formula. The next section informally describes these steps, details can be found in appendix.

4.4 RIPR is in NP

In this section, we describe a NP procedure to solve an instance (k, Σ, h, R) of *RIPR*. First, we argue that without loss of generality, we can assume the instance to be of a more specific form. We then sketch a procedure and give some arguments of correctness. A detailed and more complete proof can be found in the section 3 of [12] joint in the appendix.

Given a *RIPR* instance (k, Σ, h, R), we consider the instance $(k, \Sigma \cup \{r_1, \ldots, r_k\}, h', R')$ defined as follows. For $i \in [k]$, let R_i is the i-th reset matrix, *i.e.* $R_i = I_k - E_i$, and by convention let $R_0 = I_k$. For any $a \in \Sigma$, we know there are $i_a \in [0, k]$ and $\vec{b}_a \in \mathbb{Z}^k$ such that for all $\vec{v} \in \mathbb{Z}^k$, $h(a)(\vec{v}) = R_{i_a}\vec{v} + \vec{b}_a$. Now, h' and R' are such that for any $\vec{v} \in \mathbb{Z}^k$:

- for $a \in \Sigma$, $h'(a)(\vec{v}) = \vec{v} + \vec{b}_a$
- for $i \in [k]$, $h'(r_i)(\vec{v}) = R_i\vec{v}$
- R' is the language R where every occurrence of a letter $a \in \Sigma$ has been substituted for the word ar_{i_a}, except if $i_a = 0$.

These two problems obviously have the same solutions. Intuitively, the second form of the instance just isolate the resets part from the additive part (both in h and R). This clear distinction between the additive part, which we known can be solve in NP, and the resets part allows us to deal more precisely with the resets, in order to "hide" them.

Now, assume we are given such an instance of *RIPR*, that is an instance (k, Σ', h, R) where Σ' can be split in $\Sigma' = \Sigma \cup \{r_1, \ldots, r_k\}$ such that for any $i \in [k]$ and $\vec{v} \in \mathbb{Z}^k$, $h(r_i)(\vec{v}) = R_i\vec{v}$; and for any $a \in \Sigma$ there is $\vec{b} \in \mathbb{Z}^k$ such that $h(a)(\vec{v}) = \vec{v} + \vec{b}$. The key to the NP procedure is that for a word $\sigma_1 r_1 \sigma_2 \in \Sigma'$, the first component of $h(\sigma_1 r_1 \sigma_2)(\vec{v})$ is the same as the first component of $h(\sigma_2)(\vec{v})$. In consequence, only the last occurrence of a reset r_i has an importance for the i-th component. This motivates the following decomposition.

Lemma 2. *Given $\sigma \in \Sigma'^* = (\Sigma \cup \{r_1, \ldots, r_k\})^*$, there are an integer $p \in [k]$, a permutation $f : [k] \to [k]$ and p words σ_i such that*

$$\sigma = \sigma_p r_{f(p+1)} \sigma_{p+1} \cdots r_{f(k)} \sigma_k$$

where for all $i \in [p, k]$, $\sigma_i \in \Sigma \cup \{r_{f(i+1)}, \ldots, r_{f(k)}\}$

This definition decomposes a word σ in at most k parts, which is polynomial of the input. This decomposition is our new Parikh image: the generalized Parikh image consists of at most $k + 1$ vectors, and a permutation. Generalizing the construction provided in [20] to build an existential Presburger formula expressing the Parikh image of a given regular language is not difficult. Finally, we obtained the same three ingredients as for the NP-membership proof for *RIP*. Details on this procedure can be found in appendix, in section 3 of [12].

4.5 RIP_{dir} is EXPSPACE-complete

In this section, we prove that RIP_{dir} and coverability for VASS are inter-reducible. The second problems is known to be EXPSPACE-complete [17], which gives the complexity of RIP_{dir}. The intuition behind the proof is once again the same: the regular language R is just a control state structure, and the morphism h is just the actions on counters performed by transitions.

Proposition 1. *There is a logarithmic-space reduction from RIP_{dir} to coverability for VASS*

Proof. Given an instance $(k, \Sigma, h : \Sigma^* \to \mathbb{Z}^k, R)$ of RIP_{dir}, and $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ an automaton that recognizes R, we can assume that F is a singleton: otherwise it is enough to add a new state q_f, a new letter e, and transitions from states in F to q_f labeled by e, then define $h(e) = \vec{0}$. we now consider the VASS $\mathcal{V} = (Q, k, \Delta')$ where Δ' is given by: for any $p, q \in Q$, $a \in \Sigma$, $(p, a, q) \in \Delta$ iff $(p, h(a), q) \in \Delta'$. We claim that there is $\sigma \in R$ such that $h(\tau) \geq \vec{0}$ for any prefix τ of σ if and only if there is a run in $\mathcal{V}$ from $(q_0, \vec{0})$ to $(q_f, \vec{v})$ for some $\vec{v} \in \mathbb{N}^k$, that is, if and only if q_f is coverable from $(q_0, \vec{0})$ in $\mathcal{V}$.

Indeed one can easily show by induction over the length of σ that: there is a run from q_0 to q in $\mathcal{A}$ such that for every prefix τ of σ, $h(\tau) \geq \vec{0}$ if and only if there is a run from $(q_0, \vec{0})$ to $(q, h(\sigma))$ in $\mathcal{V}$. Applying this for a $\sigma \in R$, we obtain the desired result since $\mathcal{A}$ has only final state q_f. □

Proposition 2. *There is a logarithmic-space reduction from Coverability for VASS to RIP_{dir}*

Proof. Given a VASS $\mathcal{V} = (Q, k, \Delta)$ and two states q_0, q_f, consider the following RIP_{dir} instance: $(k, \Delta, h : \Delta^* \to \mathbb{Z}^k, R)$ where:

- For any $\delta = (p, \vec{v}, q) \in \Delta$, $h(\delta) = \vec{v}$
- R is the language recognized by the automaton $\mathcal{A} = (Q, \Delta, \Delta, q_0, \{q_f\})$

Obviously, the same invariant as in the previous proof holds, hence the result. □

These two reductions emphasizes how RIP_{dir} simply is a reformulation of the coverability problem for VASS, in the same way that $PEP_{dir}$is a simple reformulation of the reachability problem for Lossy Channel Systems.

4.6 $\mathbf{RIP}_{codir}$ is EXPSPACE-complete

In the previous section, RIP_{dir} has been proved EXPSPACE-complete. The same complexity holds for RIP_{codir}, since the two problems are inter-reducible. The proof is completely inspired of the proof that PEP_{dir} and PEP_{codir} are equivalent: the idea is to read the run backward, so that suffixes become prefixes. Since addition is commutative and associative, the final value of the counters is preserved (*c.f.* lemma 1).

Given (k, Σ, h, R) an instance of RIP_{dir}, we consider the instance $(k, \Sigma, h, \tilde{R})$ of RIP_{codir}, where $\tilde{R}$ is the language of mirror words in R. Given an automaton recognizing R, an automaton recognizing $\tilde{R}$ can be built in logarithmic-space, it is sufficient to replace every transition (p, a, q) by (q, a, p) and inverting initial and final states.

Now, by lemma 1, since σ and $\tilde{\sigma}$ have the same Parikh image, $h(\sigma) = h(\tilde{\sigma})$. Moreover, τ is a prefix of σ if and only if $\tilde{\tau}$ is a **suffix** of $\tilde{\sigma}$. These two facts ensure the correction of the reduction. The reduction from RIP_{codir} to RIP_{dir} is exactly the same.

4.7 $\mathbf{RIPR}_{codir}$ is in EXPSPACE

Both in the case of *PEP* and *RIP*, the directed and codirected versions of the problem turned out to be equivalent, because the target structure of the morphism(s) enjoyed some property of stability when considering the mirror operation on words of the source structure. For instance, a morphism $h : \Sigma^* \to \mathbb{Z}^k$ satisfies $h(\sigma) = h(\tilde{\sigma})$. This is no longer the case when the target structure is $\mathbb{Z}^k \to \mathbb{Z}^k$ equipped with the composition law, as this law is not commutative. As a result, there are no reason for $RAIP_{dir}$ and $RAIP_{codir}$ (resp. $RIPR_{dir}$ and $RIPR_{codir}$) to be equivalent. And indeed they are not in the case of *RIPR*: $RIPR_{dir}$is a $\mathbf{F}_\omega$-complete problem, as proved in subsection 4.8, while $RIPR_{codir}$ is only EXPSPACE-complete, as proved

in this subsection. The EXPSPACE part is by a reduction to RIP_{dir}, while the EXPSPACE-hard part is trivial by the natural reduction from RIP_{codir} to $RIPR_{codir}$.

Proposition 3. *There is a logarithmic-space reduction from $RIPR_{codir}$ to RIP_{dir}*

Proof. As in the case of *RIPR*, we can without loss of generality assume an instance of $RIPR_{codir}$ to be of the form (k, Σ', h, R) with $\Sigma' = \Sigma \cup \{r_1, \ldots, r_k\}$, $h(r_i)$ being the i-th reset and $h(a)$ being only additive for $a \in \Sigma$. With a similar reduction, we can further assume $\Sigma' = \Sigma_1 \cup \Sigma_2 \cup \cdots \cup \Sigma_k \cup \{r_1, \ldots, r_k\}$, where all the Σ_i are indexed copies of an alphabet Σ, such that $h(a_i)(\vec{v}) = \vec{v} + \lambda_{ai}\vec{e_i}$; in other words, $h(a_i)$ is not only additive, but moreover only adds on the i-th component.

From such an instance, we can build the following RIP_{dir} instance: $(3k, \Sigma'' \cup \{\#\}, h', \# \cdot R')$, where $R' \subset \Sigma''$.

- $\Sigma'' = (\bigcup_{i\in[k]} \Sigma_i) \cup \{u_i\texttt{++}, u_i\texttt{--}, v_i\texttt{++}, v_i\texttt{--} \mid i \in [k]\}$
- R' is the mirror language of R, $\tilde{R}$, where occurrences of $a_i \in \Sigma_i$ are replaced by the regular expression $e_1 = (u_i\texttt{--} \cdot a_i \cdot u_i\texttt{++}) + (v_i\texttt{--} \cdot v_i\texttt{++})$, and occurrences of r_i are replaced by the regular expression $e_2 = (u_i\texttt{--} \cdot v_i\texttt{++}) + (v_i\texttt{--} \cdot v_i\texttt{++})$
- For $a_i \in \Sigma_i$, $h'(a_i) = h(a_i)$ ($h'(a_i)$ is null over the $2k$ extra dimensions)
- $h'(u_i\texttt{++})(\vec{v}) = \vec{v} + \vec{e}_{k+i}$, $h'(u_i\texttt{--})(\vec{v}) = \vec{v} - \vec{e}_{k+i}$
- $h'(v_i\texttt{++})(\vec{v}) = \vec{v} + \vec{e}_{2k+i}$, $h'(v_i\texttt{--})(\vec{v}) = \vec{v} - \vec{e}_{2k+i}$
- $h'(w_i\texttt{++})(\vec{v}) = \vec{v} + \vec{e}_{2k+i}$, $h'(w_i\texttt{--})(\vec{v}) = \vec{v} - \vec{e}_{2k+i}$
- $h'(\#)(\vec{v}) = \vec{v} + \sum_{i=1}^{k} \vec{e}_{k+i}$

Intuitively, a word in R' is just a word in R backward, so that suffixes become prefixes, this way, the codirected constraint becomes a directed constraint. Besides, observe that as in the case of *RIPR*, only the last reset on a component has an importance: $h(\sigma_1 r_i \sigma_2)(\vec{0})(i) = h(r_i\sigma_2)(\vec{0})(i)$. In order to keep track of resets on dimension i for $i \in [k]$, component $k+i$ is equal to 1 when no reset has occurred on the i-th component and 0 otherwise. Component $2k + i$ behaves as the boolean negation of component $k + i$. Together, component $k + i$ and $2k + i$ are able to simulate zero-tests: only one of the two can be decreased. Letters $u_i\texttt{++}$ and $u_i\texttt{--}$ are used to modify component $k + i$, while letters $v_i\texttt{++}$ and $v_i\texttt{--}$ are used to modify component $2k + i$.

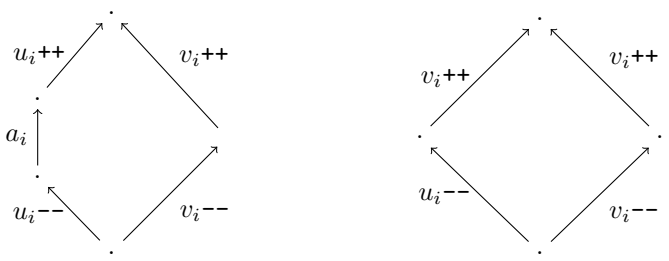

Figure 2: Diamonds to recognize the expressions e_1 and e_2

Correction of the reduction:

$$\exists \gamma \text{ solution to the original } RIPR_{codir} \text{ instance } \Leftrightarrow \exists \sigma \text{ solution to the } RIP_{dir} \text{ instance}$$

($\Rightarrow$) This direction simply is a simulation: we inductively define a function $f : \Sigma'^* \to \Sigma''^*$ such that for any $\gamma \in \Sigma'^*$ which is a solution to the $RIPR_{codir}$ instance, $\#f(\gamma)$ is a solution to the RIP_{dir} instance. f is defined as follows: for any $i \in [k]$

- $f(\epsilon) = \epsilon$
- Given $\gamma \in \Sigma'^*$ with no occurrence of r_i and $a_i \in \Sigma_i$, $f(a_i \cdot \gamma) = f(\gamma) \cdot u_i\texttt{--}a_iu_i\texttt{++}$
- Given $\gamma \in \Sigma'^*$ in which r_i occurs at least once and $a_i \in \Sigma_i$, $f(a_i \cdot \gamma) = f(\gamma) \cdot v_i\texttt{--}v_i\texttt{++}$
- Given $\gamma \in \Sigma'^*$ with no occurrence of r_i, $f(r_i \cdot \gamma) = f(\gamma) \cdot u_i\texttt{--}v_i\texttt{++}$
- Given $\gamma \in \Sigma'^*$ in which r_i occurs at least once, $f(ri \cdot \gamma) = f(\gamma) \cdot v_i\texttt{--}v_i\texttt{++}$

Now, to show that f maps solutions to solutions, we show by induction on γ the following five invariants: for any $i \in [k]$

Invariant 1 If r_i does not occur in γ then $h'(\#f(\gamma))(\vec{0})(k+i) = 1$

Invariant 2 If r_i occurs in γ then $h'(\#f(\gamma))(\vec{0})(k+i) = 0$

Invariant 3 $h'(\#f(\gamma))(\vec{0})(k+i) = 1 - h'(\#f(\gamma))(\vec{0})(2k+i)$

Invariant 4 $h(\gamma)(\vec{0})(i) = h'(\#f(\gamma))(\vec{0})(i)$

Invariant 5 If every suffix γ' of γ satisfies $h(\gamma)(\vec{0}) \geq \vec{0}$ then every prefix of $\#f(\gamma)$ satisfies $h'(\#f(\gamma))(\vec{0}) \geq \vec{0}$.

The last invariant applied to some $\gamma \in R$ clearly entails this direction of the correction of the reduction.

- If γ is empty, the property holds since $h'(\#)(\vec{0})$ satisfies the five invariants.
- If r_i does not occur in γ, then $h'(\#f(a_i\gamma))(\vec{0}) = h'(u_i\texttt{--}a_iu_i\texttt{++})(\vec{0}) + h'(\#f(\gamma))(\vec{0})$. Since r_i does not occur in γ, $h'(\#f(\gamma))(\vec{0})(k+i) = 1$ by induction hypothesis. Therefore $h'(\#f(a_i\gamma))(\vec{0})(k+i) = 1$, hence $h'(\#f(\gamma))(\vec{0})(k+j) = h'(\#f(a_i\gamma))(\vec{0})(k+j)$ for any $j \in [2k]$, which gives the three first invariant by induction hypothesis. Moreover, r_i does not occur in γ, which means that $h(a_i\gamma)(\vec{0})(i) = h(a_i)(\vec{0}) + h(\gamma)(\vec{0})$, and the same relation holds for h', thus the forth invariant holds. Finally, any component except the i-th and $(k+i)$-th ones are positive by induction hypothesis, the i-th one is positive assuming that every suffix of $a_i\gamma$ makes h positive by invariant **4**, and as noted earlier $h'(\#f(\gamma))(\vec{0})(k+i) = 1$, which ensures that $h'(\#f(\gamma) \cdot u_i\texttt{--})(\vec{0})(k+i) = 0 \geq 0$.
- If r_i occurs in γ, then $h'(\#f(a_i\gamma))(\vec{0}) = \vec{0} + h'(\#f(\gamma))(\vec{0})$, thus the three first invariant holds by induction hypothesis. Moreover, $h(a_i\gamma)(\vec{0}) = h(\gamma)(\vec{0})$. Formally, $\gamma = \gamma_1 r_i \gamma_2$, and $h(a_i\gamma_1 r_i\gamma_2)(\vec{0}) = h(r_1\gamma_2)(h(a_i\gamma_1)(\vec{0})) = h(\gamma_2)(\vec{0}) = h(\gamma)(\vec{0})$. Hence the forth invariant holds. Finally, since r_i occurs in γ, invariant **2** and **3** of the induction hypothesis gives $h'(\#f(\gamma))(\vec{0})(2k+i) = 1$, which justifies that h' stays positive under the action of $v_i\texttt{--}v_i\texttt{++}$.

- If r_i does not occur in γ, then $h'(\#f(r_i\gamma))(\vec{0}) = \vec{e}_{2k+i} - \vec{e}_{k+i} + h'(\#f(\gamma))(\vec{0})$. Thus, the three first invariants hold. Moreover, $h(r_i\gamma)(\vec{0}) = h(\gamma)(\vec{0})$ and the first k components are unchanged in h', so the forth invariant holds. Finally, since r_i does not occur in γ, $h'(\#f(\gamma))(\vec{0})(k+i) = 1$, which entails the last invariant.
- If r_i occurs in γ, then $h'(\#f(r_i\gamma))(\vec{0}) = \vec{0} + h'(\#f(\gamma))(\vec{0})$, hence the three first invariants hold, and for the same reason as in the previous case, the forth one holds as well. Finally, since r_i occurs in γ, $h'(\#f(\gamma))(\vec{0})(2k+i) = 1$, which justifies that h' remains positive under the action of $v_i\texttt{--}v_i\texttt{++}$.

($\Leftarrow$) Given $\mathcal{A}$ an automaton that recognizes R, it is easy to build an automaton $\mathcal{A}'$ that recognizes R', by reversing the direction of every edge in $\mathcal{A}$, swapping initial and final state, and then replacing every edge by the corresponding "diamond" from figure 2. Now define $\mathcal{P}(\mathcal{A})$ the set of all runs in $\mathcal{A}$ that are suffix of some accepting run, *i.e.* the set of runs recognizing $\mathrm{Suff}(R)$. Similarly, define $\mathcal{P}(\mathcal{A}')$ the set of all runs in $\mathcal{A}'$ that are **prefix** of some accepting run, **and** that moreover do not end inside a diamond. In this configuration, there is a natural map $f : \mathcal{P}(\mathcal{A}') \to \mathcal{P}(\mathcal{A})$: a part of a run using either the left or right branch of a diamond is mapped to the original edge of $\mathcal{A}$ that was replaced by this diamond in $\mathcal{A}'$. A run τ of length n in $\mathcal{P}(\mathcal{A})$ has exactly 2^n antecedents by f, since every step in τ corresponds to two possible paths in $\mathcal{A}'$ (the two branches of the diamond). Also define $\mathcal{A}'(\rho) \in \Sigma''^*$ as the word read by the run $\rho \in \mathcal{P}(\mathcal{A}')$ in the automaton $\mathcal{A}'$, and similarly $\mathcal{A}(\tau) \in \Sigma'^*$. Note that f is surjective, thus every word in $\mathrm{Suff}(R)$ can be obtained as $\mathcal{A}'(f(\rho))$ for some $\rho \in \mathcal{P}(\mathcal{A}')$.

Observe that there is no such mapping from Σ''^* to Σ'^*, which motivates that we work on runs, and not simply on words. For instance, if $R = a_i + r_i + b_i$, then in the corresponding $\mathcal{A}'$, the word $\sigma = v_i\texttt{--} \cdot v_i\texttt{++}$ is recognized by three different runs: the one using the right branch in the first diamond, the second one using the right branch in the second diamond, and the third one using the right branch in the third diamond. The images by f of these three runs read in $\mathcal{A}$ the three different words a_i, r_i and b_i. Therefore, each **accepting run** in $\mathcal{A}'$ reading a solution to the RIP_{dir} instance can be mapped to a solution to the $RIPR_{codir}$ instance, but a solution to the RIP_{dir} instance can be mapped to potentially several solutions. More precisely, we show that if a run $\rho \in \mathcal{A}'$ such that $\#\mathcal{A}'(\rho)$ is a solution to the RIP_{dir} instance, then $\mathcal{A}(f(\rho))$ is a solution to the original $RIPR_{codir}$ instance.

Given $\sigma \in R'$ such that $h'(\#\mu)(\vec{0}) \geq \vec{0}$ for any $\mu \in \mathrm{Pref}(\sigma)$, and a run ρ in $\mathcal{A}'$ recognizing σ, let $\tau = f(\rho)$ and $\gamma = \mathcal{A}(\tau)$. The following invariants hold: for all $\rho' \in \mathcal{P}(\mathcal{A}')$ prefix of ρ, let $\mu = \mathcal{A}'(\rho')$ and $\nu = \mathcal{A}(f(\rho'))$, and for any number $i \in [k]$:

Invariant 1 $h'(\#\mu)(\vec{0})(k+i) \in \{0, 1\}$

Invariant 2 $h'(\#\mu)(\vec{0})(k+i) = 1 - h(\#\mu)(\vec{0})(2k+i)$

Invariant 3 $h'(\#\mu)(\vec{0})(k+i) = 0$ if and only if ν) does not contain the letter $r_i \in \Sigma'$

Invariant 4 $h'(\#\mu)(\vec{0})(i) = h(\nu)(\vec{0})(i)$

Once these invariants proved, the forth invariant entails what needs to be proved: since $\#\sigma$ is a directed solution, $h'(\#\mu)(\vec{0}) \geq \vec{0}$, which means $h(\nu)(\vec{0}) \geq \vec{0}$. Since by construction of $\mathcal{A}'$, every suffix of γ is indeed obtained this way, γ is a codirected solution.

We show these four invariants by recursion on the length of μ. Initially, all four invariants hold for $h(\#)(\vec{0})$. If they hold for a prefix μ, then we must show that they hold for

1. $\mu_1 = \mu \cdot u_i\texttt{--} \cdot a_i \cdot u_i\texttt{++}$,
2. $\mu_2 = \mu \cdot v_i\texttt{--} \cdot v_i\texttt{++}$,

3. $\mu_3 = \mu \cdot u_i\texttt{--} \cdot v_i\texttt{++}$

1. By induction hypothesis, we know $h'(\#\mu)(\vec{0})(k+i) \in \{0,1\}$. If it is equal to 0, then $u_i\texttt{--}$ cannot be read: we would have $h(\#\mu u_i\texttt{--})(\vec{0})(i) = -1$, which is impossible since σ is a directed solution. Hence it is equal to 1, and $h(\#\mu)(\vec{0})$ and $h(\#\mu \cdot r_i\texttt{--} \cdot a_i \cdot r_i\texttt{++})(\vec{0})$ differs only on the i-th component, thus the three first invariants hold for μ_1. Moreover, ν_1 can only be $a_i \cdot \nu$, and since r_i does not occur in ν by induction hypothesis, the forth invariant is satisfied by μ_1.

2. This time, for $h'(\#\mu_2)(\vec{0})(2k+i)$ to be positive, we must have $h'(\#\mu)(\vec{0})(2k+i) = 1$, and by the third invariant, r_i occurs in ν. As mentioned earlier, the word $v_i\texttt{--}v_i\texttt{++}$ can be obtained for different ν_2: $\nu_2 = a_i\nu$ for any $a_i \in \Sigma_i$, or $\nu_2 = r_i\nu$. But since r_i occurs in ν, all possible ν_2 satisfy invariant **4**.

3. The last case is handled without difficulties with the same arguments seen in the previous cases.

□

4.8 RIPR$_{dir}$ and RAIP$_{dir}$ are F$_\omega$-complete

As in the case of RIP_{dir}, $RIPR_{dir}$ simply is a reformulation of the coverability problem for VASS with resets, which is equivalent to coverability/reachability for Lossy Counter Machines. Thus $RIPR_{dir}$ is $\mathbf{F}_\omega$-complete. Besides, $RAIP_{dir}$ is, also by the same reduction, just a reformulation of the coverability problem for VASS with affine updates. Since it is known how to simulate affine updates only with resets, the two problems are equivalent, and $RAIP_{dir}$ is $\mathbf{F}_\omega$-complete as well.

4.9 RAIP is Undecidable

Undecidability of $RAIP$ has been announced in [7]. This result has been obtained by J. Reichert and has not yet appeared in written format. For the sake of completeness, here we first repeat Reichert's argument. Note that the following reduction gives undecidability not only of $RAIP$ in its general form, but even for a fixed k greater than 4, and with only diagonal matrices.

Undecidability is shown via reduction from the undecidable Post Correspondence Problem (PCP). Given $u_1, \ldots, u_n, v_1, \ldots, v_n \in \{0,1\}^*$, PCP asks whether there are some $i_1, \ldots, i_p$ ($p > 0$) such that $u_{i_1} \cdots u_{i_p} = v_{i_1} \cdots v_{i_p}$. Below, we define a $RAIP$ instance $(4, \{0, 1, \tilde{0}, \tilde{1}, \#\}, h, R)$ that has a solution if, and only if, there is a solution to the above PCP instance:

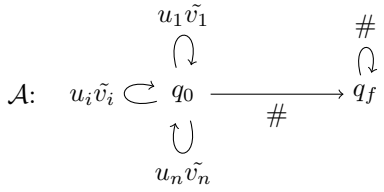

Figure 3: The automaton $\mathcal{A}$ recognizing R

$\mathcal{A}$ is an automaton describing the regular language R. It has n self-loops on q_0, and each of these loops is labeled by a word $w = u_i\tilde{v_i}$. This, of course, actually corresponds to a path with $|w|$ states such that the

path reads w. We now define h as:

$$h(0)(\vec{v}) = \begin{pmatrix} 2&0&0&0\\0&1&0&0\\0&0&2&0\\0&0&0&1 \end{pmatrix}\vec{v} \qquad h(\tilde{0})(\vec{v}) = \begin{pmatrix} 1&0&0&0\\0&2&0&0\\0&0&1&0\\0&0&0&2 \end{pmatrix}\vec{v}$$

$$h(1)(\vec{v}) = \begin{pmatrix} 2&0&0&0\\0&1&0&0\\0&0&2&0\\0&0&0&1 \end{pmatrix}\vec{v} + \begin{pmatrix} 1\\0\\-1\\0 \end{pmatrix} \qquad h(\tilde{1})(\vec{v}) = \begin{pmatrix} 1&0&0&0\\0&2&0&0\\0&0&1&0\\0&0&0&2 \end{pmatrix}\vec{v} + \begin{pmatrix} 0\\1\\0\\-1 \end{pmatrix}$$

$$h(\#)(\vec{v}) = \vec{v} + \begin{pmatrix} -1\\-1\\1\\1 \end{pmatrix}$$

The idea is that a word $\sigma \in R$ must end by a sequence of #. Moreover, h is such that for any word σ, the two first component of $h(\sigma)(\vec{0})$ are the opposite of the two last component. Therefore, the only way to obtain $h(\sigma)(\vec{0}) \geq \vec{0}$ is by having $h(\sigma)(\vec{0}) = \vec{0}$. This technique of doubling the dimension to handle reachability with coverability can actually be generalized to any $\mathbb{Z}$-counter machine, as proved in [12]. From these two observations we deduce that a run ρ reading a word σ in $\mathcal{A}$ must make the two first components equal (thus the two last ones as well) when leaving state q_0. In other words, the largest prefix σ' of σ that does not contain the symbol # must be such that $h(\sigma')(\vec{0})(1) = h(\sigma')(\vec{0})(2)$. Thinking of counter values encoded in binary, the counters represent the concatenation of the u_i and the v_i, respectively, since in binary, multiplying by 2 corresponds to concatenating 0, and multiplying by 2 and adding 1 corresponds to concatenating 1. Looping non-deterministically on q_0, the machine "guesses" an order to make the two counters, *i.e.*, words, equal.

Note that the only matrices appearing in $\mathcal{A}$ are diagonal, and of dimension 4. Thus *RAIP* is already undecidable for diagonal matrices of fixed dimension 4.

4.10 EAIP is Undecidable

In this subsection, we prove that *EAIP* is undecidable, by reducing the problem *RAIP*. Given an instance (k, Σ, h, R) of *RAIP*and an automaton $\mathcal{A} = (Q, \Sigma, \Delta, q_0, F)$ for R, we consider the instance $(k + |Q| + 3, \vec{0}, \vec{0}, \Delta \cup \{\perp\}, h')$ of *EAIP*, where h' is given below. Given $\vec{v} \in \mathbb{Z}^{k+|Q|+3}$, define $\vec{v}(i)$ for $i \in [k]$ as usual, *i.e.* the i-th component of $\vec{v}$, $\vec{v}(q)$ for $q \in Q$ as the $(k+i)$-th component of $\vec{v}$, if q is the i-th state of Q, and $\vec{v}(F), \vec{v}(G)$ and $\vec{v}(H)$ for the three last components. Given $\vec{v} \in \mathbb{Z}^{k+|Q|+3}$, h' is defined as:

- Let $\vec{w} = h'(\perp)(\vec{v})$, we have: $\vec{w}(i) = 0$ for $i \in [k]$, $\vec{w}(q) = -1$ if $q = q_0$, and 0 otherwise. The three last component are given by $\vec{w}(F) = \vec{v}(F) - 1$, $\vec{w}(G) = 2\vec{v}(G)$ and $\vec{w}(H) = 1$.
- For $\delta = (p, a, p') \in \Delta$, let $\vec{w} = h'(\delta)(\vec{v})$. The k first components of $\vec{w}$ are those of $h(a)(\vec{v})$. The $|Q|$ next ones are given by $\vec{w}(q) = 2\vec{v}(q) + 2(p = q) - (p' = q)$, where given two states $p, q \in Q$, $(p = q)$ is equal to 1 if $p = q$ and 0 otherwise. Finally, $\vec{w}(F) = \vec{v}(F) + (p' \in F)$, where $q \in F$ equals 1 if q is in F, 0 otherwise; $\vec{w}(G) = 2\vec{v}(G) + \vec{v}(H) - 1$; $\vec{w}(H) = \vec{v}(H) - (p' \in F)$.

The proof of correction is not detailed. It is similar to the one given for $RIPR_{codir}$, the invariants being given by the following intuition: a word $\sigma \in \Sigma$ is a solution to the *RAIP* instance if and only if any accepting

run ρ for σ in $\mathcal{A}$ is a solution to the *EAIP* instance. The k first components in h' store the value of the morphism h. The $|Q|$ next components ensure that ρ is a valid run in $\mathcal{A}$. The F component makes sure that h' is negative until the run reaches a final state. The reduction assumes that all final states in $\mathcal{A}$ are deadlocks, *i.e.* a final state is only reached when the run is over, not before. The idea to forbid transitions is that the effect of the multiplication by two on a negative component cannot be compensate to be positive ever again, which acts as a zero test.

5 Conclusion

During this internship, we reformulated a classical coverability problem for some class of (positive) counter machines as a variant of the Post Correspondence Problem, where we look at the effect of words in a regular language on a morphism h ranging from words to vectors of integers. Although the two problems are the same, this new point of view highlighted new variants of counter machines and counter machines problems. The first variant is the class of counter machines where counters are allowed to take negative values. Even if the idea does not seem exotic, the class have not received much attention. We filled this gap by submitting an article, *Integer Vector Addition Systems with States* [12] whose first main result is the NP upper bound for reachability and coverability of so called $\mathbb{Z}$-VASS with resets, *i.e.* VASS whose counters can take arbitrary values in $\mathbb{Z}$, equipped with an operation of reset. Not only the low complexity compared to the natural number version, which is Ackermann-complete for coverability, and undecidable for reachability, is surprising, but also the fact that adding resets to the model do not increase the complexity, since reachability and coverability for regular $\mathbb{Z}$-VASS already is NP-complete. The second main result of this article is the exact complexity of the inclusion problem for $\mathbb{Z}$-VASS (with or without resets).

The second variant is the codirected versions of the problems. Although it seems natural to wonder what effects replacing prefixes by suffixes can have in the word morphism formulation, it is unlikely to think of the codirected problems from the counter machine point of view. And indeed, these problems have never been considered. This new formulation of a known problem therefore allowed new variations. However, the problems are not motivated (yet ?), which is why they are not mentioned in the article.

Moreover, many more problems are suggested by this single reformulation. The most challenging one would be to find a class $\mathcal{A}$ of matrices that makes $RAIP(\mathcal{A})$ $\mathbf{F}_\omega$-complete. We know that for any $\{I_k - E_i \mid i \in [k]\} \subseteq \mathcal{A} \subseteq \mathcal{D}_k$ (the set of diagonal matrices), $RAIP(\mathcal{A})_{\text{dir}}$ is a $\mathbf{F}_\omega$-complete problem, and $RAIP(\mathcal{A})$ is NP-hard. It would be interesting to find such a class for which $RAIP(\mathcal{A})$ is a $\mathbf{F}_\omega$-complete problem, it would satisfy the same equivalence as between *PEP* and PEP_{dir}. One good candidate could be the set of matrices with only one "1" per row and column, zeros elsewhere.

We also let some problems unsolved, as it is the case for $RAIP_{codir}$, but also *RAIP* for a fixed dimension of 2 or 3: we established undecidability for a fixed dimension of at least 4, and the one-dimensional problem is known to be PSPACE-complete [7]. This idea of fixing the dimension could be applied to problems considered: what happens to the complexity of *RIP* if the natural number k is no longer part of the input ? An other open problem has been suggested in [12], where the inclusion problem for the stateless version of $\mathbb{Z}$-VASS has been given a NP lower bound and a Π_2^P (second level of the polynomial hierarchy) upper bound.

Finally, I have spent the last month of my internship working on a different problem: the trace universality for labeled VASS. Given a VASS $\mathcal{V}$ whose transition are labeled by letters from an alphabet Σ, the trace set of $\mathcal{V}$ noted $\mathcal{T}(\mathcal{V}) \subset \pm^*$ is the set of words w such that there exists a valid run in $\mathcal{V}$ labeled by w. The problem has first been proved decidable in [14] using generic WSTS techniques, then a $\mathbf{F}_\omega$ lower bound has been given in [13] by reduction from lossy counter machines, and finally, generic wqo analysis gives a $\mathbf{F}_{\omega^\omega}$ membership. I have tried alternatively to prove either a $\mathbf{F}_{\omega^\omega}$ lower bound or a $\mathbf{F}_\omega$ upper bound, without success. However, I am now convinced that the problem actually is $\mathbf{F}_\omega$-complete, but can not yet provide

a proof for it. This would make this problem particularly useful since it would be a member of the $\mathbf{F}_\omega$-complete class that requires a finer algorithmic analysis than just bounding the bad sequences. It may then help concluding on the complexity of other problems for which the wqo analysis do not give tight upper bounds.

References

[1] Parosh Aziz Abdulla. Well (and better) quasi-ordered transition systems. *Bulletin of Symbolic Logic*, 16(4):457–515, 2010.

[2] Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Inf. Comput.*, 160(1-2):109–127, 2000.

[3] Nathalie Bertrand and Philippe Schnoebelen. Computable fixpoints in well-structured symbolic model checking. *Formal Methods in System Design*, 43(2):233–267, 2013.

[4] I. Borosh and L.B. Treybing. Bounds on positive integral solutions of linear Diophantine equations. *Proc. AMS*, 55:299–304, 1976.

[5] Pierre Chambart and Ph. Schnoebelen. Post embedding problem is not primitive recursive, with applications to channel systems. In *FSTTCS*, pages 265–276, 2007.

[6] Pierre Chambart and Ph. Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *LICS*, pages 205–216, 2008.

[7] A. Finkel, S. Göller, and C. Haase. Reachability in register machines with polynomial updates. In *Proc. MFCS*, volume 8087 of *LNCS*, pages 409–420, 2013.

[8] Alain Finkel. A generalization of the procedure of karp and miller to well structured transition systems. In *ICALP*, pages 499–508, 1987.

[9] Alain Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.

[10] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[11] S. Ginsburg and E.H. Spanier. Semigroups, Presburger formulas and languages. *Pac. J. Math.*, 16(2):285–296, 1966.

[12] Christoph Haase and Simon Halfon. Integer vector addition systems. *CoRR*, abs/1406.2590, 2014.

[13] Piotr Hofman and Patrick Totzke. Trace inclusion for one-counter nets revisited. *CoRR*, abs/1404.5157, 2014.

[14] Petr Jančar, Javier Esparza, and Faron Moller. Petri nets and regular processes. *Journal of Computer and System Sciences*, 59(3):476 – 503, 1999.

[15] Wainer S.S. Löb, M.H. Hierarchies of number-theoretic functions. ii. *Archiv für mathematische Logik und Grundlagenforschung*, 13:97–113, 1970.

[16] E. W. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. STOC*, pages 238–246, New York, NY, USA, 1981. ACM.

[17] C. Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6(2):223–231, 1978.

[18] Sylvain Schmitz. Complexity hierarchies beyond elementary. *CoRR*, abs/1312.5686, 2013.

[19] Ph. Schnoebelen. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In *Proc. MFCS*, volume 6281 of *LNCS*, pages 616–628, 2010.

[20] H. Seidl, Th. Schwentick, A. Muscholl, and P. Habermehl. Counting in trees for free. In *Proc. ICALP*, volume 3142 of *LNCS*, pages 1136–1149, 2004.