The Well Structured Problem for Presburger Counter Machines

Alain Finkel, Ekanshdeep Gupta

The Well Structured Problem for Presburger Counter Machines

Alain Finkel
LSV, ENS Paris-Saclay, CNRS, Université Paris-Saclay, France
UMI ReLaX
alain.finkel@ens-paris-saclay.fr

Ekanshdeep Gupta
Chennai Mathematical Institute, Chennai, India
UMI ReLaX
ekanshdeep@cmi.ac.in

Abstract

We introduce the *well structured problem* as the question of whether a model (here a counter machine) is well structured (here for the usual ordering on integers). We show that it is undecidable for most of the (Presburger-defined) counter machines except for Affine VASS of dimension one. However, the *strong well structured problem* is decidable for *all* Presburger counter machines. While Affine VASS of dimension one are not, in general, well structured, we give an algorithm that computes the set of predecessors of a configuration; as a consequence this allows to decide the well structured problem for 1-Affine VASS.

2012 ACM Subject Classification Theory of computation → Logic and verification; Theory of computation → Verification by model checking

Keywords and phrases Well structured transition systems, infinite state systems, Presburger counter machines, reachability, coverability

Acknowledgements The work reported was carried out in the framework of ReLaX, UMI2000 (ENS Paris-Saclay, CNRS, Univ. Bordeaux, CMI, IMSc). This work was also supported by the grant ANR-17-CE40-0028 of the French National Research Agency ANR (project BRAVAS).

1 Introduction

Context: Well Structured Transition Systems (WSTS) [9, 8] are a well-known model to solve termination, boundedness, control-state reachability and coverability problems. It is well known that Petri nets and Vector Addition Systems with States (VASS) are WSTS and that Minsky machines are not WSTS. But the characterization of counter machines which are well structured (resp. with strong monotony) is surprisingly unknown. Moreover, given a counter machine, can we decide whether it is well structured (resp. with strong monotony)? These questions are relevant since a positive answer could allow to verify *particular instances* of undecidable models like Minsky machines and counter machines. In this paper, we consider *Presburger counter machines (PCM)* where each transition between two control-states is labelled by a Presburger formula which describes how each counter is modified by the firing of the transition. The PCM model includes Petri nets, Minsky machines and most of the counter machine models studied in the literature, for example counter machines where transitions between control-states are given by affine functions having Presburger domains [3, 11].

Affine VASS: In an Affine VASS (AVASS), transitions between control-states are labelled by affine functions whose matrices have elements in $\mathbb{Z}$ (and not in $\mathbb{N}$ as usual). AVASS extends VASS (where transitions are translations) and positive affine VASS (introduced as self-modified nets in [22] and studied as affine well structured nets in [13]. [4] extends the Rackoff technique to AVASS where all matrices are larger than the identity matrix: for this subclass, coverability and boundedness are shown in EXPSPACE. The variation of VASS which may go below 0, called $\mathbb{Z}$-VASS, is studied in [15] and for their extension, *$\mathbb{Z}$-Affine* VASS, reachability is shown NP-complete for VASS with resets, PSPACE-complete for VASS with transfers and undecidable in general [2, 1]; let us remark that all $\mathbb{Z}$-Affine VASS have *positive* matrices.

Moreover AVASS allow the simulation of the zero-test so they are at least as expressive as Minsky machines. But for dimension one, AVASS are more expressive than Minsky machines: in fact, $Post^*$ is computable as a Presburger formula for 1-counter Minsky machines but this is not the case for 1-AVASS which can generate the set of all the powers of 2 (this set is not the solution of any Presburger formula).

The computation of the set Pre^* of all predecessors of a configuration is effective for 2-VASS (extended with one zero-test and resets) [12] as a Presburger formula and for pushdown automata [5] as a regular language. But the computation of Pre^* fails for 3-VASS and for Pushdown VAS since Pre^* is neither semilinear nor regular [19].

Our contributions:

We introduce two new problems related to well structured systems and Presburger counter machines. The so-called *well structured problem*: (1) given a PCM, is it a WSTS? and the *strong well structured problem*: (2) given a PCM, is it a WSTS with *strong* monotony?

We prove that the well structured problem is undecidable for PCM even if restricted to dimension one (1-PCM) with just Presburger functions (i.e. piecewise affine functions); undecidability is also verified for Affine VASS in dimension two (2-Affine VASS). The undecidability proofs use the fact that Minsky machines can be simulated by both 1-PCM and 2-Affine VASS. However, we prove the decidability of the well structured problem for 1-Affine VASS (which subsumes 1-Minsky machines). Since the strong monotony can be expressed as a Presburger formula, the strong well structured problem is decidable for *all* PCMs. These results are summarised below:

	Well Structured Problem	Strong Well Structured Problem
PCM	U	**D**
Functional 1-PCM	**U** [Theorem 14]	D
2-AVASS	U	D
2-Minsky machines	**U** [Theorem 15]	D
1-AVASS	**D** [Theorem 26]	D

We give an algorithm that computes Pre^* of a 1-AVASS and this extends a similar known result for 1-Minsky machines and 1-VASS (and for pushdown automata [5]). The computation of Pre^* allows us to give a simple proof that reachability and coverability are decidable for 1-AVASS (in fact reachability is known to be PSPACE-complete for polynomial one-register machines [10] which contains 1-AVASS). Moreover, the computation of Pre^* allows to decide the well structured problem for 1-AVASS. These results are summarised below:

	Reachability	Coverability
1-PCM (functional)	U	**U** [Corollary 19]
1-AVASS	**D** [Corollary 24]	D
d-totally positive AVASS	**D** [Theorem 29]	D
d-positive AVASS ($d \geq 2$)	**U** [Theorem 28]	**D** [WSTS]
2-AVASS	U	**U** [Corollary 18]

Outline: We introduce in Section 2 two models, well structured transition systems (WSTS) and Presburger counter machines (PCM); we show that the property for an ordering to be well is undecidable. Section 3 analyses the decidability of the well structured problems for many classes of PCM and Affine VASS. Section 4 studies the decidability of reachability and coverability for the classes studied in Section 3.

2 Counter machines and WSTS

A relation $\leq$ on a set E is a *quasi ordering* if it is reflexive and transitive; it is an ordering if moreover $\leq$ is antisymetric. A quasi ordering $\leq$ on E is a *well quasi ordering (wqo)* if for all infinite sequences of elements of E, $(e_i)_{i\in\mathbb{N}}$, there exists two indices $i<j$ such that $e_i \leq e_j$. For an ordered set $(E,\leq)$ and a subset $X \subseteq E$, the upward closure of X denoted by $\uparrow X$ is defined as follows: $\uparrow X = \{x \mid \exists y \in X \text{ such that } y \leq x\}$. X is said to be *upward closed* if $X = \uparrow X$.

2.1 Arithmetic counter machines

A *d-dim arithmetic counter machine (short, d-arithmetic counter machine or an arithmetic counter machine)* is a tuple $M = (Q, \Phi, \rightarrow)$ where Q is a finite set of control-states, Φ is a set of logical formulae with $2d$ free variables $x_1, ..., x_d, x'_1, ..., x'_d$ and $\rightarrow \subseteq Q \times \Phi \times Q$ is the transition relation between control-states. We can also without loss of generality assume that $\rightarrow$ covers Φ, i.e. Φ does not have unnecessary formulae. A configuration of M refers to an element of $Q \times \mathbb{N}^d$. The operational semantics of a d-arithmetic counter machine M is a transition system $S_M = (Q \times \mathbb{N}^d, \rightarrow)$ where $\rightarrow \subseteq (Q \times \mathbb{N}^d) \times (Q \times \mathbb{N}^d)$ is the transition relation between configurations. For a transition (q, ϕ, q') in M, we have a transition $(q; x_1, ..., x_d) \rightarrow (q'; x'_1, ..., x'_d)$ in S_M iff $\phi(x_1, ..., x_d, x'_1, ..., x'_d)$ holds. Note that we are slightly abusing notation by using the same $\rightarrow$ for both M and S_M. We may omit Φ from the definition of a counter machine if it is clear from context.

A d-dim arithmetic counter machine M with *initial configuration* c_0 is defined by the tuple $M = (Q, \Phi, \rightarrow, c_0)$ where $(Q, \Phi, \rightarrow)$ is a d-arithmetic counter machine and $c_0 \in Q \times \mathbb{N}^d$ is the initial configuration. An arithmetic counter machine is *effective* if the transition relation is decidable (there is a decidable procedure to determine if there is a transition $x \rightarrow y$ between any two configurations x, y) and this is the case when it is given by an algorithm, a recursive relation, or decidable first order formulae (for instance Presburger formulae). An arithmetic counter machine is said to be *functional* if each formula in Φ that labels a transition in M defines a partial function.

Most usual counter machines can be expressed with Presburger formulae. It is well known that Presburger arithmetic with congruence relations without quantifiers is equivalent in expressive power to standard Presburger arithmetic [14].

▶ **Definition 1.** *A* Presburger counter machine (PCM) *is an arithmetic counter machine $M = (Q, \Phi, \rightarrow)$ such that Φ is a set of Presburger formulae with congruence relations without quantifiers.*

▶ **Proposition 2.** *[6] (proof in Appendix) The property for a d-dim PCM to be* functional *is decidable in NP.*

Minsky machines with d counters are d-PCM $M = (Q, \Phi, \rightarrow)$ where Φ consists of either translations with upwards closed guards, or formulae of the form $\wedge_{i=1}^{d}(x_i = x'_i) \wedge x_k = 0$ for varying k (zero-tests). *Vector Addition Systems with States (VASS)* are Minsky machines

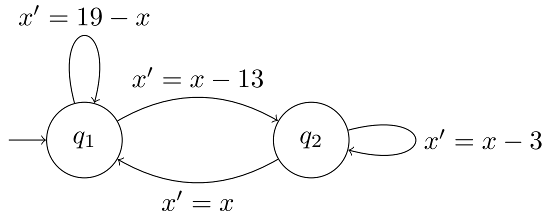


Figure 1 The counter machine M_1

without zero-tests. An *Affine VASS* with d counters (d-AVASS) is a d-PCM where each transition is labelled by a formula equivalent to an affine function of the form $f(x) = Ax + b$ where $A \in M_d(\mathbb{Z})$ is a $d \times d$ matrix over $\mathbb{Z}$ and $b \in \mathbb{Z}^d$. The domain of such a function would be the (Presburger) set of all $x \in \mathbb{N}^d$ such that $Ax + b \in \mathbb{N}^d$. For convenience, we will denote d-AVASS transitions by a pair $(A, b) \in M_d(\mathbb{Z}) \times \mathbb{Z}^d$. Note that AVASS is an extension of VASS where transitions are not labelled by vectors but by affine functions (A_i, b_i). Let us define positive and totally-positive AVASS. A *positive* AVASS S is an AVASS such that every matrix A_i of S is positive. This model has been studied for instance in [13]. A *totally-positive* AVASS S is a positive AVASS such that every vector b_i of S is positive. For totally positive AVASS, an instance of the boundedness problem has been shown decidable in [13]. Note that we say something is positive if it is greater than or equal to 0, not strictly greater than 0.

▶ **Example 3.** The machine M_1 in Figure 1 is a 1-AVASS but it is not a 1-VASS because there is a negative transition from q_1 to q_1.

▶ **Proposition 4.** *[6] Checking whether a given PCM is a VASS, AVASS, positive AVASS or a totally positive AVASS is decidable.*

2.2 Well structured transition systems

A *transition system* is a tuple $S = (X, \rightarrow)$ where X is a (potentially infinite) set of configurations and $\rightarrow \subseteq X \times X$ is the transition relation between configurations. We denote by $\xrightarrow{*}$ the reflexive and transitive closure of $\rightarrow$. For a subset $S \subseteq X$, we denote by $Pre(S) := \{t \mid t \rightarrow s \text{ for some } s \in S\}$, and $Pre^*(S) := \{t \mid t \xrightarrow{*} s \text{ for some } s \in S\}$. Similarly for $Post(S)$ and $Post^*(S)$.

An *ordered transition system* $S = (X, \rightarrow, \leq)$ is a transition system $(X, \rightarrow)$ with a quasi-ordering $\leq$ on X. Given two configurations $x, y \in X$, x is said to *cover* y if there exists a configuration $y' \geq y$ such that $x \xrightarrow{*} y'$. An ordered transition system $S = (X, \rightarrow, \leq)$ is *monotone*, if for all configurations $s, t, s' \in X$ such that $s \rightarrow t$, $s' \geq s$ implies that s' covers t. S is *strongly monotone* if for all configurations $s, t, s' \in X$ such that $s \rightarrow t$, $s' \geq s$ implies that there exists $t' \geq t$ such that $s' \rightarrow t'$.

▶ **Definition 5.** *[8] A* well structured transition system *(WSTS) is an ordered transition system $S = (X, \rightarrow, \leq)$ such that $(X, \leq)$ is a wqo and S is monotone.*

The *coverability problem* is to determine, given two configurations s and t, whether there exists a configuration t' such that $s \xrightarrow{*} t' \geq t$ (s covers t). This problem is one often studied alongside well-structuredness.

Let us consider the usual wqo $\leq$ on $Q \times \mathbb{N}^d$ associated with a d-counter machine $M = (Q, \rightarrow)$: $(q_1; x_1, x_2, ..., x_d) \leq (q_2; y_1, ..., y_d) \iff (q_1 = q_2) \wedge (\wedge_{i=1}^{d} x_i \leq y_i)$.

We say that an arithmetic counter machine $M = (Q, \Phi, \to)$ is *well structured* (or is a WSTS) iff its associated transition system S_M is a WSTS under the usual ordering. Since the usual ordering on $(Q \times \mathbb{N}^d, \leq)$ is a wqo, let us remark that the associated ordered transition system $S_M = (Q \times \mathbb{N}^d, \to, \leq)$ is a WSTS iff S_M is monotone.

Given a counter machine $M = (Q, \to)$, the *control-state reachability problem* is that given a configuration $(q; n_1, ..., n_d)$, and a control-state q' whether there exist values of counters $(m_1, ..., m_d)$ such that $(q; n_1, ..., n_d) \xrightarrow{*} (q'; m_1, ..., m_d)$. In this case, we often say that q' is *reachable* from $(q; n_1, ..., n_d)$.

We introduce two new problems related to WSTS and Presburger counter machines.

- The *well structured problem*: given a PCM, is it a WSTS?
- The *strong well structured problem*: given a PCM, is it a WSTS with strong monotony?

▶ **Example 6.** The machine M_1 (Figure 1) is not strongly monotone since we have: $(q_1, 0) \xrightarrow{x'=19-x} (q_1, 19)$. However, we see that $Post^*(q_1, 10) = \{(q_1, 9), (q_1, 10)\}$. Therefore we can deduce that $(q_1, 10)$ cannot cover $(q_1, 19)$. Hence M_1 is not well structured. We give, in Section 4, an algorithm for deciding whether a 1-AVASS is well structured.

It is shown in [8] that *almost* every transition system can be turned into a WSTS for the *termination ordering* which is not, in general, decidable. So the problem is not only to decide whether a system is a WSTS in general; we have to choose a *decidable* ordering. We show that deciding whether arbitrary (non-effective) transition systems are well-structured for the usual (decidable) ordering on natural numbers is undecidable.

▶ **Proposition 7.** *(proof in Appendix) The well structured problem for* 1*-arithmetic counter machines is undecidable.*

We now show that restricting to *effective* transition systems does not allow us to decide the property of being a WSTS.

▶ **Corollary 8.** *The well structured problem (for the usual ordering on* $\mathbb{N}$*) for effective transition systems whose set of configurations is included in* $\mathbb{N}$ *is undecidable.*

Proof. There exists a reduction from the Halting Problem as follows:

Given a Turing machine M, we define a transition system $S_M = (\mathbb{N}, \to_M)$ as follows: If $(m = 0) \vee (M$ does not halt in m steps), then, for all n, there is a transition $m \to_M n$. Hence this transition relation $\to_M$ is decidable. Now, if M does not halt, then there is a transition $m \to_M n$ for all $m, n \in \mathbb{N}$. This satisfies monotony, hence in this case, S_M is a WSTS. However, if M halts in exactly m steps, then there is no transition from $m + 1$ but there is, in any case, a transition from 0 to n for all n. Hence in this case, S_M is not a WSTS. Therefore, S_M is a WSTS iff T does not halt. ◀

2.3 Testing whether an ordering is well

In the previous results, the usual well ordering on natural numbers is not necessarily the unique *decidable* ordering when considering the well structured problem for counter machines. Let $\leq$ be a decidable quasi ordering relation on $\mathbb{N}^d$. If we are interested in whether a counter machine with this ordering is WSTS, it raises the natural question of whether we can decide if $\leq$ is a wqo. Unfortunately, but unsurprisingly, we first show that this property is undecidable in dimension one $(d = 1)$.

▶ **Proposition 9.** *(proof in Appendix) The property for a decidable ordering on* $\mathbb{N}$ *to be a well ordering is undecidable.*

Let us study the case of *Presburger-definable orderings* in $\mathbb{N}$. Among many equivalent characterizations of wqo, we know that a quasi ordering is well iff it satisfies well-foundedness and the finite anti-chain property. Both of these properties can be expressed using monadic second order variables. But, it is shown in [17] that Presburger Arithmetic with a single monadic variable becomes undecidable. Hence, this cannot directly be used to check if a Presburger-definable ordering is a *wqo*. However, we still have the following result:

▶ **Proposition 10.** *(proof in Appendix) The property for a Presburger relation on $\mathbb{N}$ to be a well quasi ordering is decidable.*

3 The well structured problem for PCM

In the sequel, whenever we talk about PCM being WSTS, we will consider the usual ordering on $Q \times \mathbb{N}^d$ defined in subsection 2.2. We introduce a general technique to prove undecidability of checking whether a counter machine of some class is a WSTS. Let S_0 be the class of machines we are interested in. We will show reduction from reachability in Minsky machines.

▶ **Lemma 11.** *Suppose we have a procedure which takes a 2 counter Minsky machine with initial state $M = (Q, \rightarrow, q_0)$ and a control-state q_1 as input and generates a machine N of class S_0 which satisfies the following two requirements:*

- *All control-states in M are reachable implies N is a WSTS.* *(1)*
- *N is a WSTS implies q_1 is reachable in M from $(q_0; 0, 0)$.* *(2)*

Then, the well structured problem for S_0 is undecidable.

Proof. Suppose that the well structured problem for S_0 is decidable. We will use the above procedure to get an algorithm for Minsky machine reachability. Fix (M, q_1), where $M = (Q, \rightarrow_M, q_0)$. We want to check if q_1 is reachable from $(q_0; 0, 0)$.

Let $|Q| = n$. Consider all 2^{n-2} subsets $Q' \subseteq Q$ satisfying that $\{q_0, q_1\} \subseteq Q'$. For each such Q', let $\rightarrow_{Q'}$ denote the restriction of $\rightarrow_M$ to the set $Q' \times Q'$. Hence, we can associate a Minsky machine $M' = (Q', \rightarrow_{Q'}, q_0)$ to each such subset Q'. We call M' a *sub-machine* of M corresponding to Q'.

Now, for each sub-machine M', we consider the machine N' of class S_0, generated by the given procedure from (M', q_1). If there exists M' such that N' is a WSTS, then we have that q_1 is reachable in M' (by condition (2)), hence in M.

On the other hand, if q_1 was reachable in M, then let $Q_{reach} \subseteq Q$ be the set of all control-states of M which are reachable from $(q_0; 0, 0)$. Let its corresponding sub-machine be M'. Since all control-states of M' are reachable (by choice of Q_{reach}), therefore the corresponding N' will be a WSTS (by condition (1)).

Hence, q_1 is reachable in M from $(q_0; 0, 0)$ iff there exists a subset $Q' \subseteq Q$ satisfying that $\{q_0, q_1\} \subseteq Q'$ such that the corresponding sub-machine M' is a WSTS. Since there are only 2^{n-2} such subsets, we can check all of them to decide whether q_1 is reachable in M.

Hence, we have given an algorithm to check reachability in Minsky machine. Therefore, the well structured problem for S_0 is undecidable. ◀

We will use Lemma 11 to prove that the well structured problem for functional 1-dim PCMs is undecidable. To apply Lemma 11, we need to give an algorithm which takes a Minsky machine $M = (Q, \rightarrow_M, q_0)$ and a control-state q_1, and generates a functional 1-dim PCM N_1 satisfying conditions (1) and (2).

Construction of a functional 1-dim PCM N_1:

Let (M, q_0) be given. The procedure to generate a 1-dim PCM N_1 is as follows:

Let $v_p(n)$ denote the largest power of p dividing n. For $M = (Q, \rightarrow_M, q_0)$, we define the 1-PCM $N_1 = (Q, \rightarrow_N, (q_0, 1))$ with the same set Q of control-states. We will represent the values of the two counters (m, n) by the one-counter values $2^m 3^n c$ for any c such that $v_2(c) = v_3(c) = 0$. Conversely, a configuration (q, n) of N_1 will correspond to $(q; v_2(n), v_3(n))$ of M. Note that, we are allowing multiplication by constants c in N_1 as long as $v_2(n)$ and $v_3(n)$ remain unchanged.

Increment/decrement of counters corresponds to multiplication/division by 2 and 3 which is Presburger expressible. Similarly, zero-test corresponds to checking divisibility by 2 and 3 which is again Presburger-expressible. So first, for each transition in $\rightarrow_M$, we add the corresponding transition to $\rightarrow_N$.

Now, to get the suitable properties of conditions (1) and (2), we will add two more types of transitions to $\rightarrow_N$. For each control-state q, we add a transition $(q, x'_1 = 6x_1 + 1, q_0)$ to $\rightarrow_N$. We shall call it a *"reset-transition"* because $v_2(6x_1 + 1) = v_3(6x_1 + 1) = 0$, so this transition corresponds to a counter-reset in M from anywhere regardless of our present configuration. Note that such a transition would not change the reachability set in M. This "reset-transition" is crucial in forcing well-structuredness in N. Also, we add a transition $(q_0, (x_1 = 0 \wedge x'_1 = 0), q_1)$ to $\rightarrow_N$ to ensure condition (2). Since the configuration $(q_0, 0)$ cannot be reached from the initial configuration $(q_0, 1)$ during any run of N_1, this will also not affect the reachability set of N_1. Note that, all of our transitions are functional, hence N_1 is a functional 1-dim PCM.

Now, we show that the construction of N_1 satisfies conditions (1) and (2).

▶ **Lemma 12.** *The functional 1-dim PCM N_1 satisfies condition (1).*

Proof. Suppose that all control-states of M are reachable from $(q_0; 0, 0)$. Then we claim that N_1 will be a WSTS. Suppose there is a transition $(q, n) \rightarrow_N (q', m)$ and (q, n') is a configuration with $(q, n') \geq (q, n)$. Hence we want to show existence of some path $(q, n') \xrightarrow{*}_N (q', m') \geq (q', m)$.

Case 1: The transition $(q, n) \rightarrow_N (q', m)$ is a "reset-transition". Hence $q' = q_0$ and $m = 6n + 1$. In this case, note that since $n' \geq n$, the transition $(q, n') \rightarrow_N (q_0, 6n' + 1) \geq (q_0, m)$ satisfies the requirement.

Case 2: The transition $(q, n) \rightarrow_N (q', m)$ is not a "reset-transition". In this case, $m \leq 3n$ because the above transition corresponds, in M to an increment/decrement in c_1 or c_2 or a zero-test. In each case, we can check that $m \leq 3n$. Let there be a path $(q_0; 0, 0) \xrightarrow{*}_M (q'; n_1, n_2)$ in M for some n_1, n_2. Such a path exists because all control-states in M are reachable. Hence, we take the "reset-transition" $(q, n') \rightarrow_N (q_0, 6n' + 1)$ and follow the corresponding path $(q_0, 6n' + 1) \xrightarrow{*}_N (q', 2^{n_1} 3^{n_2} (6n' + 1)) \geq (q', 3n) \geq (q', m)$. Hence we have again shown monotony to prove that N_1 is a WSTS.

Hence we have shown that if all control-states of M are reachable, then N_1 is monotone. ◀

▶ **Lemma 13.** *The functional 1-dim PCM N_1 satisfies condition (2).*

Proof. Since there is a transition $(q_0, 0) \rightarrow_N (q_1, 0)$, we deduce that if N_1 is a WSTS, then $(q_0, 1) \xrightarrow{*}_N (q_1, n)$ for some n by monotony because $(q_0, 0) \leq (q_0, 1)$. Also note that since N_1 simulates M, hence reachability of q_1 in N_1 implies that q_1 is reachable from $(q_0; 0, 0)$ in M. ◀

Since we have provided a construction of functional 1-dim PCM N_1 satisfying conditions (1) and (2), from Lemma 11 we have that:

▶ **Theorem 14.** *The well structured problem for functional 1-dim PCMs is undecidable.*

Similarly, we can use Lemma 11 to show this result for 2 counter Minsky machines. This construction is done in the Appendix due to lack of space.

▶ **Theorem 15.** *(proof in Appendix) The well structured problem for* 2*-dim Minsky machines is undecidable.*

Now, we make the observation that we can perform zero-tests using affine functions. The basic idea is that a transition $x' = -x$ is only satisfied by a counter whose value is 0. Increments/decrements can already be implemented in 2-AVASS since translations are affine functions. A zero test on the first counter can be done by having a transition labelled by $\left(\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)$, and similarly for second counter. Since we can implement both increment/decrements and zero-tests with 2-AVASS, we can simulate 2-counter Minsky machines with 2-AVASS. Note that we can extend this result to d-AVASS simulating d-counter Minsky machines.

As a direct consequence of this and Theorem 15, we have that:

▶ **Corollary 16.** *The well structured problem for* 2*-AVASS is undecidable.*

However, if we consider strong monotony instead of monotony, the above undecidability results can be turned into a decidability result. Strong monotony can be expressed in Presburger arithmetic as follows:

$$\bigwedge_{\phi\in\Phi} (\forall x_1 ... \forall x_d \forall x'_1 ... \forall x'_d \forall y_1 ... \forall y_d ((\bigwedge_{i=1}^{d} x_i \leq y_i) \wedge \phi(x_1, ..., x_d, x'_1, ..., x'_d)$$
$$\implies (\exists y'_1 ... \exists y'_d (\bigwedge_{i=1}^{d} x'_i \leq y'_i) \wedge \phi(y_1, ..., y_d, y'_1, ..., y'_d))))$$

Since Presburger arithmetic is decidable, the strong well structured problem for d-PCM is decidable.

▶ Remark 17. The validity of the formula of strong monotony can also be decided for extended PCM defined in decidable extensions of Presburger Arithmetic.

4 Decidability results for 1-AVASS

Now, let us look at some reachability and coverability results for the various models of AVASS. First, we can simulate 2-counter Minsky machines with 2-AVASS. Since coverability and reachability are undecidable for 2-counter Minsky machines, we directly have the following result:

▶ **Corollary 18.** *Control-state reachability, hence coverability is undecidable for* 2*-AVASS.*

Similarly, we showed in Construction of functional 1-PCM N_1 that we can also simulate 2-counter Minsky machines with functional 1-PCM. Hence, we also have the following:

▶ **Corollary 19.** *Control-state reachability, hence coverability is undecidable for functional* 1*-PCM.*

Now, let us examine the case of 1-AVASS. For 1-AVASS, reachability and consequently coverability is decidable from work done in [10]. We show that checking whether it is a WSTS is also decidable. Moreover, we give a simpler proof of decidability of reachability and coverability.

Given $M = (Q, \to)$ a 1-AVASS and a final configuration (q_f, n_f) that we want to check reachability for, we present Algorithm 1 which computes $Pre^*(q_f, n_f)$ as a Presburger formula. A transition $(q, x' = ax + b, q')$ is *positive* if $a \geq 0$. Let a cycle/path in M be called *positive* if all transitions are positive. A cycle $(q_1, ..., q_k, q_1)$ is called a *simple cycle* if $q_1, ..., q_k$ are all pairwise distinct.

Let us denote by Pre_q the set $Pre^*(q_f, n_f) \cap (\{q\} \times \mathbb{N})$. For a transition $t = (q, x' = ax + b, q')$ and a given subset of $X \subseteq \mathbb{N}$, let $Pre^t(X)$ denote $\{n : an + b \in X\}$. For a simple cycle c rooted at q with an effective guard and transition, extend the above notation $Pre^{c^i}(X)$ for i repetitions of the cycle. Then, let $Pre^{c^*}(X) := \cup_{i \in \mathbb{N}} Pre^{c^i}(X)$. We will conveniently replace X by a formula which denotes a subset of $\mathbb{N}$.

Algorithm 1 Algorithm for computing $Pre^*(q_f, n_f)$ in 1-AVASS

procedure COMPUTEPRE*
    for all $q \in Q$ **do**
        $\phi_q \equiv \perp$
    $\phi_{q_f} \equiv (n = n_f)$
    for all $q \in Q$ **do**
        for all simple cycles c rooted at q **do**
            c.transition = SIMPLIFYTRANSITION(c)
            c.guard = COMPUTEGUARD(c)
    notFinished = True
    while notFinished **do**
        notFinished = False
        for all $q \in Q$ **do**
            $\phi' = \phi_q$
            for all transitions $t = (q, x' = ax + b, q') \in \to$ **do**
                EXPLORETRANSITION(t)
            for all simple cycles c containing q **do**
                EXPLORECYCLE(c)
            if $\phi' \neq \phi_q$ **then** ▷ Check equality as Presburger formulae
                notFinished = True

The algorithm will keep a variable ϕ_q for each control-state $q \in Q$ which will store a Presburger formula (with one free variable n) denoting the currently discovered subset of Pre_q. Let this be denoted by $[\![\phi_q]\!]$, i.e. $[\![\phi_q]\!] := \{n : \phi_q(n)\}$. For uniformity, we can assume that ϕ_q is a disjunction of formulae of form $range \wedge mod$ where $range \equiv (r \leq n \leq s)$ (s possibly ∞) and $mod \equiv (n =_{d_q} d)$.

We initially simplify each simple cycle into a meta-transition which is the composition of all individual transitions in the cycle. We will also compute the guard of a cycle. Since each positive transition has an upward closed guard and each negative transition has a downward closed guard, the guard of a cycle will be of the form $r \leq n \leq s$ for some $r, s \in \mathbb{N}$ (s possibly ∞). Hence, we will only consider a cycle in terms of its guard and its meta-transition.

We use two main procedures in COMPUTEPRE*:

1. ExploreTransition: Given a transition $t = (q, x' = ax + b, q')$, it computes $Pre^t(\phi_{q'})$ and appends it to ϕ_q.
2. ExploreCycle: Given a simple cycle c rooted at q, it computes $Pre^{c^*}(\phi_q)$ and appends it to ϕ_q.

▶ **Lemma 20.** *(proof in Appendix) For any transition t, and any simple cycle c, given ϕ_q, $Pre^t(\phi_q)$ and $Pre^{c^*}(\phi_q)$ are both Presburger expressible and effectively computable.*

With this lemma the algorithm is well-defined. Now let us prove the termination and the correctness of the algorithm.

▶ **Proposition 21.** *Algorithm* computePre* *terminates.*

Proof. For each q, we will show that Pre_q can be obtained in finitely many iterations of the algorithm. Let $q \in Q$ be arbitrary.

Case 1: Pre_q is finite:
Each value will be discovered in finitely many iterations, hence Pre_q will be obtained in finitely many iterations.
Case 2: Pre_q is infinite:
Since we are talking about reaching (q_f, n_f), we note that the only transitions which can decrease arbitrarily large values are transitions of the form $x' = b$ or $x' = x - a, a > 0$. Hence, since Pre_q has arbitrarily large values, and each run has to reach n_f (i.e. has to be decreased), we can see that there must either be a transition $x' = b$, or a positive cycle with meta-transition $x' = x - a$, reachable from q through a positive path.

Case 2.i: There is a transition $x' = b$:
In this case, there exists N such that for all $n \geq N$, the same path suffices. In this case, once the aforementioned path is discovered, $\{n : n \geq N\}$ becomes a subset of $[\![\phi_q]\!] \subseteq Pre_q$, which leaves finitely many values in $Pre_q \setminus [\![\phi_q]\!]$, which can again be discovered by finitely many additional runs.
Case 2.ii: There are positive cycles with meta-transition $x' = x - a$:
The idea is that we will cover Pre_q when we compute Pre^{c^*} for such a cycle c. This is because for such a cycle, all that matters is the value of the counter modulo a. Since there are only finitely many distinct values modulo a, these will again be discovered in finitely many runs. Hence, each cycle will be discovered in finitely many runs. Therefore since there are finitely many simple cycles, the corresponding values of Pre_q will also be discovered in finitely many runs.

Hence, for all q, in finitely many runs we will get $Pre_q = [\![\phi_q]\!]$. At such a point, the algorithm has to stop, hence termination is guaranteed. ◀

▶ **Theorem 22.** *(Correctness) Given a 1-AVASS $M = (Q, \rightarrow)$ and a configuration (q, n), the algorithm* computePre* *computes $Pre^*(q, n)$ as a Presburger formula.*

Proof. We will show that Algorithm 1 upon termination will always have $[\![\phi_q]\!] = Pre_q$.

That $[\![\phi_q]\!] \subseteq Pre_q$ should be clear. Suppose the algorithm terminates with $[\![\phi_q]\!] \subsetneq Pre_q$ for some $q \in Q$. For some value $n \in Pre_q \setminus [\![\phi_q]\!]$, consider a path which covers (q_2, n_2), say the path is $(q, n) \rightarrow (p_1, n_1) \rightarrow ... \rightarrow (p_m, n_m) \rightarrow (q_2, n')$. In such a path, consider the largest i, such that $n_i \notin [\![\phi_{p_i}]\!]$. Now, in the last iteration of the algorithm, since $n_{i+1} \in [\![\phi_{p_{i+1}}]\!]$ (by choice of i), hence, we will explore the edge to include $n_i \in [\![\phi_{p_i}]\!]$. Hence, the algorithm would not have terminated. Contradiction. Hence, when the algorithm terminates, $[\![\phi_q]\!] = Pre_q$. ◀

▶ **Example 23.** Let us consider machine M_1 in Figure 1. Suppose we want to compute $Pre^*(q_1, 19)$. We begin with $\phi_{q_1} \equiv (n = 19)$, $\phi_{q_2} \equiv \bot$. If we apply EXPLORETRANSITION to the transition $(q_2, (x' = x), q_1)$, we will get $\phi_{q_2} \equiv (n = 19)$. If we now apply EXPLORECYCLE to the cycle $(q_2, x' = x - 3, q_2)$, we will get $\phi_{q_2} \equiv (n \geq 19 \wedge n =_3 1)$. Continuing like this, we end up with $\phi_{q_1} \equiv (n \in \{0, 3, 6, 19\} \vee (n \geq 13 \wedge n =_3 1) \vee (n \geq 32 \wedge n =_3 2) \vee (n \geq 45 \wedge n =_3 0))$ and $\phi_{q_2} \equiv (n \geq 0 \wedge n =_3 0) \vee (n \geq 19 \wedge n =_3 1) \vee (n \geq 32 \wedge n =_3 2)$. This is $Pre^*(q_1, 19)$.

▶ **Corollary 24.** *(proof in Appendix) [10] Reachability (hence coverability and control-state reachability) for* 1*-AVASS is decidable.*

▶ Remark 25. Algorithm 1 also works if we extend the model of 1-AVASS with Presburger guards at each transition. Hence, reachability, coverability and the well-structured problem are all decidable for this model as well.

It could be useful to determine whether an 1-AVASS is a WSTS (with strict monotony) because if it is the case, it will allow to decide other problems like the boundedness problem that is not immediately a consequence of the computability of $Pre^*(\uparrow(q, n))$. Since we can compute $Pre^*(q, n)$, we can also compute $Pre^*(\uparrow(q, n))$ by the same technique as in Corollary 24 (check Appendix). This can be used to determine whether a given 1-AVASS is a WSTS as follows.

▶ **Theorem 26.** *The well structured problem is decidable for* 1*-AVASS.*

Proof. First we show that M is a WSTS, iff for all negative transitions $(q_1, (x' = ax+b), q_2)$, the set $\{q_1\} \times \mathbb{N}$ is a subset of $Pre^*(\uparrow(q_2, b))$. For any negative transition $(q_1, (x' = ax+b), q_2)$, we have $(q_1, 0) \to (q_2, b)$. If M is a WSTS, by monotony, for any $n \geq 0$, there exists a path $(q_1, n) \xrightarrow{*} (q_2, b') \geq (q_2, b)$ because $(q_1, n) \geq (q_1, 0)$. This implies that $\{q_1\} \times \mathbb{N}$ is a subset of $Pre^*(\uparrow(q_2, b))$.

In the other direction, let there be a transition $(q_1, n) \to (q_2, an+b)$ and $(q_1, n') \geq (q_1, n)$. If the transition is positive, i.e. $a \geq 0$, then we directly have the transition $(q_1, n') \to (q_2, an'+b) \geq (q_2, an+b)$. If the transition is negative, then we have that $(q_2, an+b) \leq (q_2, b)$. Since $(q_1, n') \in Pre^*(\uparrow(q_2, b))$ (by hypothesis, since it is a negative transition), hence we have that $(q_1, n') \xrightarrow{*} (q_2, b') \geq (q_2, b) \geq (q_2, an + b)$. Hence, M is monotone. Therefore, M is a WSTS iff for all negative transitions $(q_1, (x' = ax + b), q_2)$, the set $\{q_1\} \times \mathbb{N}$ is a subset of $Pre^*(\uparrow(q_2, b))$.

Now, since $Pre^*(\uparrow(q, n))$ is computable, we can check that for each negative transition $(q_1, (x' = ax + b), q_2)$, the set $\{q_1\} \times \mathbb{N}$ is a subset of $Pre^*(\uparrow(q_2, n))$ to determine whether M is a WSTS or not. ◀

▶ **Example 27.** Let us consider machine M_1 in Figure 1 and its negative transition $(q_1, x' = 19 - x, q_1)$. We observe that the set $Pre^*(\uparrow(q_1, 19)) = \{q_1, q_2\} \times \{n : n \geq 19\}$ does not contain $\{q_1\} \times \mathbb{N}$, hence machine M_1 is not a WSTS. However, in this example (Figure 1), if we replace the transition $(q_1, (x' = x - 13), q_2)$ by $(q_1, (x' = x + 1), q_2)$, we will get a new machine M_2 which is still not a 1-VASS, but it is a WSTS.

Let us focus our attention to positive AVASS now. We know that for positive 1-AVASS reachability is decidable from Corollary 24. We show that reachability is undecidable for positive 2-AVASS by reduction from Post's Correspondence Problem (PCP) [16]. Our result completes the view about decidability of reachability for VASS extensions in small dimensions. As a matter of fact, reachability is undecidable for VASS with two resets in dimension 3 (to adapt the proof in [7]), hence for positive 3-AVASS but it is decidable for VASS with

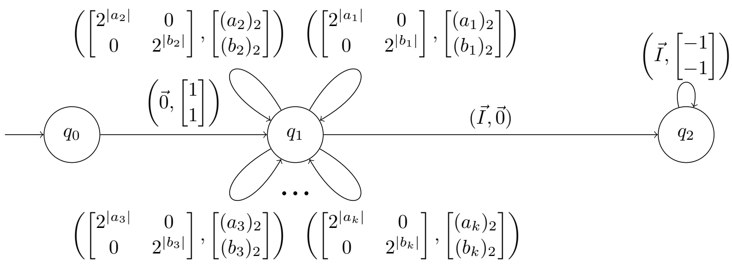


Figure 2 Construction for undecidability of reachability for positive 2-AVASS by reduction from PCP.

two resets in dimension 2 [12]. If we replace resets by affine functions, reachability becomes undecidable in dimension two.

Reichert gives in [21] a reduction from the Post correspondence problem to reachability in a subclass of 2-AVASS and we may remark that his proof is still valid for *positive* 2-AVASS. Blondin, Haase and Mazowiecki made some similar observations [1] for subclasses of $3-\mathbb{Z}$-AVASS, with positive matrices. Our proof is essentially the same as [21].

▶ **Theorem 28.** *Reachability is undecidable for positive* 2*-AVASS.*

Proof. Suppose we are given an instance of PCP, i.e. we are given $a_1, ..., a_k, b_1, ..., b_k \in \{0,1\}^*$ for some $k \in \mathbb{N}$. We want to check if there exists some sequence of numbers $n_1, ..., n_\ell \in \{1, ..., k\}$ such that $a_{n_1}...a_{n_\ell} = b_{n_1}...b_{n_\ell}$ (concatenated as strings).

We will construct the positive 2-AVASS as demonstrated in Figure 2, where $|a_i|$ refers to the length of the string, and $(a_i)_2$ refers to the number encoded by the string a_i if read in binary (most significant digit to the left). The idea is that we use the two counters to store the value of $(a_{n_1}...a_{n_\ell})_2$ and $(b_{n_1}...b_{n_\ell})_2$ for any $n_1, ..., n_\ell$. But we first increment each counter to keep track of leading zeroes. Now, the configuration $(q_2; 0, 0)$ is reachable from $(q_0; 0, 0)$ in the positive 2-AVASS described in Figure 2 iff the given PCP has an affirmative answer. Hence, checking reachability in positive d-AVASS is undecidable for $d \geq 2$. ◀

Also, we note that positive-AVASS are well-structured with strong monotony. Hence coverability is decidable [13]. If we look at totally-positive AVASS, we can see that coverability is already decidable by the same argument. However, reachability is also decidable.

▶ **Theorem 29.** *Reachability is decidable in totally-positive AVASS for any dimension.*

Proof. Let $M = (Q, \rightarrow)$ be a totally-positive d-AVASS. Given $(q_0; n_1, ..., n_d)$, suppose we want to check reachability of $(q_f; m_1, ..., m_d)$. Let $N = \max\{m_1, ..., m_d\}$. Let $f_N : \mathbb{N} \rightarrow \{1, ..., N, \omega\}$ be the function which is identity on $\{1, ..., N\}$ and maps $\{N+1, ...\}$ to ω. Extend this function to the set $\mathbb{N}^d$ component-wise. Since M is totally-positive, we can restrict our search space from $Q \times \mathbb{N}^d$ to $Q \times \{0, ..., N, \omega\}^d$ by applying f_N to each configuration and using the following arithmetic rules: $0.\omega = 0$, and for all $k \geq 1$, $k.\omega = \omega$ and $\omega + k = \omega$.

We claim that if $(q_f; m_1, ..., m_d)$ is reachable, then it is reachable in this restricted search-space. This follows from the fact that given any element $(n_1, ..., n_d)$ of $\mathbb{N}^d$, and a totally positive transition $t = (A, b)$, we will have that $t(f_N(n_1, ..., n_d)) = f_N(t(n_1, ..., n_d))$ (t acts on $f_N(n_1, ..., n_d)$ to give an element in $\{0, ..., N, \omega\}^d$). This is because a totally positive transition cannot decrease a value other than by multiplying it by 0, hence any value greater than N will continue to be greater than N. Also note that, by choice of N, $f_N(m_1, ..., m_d) = (m_1, ..., m_d)$.

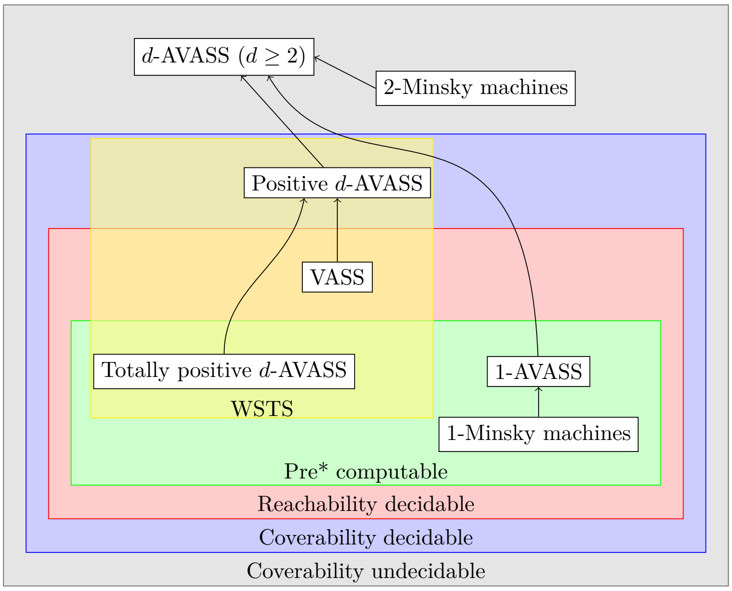


Figure 3 Showing reachability and coverability results for various AVASS models.

Once we have this, we can make an induction on the length of the path to see that if $(q_f; m_1, ..., m_d)$ is reachable, it is reachable in the restricted search-space $Q \times \{0, ..., N\}^d$.

Since $Q \times \{0, ..., N, \omega\}^d$ is finite, this shows decidability of reachability. ◀

5 Conclusion and perspective

We introduced two variants of the well structured problem for PCM and we solve it for many classes of PCMs. Moreover, we answer the decidability questions for reachability and coverability for classes of PCMs and AVASSs (we summarise the results of Section 4 in Figure 3).

Many open problems can be attacked like the complexity of reachability for 1-AVASS (reachability is NP for 1-VASS and PSPACE for polynomial VASS), the size of Pre^* of a 1-AVASS (and its relation with the theory of flattable VASS [18]), and the decidability of the property for a Presburger relation on $\mathbb{N}^d$ to be a well-quasi ordering for $d \geq 2$.

We also open the way to study the decidability of the well structured problems (for various orderings) for many other models like pushdown counter machines, FIFO automata, Petri nets extensions. For instance, we wish to solve the *well structured problems* for FIFO automata. We know that *lossy* FIFO automata are well structured (for the subword ordering) but what is the class of *perfect* FIFO automata which is well structured (for the prefix ordering)?

References

1 Michael Blondin, Christoph Haase, and Filip Mazowiecki. Affine extensions of integer vector addition systems with states. In Sven Schewe and Lijun Zhang, editors, *29th International Conference on Concurrency Theory, CONCUR 2018, September 4-7, 2018, Beijing, China*, volume 118 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

2 Michael Blondin and Mikhail Raskin. The complexity of reachability in affine vector addition systems with states. 2019.

3 Bernard Boigelot and Pierre Wolper. Symbolic verification with periodic sets. In *Computer Aided Verification, 6th International Conference, CAV '94, Stanford, California, USA, June 21-23, 1994, Proceedings*, pages 55–67, 1994.

4 Rémi Bonnet, Alain Finkel, and M. Praveen. Extending the rackoff technique to affine nets. In Deepak D'Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, volume 18 of *LIPIcs*, pages 301–312. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

5 Ahmed Bouajjani, Javier Esparza, Alain Finkel, Oded Maler, Peter Rossmanith, Bernard Willems, and Pierre Wolper. An efficient automata approach to some problems on context-free grammars. *Information Processing Letters*, 74(5-6):221–227, June 2000.

6 Stéphane Demri, Alain Finkel, Valentin Goranko, and Govert van Drimmelen. Model-checking ctl* over flat presburger counter systems. *Journal of Applied Non-Classical Logics*, 20(4):313–344, 2010.

7 Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. Reset nets between decidability and undecidability. In Kim G. Larsen, Sven Skyum, and Glynn Winskel, editors, *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115, Aalborg, Denmark, July 1998. Springer.

8 A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1):63 – 92, 2001. ISS.

9 Alain Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–179, 1990.

10 Alain Finkel, Stefan Göller, and Christoph Haase. Reachability in register machines with polynomial updates. In Krishnendu Chatterjee and Jiří Sgall, editors, *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science (MFCS'13)*, volume 8087 of *Lecture Notes in Computer Science*, pages 409–420, Klosterneuburg, Austria, August 2013. Springer.

11 Alain Finkel and Jérôme Leroux. How to compose presburger-accelerations: Applications to broadcast protocols. In *FST TCS 2002: Foundations of Software Technology and Theoretical Computer Science, 22nd Conference Kanpur, India, December 12-14, 2002, Proceedings*, pages 145–156, 2002.

12 Alain Finkel, Jérôme Leroux, and Grégoire Sutre. Reachability for two-counter machines with one test and one reset. In Sumit Ganguly and Paritosh Pandya, editors, *Proceedings of the 38th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'18)*, Leibniz International Proceedings in Informatics, pages 31:1–31:14, Ahmedabad, India, December 2018. Leibniz-Zentrum für Informatik.

13 Alain Finkel, Pierre McKenzie, and Claudine Picaronny. A well-structured framework for analysing Petri net extensions. *Information and Computation*, 195(1-2):1–29, November 2004.

14 Christoph Haase. A survival guide to presburger arithmetic. *ACM SIGLOG News*, 5(3):67–82, July 2018.

15 Christoph Haase and Simon Halfon. Integer vector addition systems with states. In Joël Ouaknine, Igor Potapov, and James Worrell, editors, *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*, volume 8762 of *Lecture Notes in Computer Science*, pages 112–124. Springer, 2014.

16 Vesa Halava. Another proof of undecidability for the correspondence decision problem - had I been emil post. *CoRR*, abs/1411.5197, 2014.

17 Matthias Horbach, Marco Voigt, and Christoph Weidenbach. The universal fragment of presburger arithmetic with unary uninterpreted predicates is undecidable. *CoRR*, abs/1703.01212, 2017.

18 Jérôme Leroux and Grégoire Sutre. On flatness for 2-dimensional vector addition systems with states. In Philippa Gardner and Nobuko Yoshida, editors, *CONCUR 2004 - Concurrency Theory, 15th International Conference, London, UK, August 31 - September 3, 2004, Proceedings*, volume 3170 of *Lecture Notes in Computer Science*, pages 402–416. Springer, 2004.

19 Jérôme Leroux, Grégoire Sutre, and Patrick Totzke. On the coverability problem for pushdown vector addition systems in one dimension. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, pages 324–336, 2015.

20 Marvin L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967.

21 Julien Reichert. *Reachability games with counters : decidability and algorithms. (Décidabilité et complexité de jeux d'accessibilité sur des systèmes à compteurs).* PhD thesis, École normale supérieure de Cachan, France, 2015.

22 Rüdiger Valk. Self-modifying nets, a natural extension of petri nets. In Giorgio Ausiello and Corrado Böhm, editors, *Automata, Languages and Programming, Fifth Colloquium, Udine, Italy, July 17-21, 1978, Proceedings*, volume 62 of *Lecture Notes in Computer Science*, pages 464–476. Springer, 1978.

6 Appendix

6.1 Section 2

▶ **Proposition 2.** *[6] The property for a d-dim PCM to be* functional*, is decidable in NP.*

Proof. Let $M = (Q, \Phi, \rightarrow)$ be a given d-dim PCM. Functionality can be expressed in Presburger arithmetic as follows:

$$\bigwedge_{\phi \in \Phi} (\forall x_1 ... \forall x_d \forall x'_1 ... \forall x'_d \forall x''_1 ... \forall x''_d$$

$$(\phi(x_1, ..., x_d, x'_1, ..., x'_d) \wedge \phi(x_1, ..., x_d, x''_1, ..., x''_d) \implies \bigwedge_{i=1}^{d} x'_i = x''_i)$$

Hence, the validity of this formula can be decided, and so, functionality is decidable. ◀

▶ **Proposition 7.** *The well structured problem for* 1*-arithmetic counter machines is undecidable.*

Proof. Since first order (FO) logic is undecidable, we can have a reduction from decidability of FO to checking whether a given arithmetic counter machine is well-structured. Let ϕ be a given FO formula with no free variables. Define the 1-arithmetic counter machine $M = (\{q_0\}, \{\phi_0\}, \rightarrow)$, where $\phi_0 = ((x_1 = 0 \wedge y_1 = 2) \vee \phi)$ and $\rightarrow = \{(q_0, \phi_0, q_0)\}$ and let S_M be its associated transition system. Hence, if ϕ is a tautology, then for all $m, n \geq 0$, the transition $(q_0, m) \rightarrow (q_0, n)$ exists in S_M. Hence S_M is a well-structured transition system. However, if ϕ is false, then S_M is not well-structured since there is a transition $(q_0, 0) \rightarrow (q_0, 2)$, but there is no transition from $(q_0, 1) \geq (q_0, 0)$ which violates monotony. Hence S_M is a WSTS iff ϕ is a tautology. ◀

▶ **Proposition 9.** *The property for a decidable ordering on* $\mathbb{N}$ *to be a well ordering is undecidable.*

Proof. We will have a reduction from Halting Problem to show undecidability of checking whether a relation on $\mathbb{N}$ is a wqo.

Let M be a Turing machine and $\leq_M$ its associated decidable relation defined as follows. For all i, j: we have $i \leq_M i$ and $i \leq_M i + j$ iff M does not halt in $i + j$ steps and $i + j \leq_M i$ if M halts in at most $i + j$ steps; hence $\mathbb{N}$ is totally ordered by the decidable ordering $\leq_M$. If M does not halt, we have $1 \leq_M 2 \leq_M ... \leq_M i \leq_M i + 1 \leq_M ...$ so $(\mathbb{N}, \leq_M)$ is a *well ordering*. If M halts in exactly n steps, then there is an infinite strictly decreasing sequence $n >_M n + 1 >_M n + 2 >_M ...$, hence $(\mathbb{N}, \leq_M)$ is not a well ordering because it is not well-founded. Therefore, checking whether an ordering $\leq$ encodes a well ordering is undecidable. ◀

▶ **Proposition 10.** *The property for a Presburger relation on* $\mathbb{N}$ *to be a well-quasi ordering is decidable.*

Proof. We can check if a Presburger formula encodes a quasi ordering since reflexivity and transitivity are Presburger-expressible. Consider a Presburger formula ϕ with two free variables x and y, encoding a relation on $\mathbb{N}$. We can assume ϕ is a quasi ordering. We want to determine if it is a wqo. We can do quantifier elimination to arrive at ϕ_0, a quantifier-free formula over the predicates $\{+, \leq, <, =, >, \geq, =_n\}$ where $a =_n b$ iff $a = b \mod n$.

We write ϕ_0 in disjunctive normal form. Hence, ϕ_0 will be written as a disjunction of formulae ψ_i where each ψ_i can be written in the form $(\phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4 \wedge \phi_5 \wedge \phi_6 \wedge \phi_7 \wedge \phi_8 \wedge \phi_9)$ where:

- ϕ_1 is a conjunction of formulae of form $ax + by \leq n$ with $a \geq 1, b \geq 1$
- ϕ_2 is a conjunction of formulae of form $ax + by \leq n$ with $a \geq 1, b \leq 1$
- ϕ_3 is a conjunction of formulae of form $ax + by \geq n$ with $a \geq 1, b \geq 1$
- ϕ_4 is a conjunction of formulae of form $ax + by \geq n$ with $a \geq 1, b \leq 1$
- ϕ_5 is a conjunction of formulae of form $ax \geq n$ with $a \geq 1$
- ϕ_6 is a conjunction of formulae of form $ax \leq n$ with $a \geq 1$
- ϕ_7 is a conjunction of formulae of form $by \geq n$ with $b \geq 1$
- ϕ_8 is a conjunction of formulae of form $by \leq n$ with $b \geq 1$
- ϕ_9 is a conjunction of formulae of the form $ax + by =_n p$ with $a, b \geq 0$.

Let ψ_0 be one conjunctive clause in the DNF for ϕ_0. Hence $\psi_0 = \psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4 \wedge \psi_5 \wedge \psi_6 \wedge \psi_7 \wedge \psi_8 \wedge \psi_9$ where each ψ_i conforms to above specifications. We show that we only need to be concerned with ψ_9. This is because, suppose $N = (n_1, n_2, n_3, ...)$ is a sequence for which we need to produce an ascending pair to check wqo of ϕ. For each n_i, there are only finitely many $n_j, j > i$ such that $\psi_1(n_i, n_j)$ is satisfied. In this case, we can remove those finitely many n_j. We do this for all n_i to get a sequence $N' = (n'_1, n'_2, n'_3, ...)$ thus rendering ψ_1, hence ψ_0 unsatisfiable by $x, y \in N'$. Clearly, if N' has an ascending pair, so does N. Hence it is sufficient to check existence of ascending pair in N'. Hence, if ψ_1 is non-empty, then we can simply disregard ψ_0 since it will never be satisfied in N'.

We can make a similar argument for ψ_4. For ψ_2 and ψ_3, we make the dual argument, that for all n_i, there are only finitely many $n_j, j > i$ such that $\psi_3(n_i, n_j) \wedge \psi_4(n_i, n_j)$ is *not* satisfied. Once we remove all such offending pairs to get N', we will get that every pair in N' satisfies $\phi_3 \wedge \phi_4$, hence we can also remove them from consideration.

For ψ_5 and ψ_7 there are only finitely many n_i such that $\psi_5(n_i, n)$ or $\psi_7(n, n_i)$ is satisfiable for any $n \in \mathbb{N}$. We can remove these n_i to render ψ_5 and ψ_7 redundant. For ψ_6 and ψ_9 we again make the dual argument, removing the finitely many n_i which can satisfy ψ_5 and ψ_7. Hence, we can disregard each ψ_i.

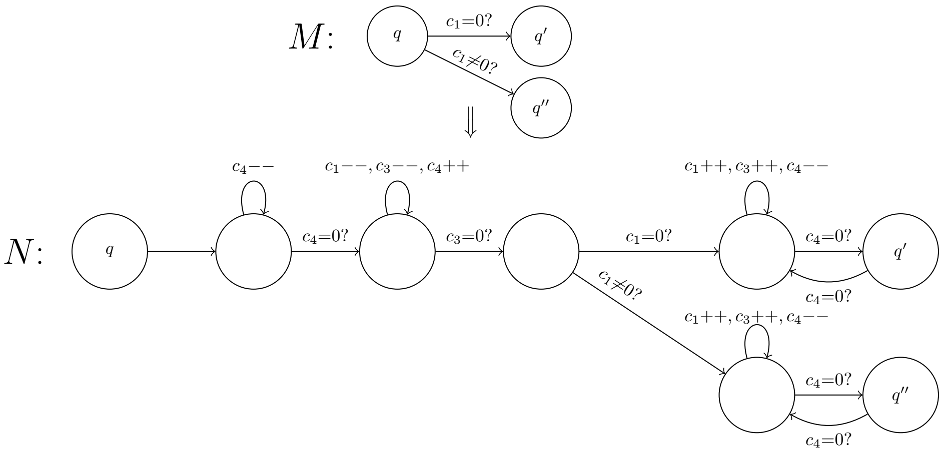
So, now, for the given formula ϕ and a sequence N, we will get a sequence N' subsequence of original sequence, and a new formula ϕ' comprising of disjunctions of formulae of form ϕ_5, where each formula ψ_i consists of conjunctions which look like $(a_{i,j}x + b_{i,j}y =_{n_{i,j}} c_{i,j})$. We now let $N = \mathrm{lcm}_{\forall i \forall j}\{n_{i,j}\}$. We can transform ϕ' to comprise entirely of formulae of form $a_{i,j}x + b_{i,j}y =_N c_{i,j}$ by multiplying each individual formula and breaking it into disjunctions.

Now, let ψ_i consist of conjunctions of $(a_{i,j}x + b_{i,j}y =_{n_{i,j}} c_{i,j})$. By checking for all $x \in \{1, ..., N\}, y \in \{1, ..., N\}$, we can reduce this to disjunctions of ψ_i of the form $(x =_N a_i \wedge y =_N b_i)$.

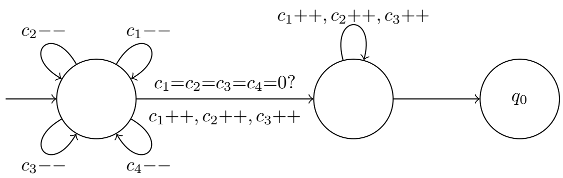
Now, we look for $n_0 \in \{1, ..., N\}$ such that $\neg\phi'(n_0, n_0)$. In other words, $\nexists$ disjunction $\psi_i \equiv (x =_N n_0 \wedge y =_N n_0)$. This can again be done finitely.

Now, the claim is that ϕ is a wqo iff no n_0 exists. Clearly, if such an element exists, then ϕ is not a wqo, because we can take an M such that $N|M$ and M is large enough such that we can ignore all subformulae of the form $\phi_1, ..., \phi_4$, and then $(M + n_0, M + N + n_0, (M + 2N) + n_0, ..., (M + aN) + n_0, ...)$ has no ascending pair.

Similarly, if no such element exists, then given any sequence $(n_1, n_2, ...)$, we can remove small enough terms, and then look at each term $\mod N$. Since no such element exists, we will have some repetition in the infinite sequence $\mod N$ and that will give us an ascending pair.



(a) Showing equivalent circuits for zero-tests in M



(b) "Reset-circuit" for N_2

Figure 4 Construction of a 4 counter Minsky machine N_2

Thus we have demonstrated a decision procedure for whether a Presburger formula ϕ encodes a wqo. ◀

6.2 Section 3

Well structured problem for 2-Minsky machines

We shall first show a construction for 4 counter Minsky machines satisfying conditions (1) and (2).

Construction of a 4 counter Minsky machine N_2:

Let (M, q_1) be given. We will use 4 counters to simulate M in such a way that we can get all the desirable properties. We will use a configuration $(q; c_1, c_2, c_3, c_4)$ of N_2 to correspond to the configuration $(q; c_1 - c_3, c_2 - c_3)$ of M.

The procedure to generate a 4-counter Minsky machine N_2 is as follows:

Let $M = (Q, \rightarrow_M, q_0)$. We define $N_2 = (Q_0, \rightarrow_N, (q_0; 1, 1, 1, 0))$ where Q_0 is a superset of Q as will be made clear. To get N_2, we will make the following modifications to M:

- Replace zero-tests in M with a circuit which checks if the respective counter equals c_3. We will use c_4 and add required additional control-states to implement such a test as illustrated in figure 4a.
- Now, from each control-state q, including the new ones added in previous step, add another circuit as illustrated in figure 4b which allows one to reach $(q_0; n, n, n, 0)$ for any $n \geq 1$. Note that q_0 is the initial state of M. We shall again call it a "reset-circuit" since $(q_0; n, n, n, 0)$ in N_2 corresponds to $(q_0, 0, 0)$ in M. Hence this transition acts as a

counter-reset. Note that adding such a transition to M will not affect its reachability set. The "reset-circuit" will be used to ensure condition (1).

- Finally, add a transition $(q_0; 0,0,0,0) \rightarrow_N (q_1; 0,0,0,0)$. This is to ensure property (2). Note that we can never reach $(q_0; 0,0,0,0)$ from the initial configuration $(q_0; 1,1,1,0)$ in any run of N_2 since the "reset-circuit" resets to $(q_0; n,n,n,0)$ for $n \geq 1$. Hence adding this transition does not affect the reachability set of N_2.

Now, we will show that the above construction indeed satisfies conditions (1) and (2).

▶ **Lemma 30.** *The 4 counter Minsky machine N_2 satisfies condition (1).*

Proof. Suppose all control-states are reachable in M. Let there be a transition $(q; n_1, n_2, n_3, n_4) \rightarrow_N (q'; m_1, m_2, m_3, m_4)$ and a configuration $(q; n'_1, n'_2, n'_3, n'_4) \geq (q; n_1, n_2, n_3, n_4)$. Since all control-states in M are reachable, there exists a path $(q_0; 0, 0) \xrightarrow{*}_M (q'; n_1, n_2)$. Choose $N \geq \max\{m_1, m_2, m_3, m_4\}$, and take the "reset-circuit" $(q; n'_1, n'_2, n'_3, n'_4) \xrightarrow{*}_N (q_0; N, N, N, 0)$. Then follow the corresponding path to $(q'; N + n_1, N + n_2, N, 0) \geq (q'; m_1, m_2, m_3, m_4)$ to satisfy monotony. Hence, if all control-states in M are reachable it implies that N_2 is a WSTS. ◀

▶ **Lemma 31.** *The 4 counter Minsky machine N_2 satisfies condition (2).*

Proof. Since N_2 is a WSTS, $(q_0; 0,0,0,0) \rightarrow_N (q_1; 0,0,0,0) \implies (q_0; 1,1,1,0) \xrightarrow{*} (q_1; n_1, n_2, n_3, n_4)$ which implies q_1 is reachable in M since N_2 simulates M. ◀

▶ **Theorem 15.** *The well structured problem for 2-dim Minsky machines is undecidable.*

Proof. Since we have given the appropriate construction, by Lemma 11, we have that checking whether 4-counter Minsky machines are WSTS is undecidable.

We will use the fact that a d-dim Minsky machine can be simulated by a 2-counter Minsky machine [20]. Let N_2 be the corresponding 4-counter Minsky machine for (M, q_1) which satisfies conditions (1) and (2). Let N'_2 be the 2-counter Minsky machine which simulates N_2. Let N''_2 be the 2-counter machine obtained by adding another "reset" circuit to N'_2 which allows reachability to $(q_0; 0, 0)$ from any configuration. Then, N''_2 will satisfy conditions (1) and (2) because N''_2 is simulating N_2 which satisfies conditions (1), (2) and we are allowing reset to initial configuration in N''_2. Hence, by Lemma 11 again we have that the well structured problem for 2-dim Minsky machines is undecidable. ◀

6.3 Section 4

▶ **Lemma 20.** *For any transition t, and any simple cycle c, given ϕ_q, $Pre^t(\phi_q)$ and $Pre^{c^*}(\phi_q)$ are both Presburger expressible and effectively computable.*

Proof. Let $\phi \equiv \vee_{i=1}^k (range_i \wedge mod_i)$. First we note that $Pre^{c^*}(\phi_q) = \cup_{i=1}^k Pre^{c^*}(range_i \wedge mod_i)$, so we will focus on a single $range \wedge mod$ clause in ϕ.

For a transition t, we can compute Pre^t by looking at each $range \wedge mod$ clause in ϕ_q, and computing its inverse.

Similarly, we can compute Pre^{c^i} for any $i \in \mathbb{N}$. We will show that $Pre^{c^*}(range \wedge mod)$ is also computable.

Given a cycle c rooted at q, with guard $r \leq n \leq s$ (s possibly ∞) and the meta-transition $y = ax + b$, let $range \equiv (r_1 \leq n \leq s_1)$ and $mod \equiv (n =_d d_1)$. By $Post^{c^i}(n)$ we denote i successive applications of a cycle to n. To show that $Pre^{c^*}(\phi_q)$ is effectively computable, we will look at multiple cases:

Case 1: $s < \infty$.
In this case, since the cycle can only be fired by a finite number of inputs, we can simply take each input n and compute $Post^{c^i}(n)$ for all i till c can no longer be activated, or it repeats. Then, we can decide whether or not $n \in Pre^{c^*}(range \wedge mod)$ based on if any of the reachable values satisfy $range \wedge mod$.
Case 2: $s = \infty$, $a \leq 1$.
Note that since the guard is upward closed, it implies that the cycle is positive. Hence the net transition is $x' = ax + b$ for $a \geq 0$. If $a = 0$, we are done. If $a = 1$, then the cycle is a translation. In this case, $Pre^{c^*}(range \wedge mod)$ is again computable using modulo relations.
Case 3: $s = \infty$, $a \geq 2$.
Let $N = \left\lceil \frac{-b}{a-1} \right\rceil$. The first thing to observe is that for all $n > N$, $Post^c(n) > n$. Hence, upon repeated application, n keeps increasing. If $s_1 < \infty$, then $n \geq \max\{s_1, N\} \implies n \notin Pre^{c^*}(range \wedge mod)$. With only finitely many values left to consider, we can again compute $Post^{c^i}(n)$ for all $r \leq n \leq \max\{s_1, N\}$ to determine whether $n \in Pre^{c^*}(range \wedge mod)$. If $s_1 = \infty$, then for values $n \geq N$, we only need to be concerned with their value $\mod d$. Hence, we can first compute the set of values $\ell_1, ..., \ell_k \mod d$ such that for some j, we have $Post^{c^j}(\ell_i) =_d d_1$. Then we know that $(n \geq N \wedge n =_d \ell_i) \implies n \in Pre^{c^*}(range \wedge mod)$. For $n \leq N$, we can again compute $Post^{c^i}(n)$ to determine whether $n \in Pre^{c^*}(range \wedge mod)$. Since it does not go off to infinity, it will again terminate or repeat.

Thus, we have shown that for any cycle c, we can compute $Pre^{c^*}(\phi_q)$. ◀

▶ **Corollary 24.** *Reachability (hence coverability and control-state reachability) for* 1*-AVASS is decidable.*

Proof. Suppose we want to check reachability of (q_2, n_2) from (q_1, n_1). Once we have computed $Pre^*(q_2, n_2)$, we can check easily whether $(q_1, n_1) \in Pre^*(q_2, n_2)$ to solve reachability. Once we have reachability, we can show a reduction from coverability to reachability to show that coverability is also decidable for 1-AVASS, as follows. Suppose we want to check coverability of (q_2, n_2). We can add two new control-states q_3 and q_4 and add the transitions $(q_2, (x' = x - n_2), q_3)$ and $(q_3, (x' = 0), q_4)$. Now, $(q_4, 0)$ is reachable iff (q_2, n_2) coverable. ◀