Lossy Counter Machines Decidability Cheat Sheet*

Philippe Schnoebelen

LSV, ENS Cachan, CNRS
61, av. Pdt. Wilson, F-94230 Cachan, France
www.lsv.ens-cachan.fr/~phs

Abstract. Lossy counter machines (LCM's) are a variant of Minsky counter machines based on weak (or unreliable) counters in the sense that they can decrease nondeterministically and without notification. This model, introduced by R. Mayr [TCS 297:337-354 (2003)], is not yet very well known, even though it has already proven useful for establishing hardness results.

In this paper we survey the basic theory of LCM's and their verification problems, with a focus on the decidability/undecidability divide.

1 Introduction

Lossy counter machines are a weakened version of Minsky counter machines. They were introduced by Richard Mayr [38,39] as a simpler version of lossy channel systems, using counters holding numerical values rather than channels recording sequences of messages in transit. Mayr proved that finiteness and uniform termination are undecidable for lossy counter machines and used this to derive various undecidability results, e.g. in [11].

Lossy counter machines are hard. Since then, lossy counter machines have been used in a variety of situations, sometimes under the guise of *counter automata with incrementation errors* [19]. Mostly, they have been used in reductions proving *hardness*, i.e., complexity lower bounds. This relies on two kinds of results. Firstly, some problems that are undecidable for Minsky machines remain undecidable for the weaker lossy counter machines. This can be used for undecidability proofs in situations where it is easier to encode lossy counters than reliable ones, e.g., as in [19,16]. Secondly, some problems that are decidable for lossy counters machines are still Ackermann-hard, i.e., they require nonprimitive-recursive time and space [43,44]. This can be used to show Ackermann-hardness of problems that are decidable but rich enough to encode lossy counters, see [18,19,32,24,46] for examples.

A survey for lossy counter machines. In this paper, we survey the main decidability and undecidability results on lossy counter machines. Most areas have not yet been investigated deeply, and some have only been superficially visited. As a consequence, our survey looks sometimes more like a road map for future research than as a record of past achievements.

* Work supported by the Agence Nationale de la Recherche, grant ANR-06-SETIN-001.

We strove for simplicity. Most decidability results can be proven by elementary arguments, relying only on generic properties like strong monotonicity of steps (Fact 2.1), the wqo property (Fact 2.3) and basic features of semilinear sets. These proofs are simpler and more versatile than the algorithms provided in, e.g., [5,28]. For undecidability, all our proofs share a single and very simple gadget, "putting a Minsky machine on a budget", making them conceptually simpler.

In this "survey" we do not always point to the earliest existing reference for each and every stated theorem. Mostly this is because these results are new, or presented in a new and extended form, or with a new and simplified proof. In general, the results come from [11,42,39] when they are specific to lossy counter machines. Some results have been first shown for lossy channel systems [15,7,6] or even well-structured systems [25,26,5,28,29].

Outline of the paper. We define counter machines, both reliable and lossy, in Section 2. We handle reachability properties in Section 3, termination and inevitability properties in Section 4, liveness properties in Section 5, finiteness properties in Section 6. All the decidability results given in these first sections are proven along the way, while the proof of the undecidability results are delayed until Section 7 where they are handled uniformly. Finally, we gather in Section 8 a few extra results on issues that are less central, or more recent, in the theory of lossy counter machines. Finally, and for the sake of completeness, the complexity of decidable problems is briefly discussed in Section 9.

2 Counter Machines

Counter machines are a model of computation where a finite-state control acts upon a finite number of *counters*, i.e., storage locations that hold natural numbers. The computation steps are usually restricted to simple tests and updates. For Minsky counter machines, the tests are zero-tests and the updates are incrementations and decrementations. Formally, a *(Minsky) counter machine* is a tuple $M = (Loc, C, \Delta)$ where $Loc = \{\ell_1, \ldots, \ell_m\}$ is finite set of *locations*, $C = \{c_1, \ldots, c_n\}$ is a finite set of *counters*, and $\Delta \subseteq Loc \times OP(C) \times Loc$ is a finite set of transition rules carrying operations from a set $OP(C) \overset{\text{def}}{=} C \times \{\texttt{++}, \texttt{--}, \texttt{=0?}\}$.

In pictorial representations, a counter machine is usually depicted as a directed graph where transition rules are $OP(C)$-labeled edges between control locations, see Fig. 1 for a simple example. An operation of the form `c++` denotes the incrementation of counter `c`, while `c--` denotes its decrementation. Decrementations are only firable when the counter at hand holds a strictly positive value, as is formally stipulated in the operational semantics. Operations of the form `c=0?` are tests used to restrict transition steps.

2.1 Operational Semantics

Let $M = (Loc, C, \Delta)$ be a counter machine. A *configuration* of M is some $\sigma = \langle \ell, \boldsymbol{a} \rangle \in Conf \overset{\text{def}}{=} Loc \times \mathbb{N}^C$, i.e., a *current control location* ℓ and a C-indexed vector $\boldsymbol{a}$ of natural numbers (one *current value* for each counter in C). If we assume, as we shall do from now on, that $C = \{\texttt{c}_1, \ldots, \texttt{c}_n\}$, we may identify $\mathbb{N}^C$ with $\mathbb{N}^n$ and write σ under the form

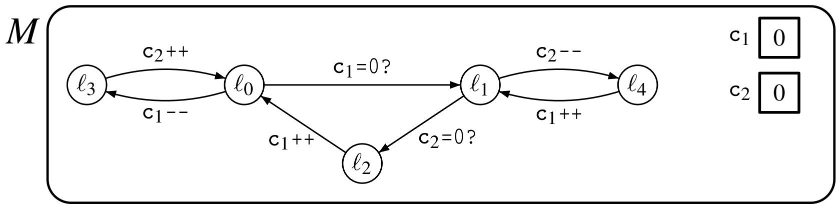

Fig. 1. M: a counter machine that enumerates all pairs $(a_1, a_2) \in \mathbb{N}^2$

$\langle \ell, a_1, \ldots, a_n \rangle$. We sometimes use counter names as positional indexes when there is a need for disambiguation, e.g., writing $\langle \mathbf{0}, \mathtt{c}_k : 1, \mathbf{0} \rangle$ for the k-th unit vector.

The operational semantics of M is given under the form of transitions between its configurations. Formally, there is a *transition* (also called a *step*) $\sigma \xrightarrow{\delta}_{\text{std}} \sigma'$ if, and only if, σ is some $\langle \ell, a_1, \ldots, a_n \rangle$, σ' is some $\langle \ell', a'_1, \ldots, a'_n \rangle$, $\delta = (\ell, op, \ell')$ and either:

– op is $\mathtt{c}_k\mathtt{=0?}$ (zero test): $a_k = 0$, and $a'_i = a_i$ for all $i = 1, \ldots, n$, or
– op is $\mathtt{c}_k\mathtt{--}$ (decrementation): $a'_k = a_k - 1$, and $a'_i = a_i$ for all $i \neq k$, or
– op is $\mathtt{c}_k\mathtt{++}$ (incrementation): $a'_k = a_k + 1$, and $a'_i = a_i$ for all $i \neq k$.

As usual, we write $\sigma \rightarrow_{\text{std}} \sigma'$ when $\sigma \xrightarrow{\delta}_{\text{std}} \sigma'$ for some $\delta \in \Delta$. Chains $\sigma_0 \rightarrow_{\text{std}} \sigma_1 \rightarrow_{\text{std}} \cdots \rightarrow_{\text{std}} \sigma_k$ of consecutive steps, also called *runs*, are denoted $\sigma_0 \xrightarrow{*}_{\text{std}} \sigma_k$, and also $\sigma_0 \xrightarrow{+}_{\text{std}} \sigma_k$ when $k > 0$. For example, M from Fig. 1 has a run:

$$\langle \ell_0,0,0 \rangle \rightarrow_{\text{std}} \langle \ell_1,0,0 \rangle \rightarrow_{\text{std}} \langle \ell_2,0,0 \rangle \rightarrow_{\text{std}} \langle \ell_0,1,0 \rangle \rightarrow_{\text{std}} \langle \ell_3,0,0 \rangle \rightarrow_{\text{std}} \langle \ell_0,0,1 \rangle$$
$$\rightarrow_{\text{std}} \langle \ell_1,0,1 \rangle \rightarrow_{\text{std}} \langle \ell_4,0,0 \rangle \rightarrow_{\text{std}} \langle \ell_1,1,0 \rangle \rightarrow_{\text{std}} \langle \ell_2,1,0 \rangle \rightarrow_{\text{std}} \langle \ell_0,2,0 \rangle \rightarrow_{\text{std}} \langle \ell_3,1,0 \rangle$$

For a vector $\boldsymbol{a} = (a_1, \ldots, a_n)$, or a configuration $\sigma = \langle \ell, \boldsymbol{a} \rangle$, we let $|\boldsymbol{a}| = |\sigma| \overset{\text{def}}{=} \sum_{i=1}^{n} a_i$ denote its *size*. For $N \in \mathbb{N}$, we say that a run $\sigma_0 \rightarrow_{\text{std}} \sigma_1 \rightarrow_{\text{std}} \cdots \rightarrow_{\text{std}} \sigma_k$ is *N-bounded* if $|\sigma_i| \leq N$ for all $i = 0, \ldots, k$.

The above definitions use a “std” subscript when writing steps to emphasize that they rely on the usual, standard, operational semantics of counter machines, where the behavior is *reliable*. We now introduce lossy counter machines as counter machines with a different semantics.

2.2 Lossy Counter Machines

In lossy counter machines, the contents of the counters may decrease non-deterministically (the machine can “leak”, or “lose data”). This behavior is not under the control of the machine, i.e., it can be seen as some inherent non-determinism. Furthermore, the lossy machine does not have any direct way of noticing if/when a loss occurs. Hence lossy counter machines are less powerful than standard, reliable, counter machines.

Technically, it is more convenient to see lossy machines as counter machines with a different operational semantics (and not as a special class of machines): thus it is possible to use simultaneously the two semantics and to relate them.

Formally, this is defined via the introduction of a partial ordering between the configurations of M:

$$\langle \ell, a_1, ..., a_n \rangle \leq \langle \ell', a'_1, ..., a'_n \rangle \overset{\text{def}}{\Leftrightarrow} \ell = \ell' \wedge a_1 \leq a'_1 \wedge \cdots \wedge a_n \leq a'_n.$$

One way to read $\sigma \leq \sigma'$ is to see σ as the result of some losses (possibly none) in σ'.

Now "lossy" steps, denoted $\sigma \xrightarrow{\delta}_{\text{lossy}} \sigma'$, are given by the following definition:

$$\sigma \xrightarrow{\delta}_{\text{lossy}} \sigma' \overset{\text{def}}{\Leftrightarrow} \exists \theta, \theta' : (\sigma \geq \theta \wedge \theta \xrightarrow{\delta}_{\text{std}} \theta' \wedge \theta' \geq \sigma'). \tag{†}$$

Note that reliable steps are a special case of lossy steps:

$$\sigma \rightarrow_{\text{std}} \sigma' \text{ implies } \sigma \rightarrow_{\text{lossy}} \sigma'. \tag{‡}$$

An immediate corollary of (†) is the so-called "monotonicity of steps" property:

Fact 2.1 ((Strong) Monotonicity)
1. Assume $\sigma \rightarrow_{lossy} \tau$. Then $\sigma' \rightarrow_{lossy} \tau'$ for all $\sigma' \geq \sigma$ and all $\tau' \leq \tau$.
2. Assume $\sigma \xrightarrow{+}_{lossy} \tau$. Then $\sigma' \xrightarrow{+}_{lossy} \tau'$ for all $\sigma' \geq \sigma$ and all $\tau' \leq \tau$.

Remark 2.2. Here the adjective "strong" emphasizes the fact that the existence of some step $\sigma \rightarrow_{\text{lossy}} \tau$ implies the existence of $\sigma' \rightarrow_{\text{lossy}} \tau$ for all $\sigma' \geq \sigma$, (rather than *some* $\sigma' \rightarrow_{\text{lossy}} \tau'$) and, symmetrically, the existence of $\sigma \rightarrow_{\text{lossy}} \tau'$ for all $\tau' \leq \tau$. □

2.3 Dickson's Lemma

The configuration ordering enjoys the following key property:

Fact 2.3 (Wqo). *$(Conf, \leq)$ is a well-quasi-ordering.*

This is otherwise known as Dickson's Lemma. It means that any infinite sequence $\sigma_0, \sigma_1, \sigma_2, \ldots$ of configurations contains an infinite increasing subsequence $\sigma_{i_0} \leq \sigma_{i_1} \leq \sigma_{i_2} \leq \cdots$. Equivalently, not only is the ordering well-founded (there is no infinite decreasing sequence $\sigma_0 > \sigma_1 > \sigma_2 > \cdots$) but every linearisation is well-founded. In particular, there is no infinite set of pairwise incomparable configurations. See [34] for more information.

It is the combination of monotonicity of steps with the wqo-property that turns lossy counter machines into what are called *well-structured transition systems* [28,5].

2.4 Semilinear Sets of Configurations

A set of configurations $R \subseteq Conf$ is *linear* if it can be written under the form

$$R = \{\langle \ell, \boldsymbol{a} + k_1.\boldsymbol{b}_1 + \cdots + k_m.\boldsymbol{b}_m \rangle \mid k_1, \ldots, k_m \in \mathbb{N}\}$$

for some *base configuration* $\langle \ell, \boldsymbol{a} \rangle$ and some finite set of increments $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m \in \mathbb{N}^n$. For example the upward-closure $\uparrow\sigma \overset{\text{def}}{=} \{\theta \in Conf \mid \theta \geq \sigma\}$ of a single configuration is linear, with σ itself as the base, and n unit vectors, one per counter, as increments. A second example is the singleton set $\{\sigma\}$, linear too, with same base but no increments.

A set $R \subseteq Conf$ is *semilinear* if it is a finite union $R = L_1 \cup \cdots \cup L_k$ of linear sets. In particular, the empty set is semilinear (take $k = 0$) and $Conf$ itself is semilinear as $\bigcup_{\ell \in Loc} \uparrow\langle \ell, \mathbf{0}\rangle$.

It is well-known that semilinear sets are exactly the sets that can be denoted by Presburger formulae (effective translations between the two representations exist) and that they are closed under complement, intersection, projection, etc., all this in an effective way [33,37]. Slightly abusing notations, we shall use letters like $X, Y, \ldots$ to denote semilinear sets of configurations and, at the same time, to denote their finitary descriptions (e.g., Presburger formulae, or bases *cum* increments) that can be given as input to algorithms.

Not all sets of configurations are semilinear but many interesting sets can be denoted by Presburger formulae (e.g., the set of all configurations whose size satisfy a Presburger constraint) and thus are semilinear.

The following is even more important for our purposes:

Fact 2.4 (Order-closed sets are semilinear). *If $R \subseteq Conf$ is upward-closed or downward-closed, it is semilinear.*

Indeed, by the wqo-property, an upward-closed R has *finitely many* minimal elements, hence can be written $R = \cup_{\sigma \in \min(R)} \uparrow\sigma$ which is semilinear. For a downward-closed R, we observe that its complement is upward-closed, hence semilinear, and rely on the fact that the complement of a semilinear set is semilinear.

3 Reachability and Safety

From now on, we omit the "lossy" subscript and write $\sigma \rightarrow \sigma'$ instead of $\sigma \rightarrow_{\text{lossy}} \sigma'$. This is because the lossy steps are our main objects. We only revert to the fully explicit notation when it is necessary to consider both reliable and lossy steps at the same time (for example in Section 7).

3.1 Post-sets and Pre-sets

For $R \subseteq Conf$, we let $Post(R) \overset{\text{def}}{=} \{\sigma' \mid \exists \sigma \in R : \sigma \rightarrow \sigma'\}$ denote the set of *immediate successors* of configurations in R. Similarly, we let $Post^*(R)$ and $Post^+(R)$ denote the set of configurations reachable from R through an arbitrary number (resp. strictly positive number) of steps. Similarly, $Pre(R)$, $Pre^*(R)$, and $Pre^+(R)$ denote sets of *predecessors* configurations, from which a configuration in R can be reached.

A consequence of monotonicity (Fact 2.1) is the following order-closure property:

Fact 3.1. *For any $R \subseteq Conf$, $Post(R)$ and $Post^+(R)$ are downward-closed sets, while $Pre(R)$ and $Pre^+(R)$ are upward-closed sets.*

Corollary 3.2. *For any $R \subseteq Conf$, $Pre(R)$, $Pre^+(R)$, $Post(R)$ and $Post^+(R)$ are semilinear.*

Furthermore, if R itself is semilinear, then $Post^*(R)$ and $Pre^*(R)$ too are semilinear.

Here the first point is just an applications of Fact 2.4 while the second point stems from $Post^*(R) = R \cup Post^+(R)$ and symmetrically for $Pre^*(R)$.

Note that, if R is semilinear, one can compute $Post(R)$ and $Pre(R)$ uniformly from R (and M). This has little to do with lossiness: counter machines is a low-level computational model with simple operational semantics for single steps. For counter machines, the one-step relations $\rightarrow_{\mathrm{std}}$ and $\rightarrow_{\mathrm{lossy}}$, seen a subsets of $Conf \times Conf$, are semilinear (and easily read out of M).

3.2 Reachability Problems

The main question is the decidability of a general form of reachability questions, that we call *general reachability* in order to distinguish it from its less general variants.

General_Reachability:

Given: a LCM M, two semilinear sets of configurations X and Y.

Question: does there exist $\sigma_1 \in X$ and $\sigma_2 \in Y$ such that $\sigma_1 \xrightarrow{*} \sigma_2$? In such a case, we write $X \xrightarrow{*} Y$.

Equivalently: Does $Post^*(X) \cap Y \neq \varnothing$? Does $Pre^*(Y) \cap X \neq \varnothing$?

In the literature, reachability problems often appear in other forms:

Configuration_Reachability: does $\sigma_0 \xrightarrow{*} \sigma_t$ for given starting configuration σ_0 and target configuration σ_t?

Location_Reachability: is there some $\boldsymbol{a} \in \mathbb{N}^n$ such that $\sigma_0 \xrightarrow{*} (\ell, \boldsymbol{a})$ for given σ_0 and target location $\ell \in Loc$?

Coverability: is there some $\sigma \geq \sigma_t$ such that $\sigma_0 \xrightarrow{*} \sigma$ for given σ_0 and target configuration to be covered σ_t?

Safety: does $Post^*(X_0) \subseteq X_s$ for given semilinear set of starting configurations X_0 and semilinear set of "safe" configurations X_s?

Obviously, all these problems are special cases of General_Reachability (or of its complement in the case of Safety), hence are easier. We observe that location reachability is a special case of coverability, and that coverability and single-configuration reachability almost coincide since, thanks to Fact 2.1, one can cover σ_g from σ_0 if, and only if, σ_g is reachable from σ_0 or is already covered by it (i.e., $\sigma_0 \geq \sigma_g$).

3.3 Decidability of Reachability

Theorem 3.3. General_Reachability *is decidable for lossy counter machines.*

First observe that general reachability is r.e. (it is enough to guess a run and check it, which amounts to simulating M) so that there only remains to show that non-reachability is r.e. too.

For this, we rely on semilinear invariants. An *inductive invariant*, or just "an invariant", is a set of configurations I such that $Post(I) \subseteq I$ or, equivalently, $Pre(J) \subseteq J$ letting $J \overset{\mathrm{def}}{=} Conf \setminus I$.

Classically, invariants are used to prove safety properties, relying on the following fact: if $R \subseteq I$ for some invariant I, then $Post^*(R) \subseteq I$. They can be used as negative witnesses for general reachability: finding an invariant I that contains X and does not intersect Y proves that one cannot reach Y from X, written $\neg(X \xrightarrow{*} Y)$ for short.

This method is *complete* since, if $\neg(X \xrightarrow{*} Y)$, this is certainly witnessed by invariants, the smallest one being $Post^*(X)$ and the largest being $Conf \setminus Pre^*(Y)$ [45]. The method can be made *effective* by restricting to semilinear invariants. Only considering semilinear invariants allows enumerating candidates sets I and it allows checking that a candidate I is indeed an invariant, that it contains X, and does not intersect Y. Restricting to semilinear invariant does not hinder completeness since, e.g., $Post^*(X)$ and $Conf \setminus Pre^*(Y)$ are semilinear (by Coro. 3.2).

Finally, general reachability is co-r.e., and being r.e. too, is decidable.

Remark 3.4. We observe that the key ingredient for the above proof is simply that the reachability sets $Post^*(X)$ are "regular" in some sense (for LCM's, they are semilinear) and that the one-step image $Post(X)$, or the pre-image $Pre(X)$, is semilinear too and can be computed effectively from X. This proof technique is quite general and applies to many different situations. For example, the same argument was used for reversible Petri nets in [14], or for 3-dim VASS's in [36]. □

Corollary 3.5. Configuration_Reachability, Location_Reachability, Coverability, *and* Safety *are decidable for lossy counter machines.*

3.4 Reachability Logic

The reachability problems we just proved decidable can all be stated in a first-order logic of reachability, where the basic predicates are $s \rightarrow t$, $s \xrightarrow{*} t$, and $s \in X$ for X a semilinear set.

For example, Safety is written

$$\forall s \in X_0 : \forall t \in X_s : \neg(s \xrightarrow{*} t), \tag{φ_{Saf}}$$

while configuration reachability and coverability are written, respectively,

$$\exists s \in \{\sigma_0\} : \exists t \in \{\sigma_t\} : s \xrightarrow{*} t, \tag{φ_{CR}}$$

$$\exists s \in \{\sigma_0\} : \exists t \in \uparrow\sigma_t : s \xrightarrow{*} t. \tag{φ_{Cov}}$$

These examples show that it is convenient to allow a simple language of terms denoting semilinear sets, like singletons "$\{\sigma\}$" or upward-closure "$\uparrow X$". Below we also use Boolean operations, e.g., "$X \setminus Y$", and order-theoretic constructions e.g., writing "$\min(X)$" to denote the set of minimal configurations in X. In any case we only use Presburger-definable operations: they always denote semilinear sets that can be computed effectively from their semilinear operands.

The model-checking problem for reachability logic is a natural generalization of the reachability problems we considered in Section 3.2. This problem is undecidable in general but identifying the decidable fragment is certainly an interesting question that is still very open. The question is even more interesting since there is ample room for refining and extending the logic in meaningful ways (see Section 8.1 for related questions).

Regarding some of the simplest non-trivial formulae, we can already provide a few results:

$$\exists s \in X : \exists t \in Y : s \xrightarrow{*} t \quad \text{decidable} \qquad \text{(one-to-one)}$$

$$\forall s \in X : \exists t \in Y : s \xrightarrow{*} t \quad \text{decidable} \qquad \text{(from-all)}$$

$$\exists s \in X : \forall t \in Y : s \xrightarrow{*} t \quad \text{undecidable, } \Sigma^0_2\text{-complete} \qquad \text{(one-to-all)}$$

$$\forall s \in X : \forall t \in Y : s \xrightarrow{*} t \quad \text{undecidable, } \Pi^0_1\text{-complete} \qquad \text{(all-to-all)}$$

$$\forall t \in Y : \exists s \in X : s \xrightarrow{*} t \quad \text{undecidable, } \Pi^0_1\text{-complete} \qquad \text{(to-all)}$$

$$\exists t \in Y : \forall s \in X : s \xrightarrow{*} t \quad \text{decidable} \qquad \text{(all-to-same)}$$

The undecidability results in the above list will be proven later, in Section 7. We mention them now because they are an indication that we should find the decidability results a bit surprising.

Regarding the decidability results, one-to-one formulae are just general reachability and have been shown decidable above. Observe that this entails the decidability of

$$\exists s \in X : \exists t \in Y : s \xrightarrow{+} t. \qquad \text{(one-to-one')}$$

Indeed this formula, also written $X \xrightarrow{+} Y$, is equivalent to both $\mathit{Post}(X) \xrightarrow{*} Y$ and $X \xrightarrow{*} \mathit{Pre}(Y)$, and $\mathit{Post}(X)$ and $\mathit{Pre}(Y)$ are semilinear sets that can be computed effectively from X and Y (and M), see Coro. 3.2.

Regarding from-all formulae, they reduce to conjunctions of simple reachability questions with the following reasoning:

$$\begin{aligned} & \forall s \in X : \exists t \in Y : s \xrightarrow{*} t \qquad \text{(from-all)} \\ \Leftrightarrow\ & \forall s \in (X \setminus Y) : \exists t \in Y : s \xrightarrow{+} t \\ \Leftrightarrow\ & \forall s \in \min(X \setminus Y) : \exists t \in Y : s \xrightarrow{+} t \end{aligned}$$

where the last step of the reduction relies on monotonicity of lossy steps (Fact 2.1). Now, $\min(X \setminus Y)$ is some finite set $\{\sigma_1, \ldots, \sigma_k\}$ (Fact 2.3) that can be computed effectively from X and Y. Thus we have reduced a from-all formula to a finite conjunction of one-to-one' formulae.

We now turn to all-to-same formulae. The main idea is easier to understand if we consider a version where $\xrightarrow{+}$ is used:

$$\exists t \in Y : \forall s \in X : s \xrightarrow{+} t. \qquad \text{(all-to-same')}$$

One can simplify this by using monotonicity *on both sides of the steps*:[1]

$$\exists t \in Y : \forall s \in X : s \xrightarrow{+} t \quad \Leftrightarrow \quad \exists t \in \min(Y) : \forall s \in \min(X) : s \xrightarrow{+} t.$$

[1] Here it is crucial that the source is universally quantified upon and the destination is existentially quantified upon. It would not work the other way around.

Hence, letting $\min(X) = \{\sigma_1, \ldots, \sigma_k\}$ and $\min(Y) = \{\sigma'_1, \ldots, \sigma'_m\}$, we have reduced all-to-same' to $\bigvee_{j=1}^{m} \bigwedge_{i=1}^{k} \sigma_i \xrightarrow{+} \sigma'_j$, a finite disjunction of conjunctions of decidable questions.

One can now show the decidability of all-to-same formulae by adapting the above idea. One possible way is to rely on, e.g.,

$$\exists t \in Y : \forall s \in X : s \xrightarrow{*} t \;\Leftrightarrow\; \begin{pmatrix} \exists t \in \min(Y) : \forall s \in \min(X) : s \xrightarrow{+} t \\ \vee\ \exists t \in \min(X) \cap Y : \forall s \in \min(X) : s \xrightarrow{*} t \end{pmatrix}.$$

Again, we end up with a finite combination of decidable reachability questions.

3.5 Computing Co-reachability Sets

One can go beyond Theorem 3.3 and compute the co-reachability sets.

Theorem 3.6 (*Pre** is effective). *For semilinear $X \subseteq Conf$, $Pre^*(X)$ can be computed effectively as a function of X and M.*

Indeed, we know that $Pre^*(X)$ is a semilinear set X_0 that satisfies both

$$X_0 \subseteq Pre^*(X), \quad \text{i.e., } \forall s \in X_0 : \exists t \in X : s \xrightarrow{*} t, \tag{1}$$

and

$$X_0 \supseteq Pre^*(X), \quad \text{i.e., } \neg\big(\exists s \notin X_0 : \exists t \in X : s \xrightarrow{*} t\big). \tag{2}$$

These two formulae are decidable for given X and X_0: (1) is a from-all formula while (2) is a negated one-to-one formula. Thus we can effectively recognize when a given X_0 coincides with $Pre^*(X)$. There only remains to enumerate all semilinear X_0 until we encounter $Pre^*(X)$, which is bound to eventually happen.

Computing $Pre^*(X)$ is useful in many situations where just deciding reachability questions would be insufficient. For example, Theo. 3.6 lets us list, or count, the number of starting configurations that do not satisfy a given safety property.

3.6 Computing Reachability Sets

Surprisingly, it is not possible to compute $Post^*(X)$ effectively. This is captured more precisely by the following statement:

Theorem 3.7 (On computing *Post**)
1. The question whether, for semilinear X and Y, $Post^*(X) \subseteq Y$ is decidable.
2. The question whether, for semilinear X and Y, $Post^*(X) \supseteq Y$ is Π^0_1-complete.

Indeed, Point 1 is the decidability of Safety, and Point 2 is the undecidability of to-all formulae (Section 3.4).

There is a troubling lack of symmetry here, between the computable Pre^* and the non-computable $Post^*$. We stress that this situation has little to do with the specifics of counter machines. Indeed, most of the proofs above only rely on monotonicity of steps, on Dickson's Lemma, and basic assumptions on the operational semantics (e.g., Presburger-definable one-step relation) that are fulfilled by many models. The bottom line is that most decidability proofs above rely on the closure properties stated in Coro. 3.2 where the asymmetry appears: upward-closed sets have a finite basis (on which one can base algorithms), while downward-closed sets do not.[2]

4 Termination and Inevitability

In this section, we consider termination and more general inevitability properties.

4.1 Termination

Consider the following problems:

Termination**:**

Given: a LCM M and an initial configuration σ_0,
Question: does M terminate?
Equivalently: are all runs starting from σ_0 finite?

Looping**:**

Given: a LCM M and an initial configuration σ_0,
Question: may the system loop? I.e., is there a configuration σ s.t. $\sigma_0 \xrightarrow{*} \sigma \xrightarrow{+} \sigma$?

Of course, looping is a special case of non-termination. That they coincide is less usual!

Lemma 4.1. *A lossy counter machine is looping if, and only if, it does not terminate.*

Indeed, assume there is an infinite run $\sigma_0 \to \sigma_1 \to \sigma_2 \to \ldots$. The wqo property entails that there must be positions $k < l$ along this run with $\sigma_k \leq \sigma_l$. Since $\sigma_k \xrightarrow{+} \sigma_l$, monotonicity (Fact 2.1) entails $\sigma_k \xrightarrow{+} \sigma_k$ and we have a loop.

Theorem 4.2. *Termination and looping are decidable for lossy counter machines.*

The proof of Theorem 4.2 is much simpler than one would expect.

First, we observe that termination is r.e.: since the transition relation is finitely branching, we know (Kőnig's Lemma) that if all runs from σ_0 are finite, then the tree of all runs is finite and an exhaustive simulation algorithm will terminate after examining finitely many runs.

On the other hand, looping too is r.e.: one just has to guess a looping run $\sigma_0 \xrightarrow{*} \sigma_k \xrightarrow{+} \sigma_k$, which can be represented finitely and checked in finite time.

Now since looping and non-termination coincide, the two problems are r.e. and co-r.e., hence decidable.

[2] Finite representations of upward-closed sets exist but they use some kind of "limits points" [27]. For lossy counter machines, the limit points are extended configurations where some counters contain ω. These behave like directed sets of configurations, not like real individual configurations.

Remark 4.3. The beauty of this proof is that termination and looping are r.e. *very generally*. That is to say, termination is r.e. for most sensible computation models, e.g., Turing machines or Minsky counter machines, and the same is true of looping. Thus that part of the proof is totally generic. What is *specific to lossy counter machines* is that non-termination and looping coincide. Indeed, they do not usually coincide for other models where a system may have infinite runs but no looping ones. □

4.2 Inevitability

Inevitability means that all runs will eventually stumble into something. We consider a slightly more general form:

Strong_Inevitability**:**
Given: a LCM M, an initial configuration σ_0, and two semilinear sets $X_1, X_2 \subseteq Conf$ of configurations,
Question: do all runs from σ_0 stay within X_1 until they eventually visit X_2?
Equivalently: does the *CTL* formula $\mathsf{A}[X_1 \,\mathcal{U}\, X_2]$ hold in σ_0?

Observe that termination is a special case of strong inevitability (by letting $X_2 = Halt \stackrel{\text{def}}{=} Conf \smallsetminus Pre(Conf)$ be the set of all "dead" configurations, from which no move is possible).

Theorem 4.4. *Strong inevitability is decidable for lossy counter machines.*

The reasoning is similar to what we did for termination: First, strong inevitability is r.e. There remains to see that it is also co-r.e., i.e. that there are finite witnesses for non-inevitability. So assume that there is a run that violates strong inevitability, that run is either finite or infinite. If it is finite, it is a finite witness. If it is infinite, then the LCM has an infinite run that remains inside $X_1 \smallsetminus X_2$. By the wqo property, there are two configurations $\sigma_i \leq \sigma_j$ along this run. By the monotonicity property, there is a looping run $\sigma_0 \xrightarrow{*} \sigma_i \xrightarrow{+} \sigma_i$. This looping run remains inside $X_1 \smallsetminus X_2$ and is the finite witness we need.

4.3 Undecidability

The decidability of termination and inevitability is very fragile. We only give two examples:

Uniform_Termination**:**
Given: a LCM M,
Question: does M terminate from all starting configurations $\sigma \in Conf$?

Repeated_Inevitability**:**
Given: a LCM M, an initial configuration σ_0, and a semilinear set $X \subseteq Conf$ of configurations,
Question: do all runs from σ_0 visit X infinitely many times?
Equivalently: does the *ECTL* formula $\mathsf{AF}^{\infty}X$ hold in σ_0?

Theorem 4.5. Uniform_Termination *and* Repeated_Inevitability *are* Π^0_1*-complete for lossy counter machines.*

For these two problems, membership in Π^0_1 is a consequence of the results we already saw. Indeed, the complement of Uniform_Termination can be written $\exists\sigma,\sigma' \in \mathit{Conf} : \sigma \xrightarrow{*} \sigma' \xrightarrow{+} \sigma'$, or even $\exists\sigma \in \mathit{Conf} : \sigma \xrightarrow{+} \sigma$, which is in Σ^0_1, while the complement of Repeated_Inevitability is $\exists$ a run $\sigma_0 \xrightarrow{*} \sigma \xrightarrow{+} \sigma$ such that X is not visited along the $\sigma \xrightarrow{+} \sigma$ loop. Π^0_1-hardness is shown as Coro. 7.2 in Section 7.

Corollary 4.6. *The set Halt of configurations from which M must terminate cannot be computed.*

Note that, for lossy counter machines, *Halt* is both downward-closed and an invariant, and it has a decidable membership problem (Theorem 4.2).

5 Büchi and Liveness

Here we consider the following problems:

Buchi**:**
  Given: a LCM M, a configuration σ_0, and a location $\ell \in \mathit{Loc}$,
  Question: is there a run starting from σ_0 that visits ℓ infinitely many times?

Looping_on_location**:**
  Given: a LCM M, a configuration σ_0, and a location $\ell \in \mathit{Loc}$,
  Question: is there a looping run on ℓ, i.e., does $\sigma_0 \xrightarrow{*} \langle \ell, \boldsymbol{a} \rangle \xrightarrow{+} \langle \ell, \boldsymbol{a} \rangle$ for some $\boldsymbol{a}$?

At first glance, the situation with Buchi and Looping_on_location appears very similar to what we encountered in Section 4. Now, instead of just considering the existence of infinite runs, we ask for infinite runs that visit a given ℓ infinitely many times. Still, we can adapt Lemma 4.1:

Lemma 5.1. Buchi *and* Looping_on_location *coincide.*

Proof. Obviously, Looping_on_location entails Buchi. For the reverse direction, assume there exists an infinite run visiting ℓ infinitely often:

$$\sigma_0 \xrightarrow{*} \langle \ell, \boldsymbol{a}_1 \rangle \xrightarrow{+} \langle \ell, \boldsymbol{a}_2 \rangle \xrightarrow{+} \langle \ell, \boldsymbol{a}_3 \rangle \xrightarrow{+} \cdots$$

By the wqo property, there exists some $\boldsymbol{a}_i \leq \boldsymbol{a}_j$ for some $i < j$. Hence $\langle \ell, \boldsymbol{a}_j \rangle \xrightarrow{+} \langle \ell, \boldsymbol{a}_j \rangle$ by the monotonicity property. Finally, we have proven the existence of a run looping on ℓ. □

From there, we cannot prove decidability by claiming that Buchi is both r.e. and co-r.e., as we did for non-termination, It is r.e. since looping on ℓ is. But the absence of Büchi runs does not have finite witnesses, as the absence of infinite runs has. (For Minsky machines, non-termination is Σ^0_1-complete while Buchi is Σ^1_1-complete).

Finally Buchi is undecidable:

Theorem 5.2. Buchi *and* Looping_on_location *are* Σ_1^0*-complete for lossy counter machines.*

For these two equivalent problems, membership in Σ_1^0 is clear. Σ_1^0-hardness is shown as Coro. 7.4 in Section 7.

6 Finiteness of the Reachability Sets

Here we consider the following problems:

Finiteness**:**

Given: a LCM M and an initial configuration σ_0,
Question: is the reachability set $Post^*(\sigma_0)$ finite?
Equivalently: (Boundedness) is there a *bound* $B \in \mathbb{N}$ such that $|\sigma| \leq B$ for all reachable σ?

Unbounded_Run**:**

Given: a LCM M and a configuration σ_0,
Question: is there an infinite run from σ_0 that visits ever larger configurations?
Equivalently: is there a run that visits infinitely many different configurations?

The two problems are complementary since a system is unbounded if, and only if, it has an unbounded run. To see this, which is not specific to lossy counter machines, assume that $Post^*(\sigma_0)$ is infinite. Since every reachable configuration is reachable via a *pure* run, i.e., a run that does not visit any configuration twice, we conclude that there are infinitely many pure runs. By arranging them in a tree and invoking Kőnig's lemma, we conclude that there exists an infinite pure run (since all its finite prefixes are pure). Hence M has an unbounded run.

6.1 Undecidability

Finiteness is undecidable for LCM's:

Theorem 6.1. Finiteness *is* Σ_1^0*-complete and* Unbounded_Run *is* Π_1^0*-complete for lossy counter machines.*

When it first surfaced (in [39]), undecidability of Finiteness was a bit surprising in a way that is difficult to explain in retrospect. The result is now well-known and we give a direct proof in Section 7. Before undecidability was known, there were two lines of reasoning pointing to a conjecture of decidability: firstly, the fact that $Post^*(\sigma_0)$ is regular suggested that one could compute it, and secondly, one expected Karp and Miller's procedure to extend to all monotonic systems, inferring an unbounded run from an increasing prefix $\sigma_0 \xrightarrow{*} \sigma_1 \xrightarrow{+} \sigma_2$ with $\sigma_1 < \sigma_2$.

6.2 Uniform Finiteness

Uniform finiteness is to finiteness what uniform termination is to termination:

Uniform_Finiteness:
Given: a LCM M,
Question: are all the reachability sets $Post^*(\sigma)$ finite?
Equivalently: does every run in M visit only finitely many different configurations?

Mayr showed that uniform finiteness is undecidable for lossy counter machines. This result is perhaps not surprising in view of the undecidability of finiteness. However the proof is still delicate since, in the encoding showing hardness, one cannot easily anchor the considered behaviors on some given natural starting configuration.

Theorem 6.2. Uniform_Finiteness *is Π^0_2-complete for lossy counter machines.*

Here, membership in Π^0_2 is obvious since finiteness is in Σ^0_1. For Π^0_2-hardness, we refer to Section 7.

7 Proving Undecidability

Undecidability, and more generally hardness, results are almost always established by reductions. This means taking some hard computational problems and encoding it in LCM's. This encoding can be tricky since, as we noted, LCM's are hard to control because of the possibly adversarial losses. Early undecidability proofs for lossy systems (e.g. [6,39,1]) are sometimes hard to understand and then to adapt to related problems.

In this section we want to explain how the idea of "*putting a counter machine on a budget*" can be used as a simple, yet versatile and powerful, gadget allowing easy-to-understand hardness proofs.

7.1 Putting Counter Machines on a Budget

With a Minsky counter machine $M = (Loc, C, \Delta)$ we associate a derived Minsky machine denoted $M^{\text{on\_budget}}$, or M^b for short.

In essence, $M^{\text{on\_budget}}$ is obtained by adding to M an extra "budget" counter $\mathtt{B}$ and by adapting the rules of Δ so that any incrementation (resp. decrementation) in the original counters is balanced by a corresponding decrementation (resp. incrementation) on the new counter $\mathtt{B}$. Thus, *the sum of the counters remains constant* in M^b. This is a classic idea in Petri nets and counter machines. The construction is described on a schematic example (Fig. 2) that is more explicit that a formal definition. Observe that extra intermediary locations (in gray) are used, and that a step in M that increments some $\mathtt{c}_i$ will be forbidden in M^b when the budget is exhausted (instead, M^b may reach a new, terminal, bankrupt location).

This construction enjoys a few obvious properties that we now state informally (formal statements are given in [44]).

***M^b* simulate *M*:** Any reliable run $\langle \ell, \boldsymbol{a}\rangle \xrightarrow{*}_{\text{std}} \langle \ell', \boldsymbol{a}'\rangle$ of M can be simulated as some $\langle \ell, B, \boldsymbol{a}\rangle \xrightarrow{*}_{\text{std}} \langle \ell', B', \boldsymbol{a}'\rangle$ in M^b provided with some large enough budget $B \in \mathbb{N}$.

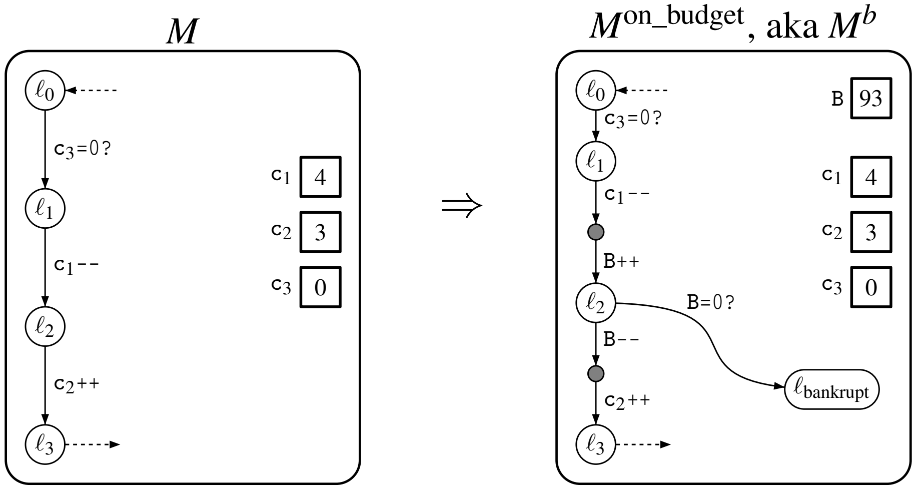

Fig. 2. From M to M^b (schematically)

M^b can only simulate M: Any reliable run in M^b can be seen as a run in M if we forget about the extra budget counter.

Counters are bounded: A lossy run $\langle \ell, \boldsymbol{a}\rangle \xrightarrow{*}_{\text{lossy}} \langle \ell', \boldsymbol{a}'\rangle$ in M^b has $|\boldsymbol{a}'| \leq |\boldsymbol{a}|$, i.e., the total sum of the counters can not increase.

Losses are visible: A lossy run $\langle \ell, \boldsymbol{a}\rangle \xrightarrow{*}_{\text{lossy}} \langle \ell', \boldsymbol{a}'\rangle$ in M^b is also a reliable run if, and only if, $|\boldsymbol{a}| = |\boldsymbol{a}'|$, i.e., if the total sum of the counters is unchanged (and the run does not bankrupt).

7.2 Undecidability of Uniform Termination

The above properties can be put to use immediately. Let M be some Minsky machine and $\sigma = \langle \ell, \boldsymbol{a}\rangle$ one of its configurations.

Proposition 7.1. *There is a loop $\langle \ell, \boldsymbol{a}\rangle \xrightarrow{+}_{std} \langle \ell, \boldsymbol{a}\rangle$ in M if, and only if, there is a $B \in \mathbb{N}$ and a loop $\langle \ell, B, \boldsymbol{a}\rangle \xrightarrow{+}_{lossy} \langle \ell, B, \boldsymbol{a}\rangle$ in M^b.*

Indeed, the loop in M is simulated in M^b by taking a large enough budget. And the loop in M^b must be a reliable run since the total sum of the counters is unchanged, hence it can be simulated in M.

Now recall that the question whether a Minsky machine has a loop $\sigma \xrightarrow{+}_{\text{std}} \sigma$ (where σ is existentially quantified upon) is undecidable, more precisely Σ^0_1-complete[3].

Corollary 7.2 (Undecidability). Uniform_Termination *is Π^0_1-hard for lossy counter machines.*

Indeed, M^b has an infinite run (starting from somewhere) if, and only if, it has a loop (from somewhere). Hence Π^0_1-hardness.

[3] This applies even if we do not restrict to configurations that are reachable from a given starting σ_0. I do not have a reference at hand but it is an easy exercise in computability theory.

7.3 Undecidability of Büchi Acceptance

We now show the undecidability of Buchi, or equivalently, of Looping_on_location, for lossy counter machines. This can be obtained by elaborating on the proof used for Coro. 7.2 above, but we find it more instructive to present another reduction that can be adapted for the next section.

Let M be a Minsky machine with a starting location ℓ_{init} and an accepting location ℓ_{end}. With M we associate a new machine M' obtained as follows (see schematics in Fig. 3): First we put M on a budget. Then we add two extra locations: ℓ_0 where B can be given any value, and ℓ_1 from which we can start M (on a budget). Finally, from ℓ_{end} it is possible to reset all counters to zero and go back to ℓ_1. This resetting uses the B (budget) counter to store the total sum the other counters had, using perhaps a few extra intermediary locations that are of no interest.

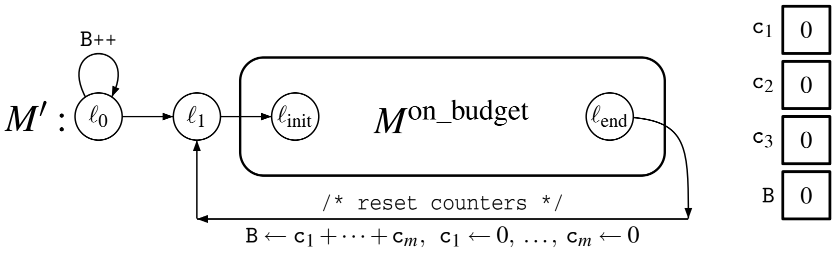

Fig. 3. Reduction for undecidability of Buchi

Proposition 7.3. *M has an accepting run $\langle \ell_{init}, \mathbf{0}\rangle \xrightarrow{*}_{std} \langle \ell_{end}, \boldsymbol{a}\rangle$ if, and only if, M' has a lossy run starting from $\langle \ell_0, \mathbf{0}\rangle$ and visiting ℓ_1 infinitely many times.*

Here, the left-to-right implication is clear: if M has an accepting run, this can be simulated by M' after it looped in ℓ_0 to start with a large enough budget. Once the accepting run has been completely simulated, M' can reset the counters, go back to ℓ_1 and repeat the simulation infinitely many times.

Reciprocally, if M' has a run that visits ℓ_1 infinitely many times, this run cannot increase the total sum of the counters once it has left ℓ_0. Hence this total sum can only decrease or stay constant. If the run is infinite, the total sum will eventually stay constant. Thus, after some time, the lossy run only has reliable steps. Since it visits ℓ_1 (and thus also ℓ_{init} and ℓ_{end}) infinitely many times, after some time its reliable steps will witness an accepting run of M.

Since the existence of an accepting run is Σ^0_1-complete for Minsky machines, we deduce:

Corollary 7.4 (Undecidability). Buchi *is Σ^0_1-hard for lossy counter machines.*

7.4 Undecidability of Finiteness

Our next reduction is a simple adaptation of the previous one (see schematics in Fig. 4). The modifications are as follows: (1) the resetting of the counters is not reached from ℓ_{end} but from the bankrupt location ℓ_{bankrupt} that M^b reaches when its budget appears to be too small (recall Fig. 2), and (2) the initial value of B cannot be chosen as large as one wants via a loop on ℓ_0: instead, B can only be incremented in the step from ℓ_1 to ℓ_{init}.

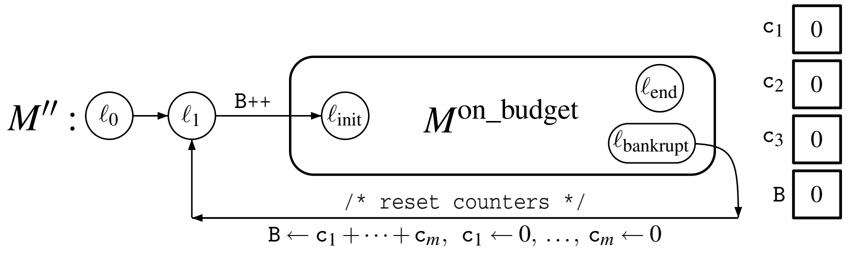

Fig. 4. Reduction for undecidability of Finiteness

Proposition 7.5. *M has an unbounded (reliable) run starting from $\langle \ell_{init}, \mathbf{0}\rangle$ if, and only if, M'' has an unbounded (lossy) run starting from $\langle \ell_0, \mathbf{0}\rangle$.*

Here again the left-to-right implication is clear. The unbounded run of M can be simulated by M''. This simulation is done in incremental stages. First M'' reaches ℓ_{init} with a low budget $B = 1$. The simulation proceeds until the budget is too low for continuing. M'' is then in the bankrupt location, resets its counters and goes back to ℓ_1. There B is incremented and the simulation can be started from scratch, this time with $B = 2$. It will now take more steps before bankrupting, resetting the counters, and starting again with a larger budget. This simulation will reenact longer and longer prefixes of the unbounded run of M, leading to a run of M'' that is itself unbounded.

The right-to-left implication is more subtle. Assume M'' has an unbounded run. Necessarily, this run visits ℓ_1 infinitely many times since this is the only way to increase the total sum of the counters. Let us write this unbounded run in the following way, isolating the places where ℓ_1 is visited:

$$\langle \ell_0, \mathbf{0}\rangle \xrightarrow{+} \langle \ell_1, \boldsymbol{a}_1\rangle \xrightarrow{+} \langle \ell_1, \boldsymbol{a}_2\rangle \xrightarrow{+} \langle \ell_1, \boldsymbol{a}_3\rangle \xrightarrow{+} \cdots$$

Zooming in a little bit on the part between two consecutive visits to ℓ_1, we see it must be some subrun π_i of the form

$$\langle \ell_1, \boldsymbol{a}_i\rangle \equiv \langle \ell_1, B_i, \mathbf{0}\rangle \to \langle \ell_{\text{init}}, 1 + B_i, \mathbf{0}\rangle \xrightarrow{*} \langle \ell_{\text{bankrupt}}, B, \boldsymbol{c}\rangle \xrightarrow{+} \langle \ell_1, B_{i+1}, \mathbf{0}\rangle \equiv \langle \ell_1, \boldsymbol{a}_{i+1}\rangle.$$

Now, $B_{i+1} \leq 1 + B_i$ since "Counters are bounded" and the sequence $B_1, B_2, \ldots$ can only increase by 1 at a time. It can also decrease (by losses) but, since the run is unbounded, it must eventually increase and for every $k \in \mathbb{N}$, there is an index i such that $B_i = k$. If now we assume that i_k is the first such index, we deduce $B_{i_k} = 1 + B_{i_k-1}$, hence the run π_{i_k-1} only uses reliable steps (indeed, "Losses are visible"). Reliable steps simulate M, hence π_{i_k-1} witnesses a run $\pi'_k \equiv \langle \ell_{\text{init}}, \mathbf{0} \rangle \xrightarrow{+} \langle \ell, \boldsymbol{c} \rangle$ for some ℓ and some $\boldsymbol{c}$ of size k. If we assume that M is deterministic, these runs are longer and longer prefixes of the infinite unbounded run of M. If M is non-deterministic, we use Kőnig's Lemma to extract an unbounded run from these ever larger finite runs.

Since the existence of an unbounded run is Π^0_1-complete for Minsky machines, we deduce:

Corollary 7.6 (Undecidability). Finiteness *is Σ^0_1-hard for lossy counter machines.*

The reduction also shows undecidability for the to-all and all-to-all formulae of the reachability logic (Section 3.4). For to-all formulae, i.e., formulae of the form $\forall t \in Y : \exists s \in X : s \xrightarrow{*} t$, we observe that by taking $X = \{\sigma_0\}$ and $Y = \{\langle \ell_1, k, \mathbf{0} \rangle \mid k \in \mathbb{N}\}$, the formula expresses the existence of an unbounded run in M''. Since in this reduction X is a singleton, the reduction also works for all-to-all formulae, of the form $\forall s \in X : \forall t \in Y : s \xrightarrow{*} t$.

7.5 Undecidability of Uniform Finiteness

We further adapt the previous reduction (see schematics in Fig. 5). Now M''' has an extra counter K that is never modified and that is used to store a value with which to reinitialize $\mathtt{c}_1$ when looping back to ℓ_1.

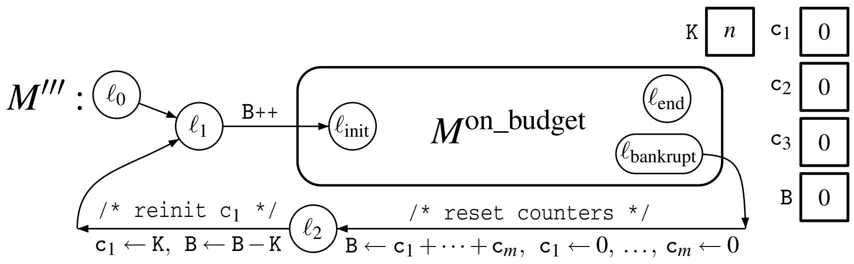

Fig. 5. Reduction for undecidability of Uniform_Finiteness

Proposition 7.7. *M has an unbounded (reliable) run starting from some $\langle \ell_{init}, \mathtt{c}_1 : n, \mathbf{0} \rangle$ if, and only if, M''' has an unbounded (lossy) run starting from some σ.*

We reason as for the proof of Prop. 7.5, with very minor adaptations.

Again, the left-to-right implication is the easier one. Assume M has an unbounded run from $\langle \ell_{\text{init}}, n, \mathbf{0} \rangle$. This can be simulated by M''' starting from $\langle \ell_{\text{init}}, B, \mathtt{K} : n, \mathtt{c}_1 : n, \mathbf{0} \rangle$, i.e., after we make sure that the extra counter K contains exactly n. As with Proposition 7.5, the simulation proceeds until the budget bankrupts. Then, M''' loops back to ℓ_1, where the budget is incremented. and the simulation starts anew. This loop back to ℓ_1 resets the counters with $c_1 = n$, using the memory K to find the value (truly, a Minsky machine needs an auxiliary storage for this copy, but $\mathtt{c}_2$ can do the job). By visiting ℓ_1 infinitely many times, this simulation manages to produce an unbounded run of M'''.

For the right-to-left implication, we assume that M''' has an unbounded run from some arbitrary σ. Since the only way to increase the total sum of the counters is to go through ℓ_1, the run must visit ℓ_1 infinitely many times, and increase the total sum of the counters by at most one between such visits. Also, since $\mathtt{K}$ can only decrease (by losses) it will eventually stays constant. Once $\mathtt{K}$ is constant (say $= n$), we have, for any $k \in \mathbb{N}$, a run like $\pi_{i_k - 1}$ above that increments B from $k-1$ to k, going from $\langle \ell_{\text{init}}, \mathtt{B} : k-1, \mathtt{K} : n, \mathtt{c}_1 : n, \mathbf{0}\rangle$ to $\langle \ell_{\text{init}}, k, n, n, \mathbf{0}\rangle$. This run only uses reliable steps and witnesses, inside the Minsky machine, a path $\langle \ell_{\text{init}}, n, \mathbf{0}\rangle \xrightarrow{*} \langle \ell, \boldsymbol{a}\rangle$ for some $\boldsymbol{a}$ of size k. Hence M has an unbounded run from $\langle \ell_{\text{init}}, n, \mathbf{0}\rangle$.

Since the question whether there exists some $n \in \mathbb{N}$ such that a Minsky machine has an unbounded run starting from $\langle \ell_{\text{init}}, n, \mathbf{0}\rangle$ is Σ^0_2-complete, we deduce:

Corollary 7.8 (Undecidability). Uniform_Finiteness *is Π^0_2-hard for lossy counter machines.*

The reduction also shows Σ^0_2-hardness of the one-to-all formulae of the reachability logic. These have the form $\exists s \in X : \forall t \in Y : s \xrightarrow{*} t$. By taking $X = \mathit{Conf}$ and $Y = \{\langle \ell_2, \mathtt{B} : k, \mathbf{0}\rangle \mid k \in \mathbb{N}\}$, the formula expresses the existence of an unbounded run in M''', i.e., the negation of uniform finiteness.

8 Further Developments

We gather in this section a few results, remarks, and pointers to the literature, regarding problems that are less central in the theory of lossy counter machines as it exists today.

8.1 Temporal Logic Model-Checking

Temporal logics [22] can express behavioral properties of systems in general, and of lossy counter machines in particular. It has been observed in the literature on lossy systems that temporal logic model-checking is generally undecidable (e.g., [6] shows the undecidability of both *CTL* model-checking and *LTL* model-checking for lossy channel systems). However, as with the reachability logic we considered in Section 3.4, the picture can be more interesting if we focus on relevant fragments of general logics.

For lossy counter machines, the $\exists$*CTL* fragment of *CTL* has a decidable model-checking problem. This fragment, also denoted $B(\mathsf{E}\mathcal{U}, \mathsf{EX})$, is the branching-time logic built on two *CTL* modalities $\mathsf{E}\mathcal{U}$ and EX. Arbitrary nesting and Boolean combinations are allowed, and we take all the semilinear sets as basic propositions.

Theorem 8.1 (Decidability of $\exists$*CTL* model-checking)
1. The problem, given a LCM M, a configuration σ, and an $\exists$CTL formula φ, whether $M, \sigma \models \varphi$, is decidable.
2. Moreover, the set $\mathsf{Mod}(\varphi) \overset{def}{=} \{\sigma \in \mathit{Conf} \mid M, \sigma \models \varphi\}$ *is a semilinear set that can be computed effectively from M and φ.*

Computing $\mathsf{Mod}(\varphi)$ is done by induction over the structure of φ. This uses standard techniques like

$$\begin{aligned}\mathsf{Mod}(\neg\varphi) &= \mathit{Conf} \smallsetminus \mathsf{Mod}(\varphi),\\ \mathsf{Mod}(\varphi \vee \psi) &= \mathsf{Mod}(\varphi) \cup \mathsf{Mod}(\psi),\\ \mathsf{Mod}(\mathsf{EX}\varphi) &= \mathit{Pre}(\mathsf{Mod}(\varphi)),\end{aligned}$$

and relies on the fact that semilinear sets are closed under complementation, union and the *Pre* operator, all this in an effective way.

For $\mathsf{Mod}(\mathsf{E}\varphi\mathcal{U}\psi)$, semilinearity is seen after one unfolding of the Until:

$$\begin{aligned}\mathsf{Mod}(\mathsf{E}\varphi\mathcal{U}\psi) &= \mathsf{Mod}(\psi \vee \varphi \wedge \mathsf{EXE}\varphi\mathcal{U}\psi)\\ &= \mathsf{Mod}(\psi) \cup \mathsf{Mod}(\varphi) \cap \mathit{Pre}(\mathsf{Mod}(\mathsf{E}\varphi\mathcal{U}\psi)).\end{aligned}$$

The last expression denotes a semilinear set since $\mathit{Pre}(\cdots)$ is always semilinear.

The computability of $\mathsf{Mod}(\mathsf{E}\varphi\mathcal{U}\psi)$ can be shown with the same technique we used, in Section 3.5, for the computability of $\mathit{Pre}^*(X)$. Alternatively, one can use backward-chaining algorithms whose termination is guaranteed by Dickson's Lemma (see [9]).

Remark 8.2. The same techniques can be used to enlarge decidability from $\exists$*CTL* to some existential fragment of the branching-time mu-calculus, where regular properties like "*there exists a run along which every even-numbered configuration is in X*" can be stated. See [11,9]. □

Regarding other temporal modalities, we know that model checking one $\mathsf{A}\varphi\mathcal{U}\psi$ formula is decidable when $\mathsf{Mod}(\varphi)$ and $\mathsf{Mod}(\psi)$ are effectively given semilinear sets (this is the decidability of Strong_Inevitability from Section 4.2) but it is *not possible* to compute $\mathsf{Mod}(\mathsf{A}\varphi\mathcal{U}\psi)$, nor even (by Coro. 4.6) $\mathsf{Mod}(\mathsf{AF}\neg\mathsf{EX}\top)$.

As a consequence, nested $\mathsf{A}\mathcal{U}$ modalities give undecidable model-checking problems (e.g., they can easily encode uniform termination).

Model-checking is also undecidable for *ECTL* modalities like EF^∞ (this is the Buchi problem from Section 5) and AF^∞ (this is repeated inevitability from Section 4.3).

8.2 Games People Play on Lossy Counter Machines

Sections 3 to 5 focused on classical reachability, inevitability, or liveness properties, but one is also interested in more general game-theoretical problems where several opponents have conflicting goals. Branching-time temporal logic is only a first step toward these new issues.

The question of checking game-theoretical properties of lossy counter machines has barely been scratched. Obviously, one could expect that undecidability is everywhere since the properties are more general. One could be wrong.

Let us illustrate this on an example. We consider a reachability game played in turn by two opponents on a single LCM. Starting from σ_0, Alice tries to reach ℓ_{end} by picking the odd-numbered lossy steps of a growing run, while Bob tries to frustrate her by choosing adversarially the even-numbered lossy steps. The decision problem is:

Reachability_Game**:**
Given: a LCM M, an initial configuration σ_0, and a goal location ℓ_{end}.
Question: does Alice have a winning strategy?

Surprisingly, this problem is very easy.

Theorem 8.3. Reachability_Game *is* PTIME-*complete for lossy counter machines.*

The paradox is explained when we realize that an optimal strategy for both players can choose to always lose all the contents of the counters at every step. Indeed, losing everything can only reduce our opponent's options (because of strong monotonicity). It also reduces our later options, but anyway the opponent will have the possibility to lose everything if it hurts us.

Finally, it is possible to solve the game by restricting to the finite graph of all configurations $\langle \ell, \mathbf{0} \rangle$ for $\ell \in Loc$, which is PTIME-complete.

Games on LCM's can be more interesting. We could decide that Bob can only play reliable steps. Or that Alice and Bob choose reliable steps while losses in the counters are chosen probabilistically by the environment, leading to games with $2\,{}^1\!/_2$ players. Or that the objective is more complex than just reachability. Many variations are possible, motivated by different situations. We refer to [3,41,9,10,2,4] for results on such games.

8.3 Equivalence Checking

Comparing two systems is a classic decision problem. In the simplest situations, the comparison criterion is an equivalence relation, sometimes a preorder.

When dealing with systems (like LCM's) that give rise to infinite-state transition systems, the behavioral equivalences one could use for verification purposes are often undecidable. The main exception is *strong bisimilarity* that has been shown decidable in many cases (and undecidable in many other cases) [13].

For lossy counter machines, equivalences are hard. One way to put it is to say that all interesting relations between lossy counter machines are undecidable, even if we only consider lossy VAS's (i.e., lossy counter machines without zero-tests). A proof for all relations between bisimilarity and trace containment can be obtained (see [42]) by adapting Jančar's classic proof for Petri nets [30]. The proof certainly extends, e.g., to all equivalences between equality of the reachability set and trace containment modulo invisibility of internal steps.

On the other hand, comparison between a lossy counter machine and a *finite transition system* is very often decidable.

This line of positive results was started by Abdulla and Kindahl [8] with the simulation preorder and the bisimulation equivalence.

It turns out that there is a generic approach to these problems: the question whether $S \preceq F$ or $S \approx F$ for some finite F can often be translated as a temporal question, whether $S \models \varphi$ for some formula $\varphi = \varphi_F^{\preceq}$ or $\varphi = \varphi_F^{\approx}$, called a *characteristic formula for F*, that states exactly what is required to be $\preceq F$ or $\approx F$. We refer to [12,31,35] for more details.

In the special case of lossy counter machines, comparison with finite systems is decidable for all the equivalences and preorders that admit characteristic formulae in $\exists$*CTL*. This is a direct corollary of Theorem 8.1. The equivalences and preorders thus covered are numerous and include, e.g., weak bisimulation and branching bisimulation.

9 Decidable but Hard

Problems that are decidable for lossy counter machines are usually very hard.

9.1 Lower Bounds for Complexity

Reachability and termination are Ackermann-hard for LCM's. We refer to [44] for a recent and simplified proof that uses the same "counter machine on a budget" gadget that we used in Section 7. Hardness extends, via obvious reductions, to most decidable problems we listed in the previous sections (one major exception is the reachability game from Section 8.2).

A finer analysis of the lower bounds shows that the most important parameter here is the *number of counters* in a lossy counter machine. The hardness proof uses a number of counters that cannot be bounded *a priori*. For a fixed number of counters, one only obtains lower bounds at a finite, primitive-recursive, level in the Fast Growing Hierarchy, see [44]. This is in accordance with what is known on upper bounds.

9.2 Upper Bounds

All along this paper, we deliberately avoided giving explicit algorithms for our decidability proofs. However, algorithms exist in the literature. Their termination arguments usually rely on the wqo property, and more precisely Dickson's Lemma. From these, upper bounds can be deduced, based on the length of bad sequences for the $(\mathbb{N}^n, \leq)$ wqo [40,17].

These upper bounds lie in the Fast Growing Hierarchy. The good news is that they closely match the known lower bounds. In particular, an Ackermann upper bound holds for most decidable problems on lossy counter machines, and this can be refined to primitive-recursive upper bounds at various levels when one restricts attention to machines with a fixed number of counters. We refer to our upcoming paper for more details [23].

10 Concluding Remarks

Lossy counter machines are a paradoxical computational model where unreliability brings decidability. At the moment, they have mostly been used as a tool for hardness results (undecidability or Ackermann-hardness). They have sometimes been used under the symmetrical guise of counters with incrementation errors [19].

In a leisurely way, we surveyed the main known results on both sides of the decidability frontier. From this, two main conclusions emerge:

1. Most decidability results rely only superficially on specific features of lossy counter machines. They can be obtained by a combination of very general properties enjoyed by most models (e.g., finitely branching non-determinism, effective one-step relation, ...) and the combination of strong monotonicity of steps with the wqo property of configurations. As a consequence, most of our decidability proofs can be easily adapted to other classes of well-structured transition systems. For example, they hold *mutatis mutandis* for lossy channel systems [7] or Reset Petri nets [20].

2. Most hardness results can be proved with the "machine on a budget" gadget. For counter systems, this gadget is used in two different ways (pioneered by [21]). It can bound the total sum of the counters, so that this sum must eventually stabilize along an infinite behavior, or can only grow in controlled ways. Then, when the sum is stabilized, the behavior must be reliable and hardness can be inherited from the Turing-powerful Minsky machines.

Acknowledgements. We thank Pierre Chambart, Jérôme Leroux and Sylvain Schmitz who greatly helped by proof-reading this paper at various stages.

References

1. Abdulla, P.A., Baier, C., Purushothaman Iyer, S., Jonsson, B.: Simulating perfect channels with probabilistic lossy channels. Information and Computation 197(1–2), 22–40 (2005)
2. Abdulla, P.A., Ben Henda, N., de Alfaro, L., Mayr, R., Sandberg, S.: Stochastic games with lossy channels. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 35–49. Springer, Heidelberg (2008)
3. Abdulla, P.A., Bertrand, N., Rabinovich, A., Schnoebelen, P.: Verification of probabilistic systems with faulty communication. Information and Computation 202(2), 141–165 (2005)
4. Abdulla, P.A., Bouajjani, A., d'Orso, J.: Monotonic and downward closed games. Journal of Logic and Computation 18(1), 153–169 (2008)
5. Abdulla, P.A., Čerāns, K., Jonsson, B., Tsay, Y.-K.: Algorithmic analysis of programs with well quasi-ordered domains. Information and Computation 160(1/2), 109–127 (2000)
6. Abdulla, P.A., Jonsson, B.: Undecidable verification problems for programs with unreliable channels. Information and Computation 130(1), 71–90 (1996)
7. Abdulla, P.A., Jonsson, B.: Verifying programs with unreliable channels. Information and Computation 127(2), 91–101 (1996)
8. Abdulla, P.A., Kindahl, M.: Decidability of simulation and bisimulation between lossy channel systems and finite state systems. In: Lee, I., Smolka, S.A. (eds.) CONCUR 1995. LNCS, vol. 962, pp. 333–347. Springer, Heidelberg (1995)
9. Baier, C., Bertrand, N., Schnoebelen, P.: On computing fixpoints in well-structured regular model checking, with applications to lossy channel systems. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, pp. 347–361. Springer, Heidelberg (2006)
10. Baier, C., Bertrand, N., Schnoebelen, P.: Verifying nondeterministic probabilistic channel systems against ω-regular linear-time properties. ACM Trans. Computational Logic 9(1) (2007)
11. Bouajjani, A., Mayr, R.: Model checking lossy vector addition systems. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 323–333. Springer, Heidelberg (1999)

12. Browne, M.C., Clarke, E.M., Grumberg, O.: Characterizing finite Kripke structures in propositional temporal logic. Theoretical Computer Science 59(1–2), 115–131 (1988)
13. Bukart, O., Caucal, D., Moller, F., Steffen, B.: Verification on infinite structures. In: Bergstra, J.A., Ponse, A., Smolka, S.A. (eds.) Handbook of Process Algebra, ch. 9, pp. 545–623. Elsevier, Amsterdam (2001)
14. Cardoza, E., Lipton, R., Meyer, A.R.: Exponential space complete problems for Petri nets and commutative subgroups. In: Proc. STOC '76, pp. 50–54. ACM Press, New York (1976)
15. Cécé, G., Finkel, A., Purushothaman Iyer, S.: Unreliable channels are easier to verify than perfect channels. Information and Computation 124(1), 20–31 (1996)
16. Chambart, P., Schnoebelen, P.: Computing blocker sets for the Regular Post Embedding Problem. In: Proc. DLT 2010. LNCS. Springer, Heidelberg (to appear, 2010)
17. Clote, P.: On the finite containment problem for Petri nets. Theoretical Computer Science 43(1), 99–105 (1986)
18. Demri, S.: Linear-time temporal logics with Presburger constraints: An overview. J. Applied Non-Classical Logics 16(3-4), 311–347 (2006)
19. Demri, S., Lazić, R.: LTL with the freeze quantifier and register automata. In: Proc. LICS 2006, pp. 17–26. IEEE Computer Society Press, Los Alamitos (2006)
20. Dufourd, C., Finkel, A., Schnoebelen, P.: Reset nets between decidability and undecidability. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 103–115. Springer, Heidelberg (1998)
21. Dufourd, C., Jančar, P., Schnoebelen, P.: Boundedness of reset P/T nets. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 301–310. Springer, Heidelberg (1999)
22. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, ch. 16, vol. B, pp. 995–1072. Elsevier, Amsterdam (1990)
23. Figueira, D., Figueira, S., Schmitz, S., Schnoebelen, P.: Ackermann and primitive-recursive upper bounds with Dickson's lemma (2010) (in preparation)
24. Figueira, D., Segoufin, L.: Future-looking logics on data words and trees. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 331–343. Springer, Heidelberg (2009)
25. Finkel, A.: A generalization of the procedure of Karp and Miller to well structured transition systems. In: Ottmann, T. (ed.) ICALP 1987. LNCS, vol. 267, pp. 499–508. Springer, Heidelberg (1987)
26. Finkel, A.: Reduction and covering of infinite reachability trees. Information and Computation 89(2), 144–179 (1990)
27. Finkel, A., Goubault-Larrecq, J.: Forward analysis for WSTS, part I: Completions. In: Proc. STACS 2009. Leibniz International Proceedings in Informatics, vol. 3, pp. 433–444. Leibniz-Zentrum für Informatik (2009)
28. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theoretical Computer Science 256(1–2), 63–92 (2001)
29. Henzinger, T.A., Majumdar, R., Raskin, J.-F.: A classification of symbolic transition systems. ACM Trans. Computational Logic 6(1), 1–32 (2005)
30. Jančar, P.: Undecidability of bisimilarity for Petri nets and some related problems. Theoretical Computer Science 148(2), 281–301 (1995)
31. Jančar, P., Kučera, A., Mayr, R.: Deciding bisimulation-like equivalences with finite-state processes. Theoretical Computer Science 258(1–2), 409–433 (2001)
32. Jurdziński, M., Lazić, R.: Alternation-free modal mu-calculus for data trees. In: Proc. LICS 2007, pp. 131–140. IEEE Comp. Soc. Press, Los Alamitos (2007)
33. Kracht, M.: A new proof of a theorem by Ginsburg and Spanier. Dept. Linguistics, UCLA (December 2002) (manuscript)
34. Kruskal, J.B.: The theory of well-quasi-ordering: A frequently discovered concept. Journal of Combinatorial Theory, Series A 13(3), 297–305 (1972)

35. Kučera, A., Schnoebelen, P.: A general approach to comparing infinite-state systems with their finite-state specifications. Theoretical Computer Science 358(2–3), 315–333 (2006)
36. van Leeuwen, J.: A partial solution to the reachability-problem for vector-addition systems. In: Proc. STOC '74, pp. 303–309. ACM Press, New York (1974)
37. Leroux, J., Point, G.: TaPAS: The Talence Presburger Arithmetic Suite. In: Kowalewski, S., Philippou, A. (eds.) Proc. TACAS 2009. LNCS, vol. 5505, pp. 182–185. Springer, Heidelberg (2009)
38. Mayr, R.: Undecidable problems in unreliable computations. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 377–386. Springer, Heidelberg (2000)
39. Mayr, R.: Undecidable problems in unreliable computations. Theoretical Computer Science 297(1–3), 337–354 (2003)
40. McAloon, K.: Petri nets and large finite sets. Theoretical Computer Science 32(1–2), 173–183 (1984)
41. Raskin, J.-F., Samuelides, M., Van Begin, L.: Games for counting abstractions. In: Proc. AVoCS 2004. Electronic Notes in Theor. Comp. Sci, vol. 128(6), pp. 69–85. Elsevier Science, Amsterdam (2005)
42. Schnoebelen, P.: Bisimulation and other undecidable equivalences for lossy channel systems. In: Kobayashi, N., Pierce, B.C. (eds.) TACS 2001. LNCS, vol. 2215, pp. 385–399. Springer, Heidelberg (2001)
43. Schnoebelen, P.: Verifying lossy channel systems has nonprimitive recursive complexity. Information Processing Letters 83(5), 251–261 (2002)
44. Schnoebelen, P.: Revisiting Ackermann-Hardness for Lossy Counter Machines and Reset Petri Nets. In: Hliněny, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 616–628. Springer, Heidelberg (2010)
45. Sifakis, J.: A unified approach for studying the properties of transitions systems. Theoretical Computer Science 18, 227–258 (1982)
46. Tan, T.: On pebble automata for data languages with decidable emptiness problem. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 712–723. Springer, Heidelberg (2009)