Computing Blocker Sets for the Regular Post Embedding Problem*

P. Chambart and Ph. Schnoebelen

LSV, ENS Cachan, CNRS
61, av. Pdt. Wilson, F-94230 Cachan, France
email: {chambart|phs}@lsv.ens-cachan.fr

Abstract. Blocker and coblocker sets are regular languages involved in the algorithmic solution of the Regular Post Embedding Problem. We investigate the computability of these languages and related decision problems.

1 Introduction

Post's Embedding Problem (shortly PEP, named by analogy with Post's Correspondence Problem) is the question whether two morphisms on words $u, v : \Sigma^* \to \Gamma^*$ agree non-trivially on some input, i.e., whether $u(x)$ is a (scattered) subword of $v(x)$ for some $x \in \Sigma^+$. The *subword* ordering, also called *embedding*, is denoted "$\sqsubseteq$": $x \sqsubseteq y \stackrel{\text{def}}{\Leftrightarrow} x$ can be obtained from y by erasing some letters, possibly all of them, possibly none.

PEP is trivial if there are no restrictions on the form of solutions. But when one looks for solutions x as above belonging to a regular language $R \subseteq \Sigma^*$, the problem (hereafter called the *Regular* Post Embedding Problem, or $\mathsf{PEP}^{\text{reg}}$) becomes very interesting: decidable but surprisingly hard [1].

The Regular Post Embedding Problem was introduced in [1, 2] where it is shown that $\mathsf{PEP}^{\text{reg}}$ is expressive enough to encode problems on lossy channel systems. In fact, encodings in both directions exist, hence $\mathsf{PEP}^{\text{reg}}$ is exactly at level $\mathcal{F}_{\omega^\omega}$ in the Fast Growing Hierarchy. Thus, although it is decidable, $\mathsf{PEP}^{\text{reg}}$ is not primitive-recursive, and not even multiply-recursive (see [3] and the references therein). Finally, $\mathsf{PEP}^{\text{reg}}$ is an abstract problem that is inter-reducible with a growing list of decidable problems having the same $\mathcal{F}_{\omega^\omega}$ complexity: metric temporal logic [4], alternating one-clock timed automata [5, 6], leftist grammars [7, 8], products of modal logics [9], etc.

Blockers and coblockers. The original decision algorithm for $\mathsf{PEP}^{\text{reg}}$ relies on so-called "*blocker*" and "*coblocker*" sets [1]. Write Sol_L for the set $\{x \in L \mid u(x) \sqsubseteq v(x)\}$ of solutions in some *constraint* language $L \subseteq \Sigma^*$ and define:

$$X_L \stackrel{\text{def}}{=} \{\alpha \in \Gamma^* \mid \forall x \in L, \alpha.u(x) \not\sqsubseteq v(x)\}, \qquad \text{(left } L\text{-blockers)}$$

$$X'_L \stackrel{\text{def}}{=} \{\alpha \in \Gamma^* \mid \forall x \in L, u(x).\alpha \not\sqsubseteq v(x)\}, \qquad \text{(right } L\text{-blockers)}$$

$$Y_L \stackrel{\text{def}}{=} \{\alpha \in \Gamma^* \mid \forall x \in L, u(x) \not\sqsubseteq \alpha.v(x)\}, \qquad \text{(left } L\text{-coblockers)}$$

$$Y'_L \stackrel{\text{def}}{=} \{\alpha \in \Gamma^* \mid \forall x \in L, u(x) \not\sqsubseteq v(x).\alpha\}. \qquad \text{(right } L\text{-coblockers)}$$

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

A key observation is that, in order to decide whether Sol_L is empty or not, it is simpler to reason about blocker and coblocker sets (see [1, Section 3] for more details on the decision algorithm). Rather than considering what are the solutions, the blocker and coblocker sets provide information on what latitude is allowed/required by the solutions, in particular by the most permissive ones. As a special case, they can tell us whether a given $\mathsf{PEP}^{\mathrm{reg}}$ instance is solvable since

$$Sol_L = \varnothing \text{ iff } \varepsilon \in X_L \text{ iff } \varepsilon \in X'_L \text{ iff } \varepsilon \in Y_L \text{ iff } \varepsilon \in Y'_L. \tag{1}$$

Working with blocker sets rather than solutions sets has two main advantages:

– First, blocker and coblocker sets behave smoothly as a function of the constraint set L. This allows compositional reasoning w.r.t. L. The "Stability Inequations" (see long version of this paper) is the main example, but there are more. For instance, assume L is the product (concatenation) of two languages: $L = L_1.L_2$. Clearly Sol_L contains $Sol_{L_1}.Sol_{L_2}$. However the containment is strict in general, and it is not possible to express Sol_L as a function of Sol_{L_1} and Sol_{L_2}. By contrast, the following holds:

$$X_{L_1.L_2} = \Gamma^* \text{ iff } \left(X'_{L_1} \cup Y_{L_2}\right) \cap \left(Y'_{L_1} \cup X_{L_2}\right) = \Gamma^*. \tag{2}$$

– Second, blocker and coblocker sets are always regular languages, unlike the Sol_L sets [10]. This makes them easier to handle algorithmically, representing them via FSA's or regular expressions. In particular, compositional reasoning as exemplified in Equation (2) can easily be turned into simple and effective algorithms.

Our contribution. In this paper we consider the computability of the blocker and coblocker sets X_R and Y_R for R a regular constraint language. This is a natural question in view of the decision algorithm for $\mathsf{PEP}^{\mathrm{reg}}$, where lower approximations of these sets are enumerated. More importantly, and as we explain in Section 7, it is another step in our attempts at enlarging the class of known decidable problems that combine Post-embedding and regular constraints.

We prove that blocker sets are not computable[1] while, quite unexpectedly, coblocker sets are computable. Concerning blocker sets, and since they cannot be computed, we consider decision problems that are weaker than computability, e.g., whether a blocker set is empty, infinite, whether is it contained in ("safety"), or contains ("cosafety"), a given set. A summary of our results will be found in Fig. 3 (section 3). In addition, we answer a question raised by [10] and prove that the regularity of Post-embedding languages is undecidable.

Comparison with existing work. This work continues our exploration of the Regular Post Embedding Problem. The problem was introduced and proved decidable in [1]. The links between lossy channels and $\mathsf{PEP}^{\mathrm{reg}}$ are clarified in [2] where it is also shown that looking for infinite solutions within an ω-regular constraint set can be reduced to

[1] Here, and in the rest of the paper, we say informally that regular sets like X_L are "*computable*" when we really mean that an index for them can be computed uniformly from an index for L.

looking for finite solutions. In [10] it is shown how to count solutions, and how to check whether a regular property entails Post embedding. That blocker sets are not computable was claimed in [1, Remark 3.8] without any details or proofs (nor comments on the difference between blocker and coblocker sets).

Outline of the paper. Section 2 recalls the necessary definitions and notations, and proves a few useful lemmas on subwords. Section 3 formally introduces the problems we address. Then Section 4 shows how to compute coblocker sets, while Section 5 considers what can be computed on blocker sets. The undecidability results in that section are proved by a reduction from lossy counter machines described in Section 6. Proofs omitted in the main text can be found in the full version of this extended abstract.

2 Notations and definitions

Words and their morphisms. We write $x, y, w, t, \sigma, \rho, \alpha, \beta, \ldots$ for words, i.e., finite sequences of letters such as $a, b, i, j, \ldots$ from alphabets $\Sigma, \Gamma, \ldots$. With $x.y$, or xy, we denote the concatenation of x and y. With ε we denote the empty word. The *length* of x is written $|x|$.

A *morphism* from Σ^* to Γ^* is a map $u : \Sigma^* \to \Gamma^*$ that respects the monoidal structure, i.e., with $u(\varepsilon) = \varepsilon$ and $u(x.y) = u(x).u(y)$. A morphism u is completely defined by its image $u(a)$, $u(b)$, ..., on $\Sigma = \{a, b, \ldots\}$. Most of the time, we shall write $u_a, u_b, \ldots$, and u_x, instead of $u(a), u(b), \ldots$, and $u(x)$.

The mirror image of a word x is denoted $\widetilde{x}$, e.g., $\widetilde{abc} = bca$. The mirror image of a language L is $\widetilde{L} \stackrel{\text{def}}{=} \{\widetilde{x} \mid x \in L\}$. The mirror image of a morphism u, denoted $\widetilde{u}$, is defined by $\widetilde{u}(a) \stackrel{\text{def}}{=} \widetilde{u(a)}$, so that $\widetilde{u}(x) = \widetilde{u(\widetilde{x})}$.

Subword ordering. Given two words x and y, we write $x \sqsubseteq y$ when x is a (scattered) *subword* of y, i.e., when x can be obtained by erasing some letters (possibly none) from y. For example, $abba \sqsubseteq \underline{ab}racad\underline{abr}\underline{a}$. The subword relation is a partial ordering, compatible with the monoidal structure: $\varepsilon \sqsubseteq x$, and $xy \sqsubseteq x'y'$ when $x \sqsubseteq x'$ and $y \sqsubseteq y'$. Higman's Lemma further states that, over a finite alphabet, the subword relation is a well-quasi-ordering, i.e., it is well-founded and all antichains (sets of incomparable words) are finite.

Section 6 relies on the following lemma (see long version of this paper for a proof):

Lemma 2.1 (Elimination Lemma).
If $xw \sqsubseteq y$ and $x' \sqsubseteq wy'$ then $xx' \sqsubseteq yy'$.
If $x \sqsubseteq yw$ and $wx' \sqsubseteq y'$ then $xx' \sqsubseteq yy'$.

Upward-closed and downward-closed languages. A language $L \subseteq \Gamma^*$ is *upward-closed* if $x \in L$ and $x \sqsubseteq y$ imply $y \in L$. It is *downward-closed* if $x \in L$ and $y \sqsubseteq x$ imply $y \in L$ (equivalently, if its complement is upward-closed). Higman's Lemma entails that upward-closed languages and downward-closed languages are regular [11]. In fact, upward-closed languages can be denoted by very simple regular expressions since

they obviously reside at level 1/2 of the Straubing-Thérien Hierarchy [12]. Downward-closed languages too can be denoted by simple regular expressions [13, 14]. In Section 4 we use "$*$-products", defined as concatenations of *atoms* that are either of the form $a+\varepsilon$ for some $a \in \Gamma$, or of the form A^* for some sub-alphabet $A \subseteq \Gamma$. For example, with $\Gamma = \{a,b,c\}$, the set of subwords of *abac* is $(a+\varepsilon).(b+\varepsilon).(a+\varepsilon).(c+\varepsilon)$ and the set of words that do not have *ab* as a subword is $\{b,c\}^*.\{a,c\}^*$. Any downward-closed language is, in a unique way, a finite union of maximal $*$-products.

3 Blockers and coblockers

In the rest of the paper, we consider a generic PEP instance given by some $u,v : \Sigma^* \to \Gamma^*$. Recall that, for a regular constraint set $R \subseteq \Sigma^*$, the (left) blocker and coblocker sets X_R and Y_R are defined by:

$$X_R \stackrel{\text{def}}{=} \{\alpha \in \Gamma^* \mid \forall x \in R, \alpha.u_x \not\sqsubseteq v_x\}, \qquad Y_R \stackrel{\text{def}}{=} \{\alpha \in \Gamma^* \mid \forall x \in R, u_x \not\sqsubseteq \alpha.v_x\}.$$

Observe that X_R is upward-closed and Y_R is downward-closed. Hence both are regular.

Remark 3.1. In the rest of the paper, starting with Def. 3.2 below, we restrict our attention to the *left* sets X_L and Y_L. This is no loss of generality in view of the symmetry between the left-handed and the right-handed notions: α is a right L-blocker (or coblocker) if, and only if, $\widetilde{\alpha}$ is a left $\widetilde{L}$-blocker (resp., coblocker) in the mirror instance $\widetilde{u},\widetilde{v}$. □

For blocker and coblocker sets, we consider questions that range in generality from just checking one α for membership, to computing the whole set.

Definition 3.2 (Decision problems for blocker and coblocker sets). *We consider questions where one is given two morphisms $u,v : \Sigma^* \to \Gamma^*$ and a regular language $R \subseteq \Sigma^*$ as inputs, with possibly some additional input in the form of a word $\alpha \in \Gamma^*$, or a regular "safe" set $S \subseteq \Gamma^*$.*

- Blockers_Membership*: does $\alpha \in X_R$?*
- Blockers_Emptiness*: does $X_R = \varnothing$?*
- Blockers_Universality*: does $X_R = \Gamma^*$?*
- Blockers_Safety*: does $X_R \subseteq S$?*
- Blockers_Cosafety*: does $S \subseteq X_R$?*
- Blockers_Finiteness*: is X_R finite?*
- Blockers_Cofiniteness*: is X_R cofinite?, i.e., is $\Gamma^* \smallsetminus X_R$ finite?*

The same decision problems CoBlockers_Membership, CoBlockers_Safety, ..., *are defined for coblocker sets.*

Finally, Blockers_Computation *and* CoBlockers_Computation *ask one to* compute *a representation of X_R (resp., Y_R) under the form of a regular expression or a FSA. (These are not decision problems).*

Remark 3.3. The restriction to *regular* safe sets S is a natural assumption that is both expressive and tractable. However, in our setting where blocker and coblocker sets are upward-closed (resp., downward-closed), the expressive power is even larger. Indeed, for any L, $X_R \subseteq L$ iff $X_R \subseteq S$ where S is the upward-closure of L. Thus, and since

	Blockers	**Coblockers**
Membership	decidable (Coro. 4.2)	decidable (Coro. 4.2)
Safety	undecidable (Theo. 5.3)	decidable (Theo. 4.3)
Cosafety	decidable (Coro. 4.2)	decidable (Coro. 4.2)
Emptiness	undecidable (Theo. 5.3)	decidable (Theo. 4.3)
Universality	decidable (Coro. 4.2)	trivial
Finiteness	undecidable (Theo. 5.3)	decidable (Theo. 4.3)
Cofiniteness	undecidable (Theo. 5.2)	trivial
Computation	no	yes (Theo. 4.3)

Fig. 1. Computability for blocker and coblocker sets. See Remark 3.5 about complexity.

the upward-closure of L is always regular, our positive results automatically apply to any class of safe sets for which the upward and downward closures can be effectively computed (e.g., context-free languages [15]). □

Remark 3.4 (Relations among problems). Safety is a general problem that subsumes Emptiness and Membership. Cosafety subsumes Universality and (non-)Membership. Blockers_Universality reduces to Blockers_Membership since $X_R = \Gamma^*$ iff $\varepsilon \in X_R$. CoBlockers_Universality is trivial since $Y_R = \Gamma^*$ iff $R = \varnothing$. Finiteness and Cofiniteness are natural counting questions. Finiteness coincides with Emptiness for blocker sets (assuming Γ is not empty) and more generally for all upward-closed sets (Cofiniteness and Universality coincide for downward-closed sets in general, and coblocker sets in particular).

There are no other obvious reductions between the above decision problems (e.g., Finiteness and Cofiniteness are in general unrelated).

Regarding computability of the blocker and coblocker sets, observe that since these sets are regular, the decidability of Safety and Cosafety would entail their computability (see also Section 4). Conversely, all the decision problems listed above can easily be answered from an FSA description of the sets. Hence our decision problems can be seen as different special cases of the general Blockers_Computation and CoBlockers_Computation problems. □

Remark 3.5 (On the complexity of blocker and coblocker problems). All the non-trivial problems listed in Def. 3.2 are more general than $\mathsf{PEP}^{\mathrm{reg}}$. This was made precise in Remark 3.4 except for CoBlockers_Finiteness, but it is easy to provide a reduction from CoBlockers_Emptiness to CoBlockers_Finiteness: add one extra symbol to Γ, ensuring that Y_R is finite iff it is empty. Hence all the above problems are at least as hard as $\mathsf{PEP}^{\mathrm{reg}}$ and none of them is multiply-recursive. □

4 Computing coblocker sets

We start with the computability results. They can be obtained via reductions to $\mathsf{PEP}^{\mathrm{reg}}$:

Lemma 4.1. Blockers_Cosafety *and* CoBlockers_Cosafety *many-one reduce to (the complement of)* $\mathsf{PEP}^{\mathrm{reg}}$.

Proof. Blockers_Cosafety: with u, v, R and S we associate a $\mathsf{PEP}^{\mathrm{reg}}$ instance $u', v' : \Sigma'^* \to \Gamma^*$ and a regular constraint $R' \subseteq \Sigma'^*$. Assume w.l.o.g. that Σ and Γ are disjoint alphabets and let $\Sigma' \stackrel{\text{def}}{=} \Sigma \cup \Gamma$. u' and v' are extensions of u and v with $u'(\gamma) = \gamma$ and $v'(\gamma) = \varepsilon$ for all $\gamma \in \Gamma$. Finally let $R' \stackrel{\text{def}}{=} S.R$, this is indeed a regular subset of Σ'^*.

Now, u', v', R' is a positive $\mathsf{PEP}^{\mathrm{reg}}$ instance iff $u'_x \sqsubseteq v'_x$ for some $x \in R'$, iff $u'_{\alpha y} \sqsubseteq v'_{\alpha y}$ for some $\alpha \in S$ and some $y \in R$, iff $u'_\alpha.u'_y \sqsubseteq v'_\alpha.v'_y$, iff $\alpha.u_y \sqsubseteq v_y$ for some $\alpha \in S$ and y, i.e., iff some $\alpha \in S$ is not in X_R, i.e., $S \not\subseteq X_R$.

CoBlockers_Cosafety: the same idea works provided we let $u'(\gamma) = \varepsilon$ and $v'(\gamma) = \gamma$. □

Since $\mathsf{PEP}^{\mathrm{reg}}$ is decidable, and thanks to Remark 3.4, Lemma 4.1 entails:

Corollary 4.2. *For blocker and coblocker sets,* Cosafety, Universality *and* Membership *are decidable.*

We are now ready to proceed to the main computability result:

Theorem 4.3. *The coblocker sets Y_R and Y'_R are computable.*

Our proof simply leverages the decidability of CoBlockers_Cosafety (Coro. 4.2) with the VJGL Lemma (here specialized to words with embeddings).

Lemma 4.4 (VJGL Lemma, see Theo. 2 of [16]). *Let $(U_i)_i$ be an enumeration of upward-closed languages on some finite alphabet. One can compute a finite representation for the U_i's if, and only if, one can decide whether $U_i \cap P = \varnothing$ for $*$-products P (when i and P are inputs).*

Here, computing "a finite representation" means computing the finite basis, i.e, the set of minimal words, but this can easily be transformed into a regular expression or an FSA representation. The VJGL-Lemma is based on a generic algorithm that, in the case of words with embedding, computes such finite bases using an oracle for non-intersection with $*$-products.

Another wording of the VJGL-Lemma is given by the following corollary.

Corollary 4.5. *1. If $(U_i)_i$ are upward-closed languages with a decidable safety problem, then they are computable.*
2. Equivalently, if $(V_i)_i$ are downward-closed languages with a decidable cosafety problem, then they are computable.

Proof. $U_i \cap P = \varnothing$ is equivalent to $U_i \subseteq (\Sigma^* \setminus P)$, a safety question. □

We now prove Theorem 4.3: The coblocker sets Y_R are downward-closed and have a decidable cosafety problem (Coro. 4.2). Hence they are computable by Coro. 4.5.2. Then Theorem 4.3 accounts for all the positive results on coblocker sets in Fig. 3.

5 Blocker sets are not computable

It is not possible to effectively compute the blocker sets X_R from given u, v, R, even though X_R is known to be regular. This is shown with Lemma 5.1, our main negative result (proved in Section 6):

Lemma 5.1. Blockers_Cofiniteness *is* Σ_1^0*-hard and* Blockers_Emptiness *is* Π_1^0*-hard.*

With Lemma 5.1, we are in a position to prove all the undecidability results in Fig. 3:

Theorem 5.2. Blockers_Cofiniteness *is* Σ_1^0*-complete.*

Proof (Sketch). Membership in Σ_1^0 can be seen by writing the cofiniteness of X_R under the form $\exists n \in \mathbb{N}, \Gamma^{\geq n} \subseteq X_R$ and relying on the decidability of Blockers_Cosafety (Coro. 4.2). □

Theorem 5.3. Blockers_Safety, Blockers_Emptiness *and* Blockers_Finiteness *are* Π_1^0*-complete.*

Proof. The Π_1^0-hardness of Blockers_Emptiness (Lemma 5.1) also applies to Blockers_Finiteness (since the two problems coincide) and Blockers_Safety (a more general problem), see Remark 3.4.

For upper bounds, we observe that Blockers_Safety (hence also Blockers_Emptiness) is in Π_1^0 since it can be written under the form $\forall \alpha \in \Gamma^*, (\alpha \in S \vee \alpha \notin X_R)$ (recall that $\alpha \notin X_R$ is decidable). □

6 Lossy counter machines

Lossy counter machines or, for short, *LCM*'s, were introduced by R. Mayr [17]. They are a variant of Minsky counter machines (with zero-test, increments and decrements) where counters are *lossy*, i.e., they may decrease non-deterministically. We only give a streamlined presentation of LCM's here and refer to [17, 18] for more details.

Let $M = (Q, C, \Delta, q_{\text{init}})$ be a Minsky counter machine with finite set of control states $Q \ni q_{\text{init}}$, finite set of counters C, and finite set of transitions rules Δ. Four counters are sufficient for our purposes so we fix $C = \{c_1, c_2, c_3, c_4\}$. A configuration of M is some $\tau = (q, n_1, n_2, n_3, n_4) \in \mathit{Conf}(M) \stackrel{\text{def}}{=} Q \times \mathbb{N}^4$, with *size*, denoted $|\tau|$, being $n_1 + n_2 + n_3 + n_4$. We (partially) order $\mathit{Conf}(M)$ with

$$(q, n_1, n_2, n_3, n_4) \leq (q', n_1', n_2', n_3', n_4') \stackrel{\text{def}}{\Leftrightarrow} q = q' \wedge n_1 \leq n_1' \wedge \cdots \wedge n_4 \leq n_4'.$$

An initial state $q_{\text{init}} \in Q$ is fixed, and the initial configuration is $\tau_{\text{init}} \stackrel{\text{def}}{=} (q_{\text{init}}, 0, 0, 0, 0)$. Observe that the only way to have $\tau \leq \tau_{\text{init}}$ is with $\tau = \tau_{\text{init}}$.

A transition rule δ is a directed edge between states of M, labeled by an operation $op \in OP \stackrel{\text{def}}{=} C \times \{\texttt{++}, \texttt{--}, \texttt{=0?}\}$, and denoted (q, op, q'). The rules in Δ give rise to two different transition relations between configurations. First, steps $\tau \xrightarrow{\delta} \tau'$ are defined in the expected way. Formally, with $\delta = (q_1, op, q_2)$, there is a step $(q, n_1, n_2, n_3, n_4) \xrightarrow{\delta} (q', n_1', n_2', n_3', n_4')$ if, and only if, the following three conditions are satisfied:
1. $q_1 = q$ and $q_2 = q'$;
2. op is some c_k`++` or c_k`--` or c_k`=0?`, and $n_i' = n_i$ for all $i \neq k$;
3. if op is c_k`++` then $n_k' = n_k + 1$; if op is c_k`--` then $n_k' = n_k - 1$; if op is c_k`=0?` then $0 = n_k = n_k'$.

These so-called *perfect steps* describe the operational semantics of M when its counters are not assumed to be lossy. Then a second operational semantics, with transitions denoted $\tau \xrightarrow{\delta}_{\text{lossy}} \tau'$, is derived[2] in the following way:

$$\tau \xrightarrow{\delta}_{\text{lossy}} \tau' \stackrel{\text{def}}{\Leftrightarrow} \tau \xrightarrow{\delta} \tau'' \text{ for some } \tau'' \geq \tau'. \tag{3}$$

These *lossy steps* describe the behavior of M when its counters are assumed to be lossy. In the usual way, the δ superscript on transitions is omitted when irrelevant. *Lossy runs*, denoted $\tau_0 \xrightarrow{*}_{\text{lossy}} \tau_n$, are sequences of chained lossy steps $\tau_0 \rightarrow_{\text{lossy}} \tau_1 \rightarrow_{\text{lossy}} \cdots \tau_n$. We write $\mathit{Reach}_{\text{lossy}}(M)$ for the set of configurations that can be reached via lossy runs of M, starting from τ_{init}.

We rely on known undecidability results on LCM's and use the following two problems:

LCM_Infinite**:** the question whether $\mathit{Reach}_{\text{lossy}}(M)$ is infinite, for a given LCM M;
LCM_Unbounded_Counter**:** the question whether $\mathit{Reach}_{\text{lossy}}(M)$ contains configurations with arbitrarily large values for the first counter c_1.

These two problems are a variant of one another, and they are easily seen to be interreducible. The following theorem is from [17, 18]:

Theorem 6.1. LCM_Infinite *and* LCM_Unbounded_Counter *are* Π_1^0*-complete.*

6.1 From lossy counters to Post-embedding

With a LCM $M = (Q, C, \Delta, q_{\text{init}})$ we associate a PEP instance $u, v : \Sigma^* \to \Gamma^*$ that will be used in three different reductions (with different constraint languages $R_1, R_2, R_3 \subseteq \Sigma^*$). Here $\Gamma \stackrel{\text{def}}{=} Q \cup C$ is used to encode the configurations of M: a configuration $\tau = (q, n_1, n_2, n_3, n_4)$ is encoded by the word $c_1^{n_1} c_2^{n_2} c_3^{n_3} c_4^{n_4} q$, denoted $\lceil \tau \rceil$. Observe that $\lceil \tau \rceil \sqsubseteq \lceil \tau' \rceil$ iff $\tau \leq \tau'$.

We further let $\Sigma \stackrel{\text{def}}{=} \Gamma \cup \Delta \cup OP \cup \overline{Q} \cup \overline{C}$ where $\overline{Q} = \{\overline{q} \mid q \in Q\}$ and $\overline{C} = \{\overline{c}_1, \overline{c}_2, \overline{c}_3, \overline{c}_4\}$ are copies of Q and C, with new symbols obtained by overlining the original symbols from $Q \cup C$. We define two morphisms $u, v : \Sigma^* \to \Gamma^*$ with

$$u((q, op, q')) \stackrel{\text{def}}{=} q, \qquad v((q, op, q')) \stackrel{\text{def}}{=} q', \qquad u(\overline{c}_i) \stackrel{\text{def}}{=} c_i, \qquad v(\overline{c}_i) \stackrel{\text{def}}{=} c_i,$$
$$u(c_i\texttt{++}) \stackrel{\text{def}}{=} \varepsilon, \qquad v(c_i\texttt{++}) \stackrel{\text{def}}{=} c_i, \qquad u(c_i\texttt{--}) \stackrel{\text{def}}{=} c_i, \qquad v(c_i\texttt{--}) \stackrel{\text{def}}{=} \varepsilon.$$

How u and v evaluate on the rest of Σ will be defined later when it becomes relevant.

With every transition rule $\delta = (q, op, q')$ in Δ, we associate a language $R_\delta \subseteq \Sigma^*$ given via the following regular expressions:

$$R_\delta \stackrel{\text{def}}{=} \begin{cases} \overline{c}_1{}^* \cdots \overline{c}_{k-1}{}^* \cdot op \cdot \overline{c}_k{}^* \cdots \overline{c}_4{}^* \cdot \delta & \text{if } op \text{ is } c_k\texttt{++} \text{ or } c_k\texttt{--}, \\ \overline{c}_1{}^* \cdots \overline{c}_{k-1}{}^* \cdot \overline{c}_{k+1}{}^* \cdots \overline{c}_4{}^* \cdot \delta & \text{if } op \text{ is } c_k\texttt{=0?}. \end{cases}$$

[2] Lossy steps could also be defined *directly* without deriving them from perfect steps, but the indirect definition is very convenient as it permits reasoning simultaneously on both kinds of steps for the same counter machine.

These definitions ensure that, when $x \in R_\delta$, u_x and v_x are the encodings of related configurations. We let the reader check that the following more precise statement holds:

Lemma 6.2.
1. If $x \in R_\delta$, then $u_x = \lceil \tau \rceil$ and $v_x = \lceil \tau' \rceil$ for some configurations τ, τ' such that $\tau \xrightarrow{\delta} \tau'$.
2. Reciprocally, if $\tau \xrightarrow{\delta} \tau'$, then $\lceil \tau \rceil = u_x$ and $\lceil \tau' \rceil = v_x$ for some (unique) $x \in R_\delta$.

We further define $R_\Delta \stackrel{\text{def}}{=} \bigcup_{\delta \in \Delta} R_\delta$ and $R_M \stackrel{\text{def}}{=} (R_\Delta)^*$: these languages are regular.

Lemma 6.3. *Let $\alpha \in \Gamma^*$. If $u_x.\alpha \sqsubseteq \lceil \tau_{init} \rceil.v_x$ for some $x \in R_M$, then $\alpha \sqsubseteq \lceil \tau \rceil$ for some $\tau \in Reach_{lossy}(M)$.*

Proof. We assume $\alpha \neq \varepsilon$ and $x \neq \varepsilon$, otherwise $\alpha \sqsubseteq \lceil \tau_{\text{init}} \rceil$ trivially. Thus $x \in R_M$ must be of the form $x = x_1 \dots x_n$ with $n > 0$ and $x_i \in R_\Delta$ for all $i = 1, \dots, n$. By Lemma 6.2, u_x is some $\lceil \tau_0 \rceil . \lceil \tau_1 \rceil \dots \lceil \tau_{n-1} \rceil$ and v_x is some $\lceil \tau'_1 \rceil . \lceil \tau'_2 \rceil \dots \lceil \tau'_n \rceil$ such that, for all $i = 1, \dots, n$, $\tau_{i-1} \to \tau'_i$ is a perfect step of M.

We now use the assumption that $u_x.\alpha \sqsubseteq \lceil \tau_{\text{init}} \rceil.v_x$. Since $\alpha \neq \varepsilon$, u_x embeds into a strict prefix, denoted w, of $\lceil \tau_{\text{init}} \rceil.v_x$. Note that u_x contains $n > 0$ symbols from Q and ends with one of them, while w has at most n (it is shorter than $\lceil \tau_{\text{init}} \rceil.v_x$ that has $n+1$ symbols from Q and ends with one of them). Hence w necessarily has n symbols from Q and $u_x.\alpha \sqsubseteq \lceil \tau_{\text{init}} \rceil.v_x$ can be decomposed as $\lceil \tau_i \rceil \sqsubseteq \lceil \tau'_i \rceil$ (i.e., $\tau_i \leq \tau'_i$) for all $i = 1, \dots, n-1$, with also $\lceil \tau_0 \rceil \sqsubseteq \lceil \tau_{\text{init}} \rceil$ (hence $\tau_0 = \tau_{\text{init}}$) and $\alpha \sqsubseteq \lceil \tau'_n \rceil$. Combining with $\tau_{i-1} \to \tau'_i$ we deduce $\tau_{i-1} \to_{\text{lossy}} \tau_i$ for $i = 1, \dots, n-1$. Finally $\tau_{\text{init}} = \tau_0 \to_{\text{lossy}} \tau_1 \cdots \to_{\text{lossy}} \tau_{n-1} \to \tau'_n$ is a lossy run of M, so that $\tau'_n \in Reach_{\text{lossy}}(M)$. □

There is a converse to Lemma 6.3:

Lemma 6.4. *If $\tau \in Reach_{lossy}(M)$, there exists some $x \in R_M$ such that $u_x.\lceil \tau \rceil \sqsubseteq \lceil \tau_{init} \rceil.v_x$.*

Proof. Since $\tau \in Reach_{\text{lossy}}(M)$ there exists a lossy run $\tau_{\text{init}} = \tau_0 \to_{\text{lossy}} \tau_1 \to_{\text{lossy}} \cdots \tau_n = \tau$. We show, by induction on $i = 0, 1, \dots, n$, that $u_{x_i}.\lceil \tau_i \rceil \sqsubseteq \lceil \tau_{\text{init}} \rceil.v_{x_i}$ for some $x_i \in R_M$.

The base case, $i = 0$, is dealt with $x_0 = \varepsilon$ since $\tau_0 = \tau_{\text{init}}$.

For the case $i > 0$, we know by ind. hyp. that there is some $x_{i-1} \in R_M$ with

$$u_{x_{i-1}}.\lceil \tau_{i-1} \rceil \sqsubseteq \lceil \tau_{\text{init}} \rceil.v_{x_{i-1}}. \tag{4}$$

The lossy step $\tau_{i-1} \to_{\text{lossy}} \tau_i$ implies the existence of a perfect step $\tau_{i-1} \to \tau'$ with $\tau' \geq \tau_i$ (Equation (3)). Thus $\lceil \tau_{i-1} \rceil = u_y$ and $\lceil \tau' \rceil = v_y$ for some $y \in R_\Delta$ (Lemma 6.2).

From $\tau_i \leq \tau'$, we deduce

$$u_y.\lceil \tau_i \rceil \sqsubseteq \lceil \tau_{i-1} \rceil.v_y. \tag{5}$$

We now put together Equations (4) and (5). The Elimination Lemma yields

$$u_{x_{i-1}}.u_y.\lceil \tau_i \rceil \sqsubseteq \lceil \tau_{\text{init}} \rceil.v_{x_{i-1}}.v_y, \tag{6}$$

so that setting $x_i \stackrel{\text{def}}{=} x_{i-1}.y$ concludes our proof. We observe that $x_i \in R_M$ since $x_{i-1} \in R_M$ and $y \in R_\Delta$. □

6.2 Reducing LCM_Infinite and LCM_Unbounded_Counter to blockers problems

For the next step in the reduction, we extend u and v on $Q \cup C$ ($= \Gamma$) with

$$u(\gamma) \stackrel{\text{def}}{=} \pi_1(\gamma) = \begin{cases} c_1 & \text{if } \gamma = c_1, \\ \varepsilon & \text{if } \gamma \in \Gamma \setminus \{c_1\}, \end{cases} \qquad v(\gamma) \stackrel{\text{def}}{=} \gamma \text{ for all } \gamma \in \Gamma.$$

When $\alpha \in \Gamma^*$, we shall write $\pi_1(\alpha)$ rather than u_α to emphasize the fact that u only retains the c_1 symbols of α and erases the rest. Below, we rely on a few obvious properties of this erasing morphism, such as $\pi_1(\alpha) \sqsubseteq \alpha$, or $\pi_1(\alpha\beta) = \pi_1(\beta\alpha)$, and in particular the following:

Fact 6.5 *For all $\beta \in \Gamma^*$ and $x, y \in \Sigma^*$, $x.c_1.\pi_1(\beta) \sqsubseteq y.\beta$ implies $x.c_1 \sqsubseteq y$.*

Finally, we let $R_1 \stackrel{\text{def}}{=} q_{\text{init}}.R_M$ and $R_2 \stackrel{\text{def}}{=} R_1.\Gamma^*$. This provides two different reductions, with properties captured by Lemmas 6.6 and 6.8.

Lemma 6.6. *Let $\alpha \in \Gamma^*$. The following are equivalent:*
(1) $\alpha \notin X'_{R_1}$,
(2) there exists $x \in R_1$ such that $u_x.\alpha \sqsubseteq v_x$,
(3) there exists $\tau \in Reach_{lossy}(M)$ such that $\alpha \sqsubseteq \lceil \tau \rceil$.

Proof (Sketch). (1) $\Leftrightarrow$ (2) by definition of X'_{R_1}. Then, given the definitions of R_1, u and v, Lemma 6.3 shows "(2) $\Rightarrow$ (3)" (note that $u(q_{\text{init}}) = \varepsilon$ and $v(q_{\text{init}}) = q_{\text{init}} = \lceil \tau_{\text{init}} \rceil$). Finally, Lemma 6.4 shows "(3) $\Rightarrow$ (2)". □

In particular, X'_{R_1} is cofinite iff M does not satisfy LCM_Infinite.

Corollary 6.7. Blockers_Cofiniteness *is Σ_1^0-hard.*

Lemma 6.8. *Let $\alpha \in \Gamma^*$. The following are equivalent:*
(1) $\alpha \notin X'_{R_2}$,
(2) there exists $y \in R_2$ such that $u_y.\alpha \sqsubseteq v_y$,
(3) there exists $\tau \in Reach_{lossy}(M)$ such that $\pi_1(\alpha) \sqsubseteq \pi_1(\lceil \tau \rceil)$.

Proof. (1) $\Leftrightarrow$ (2) by definition of X'_{R_2}.
(3) $\Rightarrow$ (2): Assume $\pi_1(\alpha) \sqsubseteq \pi_1(\lceil \tau \rceil)$ for some $\tau \in Reach_{\text{lossy}}(M)$. Then, $\pi_1(\alpha) \sqsubseteq \lceil \tau \rceil$ so that, by Lemma 6.6, there exists some $x \in R_1$ with $u_x.\pi_1(\alpha) \sqsubseteq v_x$. Appending α to the right yields $u_x.\pi_1(\alpha).\alpha = u_x.u_\alpha.\alpha \sqsubseteq v_x.\alpha = v_x.v_\alpha$. Letting $y \stackrel{\text{def}}{=} x.\alpha$ ($\in R_2$) proves (2).
(2) $\Rightarrow$ (3): Assume $u_y.\alpha \sqsubseteq v_y$ for some $y \in R_2$ of the form $x.\beta$ with $x \in R_1$ and $\beta \in \Gamma^*$. We assume $\pi_1(\alpha) \neq \varepsilon$ since otherwise $\pi_1(\alpha) \sqsubseteq \pi_1(\lceil \tau_{\text{init}} \rceil)$ holds trivially. From $u_y.\alpha \sqsubseteq v_y$, we deduce

$$u_x.\pi_1(\alpha).\pi_1(\beta) = u_x.\pi_1(\beta).\pi_1(\alpha) = u_y.\pi_1(\alpha) \sqsubseteq u_y.\alpha \sqsubseteq v_y = v_x.v_\beta = v_x.\beta.$$

From $u_x.\pi_1(\alpha).\pi_1(\beta) \sqsubseteq v_x.\beta$, one deduces $u_x.\pi_1(\alpha) \sqsubseteq v_x$ (using Fact 6.5 and the assumption that $\pi_1(\alpha) \neq \varepsilon$). Thus there exists a $\tau \in Reach_{\text{lossy}}(M)$ with $\pi_1(\alpha) \sqsubseteq \lceil \tau \rceil$ (Lemma 6.3), hence $\pi_1(\alpha) \sqsubseteq \pi_1(\lceil \tau \rceil)$. □

In other words, $\alpha \notin X'_{R_2}$ iff there is a reachable configuration where the c_1 counter is larger than, or equal to, the number of c_1 symbols in α. Thus $X'_{R_2} = \varnothing$ iff M satisfies LCM_Unbounded_Counter.

Corollary 6.9. Blockers_Emptiness *is* Π^0_1*-hard.*

As an aside, the reduction from LCM's can be used to prove Theo. 6.11 below. The regularity problem for Post-embedding languages is a natural question since Sol_R is not always regular, and since comparisons with a regular S are possible:

Theorem 6.10 ([10]). *The questions, for $S \subseteq \Sigma^*$ a regular language, whether $S \subseteq Sol_R$, and whether $Sol_R \subseteq S$, are decidable.*

Theorem 6.11. *The question whether, for $u, v : \Sigma^* \to \Gamma^*$ and a regular $R \subseteq \Sigma^*$, Sol_R is a regular language, is Σ^0_1-complete.*

The proof for Σ^0_1-hardness simply adapts our previous reduction, providing u, v and R such that Sol_R is regular iff $Reach_{\text{lossy}}(M)$ is finite, then relying on Theo. 6.1.

7 Concluding remarks

The decidability of $\mathsf{PEP}^{\text{reg}}$ is the decidability of existential questions of the form

$$\exists x \in R : u(x) \sqsubseteq v(x) \tag{Q1}$$

for regular R's. This result is fragile and does not extend easily. When one looks for solutions satisfying more expressive constraints, e.g., deterministic context-free, or also Presburger-definable, the problem becomes undecidable [1]. In another direction, combining two embeddings quickly raises undecidable questions, e.g., the following questions are undecidable [10, Theo. 4.1]:

$$\exists x \in \Sigma^+ : (u_1(x) \sqsubseteq v_1(x) \wedge u_2(x) \sqsubseteq v_2(x)), \tag{Q2}$$

$$\exists x \in \Sigma^+ : (u_1(x) \sqsubseteq v_1(x) \wedge u_2(x) \not\sqsubseteq v_2(x)). \tag{Q3}$$

Remark that, by Theorem 6.10, the following universal question is decidable [10]:

$$\forall x \in R : u(x) \sqsubseteq v(x). \tag{Q4}$$

This suggests considering questions like

$$\forall x \in R\, \exists x' \in R' : u(xx') \sqsubseteq v(xx'), \tag{Q5}$$

$$\exists x \in R\, \forall x' \in R' : u(xx') \sqsubseteq v(xx'). \tag{Q6}$$

The undecidability of (Q5) is clear since already Blockers_Emptiness is undecidable. The (un?)decidability of (Q6) is still open. We believe blockers and coblockers may play a useful role here. Indeed, by analogy with blockers, we may define

$$A_R \stackrel{\text{def}}{=} \{\alpha \mid \forall x \in R, \alpha.u(x) \sqsubseteq v(x)\}, \qquad B_R \stackrel{\text{def}}{=} \{\beta \mid \forall x \in R, u(x) \sqsubseteq \beta.v(x)\}.$$

Note that membership in A_R (or in B_R), being an instance of (Q5), is decidable. Furthermore, B_R is upward-closed and A_R is finite (unless R is empty). Now, the following observation:

$$\big(\exists x \in R\, \forall x' \in R' : u(xx') \sqsubseteq v(xx')\big) \quad \text{iff} \quad \big((A_{R'} \smallsetminus Y_R) \cup (B_{R'} \smallsetminus X_R) \neq \varnothing\big)$$

provides a direct link between (Q6) and blocker-like languages. We leave this as a suggestion for future investigations.

References

1. P. Chambart and Ph. Schnoebelen. Post embedding problem is not primitive recursive, with applications to channel systems. In *Proc. FST&TCS 2007*, volume 4855 of *Lecture Notes in Computer Science*, pages 265–276. Springer, 2007.
2. P. Chambart and Ph. Schnoebelen. The ω-Regular Post Embedding Problem. In *Proc. FOSSACS 2008*, volume 4962 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2008.
3. P. Chambart and Ph. Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *Proc. LICS 2008*, pages 205–216. IEEE Comp. Soc. Press, 2008.
4. J. Ouaknine and J. Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Comp. Science*, 3(1):1–27, 2007.
5. P. A. Abdulla, J. Deneux, J. Ouaknine, K. Quaas, and J. Worrell. Universality analysis for one-clock timed automata. *Fundamenta Informaticae*, 89(4):419–450, 2008.
6. S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. Computational Logic*, 9(2), 2008.
7. T. Jurdziński. Leftist grammars are nonprimitive recursive. In *Proc. ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 51–62. Springer, 2008.
8. P. Chambart and Ph. Schnoebelen. Toward a compositional theory of leftist grammars and transformations. In *Proc. FOSSACS 2010*, volume 6014 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2010.
9. D. Gabelaia et al. Non-primitive recursive decidability of products of modal logics with expanding domains. *Annals of Pure and Applied Logic*, 142(1–3):245–268, 2006.
10. P. Chambart and Ph. Schnoebelen. Pumping and counting on the Regular Post Embedding Problem. In *Proc. ICALP 2010*, Lecture Notes in Computer Science. Springer, 2010.
11. L. H. Haines. On free monoids partially ordered by embedding. *J. Combinatorial Theory*, 76:94–98, 1969.
12. J.-E. Pin and P. Weil. Polynomial closure and unambiguous product. *Theory of Computing Systems*, 30(4):383–422, 1997.
13. A. Finkel and J. Goubault-Larrecq. Forward analysis for WSTS, part I: Completions. In *Proc. STACS 2009*, Leibniz International Proceedings in Informatics, pages 433–444, 2009.
14. P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004.
15. J. van Leeuwen. Effective constructions in well-partially-ordered free monoids. *Discrete Mathematics*, 21(3):237–252, 1978.
16. J. Goubault-Larrecq. On a generalization of a result by Valk and Jantzen. Research Report LSV-09-09, Laboratoire Spécification et Vérification, ENS Cachan, France, May 2009.
17. R. Mayr. Undecidable problems in unreliable computations. *Theoretical Computer Science*, 297(1–3):337–354, 2003.
18. Ph. Schnoebelen. Lossy counter machines: A survey. In *Proc. RP 2010*, Lecture Notes in Computer Science. Springer, 2010.