Post Embedding Problem Is Not Primitive Recursive, With Applications To Channel Systems*

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. We introduce PEP, the Post Embedding Problem, a variant of PCP where one compares strings with the subword relation, and $\mathsf{PEP}^{\mathrm{reg}}$, a further variant where solutions are constrained and must belong to a given regular language. $\mathsf{PEP}^{\mathrm{reg}}$ is decidable but not primitive recursive. This entails the decidability of reachability for unidirectional systems with one reliable and one lossy channel.

Keywords: Post correspondence problem; Lossy channel systems; Higman's Lemma.

1 Introduction

Post correspondence problem, or shortly PCP, can be stated as the question whether two morphisms $u,v:\Sigma^*\to\Gamma^*$ agree non-trivially on some input, i.e., whether $u(\sigma)=v(\sigma)$ for some non-empty $\sigma\in\Sigma^+$. This undecidable problem plays a central role in computer science because it is very often easier and more natural to prove undecidability by reduction from PCP than from, say, the halting problem for Turing machines.

In this paper we introduce PEP, a variant of PCP where one asks whether $u(\sigma)$ is a *subword* of $v(\sigma)$ for some σ. The subword relation, also called embedding, is denoted "$\sqsubseteq$": $w\sqsubseteq w'\overset{\text{def}}{\Leftrightarrow} w$ can be obtained from w' by erasing some letters, possibly all of them, possibly none. We also introduce $\mathsf{PEP}^{\mathrm{reg}}$, an extension of PEP where one adds the requirement that a solution σ belongs to a regular language $R\subseteq\Sigma^*$.

As far as we know, PEP and $\mathsf{PEP}^{\mathrm{reg}}$ have never been considered in the literature [13, 9]. This is probably because PEP is trivial (Prop. 3.1). However, and quite surprisingly, adding a regular constraint makes the problem considerably harder. In this paper we show that $\mathsf{PEP}^{\mathrm{reg}}$ is decidable but that it is not primitive recursive.

Channel systems. What led us to consider $\mathsf{PEP}^{\mathrm{reg}}$ are verification problems for channel systems, i.e., systems of finite-state machines that communicate asynchronously via unbounded FIFO channels. These systems are Turing-powerful in general but several restricted families or variants have decidable verification problems. For example *lossy* channel systems, where messages can be lost nondeterministically, have decidable reachability and termination problems [7, 3, 15]. For systems with one reliable channel (no message losses), reachability is easily decidable if the system is *unidirectional*: one

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

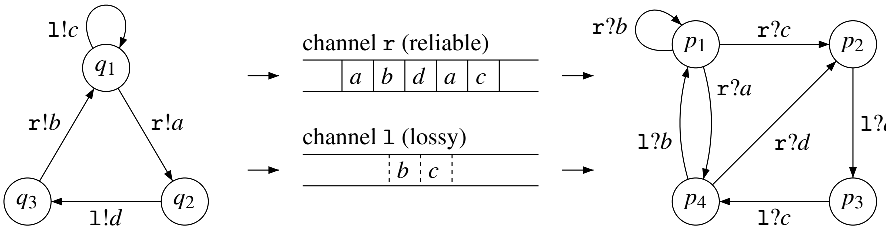

Fig. 1. A unidirectional channel system with one *r*eliable and one *l*ossy channel

sender sends messages to a receiver via the reliable channel, but no communication is possible in the other direction. With two (reliable) unidirectional channels between the sender and the receiver, reachability is undecidable. The open question that motivated our study is ReachUcs, i.e., reachability for channel systems with *unidirectional communication through one reliable and one unreliable channels*, as illustrated in Figure 1.

It is easy to reduce PEP and PEP^{reg} to ReachUcs. It turns out that reductions from ReachUcs to PEP^{reg} also exist. More surprisingly, we are able to reduce PEP^{reg} to ReachLcs, the reachability problem for (classical) lossy channel systems, and to reduce ReachLcs to ReachUcs. Finally, all three problems are equivalent.

Summary of our contributions. 1. We introduce PEP^{reg}, a new decidable variant of the PCP problem that is based on the subword relation. A surprising fact is that the regularity constraint makes PEP^{reg} very different from PEP, and highly non-trivial.
2. We prove that PEP^{reg} is equivalent to (i.e., inter-reducible with) ReachUcs and ReachLcs, two verification problems for systems of communicating automata. This provides the decidability of ReachUcs (and a new decidability proof for ReachLcs).
3. This shows that PEP^{reg} is not primitive recursive (since ReachLcs is not either [15]).

This last point is quite interesting. In recent years, several problems coming from various areas have been shown to be not primitive recursive by reductions from ReachLcs: see, e.g., [2, 4, 6, 8, 10–12]. This is a clear indication that ReachLcs and equivalent problems occupy a specific niche that had not been identified previously. Discovering a simple and natural problem like PEP^{reg} amid this class will help extend the range of problems that can be connected to the class: PEP^{reg} can be used to simplify existing reduction proofs, and make some future proofs easier to obtain.

Outline of the paper. Section 2 recalls the necessary definitions and notations. We prove that PEP^{reg} is decidable in Section 3 and explore variants and extensions in Section 4. The reductions between PEP^{reg} and ReachLcs or ReachUcs are given in sections 5 and 6. Proofs omitted for lack of space can be found in the long version of this paper [5].

2 Notations and definitions

Words. We write $u, v, w, t, \sigma, \rho, \alpha, \beta, \ldots$ for words, i.e., finite sequences of letters such as $a, b, i, j, \ldots$ from alphabets $\Sigma, \Gamma, \ldots$, and denote with $u.v$, or uv, the concatenation of u and v. The *length* of u is written $|u|$. A *morphism* from Σ^* to Γ^* is a map $h : \Sigma^* \to \Gamma^*$ that respects the monoidal structure, i.e., with $h(\varepsilon) = \varepsilon$ and $h(\sigma.\rho) = h(\sigma).h(\rho)$. A morphism h is completely defined by its image $h(1)$, $h(2)$, ..., on $\Sigma = \{1,2,\ldots\}$. We often simply write $h_1, h_2, \ldots$, and h_σ, instead of $h(1), h(2), \ldots$, and $h(\sigma)$.

Quotients. Let L be a language and m a word: $m\backslash L \stackrel{\text{def}}{=} \{w | m.w \in L\}$ is the *(right) quotient* of L by m. When $L \subseteq \Sigma^*$, we write $\mathcal{L}(L)$ for the set $\{m\backslash L \mid m \in \Sigma^*\}$ of all quotients of L. It is well-known that if R is a regular language, then $\mathcal{L}(R)$ is finite and only contains regular languages (that still have their quotients in $\mathcal{L}(R)$). $\mathcal{L}(R)$ can be built effectively from a canonical DFA for R just by varying the initial state.

Embeddings. Given two words $u = a_1 \ldots a_n$ and $v = b_1 \ldots b_m$, we write $u \sqsubseteq v$ when u is a *subword* of v, i.e., when u can be obtained by erasing some letters (possibly none) from v. For example, *abba* $\sqsubseteq$ *abracadabra*. Equivalently, $u \sqsubseteq v$ when u can be embedded in v, i.e., when there exists an order-preserving injective map $h : \{1,\ldots,n\} \to \{1,\ldots,m\}$ such that $a_i = b_{h(i)}$ for all $i = 1,\ldots,n$. It is well-known that the subword relation is a partial ordering on words, and it is a well-quasi-ordering (Higman's Lemma) when we consider words over a fixed finite alphabet. This means that any set of words has a finite number of minimal elements (minimal w.r.t. $\sqsubseteq$).

Upward-closure. A language $L \subseteq \Gamma^*$ is *upward-closed* if $u \in L$ and $u \sqsubseteq v$ imply $v \in L$. It is *downward-closed* if its complement is upward-closed. Higman's Lemma entails that upward-closed languages (hence also downward-closed languages) are regular.

Splitting words. When $u \sqsubseteq v$, we write $v[u]$ for the longest v_1 such that v is some $v_0.v_1$ with $u \sqsubseteq v_0$. Hence $v[u]$ is the longest suffix of v that can be retained if one has to remove some prefix containing u. Dually, for any u and v, we write $u\{v\}$ for the shortest u_1, such that u can be written as some $u_0.u_1$ with $u_0 \sqsubseteq v$. Hence $u\{v\}$ is the shortest suffix of u that can be obtained if one may only remove prefixes that are contained in v. Observe that $u\{v\}$ is always defined while $v[u]$ is only defined when $u \sqsubseteq v$.

When reasoning about embedding and concatenation, a natural and simple tool is the following.

Lemma 2.1 (Simple Decomposition Lemma). *If $u.w \sqsubseteq v.t$ then either $u \sqsubseteq v$ or $w \sqsubseteq t$.*

However, Lemma 2.1 only works one way. For deeper analyses, we shall need the following more powerful tool.

Lemma 2.2 (Complete Decomposition Lemma).

$$u.w \sqsubseteq v.t \ \textit{ if and only if } \begin{cases} \quad u \sqsubseteq v \textit{ and } w \sqsubseteq v[u].t \\ \textit{or } \ u \not\sqsubseteq v \textit{ and } u\{v\}.w \sqsubseteq t. \end{cases}$$

3 PEP: Post correspondence with embedding

The problem we are considering is a variant of Post correspondence problem where equality is replaced by embedding, and where an additional regular constraint is imposed over the solution.

Problem PEP$^{\text{reg}}$

Instance: Two finite alphabets Σ and Γ, two morphisms $u, v : \Sigma^* \to \Gamma^*$, and a regular language $R \subseteq \Sigma^*$.

Answer: Yes if and only if there exists a $\sigma \in R$ such that $u_\sigma \sqsubseteq v_\sigma$.

In the above definition, the regular constraint applies to σ but this is inessential and our results still hold when the constraint applies to u_σ, or v_σ, or both (see Section 4).

For complexity issues, we assume that the constraint R in a PEP$^{\text{reg}}$ instance is given as a nondeterministic finite-state automaton (NFA) $\mathcal{A}_R$. By a *reduction* between two decision problems, we mean a logspace many-one reduction. We say two problems are *equivalent* when they are inter-reducible.

PEP is the special case of PEP$^{\text{reg}}$ where R is Σ^+, i.e., where there are no constraints over the form of a non-trivial solution. As far as we know, PEP and PEP$^{\text{reg}}$ have never been considered in the literature and this is probably because PEP is trivial:

Proposition 3.1. *There is a $\sigma \in \Sigma^+$ such that $u_\sigma \sqsubseteq v_\sigma$ if and only if there is some $i \in \Sigma$ such that $u_i \sqsubseteq v_i$.*

This is a direct corollary of Lemma 2.1. A consequence is that PEP is decidable in deterministic logarithmic space.

Surprisingly, adding a regularity constraint makes the problem much harder, as will be proved later. As of now, we focus on proving the following main result.

Theorem 3.2 (Main Result). PEP$^{\text{reg}}$ *is decidable.*

In the rest of this section, we assume a given PEP$^{\text{reg}}$ instance made of $u, v : \Sigma^* \to \Gamma^*$ and $R \subseteq \Sigma^*$. We consider some $\mathcal{L}(R)$-indexed families of languages in Γ^*:

Definition 3.3 (Blocking family). *An $\mathcal{L}(R)$-indexed family $(A_L, B_L)_{L \in \mathcal{L}(R)}$ of languages in Γ^* is a* blocking family *if for all $L \in \mathcal{L}(R)$:*

$$\sigma \in L \text{ and } \alpha \in A_L \text{ imply } \alpha u_\sigma \not\sqsubseteq v_\sigma, \tag{B1}$$

$$\sigma \in L \text{ and } \beta \in B_L \text{ imply } u_\sigma \not\sqsubseteq \beta v_\sigma. \tag{B2}$$

The terminology "blocking" comes from the fact that the α prefix "blocks" solutions in L to $\alpha.u_\sigma \sqsubseteq v_\sigma$. For B_L, the situation is dual: adding $\beta \in B_L$ is not enough to allow solutions in L to $u_\sigma \sqsubseteq \beta.v_\sigma$.

There is a largest blocking family, called the *blocker* languages, or blocker family, $(X_L, Y_L)_{L \in \mathcal{L}(R)}$, given by:

$$X_L \stackrel{\text{def}}{=} \{\alpha \in \Gamma^* \mid \alpha u_\sigma \not\sqsubseteq v_\sigma \text{ for all } \sigma \in L\}, \tag{B3}$$

$$Y_L \stackrel{\text{def}}{=} \{\beta \in \Gamma^* \mid u_\sigma \not\sqsubseteq \beta v_\sigma \text{ for all } \sigma \in L\}. \tag{B4}$$

A blocking family provides information about the absence of solutions to several variants of our PEP$^{\text{reg}}$ instance. For example, the u, v, R instance itself is positive iff $\varepsilon \notin X_R$ iff $\varepsilon \notin Y_R$.

For proving that a given family is blocking, we use a criterion called "stability".

Definition 3.4 (Stable family). *An $\mathcal{L}(R)$-indexed family $(A_L, B_L)_{L\in\mathcal{L}(R)}$ of languages is* stable *iff, for all $L \in \mathcal{L}(R)$:*

1. *$A_L \subseteq \Gamma^*$ is upward-closed and $B_L \subseteq \Gamma^*$ is downward-closed,*
2. *if $\varepsilon \in L$, then $\varepsilon \notin A_L \cup B_L$,*
3. *for all $i \in \Sigma$ and $\alpha \in A_L$:*
   (a) if $\alpha.u_i \sqsubseteq v_i$ then $v_i[\alpha.u_i] \in B_{i\setminus L}$,
   (b) if $\alpha.u_i \not\sqsubseteq v_i$ then $(\alpha.u_i)\{v_i\} \in A_{i\setminus L}$,
4. *for all $i \in \Sigma$ and $\beta \in B_L$:*
   (a) if $u_i \sqsubseteq \beta.v_i$ then $(\beta.v_i)[u_i] \in B_{i\setminus L}$,
   (b) if $u_i \not\sqsubseteq \beta.v_i$ then $u_i\{\beta.v_i\} \in A_{i\setminus L}$.

Recall that A_L and B_L, being respectively upward- and downward-closed, must be regular languages. Observe also that $\varepsilon \in B_L$ iff $B_L \neq \varnothing$, while $\varepsilon \in A_L$ iff $A_L = \Gamma^*$.

Proposition 3.5 (Soundness). *A stable family is a blocking family.*

Proof. Assume that $(A_L, B_L)_{L\in\mathcal{L}(R)}$ is stable. We prove that it satisfies (B1) and (B2) by induction on the length of σ.

Base case: $\sigma = \varepsilon$. Hence $u_\sigma = v_\sigma = \varepsilon$. Assuming $\alpha u_\sigma \sqsubseteq v_\sigma$ requires $\alpha = \varepsilon$ but if $\sigma \in L$, stability implies that $\varepsilon \notin A_L$. $\sigma \in L$ also implies that B_L is empty so that $u_\sigma \not\sqsubseteq \beta v_\sigma$ is vacuously true.

Inductive case: assume that σ is some $i.\rho$ with $i \in \Sigma$ and $\rho \in \Sigma^*$. Recall that $\sigma \in L$ iff $\rho \in i\setminus L$.
Let $\alpha \in A_L$. If $\alpha u_i \sqsubseteq v_i$, then $v_i[\alpha u_i] \in B_{i\setminus L}$ by stability. Hence $u_\rho \not\sqsubseteq (v_i[\alpha u_i])v_\rho$ by ind. hyp. Then $\alpha u_\sigma = \alpha u_i u_\rho \not\sqsubseteq v_i v_\rho = v_\sigma$ by Lemma 2.2. If, on the other hand, $\alpha u_i \not\sqsubseteq v_i$, then $(\alpha u_i)\{v_i\} \in A_{i\setminus L}$ by stability, hence $(\alpha u_i)\{v_i\}u_\rho \not\sqsubseteq v_\rho$ by ind. hyp., entailing $\alpha u_\sigma \not\sqsubseteq v_\sigma$ by Lemma 2.2.
For $\beta \in B_L$ the reasoning is similar. If $u_i \sqsubseteq \beta v_i$, then $(\beta v_i)[u_i] \in B_{i\setminus L}$ by stability, hence $u_\rho \not\sqsubseteq (\beta v_i)[u_i]v_\rho$ by ind. hyp., hence $u_\sigma = u_i u_\rho \not\sqsubseteq \beta v_i v_\rho = \beta v_\sigma$ by Lemma 2.2. If, on the other hand, $u_i \not\sqsubseteq \beta v_i$, then $u_i\{\beta v_i\} \in A_{i\setminus L}$ by stability, hence $u_i\{\beta v_i\}u_\rho \not\sqsubseteq v_\rho$ by ind. hyp., hence $u_\sigma \not\sqsubseteq \beta v_\sigma$. ☐

The criterion is also sufficient:

Proposition 3.6 (Completeness). *The blocker family $(X_L, Y_L)_{L\in\mathcal{L}(R)}$ is stable.*

Proof. Clearly, as defined by (B3) and (B4) and for any $L \in \mathcal{L}(R)$, X_L is upward-closed and Y_L is downward-closed. Similarly, $\varepsilon \notin X_L$ and $\varepsilon \notin Y_L$ when $\varepsilon \in L$.

It remains to check conditions *3* and *4* for stability. We consider four cases:

3a Assume that $\alpha u_i \sqsubseteq v_i$ for some i in Σ and some α in some X_L. If, by way of contradiction, we assume that $v_i[\alpha.u_i] \notin Y_{i\setminus L}$ then, by (B4), there is some $\rho \in i\setminus L$ such that $u_\rho \sqsubseteq v_i[\alpha.u_i]v_\rho$. Thus $\alpha u_i u_\rho \sqsubseteq v_i v_\rho$ by Lemma 2.2, i.e., $\alpha u_\sigma \sqsubseteq v_\sigma$ writing σ for $i.\rho$. But, since $\sigma \in L$, this contradicts $\alpha \in X_L$.

4a A similar reasoning applies if we assume that $u_i \sqsubseteq \beta v_i$ for some i in Σ and some β in some Y_L while $(\beta v_i)[u_i] \notin Y_{i\setminus L}$: we derive from (B4) that $u_\rho \sqsubseteq (\beta v_i)[u_i]v_\rho$ for some $\rho \in i\setminus L$. Hence $u_i u_\rho \sqsubseteq \beta v_i v_\rho$ by Lemma 2.2, a contradiction since $i.\rho \in L$.

3b If we assume that $\alpha u_i \not\sqsubseteq v_i$ for $\alpha \in X_L$ and $(\alpha u_i)\{v_i\} \notin X_{i\backslash L}$ then, by (B3), there is some $\rho \in i\backslash L$ s.t. $(\alpha u_i)\{v_i\}u_\rho \sqsubseteq v_\rho$. Then $\alpha u_i u_\rho \sqsubseteq v_i v_\rho$ by Lemma 2.2, a contradiction since $i.\rho \in L$.

4b Similarly, assuming that $u_i \not\sqsubseteq \beta v_i$ while $u_i\{\beta v_i\} \notin A_{i\backslash L}$, we derive $(u_i\{\beta v_i\})u_\rho \sqsubseteq v_i v_\rho$, i.e., $u_i u_\rho \sqsubseteq \beta v_i v_\rho$, another contradiction. □

Proposition 3.7 (Stability is decidable). *It is decidable whether an $\mathcal{L}(R)$-indexed family $(A_L, B_L)_{L\in\mathcal{L}(R)}$ of regular languages is a stable family.*

Proof. We can assume that the A_L and B_L are given by DFA's. Conditions *1* and *2* of stability are easy to check.

For a given $i \in \Sigma$ and $L \in \mathcal{L}(R)$, checking condition *3a* needs only consider α's that are shorter than v_i, which is easily done.

Checking condition *3b* is trickier. One way to do it is to consider the set of all α's such that $\alpha u_i \not\sqsubseteq v_i$. This is a regular set that can be obtained effectively. Then the set of all corresponding $(\alpha u_i)\{v_i\}$ is also regular and effective (see [5]) so that we can check that it is included in $A_{i\backslash L}$.

For condition *4a*, and given some $L \in \mathcal{L}(R)$ and some $i \in \Sigma$, the set of all β's such that $u_i \sqsubseteq \beta v_i$ is regular and effective. One can then compute the corresponding set of all $(\beta v_i)[u_i]$, again regular and effective, and check inclusion in $B_{i\backslash L}$. The complement set of all β's such that $u_i \not\sqsubseteq \beta v_i$ is also regular and effective, and one easily derives the corresponding $u_i\{\beta v_i\}$'s (a finite set of suffixes of u_i), hence checking condition *4b*. □

Proof (of Theorem 3.2). Since $\mathsf{PEP}^{\mathrm{reg}}$ is r.e., it is sufficient to prove that it is also co-r.e. For this we observe that, by Propositions 3.5 and 3.6, a $\mathsf{PEP}^{\mathrm{reg}}$ instance is negative if, and only if, there exists a stable family $(A_L, B_L)_{L\in\mathcal{L}(R)}$ satisfying $\varepsilon \in A_R$. One can effectively enumerate all families $(A_L, B_L)_{L\in\mathcal{L}(R)}$ of regular languages and check whether they are stable (Proposition 3.7) (and have $\varepsilon \in A_R$). If the $\mathsf{PEP}^{\mathrm{reg}}$ instance is negative, this procedure will eventually terminate, e.g., when it considers the blocker family. □

Remark 3.8. Computing the blocker family for a negative $\mathsf{PEP}^{\mathrm{reg}}$ instance cannot be done effectively (this is a consequence of known results on lossy channel systems). Thus when the procedure described above terminates, there is no way to know that it has encountered the largest blocking family. □

4 Variants and extensions

Short morphisms. $\mathsf{PEP}^{\mathrm{reg}}_{\leq 1}$ is $\mathsf{PEP}^{\mathrm{reg}}$ with the constraint that all u_i's and v_i's have length ≤ 1, i.e., they must belong to $\Gamma \cup \{\varepsilon\}$.

Proposition 4.1. $\mathsf{PEP}^{\mathrm{reg}}$ *reduces to* $\mathsf{PEP}^{\mathrm{reg}}_{\leq 1}$.

Proof (Sketch). Let u, v, R be a $\mathsf{PEP}^{\mathrm{reg}}$ instance. For all $i \in \Sigma$, write u_i in the form $a_i^1 \dots a_i^{l_i}$ and v_i in the form $b_i^1 \dots b_i^{m_i}$. Let $k = \max\{l_i, m_i \mid i \in \Sigma\}$. One builds a $\mathsf{PEP}^{\mathrm{reg}}_{\leq 1}$ instance u', v', R' by letting $\Sigma' \stackrel{\text{def}}{=} \Sigma \times \{1, 2, \dots, k\}$, $u'(i,p) \stackrel{\text{def}}{=} a_i^p$ if $p \leq l_i$, and $u'(i,p) \stackrel{\text{def}}{=} \varepsilon$ otherwise. Similarly, $v'(i,p)$ is v_i^p, the pth letter in v_i, or ε. We now let $R' \stackrel{\text{def}}{=} h(R)$ where $h : \Sigma \to \Sigma'$ is the morphism defined by $h(i) = (i,1)(i,2)\dots(i,k)$. Finally u', v', R' is a $\mathsf{PEP}^{\mathrm{reg}}_{\leq 1}$ instance that is positive iff u, v, R is positive. □

Constraining u_σ and v_σ. $\mathsf{PEP}^{\mathrm{u\_reg}}$ is like $\mathsf{PEP}^{\mathrm{reg}}$ except that the constraint $R \subseteq \Gamma^*$ now applies to u_σ: a solution is some $\sigma \in \Sigma^*$ with $u_\sigma \in R$ (and $u_\sigma \sqsubseteq v_\sigma$). Similarly, $\mathsf{PEP}^{\mathrm{v\_reg}}$ has the constraint apply to v_σ, while $\mathsf{PEP}^{\mathrm{uv\_reg}}$ has two constraints, $R_1, R_2 \subseteq \Gamma^*$, that apply to, respectively and simultaneously, u_σ and v_σ.

Proposition 4.2. $\mathsf{PEP}^{\mathrm{uv\_reg}}$ *reduces to* $\mathsf{PEP}^{\mathrm{reg}}$.

Proof. Let u, v, R_1, R_2 be a $\mathsf{PEP}^{\mathrm{uv\_reg}}$ instance. Let $R \stackrel{\mathrm{def}}{=} u^{-1}(R_1) \cap v^{-1}(R_2)$. (Recall that the image of a regular R by an inverse morphism is regular and can easily be constructed from R.) By definition $\sigma \in R$ iff $u_\sigma \in R_1$ and $v_\sigma \in R_2$. Thus the $\mathsf{PEP}^{\mathrm{reg}}$ instance u, v, R is positive iff u, v, R_1, R_2 is. □

Reductions exist in the other direction, as the next two propositions show.

Proposition 4.3. $\mathsf{PEP}^{\mathrm{reg}}$ *reduces to* $\mathsf{PEP}^{\mathrm{v\_reg}}$.

Proof (Sketch). Let u, v, R be a $\mathsf{PEP}^{\mathrm{reg}}$ instance. W.l.o.g., we may assume that $\Sigma \cap \Gamma = \varnothing$. Define a $\mathsf{PEP}^{\mathrm{v\_reg}}$ instance u', v', R' by letting $v' : \Sigma^* \to (\Gamma \cup \Sigma)^*$ be given by $v'_i \stackrel{\mathrm{def}}{=} i.v_i$ and keeping $u' = u$ unchanged. Let $R' \stackrel{\mathrm{def}}{=} h^{-1}(R)$ where $h : (\Gamma \cup \Sigma)^* \to \Gamma^*$ is the erasing morphism that suppresses letters from Σ. Note that $v'_\sigma \in R'$ iff $\sigma = h(v'_\sigma) \in R$, so that u', v', R' is a positive $\mathsf{PEP}^{\mathrm{v\_reg}}$ instance iff u, v, R is a positive $\mathsf{PEP}^{\mathrm{reg}}$ instance. □

Proposition 4.4. $\mathsf{PEP}^{\mathrm{reg}}_{\leq 1}$ *reduces to* $\mathsf{PEP}^{\mathrm{u\_reg}}$.

Proof (Sketch). Let u, v, R be a $\mathsf{PEP}^{\mathrm{reg}}_{\leq 1}$ instance. W.l.o.g., we assume $\Sigma = \{1, 2, \ldots, k\}$ and let $\Sigma' \stackrel{\mathrm{def}}{=} \{0\} \cup \Sigma$ with $g : \Sigma'^* \to \Sigma^*$ the associated erasing morphism. We also assume $\Gamma \cap \Sigma' = \varnothing$ and let $\Gamma' \stackrel{\mathrm{def}}{=} \Gamma \cup \Sigma'$, with $h : \Gamma'^* \to \Sigma^*$ as erasing morphism.

With u, v, R, we associate a $\mathsf{PEP}^{\mathrm{u\_reg}}$ instance u', v', R' based on Σ' and Γ', and defined by $u'_0 \stackrel{\mathrm{def}}{=} \varepsilon$, $v'_0 \stackrel{\mathrm{def}}{=} 1.2\ldots k$, and, for $i \in \Sigma$, $u'_i \stackrel{\mathrm{def}}{=} i.u_i$ and $v'_i \stackrel{\mathrm{def}}{=} v_i$. Letting $R' = h^{-1}(R)$ ensures that $u'_\sigma \in R'$ iff $g(\sigma) \in R$. Clearly, if $u'_\sigma \sqsubseteq v'_\sigma$, then $u_{g(\sigma)} \sqsubseteq v_{g(\sigma)}$. Conversely, if $u_{\sigma'} \sqsubseteq v_{\sigma'}$, it is possible to find a $\sigma \in g^{-1}(\sigma')$ that satisfies $u'_\sigma \sqsubseteq v'_\sigma$: this is just a matter of inserting enough 0's at the appropriate places (and this is where we use the assumption that all u_i's and v_i's have length ≤ 1). □

Now, since $\mathsf{PEP}^{\mathrm{u\_reg}}$ and $\mathsf{PEP}^{\mathrm{v\_reg}}$ are special cases of $\mathsf{PEP}^{\mathrm{uv\_reg}}$, and since $\mathsf{PEP}^{\mathrm{reg}}_{\leq 1}$ is a special case of $\mathsf{PEP}^{\mathrm{reg}}$, Propositions 4.1, 4.2, 4.3 and 4.4 entail the following.

Theorem 4.5. $\mathsf{PEP}^{\mathrm{reg}}$*,* $\mathsf{PEP}^{\mathrm{reg}}_{\leq 1}$*,* $\mathsf{PEP}^{\mathrm{u\_reg}}$*,* $\mathsf{PEP}^{\mathrm{v\_reg}}$ *and* $\mathsf{PEP}^{\mathrm{uv\_reg}}$ *are inter-reducible.*

Context-free constraints and Presburger constraints. $\mathsf{PEP}^{\mathrm{cf}}$ is the extension of $\mathsf{PEP}^{\mathrm{reg}}$ where we allow the constraint R to be any context-free language (say, given in the form of a context-free grammar). $\mathsf{PEP}^{\mathrm{dcf}}$ is $\mathsf{PEP}^{\mathrm{cf}}$ restricted to *deterministic* context-free constraints. $\mathsf{PEP}^{\mathrm{Pres}}$ is the extension where $R \subseteq \Sigma^*$ can be any language defined by a Presburger constraint over the number of occurrences of each letter from Σ (or, equivalently, the commutative image of R is a semilinear subset of the commutative monoid $\mathbb{N}^\Sigma$).

Theorem 4.6. $\mathsf{PEP}^{\mathrm{dcf}}$*,* $\mathsf{PEP}^{\mathrm{cf}}$ *and* $\mathsf{PEP}^{\mathrm{Pres}}$ *are undecidable.*

Proof. The (classic) PCP problem reduces to $\mathsf{PEP}^{\mathrm{dcf}}$ or $\mathsf{PEP}^{\mathrm{Pres}}$ by associating, with an instance $u,v:\Sigma^*\to\Gamma^*$, the constraint $R_{\geq}\subseteq\Sigma^+$ defined by

$$\sigma\in R_{\geq}\overset{\text{def}}{\Leftrightarrow}|u_\sigma|\geq|v_\sigma| \text{ and } \sigma\neq\varepsilon.$$

Obviously, $u_\sigma\sqsubseteq v_\sigma$ and $\sigma\in R_{\geq}$ iff $u_\sigma=v_\sigma$. Observe that $R_{\geq}$ is easily defined in the quantifier-free fragment of Presburger logic. Furthermore, since $R_{\geq}$ can be recognized by a counter machine with a single counter, it is indeed deterministic context-free. ☐

5 From $\mathsf{PEP}^{\mathrm{reg}}$ to lossy channel systems

We now reduce $\mathsf{PEP}^{\mathrm{reg}}$ to $\mathsf{ReachLcs}$, the reachability problem for lossy channel systems.

Systems composed of several finite-state components communicating via several channels (all of them lossy) can be simulated by systems with a single channel and a single component (see, e.g., [15, Section 5]). Hence we define here a lossy channel system (a LCS) as a tuple $S=(Q,\mathtt{M},\{\mathtt{c}\},\Delta)$ where $Q=\{q_1,q_2,\ldots\}$ is a finite set of *control states*, $\mathtt{M}=\{a_1,a_2,\ldots\}$ is a finite *message alphabet*, $\mathtt{c}$ is the name of the single *channel*, and $\Delta=\{\delta_1,\ldots\}$ is the finite set of *transition rules*. Rules in Δ are *writing rules*, of the form $q\xrightarrow{\mathtt{c}!u}q'$ (where $u\in\mathtt{M}^*$ is any sequence of messages), or *reading rules* $q\xrightarrow{\mathtt{c}?u}q'$. We usually omit writing "$\mathtt{c}$" in rules since there is only one channel, and no possibility for confusion.

The behaviour of S is given in the form of a transition system. A *configuration* of S is a pair $\langle q,v\rangle\in Q\times\mathtt{M}^*$ of a state and a channel contents. Transitions between configurations are obtained from the rules. Formally, $\langle q,v\rangle\to\langle q',v'\rangle$ is a valid transition iff Δ contains a reading rule of the form $q\xrightarrow{?u}q'$ and $v=uv'$, or Δ contains a writing rule of the form $q\xrightarrow{!u}q'$ and $v'=vu'$ for some $u'\sqsubseteq u$. The intuition behind this definition is that a reading rule consumes u from the head of the channel while a writing rule appends a (nondeterministically chosen) subsequence u' of u, and the rest of u is lost. See, e.g., [3, 15] for more details on LCS's.

Remark 5.1. This behaviour is called *write-lossy* because messages can only be lost when they are appended to the channel, but once inside $\mathtt{c}$ they remain there until a reading rule consumes them. This is different from, e.g., *front-lossy* semantics, where messages are lost when consumed (see [14]), or from the usual definition of LCS's, where messages can be lost at any time. These differences are completely inessential when one considers questions like reachability or termination, and authors use the definition that is technically most convenient for their purpose. In this paper, as in [1], the write-lossy semantics is the most convenient one. ☐

Remark 5.2. Below we use extended rules of the form $q\xrightarrow{!u\,?v}q'$. These are a shorthand notation for pairs of "consecutive" rules $q\xrightarrow{!u}s$ and $s\xrightarrow{?v}q'$ where s is an extra intermediary state that is not used anywhere else (and that we may omit listing in Q). ☐

$\mathsf{ReachLcs}$, the *reachability problem for LCS's*, is the question, given a LCS S and two states $q, q' \in Q$, whether there exists a sequence of transitions in S going from $\langle q, \varepsilon\rangle$ to $\langle q', \varepsilon\rangle$. The rest of this section proves the following theorem.

Theorem 5.3. $\mathsf{PEP}^{\mathrm{reg}}$ *reduces to* $\mathsf{ReachLcs}$.

Remark 5.4. Since $\mathsf{ReachLcs}$ is decidable [3], Theorem 5.3 provides another proof that $\mathsf{PEP}^{\mathrm{reg}}$ is decidable. □

Let u, v, R be a $\mathsf{PEP}^{\mathrm{reg}}$ instance and $\sigma \in R$ be a solution. We say σ is a *direct* solution if $u_\rho \sqsubseteq v_\rho$ for every prefix ρ of σ. An equivalent formulation is: $\sigma = i_1 \ldots i_m$ is a direct solution iff there are words $v'_1, \ldots, v'_m$ such that:
1. $v'_k \sqsubseteq v_{i_k}$ for all $k = 1, \ldots, m$,
2. $u_{i_1} \ldots u_{i_m} = v'_1 \ldots v'_m$,
3. $|u_{i_1} \ldots u_{i_k}| \le |v'_1 \ldots v'_k|$ for all $k = 1, \ldots, m$.

A *codirect* solution is defined in a similar way, with the difference that we now require $|u_{i_1} \ldots u_{i_k}| \ge |v'_1 \ldots v'_k|$ for all $k = 1, \ldots, m$ (i.e., the u_i's are ahead of the v'_i's instead of lagging behind).

We let $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{dir}}$ and $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{codir}}$ denote the questions whether a $\mathsf{PEP}^{\mathrm{reg}}$ instance has a direct (resp. codirect) solution. Obviously, $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{dir}}$ and $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{codir}}$ are equivalent problems since an instance u, v, R has a codirect solution iff its mirror image $\widetilde{u}, \widetilde{v}, \widetilde{R}$ had a direct solution.

Proposition 5.5. $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{dir}}$ *(and* $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{codir}}$*) reduce to* $\mathsf{ReachLcs}$.

Proof (Idea). Let u, v, R be a $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{dir}}$ instance. Recall that R is given via some NFA $\mathcal{A}_R = \langle Q, \Sigma, \delta, q_{\mathrm{init}}, F\rangle$. With this instance, one associates a LCS $S = \langle Q, \Gamma, \{\mathtt{c}\}, \Delta\rangle$ with a graph structure (Q, Δ) inherited from $\mathcal{A}_R$. The difference is that an edge $r \xrightarrow{i} s$ in $\mathcal{A}_R$ gives rise to a transition rule $r \xrightarrow{!v_i\,?u_i} s$ in S. With such rules, S can write the sequence $v'_1, v'_2, \ldots$ on $\mathtt{c}$, read $u_{i_1}, u_{i_2}, \ldots$ in lock-step fashion, and finally can move from the initial configuration $\langle q_{\mathrm{init}}, \varepsilon\rangle$ to some final configuration $\langle f, \varepsilon\rangle$ with $f \in F$ iff the $\mathsf{PEP}^{\mathrm{reg}}$ instance has a direct solution. Restricting to direct solutions is what ensures that the $v'_1 \ldots v'_k$ prefix that has been written on the channel is always longer than $u_{i_1} \ldots u_{i_k}$. □

If we now look at a general solution to a $\mathsf{PEP}^{\mathrm{reg}}$ instance (more precisely a $\mathsf{PEP}^{\mathrm{reg}}_{\le 1}$ instance) it can be decomposed as a succession of alternating direct and codirect solutions to subproblems that are constrained by residuals of R.

Formally, assume u, v, R is a $\mathsf{PEP}^{\mathrm{reg}}_{\le 1}$ instance and $\sigma = i_1 \ldots i_m$ is a solution. Then there are words $v'_1, \ldots, v'_m$ with $v'_k \sqsubseteq v_{i_k}$ for $k = 1, \ldots, m$, and such that $u_{i_1} \ldots u_{i_m} = v'_1 \ldots v'_m$. Now, for $0 \le k \le m$, define $d_k \stackrel{\mathrm{def}}{=} |u_{i_1} \ldots u_{i_k}| - |v'_1 \ldots v'_k|$. Then obviously $d_0 = d_m = 0$. σ is a direct solution if $d_k \le 0$ for all k. It is codirect if $d_k \ge 0$ for all k. In general, d_k may oscillate between positive and negative values. But since all u_i's and v_i's have length ≤ 1, the difference $d_{k+1} - d_k$ is in $\{-1, 0, 1\}$. Hence d_k cannot change sign without being zero. In summary, the following holds:

Lemma 5.6. *A* $\mathsf{PEP}^{\mathrm{reg}}_{\le 1}$ *instance* u, v, R *is positive iff there are states* $q_0, q_1, \ldots, q_{2m}$ *in* $\mathcal{A}_R$ *with* $q_0 = q_{\mathrm{init}}$, $q_{2m} \in F$, *and such that, for all* $0 \le i < m$, u, v, R_{2i} *is a positive* $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{dir}}$ *instance and* u, v, R_{2i+1} *is a positive* $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{codir}}$ *instance (where* R_i *is the regular language recognized by* $\mathcal{A}_R$ *when the initial state is changed to* q_i *and the final states to* $\{q_{i+1}\}$*).*

With Lemma 5.6, one may prove Theorem 5.3 by extending the construction proving Proposition 5.5. Now the LCS looks for a sequence of alternating direct and codirect solutions. In direct mode, it proceeds as earlier until some state q_{2i+1} is reached. It may then switch to codirect mode. For this, it checks that the channel is empty (see below), guesses nondeterministically q_{2i+2}, stores q_{2i+1} and q_{2i+2} in its finite memory, and now looks for a codirect solution to u, v, R_{2i+1}. This is done by working on the mirror problem $\widetilde{u}, \widetilde{v}$, and moving backward from q_{2i+2} to q_{2i+1}. When q_{2i+1} is reached (which can be checked since it has been stored when switching mode) it is possible to switch back to direct mode, starting from state q_{2i+2} (which was stored too), again after checking that the channel is empty. The emptiness checks use standard tricks, e.g., rules $q \xrightarrow{!\#?\#} q$ that write a special symbol $\# \not\in \Gamma$ and consume it immediately.

6 Reachability for unidirectional systems

6.1 Unidirectional systems

$\mathsf{ReachUcs}$ is the reachability problem for UCS, i.e., systems of two components communicating *unidirectionally* via one reliable and one lossy channel, as illustrated in Fig. 1. A UCS has the form $S = (Q_1, Q_2, \mathtt{M}, \{\mathtt{r},\mathtt{l}\}, \Delta_1, \Delta_2)$. The Q_1, Δ_1 pair defines the sender component, with rules of the form $q \xrightarrow{\mathtt{r}!u} q'$ or $q \xrightarrow{\mathtt{l}!u} q'$. The Q_2, Δ_2 pair has rules $q \xrightarrow{\mathtt{r}?u} q'$ or $q \xrightarrow{\mathtt{l}?u} q'$, defining the receiver component. A configuration is a tuple $\langle q_1, q_2, v_1, v_2\rangle$ with control states q_1 and q_2 for the components, contents v_1 for channel $\mathtt{r}$, and v_2 for $\mathtt{l}$.

The operational semantics is as expected. A rule $q \xrightarrow{\mathtt{r}!u} q'$ (resp. $q \xrightarrow{\mathtt{l}!u} q'$) from Δ_1 gives rise to all transitions $\langle q, q_2, v_1, v_2\rangle \to \langle q', q_2, v_1 u, v_2\rangle$ (resp. all $\langle q, q_2, v_1, v_2\rangle \to \langle q', q_2, v_1, v_2 u'\rangle$ for $u' \sqsubseteq u$). A rule $q \xrightarrow{\mathtt{r}?u} q'$ (resp. $q \xrightarrow{\mathtt{l}?u} q'$) from Δ_2 gives rise to all transitions $\langle q_1, q, uv_1, v_2\rangle \to \langle q_1, q', v_1, v_2\rangle$ (resp. all $\langle q_1, q, v_1, uv_2\rangle \to \langle q_1, q', v_1, v_2\rangle$). Observe that message losses only occur when writing to channel $\mathtt{l}$.

Remark 6.1. A consequence of unidirectionality is that a run $\langle q_1, q_2, v_1, v_2\rangle \to \cdots \to \langle q_1', q_2', v_1', v_2'\rangle$ can always be reordered so that it first uses only transitions from Δ_1 that fill the channels, followed by only transitions from Δ_2 that consume from them. ☐

Theorem 6.2. [5] $\mathsf{ReachLcs}$ *reduces to* $\mathsf{ReachUcs}$.

6.2 From unidirectional systems to $\mathsf{PEP}^{\mathrm{reg}}$

We now show that $\mathsf{PEP}^{\mathrm{reg}}$ is expressive enough to encode $\mathsf{ReachUcs}$.

Theorem 6.3. $\mathsf{ReachUcs}$ *reduces to* $\mathsf{PEP}^{\mathrm{reg}}$.

Consider an $\mathsf{ReachUcs}$ instance that asks whether one can go from $\langle q_0, q_0', \varepsilon, \varepsilon\rangle$ to $\langle q_f, q_f', \varepsilon, \varepsilon\rangle$[1] in some UCS $S = (Q_1, Q_2, \mathtt{M}, \{\mathtt{r},\mathtt{l}\}, \Delta_1, \Delta_2)$. Without loss of generality,

[1] For simplification purposes, this proof considers $\mathsf{ReachUcs}$ instances where the channels are empty in the starting and ending configurations. This is no real loss of generality since the general $\mathsf{ReachUcs}$ problem easily reduces to the restricted problem.

we assume that the rules in S only read or write at most one message: formally, we write $\mathtt{M}_\varepsilon$ for $\mathtt{M} \cup \{\varepsilon\}$ and denote with $\alpha(\delta) \in \mathtt{M}_\varepsilon$ (resp. $\beta(\delta) \in \mathtt{M}_\varepsilon$) the messages that rule δ writes to, or reads from, $\mathtt{r}$ (resp. $\mathtt{l}$). Observe that whether $\alpha(\delta)$ and $\beta(\delta)$ are read or written depends on whether δ belongs to Δ_1 or Δ_2. Observe also that there is at least one ε among $\alpha(\delta)$ and $\beta(\delta)$.

Assume that the ReachUcs instance is positive and that a witness run π first uses a sequence of rules $\delta_1 \dots \delta_m \in \Delta_1^*$, followed by a sequence $\gamma_1 \dots \gamma_l \in \Delta_2^*$ (this special form is explained in Remark 6.1). Then π first writes $w = \alpha(\delta_1) \dots \alpha(\delta_m)$ to $\mathtt{r}$, then reads $w' = \alpha(\gamma_1) \dots \alpha(\gamma_l)$ from $\mathtt{r}$, and we conclude that $w = w'$. Simultaneously, it writes a subword w'' of $\beta(\delta_1) \dots \beta(\delta_m)$ to $\mathtt{l}$, and reads it in the form $\beta(\gamma_1) \dots \beta(\gamma_l)$.

We are now ready to express this as a PEP$^{\text{reg}}$ problem. Let $\Sigma \stackrel{\text{def}}{=} \Delta_1 \cup \Delta_2$ (assuming $\Delta_1 \cap \Delta_2 = \varnothing$) and $\Gamma \stackrel{\text{def}}{=} \mathtt{M}$. The morphisms are given by

$$u(\delta) \stackrel{\text{def}}{=} \begin{cases} \beta(\delta) & \text{if } \delta \in \Delta_2, \\ \varepsilon & \text{otherwise,} \end{cases} \qquad v(\delta) \stackrel{\text{def}}{=} \begin{cases} \beta(\delta) & \text{if } \delta \in \Delta_1, \\ \varepsilon & \text{otherwise.} \end{cases}$$

Now write R_1 for the set of all sequences $\delta_1 \dots \delta_m \in \Delta_1^*$ that form a connected path from q_0 to q_f in Q_1, and R_2 for the set of all sequences $\gamma_1 \dots \gamma_l \in \Delta_2^*$ that form a connected path from q'_0 to q'_f in Q_2. Let R_3 contains all rules $\delta \in \Delta_1 \cup \Delta_2$ with $\alpha(\delta) = \varepsilon$, and all sequences $\delta.\gamma$ in $\Delta_1 \Delta_2$ with $\alpha(\delta) = \alpha(\gamma)$. R_1 and R_2 are regular subsets of Γ^*, while R_3 is even finite.

We now let $R \stackrel{\text{def}}{=} (R_1 \bowtie R_2) \cap R_3^*$, where $\bowtie$ denotes the shuffle of two languages (recall that this is regularity preserving). We conclude the proof of Theorem 6.3 with:

Lemma 6.4. [5] *u, v, R is a positive* PEP$^{\text{reg}}$ *instance iff the* ReachUcs *instance is positive.*

By combining with Theorems 6.3 and 6.2 we obtain the equivalence (inter-reducibility) of our three problems: PEP$^{\text{reg}}$, ReachLcs and ReachUcs. This has two important new corollaries:

Corollary 6.5. ReachUcs *is decidable (but not primitive recursive).*

Corollary 6.6. PEP$^{\text{reg}}$ *is (decidable but) not primitive recursive.*

7 Concluding remarks

We introduced PEP$^{\text{reg}}$, a variant of Post Correspondence Problem based on embedding (a.k.a. subword) rather than equality. Furthermore, a regular constraint can be imposed on the allowed solutions, which makes the problem non-trivial.

PEP$^{\text{reg}}$ was introduced while considering ReachUcs, a verification problem for channel systems where a sender may send messages to a receiver through one reliable and one lossy channel, and where no communication is allowed in the other direction.

Our main results are (1) a non-trivial proof that PEP$^{\text{reg}}$ is decidable, and (2) three non-trivial reductions showing that PEP$^{\text{reg}}$, ReachUcs and ReachLcs are equivalent. ReachLcs is the now well-known verification problem for lossy channel systems, where all channels are lossy but where no unidirectionality restriction applies. The equivalence between the three problems has two unexpected consequences: it shows that ReachUcs is decidable, and that $\mathsf{PEP}^{\mathrm{reg}}$ is not primitive recursive. We also show that (3) $\mathsf{PEP}^{\mathrm{reg}}$ and $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{dir}}$, an important variant, are inter-reducible.

Beyond the applications to the theory of channel systems (our original motivation), the discovery of $\mathsf{PEP}^{\mathrm{reg}}$ is interesting in its own right. Indeed, in recent years the literature has produced many hardness proofs that rely on reductions from ReachLcs. We expect that such results, existing or yet to come, are easier to prove by reducing from $\mathsf{PEP}^{\mathrm{reg}}$, or from $\mathsf{PEP}^{\mathrm{reg}}_{\mathrm{dir}}$, than from ReachLcs.

References

1. P. A. Abdulla, C. Baier, S. Purushothaman Iyer, and B. Jonsson. Simulating perfect channels with probabilistic lossy channels. *Information and Computation*, 197(1–2):22–40, 2005.
2. P. A. Abdulla, J. Deneux, J. Ouaknine, and J. Worrell. Decidability and complexity results for timed automata via channel machines. In *Proc. ICALP 2005*, vol. 3580 of *Lect. Notes Comp. Sci.*, pp. 1089–1101. Springer, 2005.
3. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.
4. R. Amadio and Ch. Meyssonnier. On decidability of the control reachability problem in the asynchronous π-calculus. *Nordic Journal of Computing*, 9(2):70–101, 2002.
5. P. Chambard and Ph. Schnoebelen. Post embedding problem is not primitive recursive, with applications to channel systems. Research Report LSV-07-28, Lab. Specification and Verification, ENS de Cachan, Cachan, France, September 2007.
6. S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. In *Proc. LICS 2006*, pp. 17–26. IEEE Comp. Soc. Press, 2006.
7. A. Finkel. Decidability of the termination problem for completely specificied protocols. *Distributed Computing*, 7(3):129–135, 1994.
8. D. Gabelaia, A. Kurucz, F. Wolter, and M. Zakharyaschev. Non-primitive recursive decidability of products of modal logics with expanding domains. *Annals of Pure and Applied Logic*, 142(1–3):245–268, 2006.
9. V. Halava, M. Hirvensalo, and R. de Wolf. Marked PCP is decidable. *Theoretical Computer Science*, 255(1–2):193–204, 2001.
10. B. Konev, F. Wolter, and M Zakharyaschev. Temporal logics over transitive states. In *Proc. CADE 2005*, vol. 3632 of *Lect. Notes Comp. Sci.*, pp. 182–203. Springer, 2005.
11. S. Lasota and I. Walukiewicz. Alternating timed automata. In *Proc. FOSSACS 2005*, vol. 3441 of *Lect. Notes Comp. Sci.*, pp. 250–265. Springer, 2005.
12. 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.
13. K. Ruohonen. On some variants of Post's correspondence problem. *Acta Informatica*, 4(19):357–367, 1983.
14. Ph. Schnoebelen. Bisimulation and other undecidable equivalences for lossy channel systems. In *Proc. TACS 2001*, vol. 2215 of *Lect. Notes Comp. Sci.*, pp. 385–399. Springer, 2001.
15. Ph. Schnoebelen. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.