# Pumping and Counting on 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.** The Regular Post Embedding Problem is a variant of Post's Correspondence Problem where one compares strings with the subword relation and imposes additional regular constraints on admissible solutions. It is known that this problem is decidable, albeit with very high complexity.
We consider and solve variant problems where the set of solutions is compared to regular constraint sets and where one counts the number of solutions. Our positive results rely on two non-trivial pumping lemmas for Post-embedding languages and their complements.

## 1 Introduction

Post's Correspondence Problem, or shortly PCP, is 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 $\sigma \in \Sigma^+$. Post's *Embedding* Problem, shortly PEP, is a variant of PCP where one asks whether $u(\sigma)$ is a (scattered) subword of $v(\sigma)$ for some $\sigma$. The subword relation, 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. The *Regular* Post Embedding Problem, or $\mathsf{PEP}^{\text{reg}}$, is an extension of PEP where one adds the requirement that only solutions $\sigma$ belonging to a given regular language $R \subseteq \Sigma^*$ are admitted. PEP and $\mathsf{PEP}^{\text{reg}}$ were introduced, and shown decidable, in [2, 3].

*Regular constraints and the set of* PEP*-solutions.* The decidability of $\mathsf{PEP}^{\text{reg}}$ can be restated under the following form: it is decidable, given two morphisms $u, v : \Sigma^* \to \Gamma^*$ and a regular language $R \subseteq \Sigma^*$, whether the following holds:

$$\exists x \in R : u(x) \sqsubseteq v(x). \qquad \text{(Existence)}$$

In other words, and letting $PE(u,v) \stackrel{\text{def}}{=} \{x \in \Sigma^* \mid u(x) \sqsubseteq v(x)\}$, one can decide whether $R \cap PE(u,v) \neq \varnothing$. However, this problem has very high complexity. Here the regular language $R$, acting as a constraint on the form of solutions, plays a key role. Indeed, in the special case where $R = \Sigma^+$, the problem becomes trivial (if there are solutions, in particular length-one solutions exist) which probably explains why PEP and $\mathsf{PEP}^{\text{reg}}$ had not been investigated earlier.

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

In this paper, we prove the decidability of the following questions:

$$\forall x \in R : u(x) \sqsubseteq v(x), \qquad \text{(Universality)}$$
$$\exists^{\infty} x \in R : u(x) \sqsubseteq v(x), \qquad \text{(Infinity)}$$
$$\neg\exists^{\infty} x \in R : u(x) \not\sqsubseteq v(x). \qquad \text{(Cofiniteness)}$$

"Universality" asks whether all words in $R$ are solutions. "Infinity" asks whether $R$ contains infinitely many solutions $x$, while dually "Cofiniteness" asks whether all but finitely many $x \in R$ are solutions. Equivalently, these questions ask whether $R \subseteq PE(u,v)$, whether $R \cap PE(u,v) =_a \varnothing$, and whether $R \smallsetminus PE(u,v) =_a \varnothing$, writing $S =_a S'$ to denote the "quasi-equality" of two sets, i.e., equality up to a finite subset. As a consequence of these decidability results we can compute the number of words in $R$ that are (respectively, that are not) solutions.

These results are obtained with the help of two pumping lemmas, one for sets of solutions and one for sets of "antisolutions", i.e., words $x$ such that $u(x) \not\sqsubseteq v(x)$. These pumping lemmas are the more technically involved developments of this paper. Proving them relies on two kinds of techniques: (1) combinatorics of words in presence of the subword relation and associated operations, and (2) a miniaturisation of Higman's Lemma that gives effective bounds on the length of bad sequences.

*Related work.* The Regular Post Embedding Problem was introduced in [2, 3] where its decidability was proved. These papers also showed that $\mathsf{PEP}^{\mathrm{reg}}$ is expressive enough to encode problems on lossy channel systems, or LCS's. In fact, encoding in both directions exist, hence $\mathsf{PEP}^{\mathrm{reg}}$ is exactly at level $\mathcal{F}_{\omega^\omega}$ in the Fast Growing Hierarchy. Thus, although it is decidable, $\mathsf{PEP}^{\mathrm{reg}}$ is not primitive-recursive, and not even multiply-recursive (see [4] and the references therein).

A consequence of the above encodings is that $\mathsf{PEP}^{\mathrm{reg}}$ is an abstract problem that is inter-reducible with a growing list of decision problems that have the same $\mathcal{F}_{\omega^\omega}$ complexity: metric temporal logic [14], products of modal logics [8], leftist grammars [9, 6], data nets [11], alternating one-clock timed automata [1, 10], etc.

*On complexity.* Aiming at simplicity, our main decidability proofs do not come with explicit statements regarding the computational complexity of the associated problems. The decidability proofs can be turned into deterministic algorithms with complexity in $\mathcal{F}_{\omega^\omega}$, providing the same upper bound that already applies to $\mathsf{PEP}^{\mathrm{reg}}$. Regarding lower bounds, it is clear that "Infinity" is at least as hard as $\mathsf{PEP}^{\mathrm{reg}}$. We do not know if the same lower bound holds for "Universality" and "Cofiniteness".

*Outline of the paper.* Section 2 recalls the necessary definitions and notations. Section 3 deals with combinatorics on words with subwords. Section 4 proves the decidability of comparisons with regular sets. Then our pumping lemma is stated in Section 5 and used in Section 6 for deciding finiteness, counting, and quasi-regular questions. Sections 7 and 8 prove the two halves of the pumping lemma. Finally, a technical appendix contains all proofs omitted in the main text.

## 2 Notations and definitions

*Words and morphisms.* We write $x, y, z, t, \sigma, \rho, \alpha, \beta, \ldots$ for words, i.e., finite sequences of letters such as $a, b, c, i, j, \ldots$ from alphabets $\Sigma, \Gamma, \ldots$, and denote with $x.y$, or $xy$, the concatenation of $x$ and $y$. We let $\varepsilon$ denote the empty word. The *length* of $x$ is written $|x|$. A *morphism* from $\Sigma^*$ to $\Gamma^*$ 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 $\Sigma = \{a, b, \ldots\}$. We often simply write $u_a, u_b, \ldots$, and $u_x$, instead of $u(a), u(b), \ldots$, and $u(x)$. Finally, for a morphism $u : \Sigma^* \to \Gamma^*$, we let $K_u = \max_{a \in \Sigma} |u_a|$ denote the *expansion factor* of $u$, thus called because clearly $|u_x| \leq K_u \times |x|$.

The *mirror image* of a word $x$ is denoted $\widetilde{x}$, e.g., $\widetilde{abc} = cba$. The mirror image of a language $L$ is $\widetilde{L} \stackrel{\text{def}}{=} \{\widetilde{x} \mid x \in L\}$. It is well-known that the mirror image of a regular language is regular. For a morphism $h : \Sigma^* \to \Gamma^*$, the mirror morphism $\widetilde{h}$ is defined by $\widetilde{h}(x) \stackrel{\text{def}}{=} \widetilde{h(\widetilde{x})}$, ensuring $\widetilde{h}(\widetilde{x}) = \widetilde{h(x)}$.

*Syntactic congruence.* For a language $L$, we let $\sim_L$ denote the syntactic congruence induced by $L$: $x \sim_L y \stackrel{\text{def}}{\Leftrightarrow} \forall w, w' (wxw' \in L \Leftrightarrow wyw' \in L)$. The Myhill-Nerode Theorem states that $\sim_L$ has finite index iff $L$ is a regular language. For a regular $L$, we let $n_L$ denote the number of equivalence classes w.r.t. $\sim_L$.[1]

*Subwords and Higman's Lemma.* Given two words $x$ and $y$, we write $x \sqsubseteq y$ when $x$ is a *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, aka *embedding*, is a partial ordering on words. It is compatible with the monoidal structure:

$$\varepsilon \sqsubseteq x, \qquad (x \sqsubseteq y \wedge x' \sqsubseteq y') \Rightarrow xx \sqsubseteq yy'.$$

It is well-known (Higman's Lemma) that the subword relation is a well-quasi-ordering 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. $\sqsubseteq$).

We say that a sequence $x_1, \ldots, x_l, \ldots$ of words in $\Sigma^*$ is $n$-good if there exists indexes $i_1 < i_2 < \ldots < i_n$ such that $x_{i_1} \sqsubseteq x_{i_2} \sqsubseteq \ldots \sqsubseteq x_{i_n}$, i.e., if the sequence contains a subsequence of length $n$ that is increasing w.r.t. embedding. It is $n$-bad otherwise. Higman's Lemma states that every infinite sequence is 2-good, and even $n$-good for any $n \in \mathbb{N}$. Hence $n$-bad sequences are finite.

Higman's Lemma is often described as being "non-effective" in that it does not give any information on the length of bad sequences. Indeed, arbitrarily long bad sequences exist. However, upper bounds on the length of bad sequences can certainly be given when one restricts to "simple" sequences. Such finitary versions of well-quasi-ordering properties are called "miniaturisations" in proof-theoretical circles.

In this paper we consider a very simple miniaturisation that applies to "controlled" sequences [7]. Formally, and given $k \in \mathbb{N}$, we say that a sequence $x_1, \ldots, x_l$ of words in $\Sigma^*$ is *$k$-controlled* if $|x_i| \leq i \times k$ for all $i = 1, \ldots, l$. We shall use the following result:

[1] If the minimal complete DFA that accepts $L$ has $q$ states, then $n_L$ can be bounded by $q^q$.

**Lemma 2.1 (see App. A).** *There exists a* bounding function $H : \mathbb{N}^3 \to \mathbb{N}$ *such that, for any* $n,k \in \mathbb{N}$ *and* $l \geq H(n,k,|\Sigma|)$*, any* $k$*-controlled sequence of* $l$ *words in* $\Sigma^*$ *is* $n$*-good.*

The lemma states that if a $k$-controlled sequence is long enough, it is $n$-good. Equivalently, $n$-bad sequences are shorter that $H(n,k,|\Sigma|)$ or are not $k$-controlled.

## 3 Composing, decomposing, and iterating words and subwords

This section is devoted to the subword ordering and the way it interacts with concatenations and factorizations. It proves a few basic results, e.g., Lemma 3.7, that we have been unable to find in the technical literature [12, 13]. All missing proofs can be found in App. B.

### 3.1 Available suffixes

When $x \sqsubseteq y$, we decompose $y$ as a concatenation $y = y_1y_2$ such that $y_1$ is the *shortest* prefix of $y$ with $x \sqsubseteq y_1$. We call $y_1$ the "*used prefix*" and $y_2$ the "*available suffix*". We use $y \oslash x$ to denote the available suffix. For example, $a\underline{bc}\underline{a}bc \oslash \underline{ba} = bc$. Note that $y \oslash x$ is only defined when $x \sqsubseteq y$.

**Lemma 3.1.** *$x \sqsubseteq y$ and $x' \sqsubseteq (y \oslash x)y'$ imply $xx' \sqsubseteq yy'$.*

**Corollary 3.2.** *$x \sqsubseteq y$ implies $x(y \oslash x) \sqsubseteq y$.*

**Lemma 3.3.** *$x \sqsubseteq y$ and $xx' \sqsubseteq yy'$ imply $x' \sqsubseteq (y \oslash x)y'$.*

### 3.2 Unmatched suffixes

When $x \not\sqsubseteq y$, we decompose $x$ as a concatenation $x = x_1x_2$ such that $x_1$ is the *longest* prefix of $x$ with $x_1 \sqsubseteq y$. We call $x_1$ the "*matched prefix*" and $x_2$ the "*unmatched suffix*". We use $x \ominus y$ to denote the unmatched suffix. For example $\underline{aa}bcabc \ominus b\underline{ac}\underline{a} = bcabc$. Note that $x \ominus y$ is only defined when $x \not\sqsubseteq y$ (hence $x \ominus y \neq \varepsilon$).

**Lemma 3.4.** *$x \not\sqsubseteq y$ and $xx' \not\sqsubseteq yy'$ imply $[(x \ominus y)x'] \ominus y' = xx' \ominus yy'$.*

**Corollary 3.5.** *$x \not\sqsubseteq y$ and $xx' \not\sqsubseteq yy'$ imply $(x \ominus y)x' \not\sqsubseteq y'$.*

**Lemma 3.6.** *$x \not\sqsubseteq y$ and $xx' \sqsubseteq yy'$ imply $(x \ominus y)x' \sqsubseteq y'$.*

### 3.3 Iterating factors

**Lemma 3.7.** *$xy \sqsubseteq yz$ if, and only if, $x^ky \sqsubseteq yz^k$ for all $k \in \mathbb{N}$.*

*Proof.* We only need to prove the "$\Rightarrow$" direction. This is done by induction on the length of $y$. The cases where $y = \varepsilon$ or $x = \varepsilon$ or $k = 0$ are obvious, so we assume that $|y|$, $|x|$ and $k$ are strictly positive. There are now two cases:
1. If $x \sqsubseteq y$, we consider a factorization $y = y_1y_2$ (e.g., $y_2 = y \oslash x$ is convenient) with $x \sqsubseteq y_1$ (hence $x^k \sqsubseteq y_1^k$) and $y \sqsubseteq y_2z$. Since $|y_2| < |y|$ (because $x \neq \varepsilon$ and hence $y_1 \neq \varepsilon$),

the induction hypothesis applies and from $y_1y_2 = y \sqsubseteq y_2z$ one gets $y_1^k y_2 \sqsubseteq y_2 z^k$. Now $x^k y \sqsubseteq y_1^k y = y_1 y_1^k y_2 \sqsubseteq y_1 y_2 z^k = yz^k$.
2. If $x \not\sqsubseteq y$, we write $x = x_1x_2$ with $x_2 = x \ominus y$. Thus $x_1 \sqsubseteq y$ and $x_2 y \sqsubseteq z$. Thus there exists a factorization $z = z_1z_2$ s.t. $x_2 \sqsubseteq z_1$ (entailing $x \sqsubseteq yz_1$) and $y \sqsubseteq z_2$. Now $x^k y \sqsubseteq (yz_1)^k z_2 = yz_1(yz_1)^{k-1}z_2 \sqsubseteq yz_1(z_2z_1)^{k-1}z_2 = yz^k$. □

**Lemma 3.8.** *Assume $x \not\sqsubseteq y$, $xz \not\sqsubseteq yt$, and $x \ominus y \sqsubseteq xz \ominus yt$. Then for all $k \in \mathbb{N}$:*

$$xz^k \not\sqsubseteq yt^k. \tag{$Z_k$}$$

*Furthermore, if we let $r_k \stackrel{def}{=} xz^k \ominus yt^k$, then for all $k \in \mathbb{N}$:*

$$r_0 \sqsubseteq r_k \sqsubseteq r_{k+1}. \tag{$R_k$}$$

*Proof.* The hypothesis for the Lemma are that ($Z_0$), ($Z_1$) and ($R_0$) hold. We prove, by induction on $k$, that ($Z_k$) and ($R_{k-1}$) imply ($Z_{k+1}$) and ($R_k$).

Proof of ($Z_{k+1}$): applying Coro. 3.5 on ($Z_0$) and ($Z_1$) yields $r_0 z \not\sqsubseteq t$, hence a fortiori $r_k z \not\sqsubseteq t$ using ($R_{k-1}$). Combining with ($Z_k$) and applying Lemma 3.6 contrapositively entails $xz^k z \not\sqsubseteq yt^k t$, i.e., ($Z_{k+1}$).

Proof of ($R_k$): $r_{k+1}$ is $xz^{k+1} \ominus yt^{k+1}$. By Lemma 3.4, this is $[(xz^k \ominus yt^k)z] \ominus t$, i.e., $r_k z \ominus t$. From ($R_{k-1}$) we get $r_{k-1}z \ominus t \sqsubseteq r_k z \ominus t$. However $r_{k-1}z \ominus t = r_k$ (Lemma 3.4). Finally $r_k \sqsubseteq r_{k+1}$. □

## 4 Regular properties of sets of PEP solutions

Given two morphisms $u, v : \Sigma^* \to \Gamma^*$, a word $x \in \Sigma^*$ is called a "*solution*" (of Post's Embedding Problem) when $u_x \sqsubseteq v_x$. Otherwise it is an "*antisolution*". We let $PE(u,v)$ denote the set of solutions (for given $u$ and $v$). Note that $\varepsilon$ is always a solution.

We consider questions where we are given a PEP instance $u, v$ with $u, v : \Sigma^* \to \Gamma^*$ and a regular language $R \subseteq \Sigma^*$. The considered problems are
PEP_Inclusion: does $PE(u,v) \subseteq R$?
PEP_Containment: does $PE(u,v) \supseteq R$?
PEP_Equality: does $PE(u,v) = R$?

It is tempting to compare $PE(u,v)$ with another Post-embedding set, however:

**Theorem 4.1.** *The questions "does $PE(u,v) \cap PE(u',v') = \{\varepsilon\}$?" and "does $PE(u,v) \subseteq PE(u',v')$?" are $\Pi_1^0$-complete.*

*Proof.* $\Pi_1^0$-hardness can be shown directly by reduction from PCP. For the first question, simply let $u' = v$ and $v' = u$. Then a common solution has $u_x \sqsubseteq v_x = u'_x \sqsubseteq v'_x = u_x$, i.e., $u_x = v_x$.

For the second question we use a more subtle encoding: assume w.l.o.g. that $\Gamma$ contains two distinct symbols $a, b$ and that $u_x \neq \varepsilon$ when $x \neq \varepsilon$. Let now $u'_x \stackrel{\text{def}}{=} (ab)^{|u_x|}$ and $v'_x \stackrel{\text{def}}{=} (ba)^{|v_x|}$. Thus $u'_x \sqsubseteq v'_x$ if, and only if, $x = \varepsilon$ or $|u_x| < |v_x|$. Finally, $PE(u,v) \smallsetminus PE(u',v')$ contains the non-trivial PCP solutions. □

**Theorem 4.2.** PEP_Inclusion, PEP_Containment *and* PEP_Equality *are decidable.*

Note that, while comparisons with a regular language are decidable, regularity itself is undecidable, at least in the more general form stated here:

**Proposition 4.3 (Regularity is undecidable [5]).** *The question "is $R \cap PE(u,v)$ a regular language?" is $\Sigma_1^0$-complete.*

The remainder of this section proves Theorem 4.2.

We first observe that PEP_Inclusion and $\mathsf{PEP}^{\mathrm{reg}}$ are inter-reducible since $(u,v,R)$ is a positive instance for PEP_Inclusion if, and only if, $(u,v,\Sigma^* \setminus R)$ is a negative instance for $\mathsf{PEP}^{\mathrm{reg}}$. Hence the decidability of PEP_Inclusion follows from the decidability of $\mathsf{PEP}^{\mathrm{reg}}$, proved in [2, 3].

For the decidability of PEP_Containment (and then of PEP_Equality), we fix an instance $(u,v,R)$.

For a word $x \in \Sigma^*$, we say that $x$ is *good* if $u_x \sqsubseteq v_x$ and then we let $w_x \stackrel{\text{def}}{=} v_x \oslash u_x$, otherwise it is *bad* and then we let $r_x \stackrel{\text{def}}{=} u_x \ominus v_x$. We say that $x$ is *alive* if $xy \in R$ for some $y$, otherwise it is *dead*. Finally, we write $|R|$ for the number of states of a FSA for $R$, and let $L \stackrel{\text{def}}{=} K_v \times |R|$ be a *size threshold* (more details in the proof of Lemma 4.5).

A word $x$ is a *cut-off* if, and only if, one of the following conditions holds:

**dead cut-off:** $x$ is dead;
**subsumption cut-off:** there exists a strict prefix $x'$ of $x$ such that $x' \sim_R x$, and either
1. both $x$ and $x'$ are good, with $w_{x'} \sqsubseteq w_x$,
2. or both $x$ and $x'$ are bad, with $r_x \sqsubseteq r_{x'}$;

**big cut-off:** $x$ is alive, bad and $|r_x| > L$.

Let $T \subseteq \Sigma^*$ be the set of all words that do not have a cut-off as a (strict) prefix. $T$ is prefix-closed and can be seen as a tree.

**Lemma 4.4.** *$T$ is finite.*

*Proof.* We show that $T$, seen as a tree, has no infinite branch. Hence, and since it is finitely branching, it is finite (Kőnig's Lemma).

Assume, by way of contradiction, that $T$ has an infinite branch labeled by some $x_0, x_1, x_2, \ldots$ (and recall that every $x_i$ is a prefix of all the $x_{i+k}$'s). We show that one of the $x_i$ must be a cut-off, which contradicts the assumption.

Since the syntactic congruence $\sim_R$ has finite index, there exists an infinite subsequence $x_0, x_1, x_2, \ldots$ (renumbered for convenience) of $\sim_R$-equivalent $x_i$'s. If infinitely many of the $x_i$'s are good, one of them must be a subsumption cut-off since, by Higman's Lemma, the infinite sequence of the $w_{x_i}$'s (for good $x_i$'s) must have some $w_{x'} \sqsubseteq w_x$. If only finitely many of the $x_i$'s are good, then infinitely many of them are bad and either some $r_{x_i}$ has size larger than $L$ (hence $x_i$ is a big cut-off), or all $r_{x_i}$'s have size at most $L$, hence belong to a finite set $\Gamma^{\leq L}$, and two of them must be equal (hence there must be a subsumption cut-off). □

With the next two lemmas, we show that $T$ contains enough information to decide whether $R \subseteq PE(u,v)$.

**Lemma 4.5.** *If $T$ contains a big cut-off, then $R \not\subseteq PE(u,v)$.*

*Proof.* Assume $x$ is a big cut-off (i.e., is alive, bad, and with $|r_x| > L$) in $T$. It is alive so $xy \in R$ for some $y$. We pick the smallest such $y$, ensuring that $|y| < |R|$ (the number of states of an FSA for $R$). Since $x$ is bad, we know that $u_x \not\sqsubseteq v_x$. Note that $|v_y| \leq K_v \times |y| \leq K_v \times |R| \leq L$ so that $|v_y| < |r_x|$ and, consequently, $r_x \not\sqsubseteq v_y$. Thus, and since $r_x = u_x \ominus v_x$, applying Lemma 3.6 contrapositively gives $u_x \not\sqsubseteq v_x v_y$ and, *a fortiori*, $u_{xy} \not\sqsubseteq v_{xy}$. Finally $xy \notin PE(u,v)$. Since $xy \in R$, we conclude $R \not\subseteq PE(u,v)$. □

There is a reciprocal.

**Lemma 4.6.** *Assume that $T$ has no big cut-offs and that $(R \cap T) \subseteq PE(u,v)$. Then $R \subseteq PE(u,v)$.*

*Proof.* Consider some $x \in R$: we show that $u_x \sqsubseteq v_x$ by induction on the size of $x$. If $x \in T$ then $x \in (R \cap T) \subseteq PE(u,v)$ and we are done. If $x \notin T$, then a prefix of $x$ is a cut-off. This cannot be a big cut-off (we assumed $T$ has none) or a dead cut-off (the prefix is alive since $x \in R$). Hence this is a subsumption cut-off, caused by one of its prefixes. Finally, $x$ can be written under the form $x = x_1x_2x_3$ with $x_1x_2$ the subsumption cut-off, and $x_1$ the prefix justifying the subsumption. We know $x_2 \neq \varepsilon$ ($x_1$ is a strict prefix of the cut-off) and $x_1 \sim_R x_1x_2$. Hence $x_1x_3 \in R$ (since $x_1x_2x_3 \in R$) and $u_{x_1x_3} \sqsubseteq v_{x_1x_3}$ by induction hypothesis.

There are now two cases, depending on what kind of subsumption is at hand.
1. If $x_1$ is good then $u_{x_1} \sqsubseteq v_{x_1}$. Combining with $u_{x_1x_3} \sqsubseteq v_{x_1x_3}$ entails $u_{x_3} \sqsubseteq w_{x_1}v_{x_3}$ (Lemma 3.3). From $w_{x_1} \sqsubseteq w_{x_1x_2}$ (condition for subsumption) we deduce $u_{x_3} \sqsubseteq w_{x_1x_2}v_{x_3}$. Combining with $u_{x_1x_2} \sqsubseteq v_{x_1x_2}$ ($x_1x_2$ too is good), Lemma 3.1 yields $u_{x_1x_2}u_{x_3} \sqsubseteq v_{x_1x_2}v_{x_3}$.
2. If $x_1$ is bad, then $u_{x_1x_3} \sqsubseteq v_{x_1x_3}$ and $u_{x_1} \not\sqsubseteq v_{x_1}$ entail $r_{x_1}u_{x_3} \sqsubseteq v_{x_3}$ (Lemma 3.6). From $r_{x_1x_2} \sqsubseteq r_{x_1}$ (condition for subsumption) we deduce $r_{x_1x_2}u_{x_3} \sqsubseteq v_{x_3}$. Combined with $u_{x_1x_2} \not\sqsubseteq v_{x_1x_2}$ ($x_1x_2$ too is bad), applying Coro. 3.5 contrapositively yields $u_{x_1x_2}u_{x_3} \sqsubseteq v_{x_1x_2}v_{x_3}$.
In both cases we proved that $x_1x_2x_3 \in PE(u,v)$ as requested. □

We can now prove the decidability of PEP_Containment: the tree $T$ can be built effectively starting from the root since it is easy to see whether a word is a cut-off. The construction terminates thanks to Lemma 4.4. Once $T$ is at hand, Lemmas 4.5 and 4.6 gives an effective criterion for deciding whether $R \subseteq PE(u,v)$: it is enough to check that $T$ has no big cut-off and that all the words $x \in T$ satisfy $u_x \sqsubseteq v_x$ or do not belong to $R$.

## 5 Pumpable solutions and antisolutions

Let $u, v : \Sigma^* \to \Gamma^*$ be a given PEP instance.

**Definition 5.1.** *A triple of words $(x,y,z) \in \Sigma^*$ with $y \neq \varepsilon$ is a* pumpable solution *if $xy^kz \in PE(u,v)$ for all $k \in \mathbb{N}$.*
*It is a* pumpable antisolution *if $xy^kz \notin PE(u,v)$ for all $k \in \mathbb{N}$.*

In other words, a pumpable solution denotes an infinite subset of $PE(u,v)$ of the form $xy^*z$, while a pumpable antisolution denotes an infinite subset of its complement. Our interest in pumpable solutions and antisolutions is that they provide simple witnesses proving that $PE(u,v)$ (or its complement) is infinite.

We observe that these witnesses are effective:

**Proposition 5.2 (Decidability of pumpability).** *It is decidable whether $(x,y,z)$ is a pumpable solution, and also whether it is a pumpable antisolution.*

*Proof.* Checking that $(x,y,z)$ is a pumpable solution reduces to the PEP_Containment problem, while checking that it is not a pumpable antisolution reduces to the $\text{PEP}^{\text{reg}}$ problem (or, equivalently, PEP_Inclusion). □

We can now state our main technical result. Here (and below) we speak loosely of "a pumpable solution", when we mean "*the language denoted by* a pumpable solution".

**Lemma 5.3 (Pumping Lemma).** *Let $R \subseteq \Sigma^*$ be a regular language.*
*1. If $R \cap PE(u,v)$ is infinite, it contains a pumpable solution.*
*2. If $R \setminus PE(u,v)$ is infinite, it contains a pumpable antisolution.*

Section 7 is devoted to a proof of the Pumping Lemma for solutions, while Section 8 proves the Pumping Lemma for antisolutions. Without waiting for that, we list the main consequences on our questions.

## 6 Quasi-regular properties and counting properties

For two languages $L, L$, we say that $L$ is *quasi-included* in $L'$, written $L \subseteq_a L'$, when $L \setminus L'$ is finite, and that they are *quasi-equal*, written $L =_a L'$, when $L \subseteq_a L'$ and $L' \subseteq_a L$.

We consider the following questions, where we are given a PEP instance $u,v$ and a regular $R \subseteq \Sigma^*$:
PEP_Quasi_Inclusion: does $PE(u,v) \subseteq_a R$?
PEP_Quasi_Containment: does $PE(u,v) \supseteq_a R$?
PEP_Quasi_Equality: does $PE(u,v) =_a R$?

**Theorem 6.1.** PEP_Quasi_Inclusion, PEP_Quasi_Containment *and* PEP_Quasi_Equality *are decidable.*

*Proof.* We start with PEP_Quasi_Inclusion. This problem is co-r.e. since when $PE(u,v) \setminus R$ is infinite, there is a pumpable solution in $\Sigma^* \setminus R$ (Pumping Lemma) that can be guessed and checked (Prop. 5.2). It is also r.e. since $PE(u,v) \subseteq_a R$ iff there is a finite language $F \subseteq \Sigma^*$ s.t. $PE(u,v) \subseteq R \cup F$, which can be checked (Theo. 4.2) since $R \cup F$ is a regular language. Thus PEP_Quasi_Inclusion, being r.e. and co-r.e., is decidable.

We use the same reasoning to show that PEP_Quasi_Containment is decidable. Then PEP_Quasi_Equality is obviously decidable as well. □

We also consider counting questions where the answer is a number in $\mathbb{N} \cup \{\omega\}$:
PEP_NbSol: what is the cardinality of $R \cap PE(u,v)$?
PEP_NbAntisol: what is the cardinality $R \setminus PE(u,v)$?

**Theorem 6.2.** PEP_NbSol *and* PEP_NbAntisol *are decidable (more precisely, the associated counting functions are recursive).*

*Proof.* We start with PEP_NbSol. We can first check whether the cardinality of $R \cap PE(u,v)$ is finite by deciding whether $PE(u,v) \subseteq_a (\Sigma^* \setminus R)$ (using the decidability of PEP_Quasi_Inclusion). If we find that the cardinality is infinite, we are done. Otherwise we can enumerate all words in $R$ and check whether they are solutions. At any given stage during this enumeration, we can check whether the current set $F$ of already found solutions is complete by deciding whether $PE(u,v) \cap (R \setminus F) = \varnothing$ (using the decidability of PEP_Inclusion). We are bound to eventually find a complete set since we only started enumerating solutions in $R$ knowing there are finitely many of them.

The same method works for PEP_NbAntisol, this times using the decidability of PEP_Containment and PEP_Quasi_Containment. □

## 7 Pumping in long solutions

We start with a sufficient condition for pumpability of solutions.

**Definition 7.1.** *A triple $x,y,z \in \Sigma^*$ with $y \neq \varepsilon$ is* positive *if the following four conditions are satisfied:*

$$u_x \sqsubseteq v_x, \quad \text{(C1)} \qquad u_x u_y \sqsubseteq v_x v_y, \quad \text{(C2)}$$
$$u_x u_y u_z \sqsubseteq v_x v_y v_z, \quad \text{(C3)} \qquad (v_x \oslash u_x) \sqsubseteq (v_x v_y \oslash u_x u_y). \quad \text{(C4)}$$

**Lemma 7.2.** *If $(x,y,z)$ is positive then $(x,y,yz)$ is a pumpable solution.*

*Proof.* Assume that $(x,y,z)$ is positive, so that (C1–4) hold. Write shortly $w$ for $v_x \oslash u_x$ and $w'$ for $v_{xy} \oslash u_{xy}$. From (C1) and the definition of $w$, Coro. 3.2 yields:

$$u_x w \sqsubseteq v_x. \quad \text{(C5)}$$

From (C2), it further yields $u_x u_y w' \sqsubseteq v_x v_y$, from which (C4) entails:

$$u_x u_y w \sqsubseteq v_x v_y. \quad \text{(C6)}$$

Applying Lemma 3.3 on (C1) and (C3) (respectively on (C1) and (C6)) yields:

$$u_y u_z \sqsubseteq w v_y v_z, \quad \text{(C7)} \qquad u_y w \sqsubseteq w v_y. \quad \text{(C7}')$$

Applying Lemma 3.7 on (C7') gives

$$u_{y^k} w = (u_y)^k w \sqsubseteq w (v_y)^k = w v_{y^k} \text{ for all } k \in \mathbb{N}. \quad \text{(C8)}$$

With (C5) and (C8), Lemma 3.1 entails

$$u_x u_{y^k} w \sqsubseteq v_x v_{y^k} \text{ for all } k \in \mathbb{N}. \quad \text{(C9)}$$

With (C7) and (C9), it then entails

$$u_x u_{y^k} u_{yz} \sqsubseteq v_x v_{y^k} v_{yz} \text{ for all } k \in \mathbb{N}, \quad \text{(C10)}$$

which just states that $(x,y,yz)$ is a pumpable solution. □

We now let $n_R$ denote the number of equivalence classes induced by $\sim_R$ (Section 2). Finally, we let $H_u$ and $H_v$ denote, respectively, $H(n_R+1,K_u,|\Gamma|)$ and $H(n_R+1,K_v,|\Gamma|)$. Recall that, by definition of the $H$ function (Lemma 2.1), any $K_u$-controlled sequence of at least $H_u$ $\Gamma$-words is $(n_R+1)$-good.

**Lemma 7.3.** *If $R$ contains a solution $\sigma \in PE(u,v)$ of length $|\sigma| \geq 2H_v$ then it contains a pumpable solution.*

(Observe that this will entail, as a corollary, the first half of the Pumping Lemma since, if $R \cap PE(u,v)$ is infinite, it contains solutions $\sigma$ of arbitrarily large length.)

*Proof.* Let $\sigma \in PE(u,v)$ be a solution of length $L$: $\sigma$ has $L+1$ prefixes $x_0, x_1, \ldots, x_L$. We consider the subsequence $x_{i_1}, x_{i_2}, \ldots x_{i_l}$ of all prefixes of $\sigma$ that satisfy $u_{x_{i_j}} \sqsubseteq v_{x_{i_j}}$ (called *good prefixes*) and split the proof in three main steps.

1. We show, by induction over $j$, that the sequence $\left(v_{x_{i_j}} \oslash u_{x_{i_j}}\right)_{j=1,\ldots,l}$ is $K_v$-controlled, i.e., writing $w_j$ for $v_{x_{i_j}} \oslash u_{x_{i_j}}$, that $|w_j| \leq j \times K_v$ for all $j = 1,\ldots,l$. The base case is obvious since $i_1 = 0$ and $w_1 = \varepsilon$. For the inductive case, we consider $j > 0$ so that $x_{i_j} = x_{i_j-1}.a$ for some $a \in \Sigma$ (the $i_j$-th letter in $\sigma$). If $u_{x_{i_j-1}} \sqsubseteq v_{x_{i_j-1}}$ (hence $i_{(j-1)} = (i_j) - 1$) then $w_j = v_{x_{i_j}} \oslash u_{x_{i_j}}$ is $(v_{x_{i_j-1}}.v_a) \oslash (u_{x_{i_j-1}}.u_a)$ which cannot be longer than $(v_{x_{i_j-1}}.v_a) \oslash u_{x_{i_j-1}}$, itself not longer than $(v_{x_{i_j-1}} \oslash u_{x_{i_j-1}}).v_a$. Thus $|w_j| \leq |w_{j-1}| + K_v$ and we conclude with the induction hypothesis. If on the other hand $u_{x_{i_j-1}} \not\sqsubseteq v_{x_{i_j-1}}$, then $w_j$ is a suffix of $v_a$ hence $|w_j| \leq K_v$.

2a. Assume now that $l \geq H_v$. Then, using Lemma 2.1, we conclude that there is a further subsequence $(x_{i_{j_r}})_{r=0,\ldots,n_R}$ of $n_R+1$ prefixes of $\sigma$ such that $w_{j_0} \sqsubseteq w_{j_1} \sqsubseteq \cdots \sqsubseteq w_{j_{n_R}}$. Since $n_R$ is the index of $\sim_R$, we deduce that there exists two such prefixes $x_{i_{jp}}$ (shortly, $x$) and $x_{i_{j_{p'}}}$ (shortly, $x'$) with $x \sim_R x'$. If we write $x'$ under the form $xy$ (NB: $y \neq \varepsilon$) and $\sigma$ under the form $xyz$, we have found a positive triple $(x,y,z)$. Then Lemma 7.2 applies and shows that $xy^*yz$ is a pumpable solution. Finally, since $x \sim_R xy$, we know that $xy^*yz$ is a subset of $R$.

2b. Observe that if a prefix $x_i$ of $\sigma = x_i.y_i$ is not good, then $\widetilde{y_i}$ is a good prefix of the solution $\widetilde{\sigma} \in PE(\widetilde{u},\widetilde{v})$ of the mirror PEP problem. Hence if $\sigma$ has $l < H_v$ good prefixes, $\widetilde{\sigma}$ has $l' \geq 2H_v - l > H_v$ good ones. Then the mirror problem falls in case 2a above (we note that $\sim_R$, $n_R$, and $K_v$ do not have to be adjusted when mirroring). We deduce that there is a pumpable solution in $\widetilde{R} \cap PE(\widetilde{u},\widetilde{v})$, whose mirror is a pumpable solution in $R \cap PE(u,v)$. □

## 8 Pumping in long antisolutions

As with pumpable solutions, there is a sufficient condition for pumpability of antisolutions.

**Definition 8.1.** *A triple $x,y,z \in \Sigma^*$ with $y \neq \varepsilon$ is* negative *if the following four conditions are satisfied:*

$$u_x \not\sqsubseteq v_x, \quad \text{(D1)} \qquad u_x u_y \not\sqsubseteq v_x v_y, \quad \text{(D2)}$$

$$u_x u_z \not\sqsubseteq v_x v_z \quad \text{(D3)} \qquad u_x \ominus v_x \sqsubseteq u_{xy} \ominus v_{xy} \quad \text{(D4)}$$

**Lemma 8.2.** *If $(x,y,z)$ is negative then $(x,y,z)$ is a pumpable antisolution.*

*Proof.* Assume that $(x,y,z)$ is negative, so that (D1–4) hold. Write shortly $r$ for $u_x \ominus v_x$ and $r'$ for $u_{xy} \ominus v_{xy}$. With (D1), (D2) and (D4), Lemma 3.8 applies and yields

$$u_{xy^k} \not\sqsubseteq v_{xy^k} \text{ for all } k \in \mathbb{N}, \tag{D5}$$

with furthermore

$$u_{xy^k} \ominus v_{xy^k} \sqsubseteq u_{xy^{k+1}} \ominus v_{xy^{k+1}}. \tag{D6}$$

On the other hand, (D1) and (D3) entail $ru_z \not\sqsubseteq v_z$ by Coro. 3.5, hence $(u_{xy^k} \ominus v_{xy^k})u_z \not\sqsubseteq v_z$ by (D6). We deduce that $u_{xy^k z} \not\sqsubseteq v_{xy^k z}$. □

**Lemma 8.3.** *If $R$ contains an antisolution $\sigma \notin PE(u,v)$ of length $|\sigma| \geq 2H_u$ then it contains a pumpable antisolution.*

(As a corollary, we obtain the second half of the Pumping Lemma.)

*Proof (Sketch).* We proceed as with Lemma 7.3. Write $L$ for $|\sigma|$, and $x_0, x_1, \ldots, x_L$ for the prefixes of $\sigma$. Consider the subsequence $x_{i_1}, x_{i_2}, \ldots x_{i_l}$ of all *bad prefixes* of $\sigma$, i.e., such that $u_{x_{i_j}} \not\sqsubseteq v_{x_{i_j}}$ and define $r_j = u_{x_{i_j}} \ominus v_{x_{i_j}}$. The sequence $(r_j)_{j=1,\ldots,l}$ is $K_u$-controlled.

If $l \geq H_u$, we find two positions $1 \leq p < p' \leq l$ such that $x_{i_{j_p}} \sim_R x_{i_{j_{p'}}}$ and $r_{j_p} \sqsubseteq r_{j_{p'}}$, so that, writing $x$ for $x_{i_{j_p}}$, $x'$ for $x_{i_{j_{p'}}}$, writing $x'$ under the form $xy$, and $\sigma$ under the form $xyz$, we can apply Lemma 8.2 and deduce that $(x,y,z)$ is a pumpable antisolution. Furthermore $xy^*z$ is a subset of $R$ since $xyz = \sigma \in R$ and $xy \sim_R x$.

Observe that if a prefix $x_i$ is not bad, then, writing $\sigma$ under the form $x_i y_i$, $\widetilde{y_i}$ is a bad prefix of the antisolution $\widetilde{\sigma} \notin PE(\widetilde{u},\widetilde{v})$ of the mirror problem. Thus, if $l < H_u$, then $\widetilde{\sigma}$ has $\geq H_u$ bad prefixes in the mirror problem. Hence $\widetilde{R} \smallsetminus PE(\widetilde{u}.\widetilde{v})$ contains a pumpable antisolution, whose mirror is a pumpable antisolution in $R \cap PE(u,v)$. □

*Remark 8.4.* Lemmas 7.3 and 8.3 show that one can strengthen the statement of the Pumping Lemma. Rather than assuming that $R \cap PE(u,v)$ (respectively, $R \smallsetminus PE(u,v)$) is infinite, we only need to assume that they contain a large enough element. □

## 9 Concluding remarks

The decidability of the Regular Post Embedding Problem means that one can find out whether the inequation $u(x) \sqsubseteq v(x)$ has a solution in a given regular $R$. In this paper, we investigated more general questions pertaining to the set of solutions $PE(u,v)$. We developed new techniques showing how one can decide regular questions (does $PE(u,v)$ contain, or is it included in, a given $R$?), finiteness and quasi-regular questions (does $PE(u,v)$ satisfy a regular constraint except perhaps for finitely many elements?), and counting questions (how many elements in some $R$ are — or are not — solutions?).

It is not clear how to go beyond these positive results. One avenue we have started to explore [5] is to consider Post-embedding questions with two variables, e.g.,

$$\exists x \in R_1 \, \forall y \in R_2 : u(xy) \sqsubseteq v(xy).$$

Another direction is suggested by the pumpings lemmas we developed here. These lemmas have applications beyond the finiteness problems we considered. For example, they are useful in the study of the expressive power of $\mathsf{PEP}^{\mathrm{reg}}$-languages, i.e., languages of the form $R \cap PE(u,v)$ for some $R,u,v$. For example, using the pumping lemma we can show that $L_0 \stackrel{\text{def}}{=} \{a^n b^n \mid n \in \mathbb{N}\}$ is not a $\mathsf{PEP}^{\mathrm{reg}}$-language. Now, and since $L_1 \stackrel{\text{def}}{=} \{a^n b^{n+m} \mid n,m \in \mathbb{N}\}$ and $L_2 \stackrel{\text{def}}{=} \{a^{n+m} b^n \mid n,m \in \mathbb{N}\}$ clearly are $\mathsf{PEP}^{\mathrm{reg}}$-languages, we conclude that $\mathsf{PEP}^{\mathrm{reg}}$-languages are not closed under intersection!

## References


1. 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.
2. 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.
3. 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.
4. 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.
5. P. Chambart and Ph. Schnoebelen. Computing blocker sets for the regular Post embedding problem. In *Proc. DTL 2010*, Lecture Notes in Computer Science. Springer, 2010. To appear.
6. 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.
7. E. A. Cichon and E. Tahhan Bittar. Ordinal recursive bounds for Higman's theorem. *Theoretical Computer Science*, 201(1–2):63–84, 1998.
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. T. Jurdziński. Leftist grammars are nonprimitive recursive. In *Proc. ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pages 51–62. Springer, 2008.
10. S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. Computational Logic*, 9(2), 2008.
11. R. Lazić, T. Newcomb, J. Ouaknine, A. W. Roscoe, and J. Worrell. Nets with tokens which carry data. *Fundamenta Informaticae*, 88(3):251–274, 2008.
12. M. Lothaire, editor. *Combinatorics on words*, volume 17 of *Encyclopedia of Mathematics and Its Applications*. Cambridge Univ. Press, 1983.
13. M. Lothaire, editor. *Algebraic combinatorics on words*, volume 90 of *Encyclopedia of Mathematics and Its Applications*. Cambridge Univ. Press, 2002.
14. 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.

# A Miniaturisation of Higman's Lemma

Fix $n > 0, k, \Sigma$ and consider the set $B$ of all $k$-controlled $n$-bad finite sequences. Every subsequence of a bad sequence is bad hence $B$ is prefix-closed and the sequences can be naturally arranged in a tree, with the empty sequence at its root. The tree is finitely branching because the sequences are $k$-controlled (and $\Sigma$ is finite). If $B$ is infinite, the tree has an infinite branch (Kőnig's Lemma), that is, there exists an infinite sequence $x_1, x_2, \ldots$ for which all finite prefixes are $n$-bad. Hence the infinite sequence itself is $n$-bad, which is impossible by Higman's Lemma.

Finally, $B$ must be finite and taking $H(n,k,\Sigma)$ as the length of the longest sequence in $B$ will fulfill the requirements for Lemma 2.1. □

Clearly, the same proof can handle more elaborate notions of controlled sequences.

*Remark A.1.* The proof of Lemma 2.1 not only yields the existence of the $H$ function, but also proves its computability: the tree can be built effectively. We refer to [7, 3] for more details on the size of $H(n,k,\Sigma)$. □

# B Combinatorics on subwords

It will be convenient to recall the following obvious facts:

**Fact B.1 (Splitting)**
*1. If $xy \sqsubseteq z$ then there exists a factorization $z = z'z''$ of $z$ such that $x \sqsubseteq z'$ and $y \sqsubseteq z''$.*
*2. If $x \sqsubseteq yz$ then there exists a factorization $x = x'x''$ of $x$ such that $x' \sqsubseteq y$ and $x'' \sqsubseteq z$.*

## B.1 Proofs for available suffixes

Recall that, when $x \sqsubseteq y$, the "used prefix" is the shortest prefix $y_1$ of $y$ such that $x \sqsubseteq y_1$. Then, writing $y = y_1 y_2$, what remains, i.e., $y_2$, is called the "available suffix" and denoted $y \oslash x$.

**Fact B.2 (Monotonicity)**
*1. If $x \sqsubseteq y$, then $(yz) \oslash x = (y \oslash x)z$.*
*2. If $xx' \sqsubseteq y$, then $y \oslash (xx') = (y \oslash x) \oslash x'$.*

**Lemma B.3.** *$xz \sqsubseteq y$ implies $z \sqsubseteq y \oslash x$.*

*Proof.* If $xz \sqsubseteq y$ then $x \sqsubseteq y'$ and $z \sqsubseteq y''$ for a factorization $y'y''$ of $y$ (Fact B.1). Since $y_1$ is the shortest prefix with $x \sqsubseteq y_1$, it is a prefix of $y'$, hence $y''$ is a suffix of $y_2$. Hence $z \sqsubseteq y_2 = y \oslash x$. □

We can now prove Lemma 3.1: $x \sqsubseteq y$ and $x' \sqsubseteq (y \oslash x)y'$ imply $xx' \sqsubseteq yy'$.

*Proof.* Let $x' = x'_1 x'_2$ be a factorization with $x'_1 \sqsubseteq y \oslash x$ and $x'_2 \sqsubseteq y'$. Lemma B.3 gives $xx'_1 \sqsubseteq y$. Concatenating with $x'_2 \sqsubseteq y'$ concludes since $xx' = (xx'_1)x'_2$. □

And Lemma 3.3: $x \sqsubseteq y$ and $xx' \sqsubseteq yy'$ imply $x' \sqsubseteq (y \oslash x)y'$.

*Proof.* From $xx' \sqsubseteq yy'$, Lemma B.3 gives $x' \sqsubseteq yy' \oslash x$. But $yy' \oslash x = (y \oslash x)y'$ since $x \sqsubseteq y$ (Fact B.2). □

### B.2 Proofs for unmatched suffixes

Recall that, when $x \not\sqsubseteq y$, the "matched prefix" is the longest prefix $x_1$ of $x$ s.t. $x_1 \sqsubseteq y$. Then, writing $x = x_1x_2$, what remains, i.e., $x_2$, is called the "unmatched suffix" and denoted $x \ominus y$.

The following is immediate from the definition:

**Fact B.4** *If $x \not\sqsubseteq yz$ then $x \ominus (yz) = (x \ominus y) \ominus z$.*

**Lemma B.5.** *Assume $x \not\sqsubseteq y$. Then $x \ominus y \sqsubseteq z$ implies $x \sqsubseteq yz$.*

*Proof.* In other words, assume $x_1 \sqsubseteq y$ and $x_2 \sqsubseteq z$ and conclude $x = x_1x_2 \sqsubseteq yz$. □

Reciprocally:

**Lemma B.6.** *Assume $x \not\sqsubseteq y$. Then $x \sqsubseteq yz$ implies $x \ominus y \sqsubseteq z$.*

*Proof.* If $x \sqsubseteq yz$ then $x' \sqsubseteq y$ and $x'' \sqsubseteq z$ for a factorization $x'x''$ of $x$ (Fact B.1). However, if $x \not\sqsubseteq y$, then $x = x_1x_2$ where $x_2 = x \ominus y$ and $x_1$ is is the longest prefix of $x$ with $x_1 \sqsubseteq z$, ensuring that $x'$ is a prefix of $x_1$, hence that $x_2$ is a suffix of $x''$. Finally, $x \ominus y = x_2 \sqsubseteq x'' \sqsubseteq z$. □

**Lemma B.7.** *$x \not\sqsubseteq y$ implies $(xx') \ominus y = (x \ominus y)x'$.*

*Proof.* Since $x \not\sqsubseteq y$, the prefixes of $x$ that embed in $y$ are exactly the prefixes of $xx'$ that embed in $y$, hence their longest matched prefixes coincide. The unmatched suffixes are $x_2$ for $x \ominus y$ and $x_2x'$ for $(xx') \ominus y$. □

We can now prove Lemma 3.4: $x \not\sqsubseteq y$ and $xx' \not\sqsubseteq yy'$ imply $(x \ominus y)x' \ominus y' = xx' \ominus yy'$.

*Proof.* By applying Lemma B.7: $(x \ominus y)x' = (xx') \ominus y$ and Fact B.4: $[(xx') \ominus y] \ominus y' = (xx') \ominus (yy')$. □

And Lemma 3.6: $x \not\sqsubseteq y$ and $xx' \sqsubseteq yy'$ imply $(x \ominus y)x' \sqsubseteq y'$.

*Proof.* If $x \not\sqsubseteq y$ then $x = x_1x_2$ with $x_1 \sqsubseteq y$ the matched prefix and $x_2 = x \ominus y$. If $xx' \sqsubseteq yy'$ then there is a factorisation $xx' = zz'$ with $z \sqsubseteq y$ and $z' \sqsubseteq y'$ (Fact B.1). Hence $z$ is a prefix of $x_1$ (Lemma B.7) so that $x_2x'$ is a suffix of $z'$. We conclude since $x_2x' = (x \ominus y)x'$. □