Foundations of Software Technology and Theoretical Computer Science (2011) Submission

Approximating Petri Net Reachability Along Context-free Traces

Mohamed Faouzi Atig[1], **Pierre Ganty**[2]

[1]Uppsala University, Sweden

[2]IMDEA Software Institute, Spain

ABSTRACT. We investigate the problem asking whether the intersection of a context-free language (CFL) and a Petri net language (PNL) is empty. Our contribution to solve this long-standing problem which relates, for instance, to the reachability analysis of recursive programs over unbounded data domain, is to identify a class of CFLs called the finite-index CFLs for which the problem is decidable. The k-index approximation of a CFL can be obtained by discarding all the words that cannot be derived within a budget k on the number of occurrences of non-terminals. A finite-index CFL is thus a CFL which coincides with its k-index approximation for some k. We decide whether the intersection of a finite-index CFL and a PNL is empty by reducing it to the reachability problem of Petri nets with *weak* inhibitor arcs, a class of systems with infinitely many states for which reachability is known to be decidable. Conversely, we show that the reachability problem for a Petri net with weak inhibitor arcs reduces to the emptiness problem of a finite-index CFL intersected with a PNL.

1 Introduction

Automated verification of infinite-state systems, for instance programs with (recursive) procedures and integer variables, is an important and a highly challenging problem. Pushdown automata (or equivalently context-free grammars) have been proposed as an adequate formalism to model procedural programs. However pushdown automata require finiteness of the data domain which is typically obtained by abstracting the program's data, for instance, using the predicate abstraction techniques [2, 8]. In many cases, reasoning over finite abstract domains yields to a too coarse analysis and is therefore not precise. To palliate this problem, it is natural to model a procedural program with integer variables as a pushdown automaton manipulating counters. In general, pushdown automata with counters are Turing powerful which implies that basic decision problems are undecidable (this is true even for the case finite-state automata with counters).

Therefore one has to look for restrictions on the model which retain sufficient expressiveness while allowing basic properties like reachability to be algorithmically verified. One such restriction is to forbid the test of a counter and a constant for equality. In fact, forbidding test for equality implies the decidability of the reachability problem for the case of finite-state automata with counters (i.e. Petri nets [12, 15]).

The verification problem for pushdown automata with (restricted) counters boils down to check whether a context-free language (CFL) and a Petri net language (PNL) are disjoint or not. We denote this last problem $\mathsf{PNL} \cap \mathsf{CFL} \stackrel{?}{=} \emptyset$.

The decidability of $\mathsf{PNL} \cap \mathsf{CFL} \stackrel{?}{=} \emptyset$ is open and lies at the very edge of our comprehension of infinite-state systems. We see two breakthroughs contributing to this question. First, determining the emptiness of a PNL was known to be decidable as early as the eighties. Then, in 2006, Reinhardt [15] lifted this result to an extension of PN with inhibitor arcs (that allow to test if a counter equals 0) which must satisfy some additional topological conditions. By imposing a topology on the tests for zero, Reinhardt prevents his model to acquire Turing powerful capabilities. We call his model PNW and the languages thereof PNWL.

Our contribution to the decidability of $\mathsf{PNL} \cap \mathsf{CFL} \stackrel{?}{=} \emptyset$ comes under the form of a partial answer which is better understood in terms of underapproximation. In fact, given a PNL L_1 and the language L of a context-free grammar we replace L by a subset L' which is obtained by discarding from L all the words that cannot be derived within a given budget $k \in \mathbb{N}$ on the number of non-terminal symbols. (In fact, the subset L' contains any word of L that can be generated by a derivation that contains at most k non-terminal symbols at each derivation step.) We show how to compute L' by annotating the variables of the context-free grammar for L with an allowance. What is particularly appealing is that the coverage of L increases with the allowance. Approximations induced by allowances are non-trivial: every regular or linear language is captured exactly with an allowance of 1, L' coincides with L when the allowance is unbounded, and under commutativity of concatenation L' coincides with L for some allowance $k \in \mathbb{N}$.

We call finite-index CFL, or fiCFL for short a context-free language where each of its words can be derived within a given budget. In this paper we prove the decidability of $\mathsf{PNL} \cap \mathsf{fiCFL} \stackrel{?}{=} \emptyset$ by reducing it to the emptiness problem of PNWL. We also prove the converse reduction showing those two problems are equivalent. Hence, we offer a whole new perspective on the emptiness problem for PNWL and $\mathsf{PNL} \cap \mathsf{CFL}$.

To conclude the introduction let us mention the recent result of [1] which builds on [12] to give an alternative proof of Reinhardt's result (PNW reachability is decidable) for the particular case where one counter only can be tested for zero.

2 Preliminaries

2.1 Context-Free Languages

An *alphabet* Σ is a finite non-empty set of *symbols*. A *word* w over an alphabet Σ is a finite sequence of symbols of Σ where the empty sequence is denoted ε. We write Σ^* for the set of words over Σ. Let $L \subseteq \Sigma^*$, L defines a *language*.

A *context-free grammar* (CFG) G is a tuple $(\mathcal{X}, \Sigma, \mathcal{P})$ where $\mathcal{X}$ is a finite non-empty set of *variables* (*non-terminal letters*), Σ is an alphabet of *terminal letters*, and $\mathcal{P} \subseteq \big(\mathcal{X} \times (\mathcal{X}^2 \cup \Sigma \cup \{\epsilon\})\big)$ a finite set of *productions* (the production (X, w) may also be denoted by $X \to w$). For every production $p = (X, w) \in \mathcal{P}$, we use $head(p)$ to denote the variable X. Observe that the form of the productions is restricted, but it has been shown in [11] that every CFG can be transformed, in polynomial time, into an equivalent grammar of this form.

Given two strings $u, v \in (\Sigma \cup \mathcal{X})^*$ we define the relation $u \Rightarrow v$, if there exists a production $(X, w) \in \mathcal{P}$ and some words $y, z \in (\Sigma \cup \mathcal{X})^*$ such that $u = yXz$ and $v = ywz$. We use $\Rightarrow^*$ for the reflexive transitive closure of $\Rightarrow$. Given $X \in \mathcal{X}$, we define the language $L_G(X)$, or simply $L(X)$ when G is clear form the context, as $\{w \in \Sigma^* \mid X \Rightarrow^* w\}$. A language L is *context-free* (CFL) if there exists a CFG $G = (\mathcal{X}, \Sigma, \mathcal{P})$ and $A \in \mathcal{X}$ such that $L = L_G(A)$.

2.2 Finite-index Approximation of Context-Free Languages

Let $k \in \mathbb{N}$, $G = (\mathcal{X}, \Sigma, \mathcal{P})$ be a CFG and $A \in \mathcal{X}$. A derivation from A given by $A = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_n$ is *k-index bounded* if for every $i \in \{0, \ldots, n\}$ at most k symbols of α_i are variables. We denote by $L^{(k)}(A)$ the subset of $L(A)$ such that for every $w \in L^{(k)}(A)$ there exists a k index bounded derivation $A \Rightarrow^* w$. We call $L^{(k)}(A)$ the *k-index approximation* of $L(A)$ or more generically we say that $L^{(k)}(A)$ is a *finite-index approximation* of $L(A)$.*

Let us now give some known properties of finite-index approximations. Clearly $\lim_{k\to\infty} L^{(k)}(A) = L(A)$. Moreover, let L be a regular or linear language†, then there exists a CFG G', and a variable A' of G' such that $L(A') = L = L^{(1)}(A')$. Also Luker showed in [14] that if $L(A) \subseteq L(w_1^* \cdots w_n^*)$ for some $w_i \in \Sigma^*$, then $L^{(k)}(A) = L(A)$ for some $k \in \mathbb{N}$. More recently, [5, 7] showed some form of completeness for finite-index approximation when commutativity of concatenation is assumed. It shows that there exists a $k \in \mathbb{N}$ such that $L(A) \subseteq \Pi(L^{(k)}(A))$ where $\Pi(L)$ denotes the language obtained by permuting symbols of w for every $w \in L$. As an incompleteness result, Salomaa showed in [16] that for the Dyck language $L_{D_1^*}$ over 1-pair of parentheses there is no CFG G', variable A' of G' and $k \in \mathbb{N}$ such that $L^{(k)}(A') = L_{D_1^*}$.

Inspired by [4, 6, 5] let us define the CFG $G^{[k]}$ which annotates the variables of $\mathcal{X}$ with a positive integer bounding the index of the derivations starting with that variable.

DEFINITION 1. *Let $G^{[k]} = (\mathcal{X}^{[k]}, \Sigma, \mathcal{P}^{[k]})$ be the context-free grammar defined as follows: $\mathcal{X}^{[k]} = \left\{X^{[i]} \mid 0 \le i \le k \wedge X \in \mathcal{X}\right\}$, and $\mathcal{P}^{[k]}$ is the smallest set such that:*

- *For every $X \to Y\,Z \in \mathcal{P}$, $\mathcal{P}^{[k]}$ has the productions $X^{[i]} \to Y^{[i-1]}Z^{[i]}$ and $X^{[i]} \to Y^{[i]}Z^{[i-1]}$ for every $i \in \{1, \ldots, k\}$.*
- *For every $X \to \sigma \in \mathcal{P}$ with $\sigma \in \Sigma \cup \{\epsilon\}$, $X^{[i]} \to \sigma \in \mathcal{P}^{[k]}$ for all $i \in \{0, \ldots, k\}$.*

What follows is a consequence of several results from different papers by Esparza *et al*. For the sake of clarity we give a direct proof in the appendix.

LEMMA 2. *Let $X \in \mathcal{X}$. We have $L(X^{[k]}) = L^{(k+1)}(X)$.*

2.3 Petri nets with Inhibitor Arcs

Let Σ be a finite non-empty set, a *multiset* $\mathbf{m} \colon \Sigma \mapsto \mathbb{N}$ over Σ maps each symbol of Σ to a natural number. Let $\mathbb{M}[\Sigma]$ be the set of all multiset over Σ.

We sometimes use the following notation for multisets $\mathbf{m} = [\![q_1, q_1, q_3]\!]$ for the multiset $\mathbf{m} \in \mathbb{M}[\{q_1, q_2, q_3, q_4\}]$ such that $\mathbf{m}(q_1) = 2$, $\mathbf{m}(q_2) = \mathbf{m}(q_4) = 0$, and $\mathbf{m}(q_3) = 1$. The empty multiset is denoted $\varnothing$.

Given $\mathbf{m}, \mathbf{m}' \in \mathbb{M}[\Sigma]$ we define $\mathbf{m} \oplus \mathbf{m}' \in \mathbb{M}[\Sigma]$ to be the multiset such that $\forall a \in \Sigma \colon (\mathbf{m} \oplus \mathbf{m}')(a) = \mathbf{m}(a) + \mathbf{m}'(a)$, we also define the natural partial order $\preceq$ on $\mathbb{M}[\Sigma]$ as

*Finite-index approximations were first studied in the 60's.

†See [10] for definitions.

follows: $\mathbf{m} \preceq \mathbf{m}'$ iff there exists $\mathbf{m}^\Delta \in \mathbb{M}[\Sigma]$ such that $\mathbf{m} \oplus \mathbf{m}^\Delta = \mathbf{m}'$. We also define $\mathbf{m} \ominus \mathbf{m}' \in \mathbb{M}[\Sigma]$ as the multiset such that $(\mathbf{m} \ominus \mathbf{m}') \oplus \mathbf{m}' = \mathbf{m}$ provided $\mathbf{m}' \preceq \mathbf{m}$.

A *Petri net* with inhibitor arcs (PNI for short) $N = (S, T, F = \langle Z, I, O\rangle, \mathbf{m}_\iota)$ consists of a finite non-empty set S of *places*, a finite set T of *transitions* disjoint from S, a tuple $F = \langle Z, I, O\rangle$ of functions $Z\colon T \mapsto 2^S$, $I\colon T \mapsto \mathbb{M}[S]$ and $O\colon T \mapsto \mathbb{M}[S]$, and an *initial marking* $\mathbf{m}_\iota \in \mathbb{M}[S]$. A marking $\mathbf{m}$ ($\in \mathbb{M}[S]$) of N assigns to each place $p \in S$ $\mathbf{m}(p)$ *tokens*.

A transition $t \in T$ is *enabled at* $\mathbf{m}$, written $\mathbf{m}\,[t\rangle$, if $I(t) \preceq \mathbf{m}$ and $\mathbf{m}(p) = 0$ for all $p \in Z(t)$. A transition t that is enabled at $\mathbf{m}$ can be *fired*, yielding a marking $\mathbf{m}'$ such that $\mathbf{m}' = (\mathbf{m} \ominus I(t)) \oplus O(t)$. We write this fact as follows: $\mathbf{m}\,[t\rangle\,\mathbf{m}'$. We extend enabledness and firing inductively to finite sequences of transitions as follows. Let $w \in T^*$. If $w = \varepsilon$ we define $\mathbf{m}\,[w\rangle\,\mathbf{m}'$ iff $\mathbf{m}' = \mathbf{m}$; else if $w = u \cdot v$ we have $\mathbf{m}\,[w\rangle\,\mathbf{m}'$ iff $\exists \mathbf{m}_1\colon \mathbf{m}\,[u\rangle\,\mathbf{m}_1 \wedge \mathbf{m}_1\,[v\rangle\,\mathbf{m}'$.

From the above definition we find that $\mathbf{m}$ is a reachable marking from $\mathbf{m}_0$ if and only if there exists $w \in T^*$ such that $\mathbf{m}_0\,[w\rangle\,\mathbf{m}$. Given a language $L \subseteq T^*$ over the transitions of N, the *set of reachable states from* $\mathbf{m}_0$ *along* L, written $[\mathbf{m}_0\rangle^L$, coincides with $\{\mathbf{m} \mid \exists w \in L\colon \mathbf{m}_0\,[w\rangle\,\mathbf{m}\}$. Incidentally, if L is unspecified then it is assumed to be T^* and we simply write $[\mathbf{m}_0\rangle$ for the set of states reachable from $\mathbf{m}_0$. For clarity, we shall sometimes write the PNI in subscript, e.g. $\mathbf{m}_1 \in [\mathbf{m}_0\rangle^L_N$.

A Petri net with *weak* inhibitor arcs (PNW for short) is a PNI $N = (S, T, F = \langle Z, I, O\rangle, \mathbf{m}_\iota)$ such that there is an index function $f\colon S \mapsto \mathbb{N}$ with the property:

$$\forall p, p' \in S\colon f(p) \leq f(p') \to (\forall t \in T\colon p' \in Z(t) \to p \in Z(t)) \ . \tag{1}$$

A Petri net (PN for short) can be seen as a subclass of Petri nets with *weak* inhibitor arcs where $Z(t) = \emptyset$ for all transitions $t \in T$. In this case, we shorten F as the pair $\langle I, O\rangle$.

The *reachability* problem for a PNI $N = (S, T, F = \langle Z, I, O\rangle, \mathbf{m}_\iota)$ is the problem of deciding, for a given marking $\mathbf{m}$, whether $\mathbf{m} \in [\mathbf{m}_\iota\rangle$ holds. It is well known that reachability for Petri nets with inhibitor arcs is undecidable [9]. However, the following holds:

THEOREM 3.*[15] The reachability problem for* PNW *is decidable.*

2.4 The reachability problem for Petri nets along finite-index CFL

Let us formally define the problem we are interested in. Given: (1) a Petri net $N = (S, T, F, \mathbf{m}_\iota)$ where $T \neq \emptyset$; (2) a CFG $G = (\mathcal{X}, T, \mathcal{P})$ and $A \in \mathcal{X}$; (3) a marking $\mathbf{m}_f \in \mathbb{M}[S]$; and (4) a value $k \in \mathbb{N}$.

Does $\mathbf{m}_f \in [\mathbf{m}_\iota\rangle^{L^{(k)}(A)}$ hold ?

In what follows, we prove the interreducibility of the reachability problem for PN along finite-index CFL and the reachability problem for PNW.

3 From PN reachability along fiCFL to PNW reachability

In this section, we show that the reachability problem for Petri nets along finite-index CFL is decidable. To this aim, let us fix an instance of the problem: a Petri net $N = (S, T, F, \mathbf{m}_\iota)$ where $T \neq \emptyset$, a CFG $G = (\mathcal{X}, T, \mathcal{P})$, $\mathbf{m}_f \in \mathbb{M}[S]$, and a natural number $k \in \mathbb{N}$. Moreover, let $G^{[k]} = (\mathcal{X}^{[k]}, T, \mathcal{P}^{[k]})$ be the CFG given by def. 1.

Lemma 2 shows that $\mathbf{m}_f \in [\mathbf{m}_\iota\rangle^{L^{(k+1)}(A)}$ if and only if $\mathbf{m}_f \in [\mathbf{m}_\iota\rangle^{L(A^{[k]})}$. Then, our decision procedure, which determines if $\mathbf{m}_f \in [\mathbf{m}_\iota\rangle^{L(A^{[k]})}$, proceeds by reduction to the reachability problem for PNW and is divided in two steps. First, we reduce the question $\mathbf{m}_f \in [\mathbf{m}_\iota\rangle^{L(A^{[k]})}$ to the existence of a successful execution in the program of Alg. 1 which, in turn, is reduced to a reachability problem for PNW. Let us describe Alg. 1.

Part 1. Alg. 1 gives the procedure *traverse* in which $\mathbf{M}_\mathsf{i}$ and $\mathbf{M}_\mathsf{f}$ are global arrays of markings with index ranging from 0 to k (i.e., for every $j \in \{0, \ldots, k\}$, $\mathbf{M}_\mathsf{i}[j], \mathbf{M}_\mathsf{f}[j] \in \mathbb{M}[S]$). We say that a call $traverse(X^{[\ell]})$ *successfully returns* if there exists an execution which eventually reaches line 19 (i.e., no assert fails) and the postcondition $\mathbf{M}_\mathsf{i}[j] = \mathbf{M}_\mathsf{f}[j] = \varnothing$ for every $j \in \{0, \ldots, \ell\}$ holds. Moreover we say that a call $traverse(X^{[\ell]})$ is *proper* if $\mathbf{M}_\mathsf{i}[j] = \mathbf{M}_\mathsf{f}[j] = \varnothing$ for all $j < \ell$. Let $\ell \in \{0, \ldots, k\}$, we shall now demonstrate that a proper call $traverse(X^{[\ell]})$ successfully returns if and only if there exists $w \in L(X^{[\ell]})$ such that $\mathbf{M}_\mathsf{i}[\ell]\ [w\rangle_N\ \mathbf{M}_\mathsf{f}[\ell]$.

Algorithm 1: *traverse*

Input: A variable $X^{[\ell]} \in \mathcal{X}^{[k]}$ of $G^{[k]}$
1 **begin**
2   Let $p \in \mathcal{P}^{[k]}$ such that $head(p) = X^{[\ell]}$
3   **switch** p **do**
4   **case** $X^{[\ell]} \to \sigma$ /* $\sigma \in \Sigma \cup \{\epsilon\}$ */
5       $\mathbf{M}_\mathsf{i}[\ell] := (\mathbf{M}_\mathsf{i}[\ell] \ominus I(\sigma)) \oplus O(\sigma)$
6       $sub\_*\_to(\mathbf{M}_\mathsf{i}[\ell], \mathbf{M}_\mathsf{f}[\ell])$
7   **case** $X^{[\ell]} \to B^{[\ell]} C^{[\ell-1]}$
8       $transfer\_from\_to(\mathbf{M}_\mathsf{f}[\ell], \mathbf{M}_\mathsf{f}[\ell-1])$
9       $add\_*\_to(\mathbf{M}_\mathsf{f}[\ell], \mathbf{M}_\mathsf{i}[\ell-1])$
10       $traverse(C^{[\ell-1]})$
11   assert $\mathbf{M}_\mathsf{i}[j] = \mathbf{M}_\mathsf{f}[j] = \varnothing$ for all $j < \ell$
12       $traverse(B^{[\ell]})$
13   **case** $X^{[\ell]} \to B^{[\ell-1]} C^{[\ell]}$
14       $transfer\_from\_to(\mathbf{M}_\mathsf{i}[\ell], \mathbf{M}_\mathsf{i}[\ell-1])$
15       $add\_*\_to(\mathbf{M}_\mathsf{i}[\ell], \mathbf{M}_\mathsf{f}[\ell-1])$
16       $traverse(B^{[\ell-1]})$
17   assert $\mathbf{M}_\mathsf{i}[j] = \mathbf{M}_\mathsf{f}[j] = \varnothing$ for all $j < \ell$
18       $traverse(C^{[\ell]})$
19   **return**

Algorithm 2: *add_∗_to, sub_∗_to*

Input: src_1, src_2
begin
  Let qty s.t. $\varnothing \preceq qty$
  if *add_∗_to* **then**
    $(src_1, src_2) := (src_1, src_2) \oplus qty$
  else// *sub_∗_to*
    $(src_1, src_2) := (src_1, src_2) \ominus qty$

Algorithm 3: *transfer_from_to*

Input: src, tgt
begin
  Let qty s.t.
  $\varnothing \preceq qty \preceq src$
  $tgt := tgt \oplus qty$
  $src := src \ominus qty$

The formal statement is given at Lem. 4. We give some intuitions about Alg. 1 first.

The control flow of *traverse* matches the traversal of a derivation tree of $G^{[k]}$ such that at each node *traverse* goes first to the subtree which carries the least index. The tree traversal is implemented through recursive calls in *traverse*. To see that the traversal goes first in the subtree of least index, it suffices to look at the ordering of the recursive calls to *traverse* in the code of Alg. 1, e.g. in case the of line 7, *traverse*$(C^{[\ell-1]})$ is called before *traverse*$(B^{[\ell]})$.

Reasoning in terms of derivation trees, we have that the proper call *traverse*$(X^{[\ell]})$ returns iff there exists a derivation tree t of $G^{[k]}$ with root variable $X^{[\ell]}$ such that the sequence of transitions given by the yield of t is enabled from the marking stored in $\mathbf{M}_{\mathsf{i}}[\ell]$ and its firing yields the marking stored in $\mathbf{M}_{\mathsf{f}}[\ell]$.

Because of the least index first traversal, it turns out that the arrays $\mathbf{M}_{\mathsf{i}}$ and $\mathbf{M}_{\mathsf{f}}$ provide enough space to manage all the intermediary results.

Also, we observe that when the procedure *traverse*$(X^{[\ell]})$ calls itself with the parameter, say $B^{[\ell]}$, the call is a *tail recursive call*. This means that when *traverse*$(B^{[\ell]})$ returns then *traverse*$(X^{[\ell]})$ immediately returns. It is known from programming techniques how to implement tail recursive call without consuming space on the call stack. In the case of Alg. 1, we can do so by having a global variable to store the parameter of *traverse* and by replacing tail recursive calls with **goto** statements. For the remaining recursive calls (line 10 and 16), because the index of the callee is one less than the index of the caller, we conclude that a bounded space consisting of k frames suffices for the call stack.

Those two insights (two arrays with k entries and a stack with k frames) will be the key to show, in Part 2, that *traverse* can be implemented as a PNW.

LEMMA 4. *Let* $\ell \in \{0,\ldots,k\}$*,* $X^{[\ell]} \in \mathcal{X}^{[k]}$*, and* $\mathbf{m},\mathbf{m}' \in \mathbb{M}[S]$*. Then, the proper call traverse*$(X^{[\ell]})$ *with context* $\mathbf{M}_{\mathsf{i}}[\ell] = \mathbf{m}$ *and* $\mathbf{M}_{\mathsf{f}}[\ell] = \mathbf{m}'$ *successfully returns if and only if there exists* $w \in L(X^{[\ell]})$ *such that* $\mathbf{m}\,[w\rangle_N\,\mathbf{m}'$*.*

PROOF. **If.** We prove that if there exists $w \in L(X^{[\ell]})$ such that $\mathbf{m}\,[w\rangle\,\mathbf{m}'$ then the proper call *traverse*$(X^{[\ell]})$ with $\mathbf{M}_{\mathsf{i}}[\ell] = \mathbf{m}$ and $\mathbf{M}_{\mathsf{f}}[\ell] = \mathbf{m}'$ successfully returns.

Our proof is done by induction on the length n of the derivation of $w \in L(X^{[\ell]})$. For the case $n = 1$, we necessarily have $X^{[\ell]} \Rightarrow w = \sigma$ for some $(X^{[\ell]},\sigma) \in \mathcal{P}^{[k]}$. In this case, the proper call *traverse*$(X^{[\ell]})$ with $\mathbf{M}_{\mathsf{i}}[\ell] = \mathbf{m}$ and $\mathbf{M}_{\mathsf{f}}[\ell] = \mathbf{m}'$ executes as follows: $p = (X^{[\ell]},\sigma)$ is picked and the case of line 4 executes successfully since $\mathbf{m} = \mathbf{M}_{\mathsf{i}}[\ell]\,[\sigma\rangle\,\mathbf{M}_{\mathsf{f}}[\ell] = \mathbf{m}'$ holds. In fact, after the assignment of line 5 we have $\mathbf{M}_{\mathsf{i}}[\ell] = \mathbf{M}_{\mathsf{f}}[\ell]$. From there, the call to *sub_∗_to* can return with $\mathbf{M}_{\mathsf{i}}[\ell] = \mathbf{M}_{\mathsf{f}}[\ell] = \varnothing$ which shows that *traverse*$(X^{[\ell]})$ successfully returns.

For the case $n > 1$, we have $X^{[\ell]} \Rightarrow^n w$ which necessarily has the form $X^{[\ell]} \Rightarrow B^{[\ell]}C^{[\ell-1]} \Rightarrow^{n-1} w$ or $X^{[\ell]} \Rightarrow B^{[\ell-1]}C^{[\ell]} \Rightarrow^{n-1} w$ by def. of $G^{[k]}$. Assume we are in the latter case. Thus there exists w_1 and w_2 such that $X^{[\ell]} \Rightarrow B^{[\ell-1]}C^{[\ell]} \Rightarrow^i w_1C^{[\ell]} \Rightarrow^j w_1w_2 = w$ with $i+j = n-1$ and $\exists \mathbf{m}_1 \colon \mathbf{m}\,[w_1\rangle\,\mathbf{m}_1\,[w_2\rangle\,\mathbf{m}'$. Observe that $w_1 \in L(B^{[\ell-1]})$ and $w_2 \in L(C^{[\ell]})$ and so by induction hypothesis we find that the proper call *traverse*$(B^{[\ell-1]})$ with $\mathbf{M}_{\mathsf{i}}[\ell-1] = \mathbf{m}$, $\mathbf{M}_{\mathsf{f}}[\ell-1] = \mathbf{m}_1$ successfully returns. And so does, by induction hypothesis, the proper call *traverse*$(C^{[\ell]})$ with $\mathbf{M}_{\mathsf{i}}[\ell] = \mathbf{m}_1$, $\mathbf{M}_{\mathsf{f}}[\ell] = \mathbf{m}'$. Therefore let us consider the proper call *traverse*$(X^{[\ell]})$ with $\mathbf{M}_{\mathsf{i}}[\ell] = \mathbf{m}$, $\mathbf{M}_{\mathsf{f}}[\ell] = \mathbf{m}'$. We show it successfully returns.

First observe that the call to the procedure *traverse*$(X^{[\ell]})$ is proper. Next, at line 2, pick $p = (X^{[\ell]}, B^{[\ell-1]}C^{[\ell]})$. Then the call *transfer_from_to*$(\mathbf{M}_{\mathsf{i}}[\ell], \mathbf{M}_{\mathsf{i}}[\ell-1])$ of line 14 executes such that $\mathbf{M}_{\mathsf{i}}[\ell]$ is updated to $\varnothing$ and $\mathbf{M}_{\mathsf{i}}[\ell-1]$ to $\mathbf{m}$. Next the call to the procedure *add_∗_to*$(\mathbf{M}_{\mathsf{i}}[\ell], \mathbf{M}_{\mathsf{f}}[\ell-1])$ of line 15 executes such that both $\mathbf{M}_{\mathsf{i}}[\ell]$ and $\mathbf{M}_{\mathsf{f}}[\ell-1]$ are updated to $\mathbf{m}_1$. Recall that $\mathbf{m}\,[w_1\rangle\,\mathbf{m}_1\,[w_2\rangle\,\mathbf{m}'$.

Finally we showed above that the proper call *traverse*$(B^{[\ell-1]})$ successfully returns, the assert that follows too and finally the proper call *traverse*$(C^{[\ell]})$. Moreover it is routine to check that upon completion of *traverse*$(C^{[\ell]})$ (and therefore *traverse*$(X^{[\ell]})$) we have $\mathbf{M}_{\mathsf{i}}[j] = \mathbf{M}_{\mathsf{f}}[j] = \varnothing$ for all $j \leq \ell$.

The left case (i.e. $p = (X^{[\ell]}, B^{[\ell]}C^{[\ell-1]}) \in \mathcal{P}^{[k]}$) is treated similarly.

Only If. Here we prove that if the proper call *traverse*$(X^{[\ell]})$ successfully returns then there exists $w \in L(X^{[\ell]})$ such that $\mathbf{M}_{\mathsf{i}}[\ell]\, [w\rangle_N\, \mathbf{M}_{\mathsf{f}}[\ell]$.

Our proof is done by induction on the number n of times line 2 is executed during the execution of *traverse*$(X^{[\ell]})$. In every case, line 2 is executed at least once. For the case $n = 1$, the algorithm necessarily executes the case of line 4. The definition of $G^{[k]}$ shows that along a successful execution of *traverse*$(X^{[\ell]})$, the non deterministic choice of line 2 necessarily returns a production of the form $p = (X^{[\ell]}, \sigma) \in \mathcal{P}^{[k]}$. Therefore, a successful execution must execute line 5 and 6 and then 19 after which the postcondition $\mathbf{M}_{\mathsf{i}}[j] = \mathbf{M}_{\mathsf{f}}[j] = \varnothing$ for all $j \leq \ell$ holds. Because the postcondition holds, we find that $\mathbf{M}_{\mathsf{i}}[\ell] = \mathbf{M}_{\mathsf{f}}[\ell]$ holds before executing line 6, hence that $\mathbf{M}_{\mathsf{f}}[\ell] = \mathbf{M}_{\mathsf{i}}[\ell] \ominus I(\sigma) \oplus O(\sigma)$ before executing line 5, and finally that $\mathbf{M}_{\mathsf{i}}[\ell]\, [\sigma\rangle\, \mathbf{M}_{\mathsf{f}}[\ell]$ by semantics of transition σ and we are done.

For the case $n > 1$, the first non deterministic choice of line 2 necessarily picks $p \in \mathcal{P}^{[k]}$ of the form $(X^{[\ell]}, B^{[\ell]}C^{[\ell-1]})$ or $(X^{[\ell]}, B^{[\ell-1]}C^{[\ell]})$. Let us assume $p = (X^{[\ell]}, B^{[\ell]}C^{[\ell-1]})$, hence that the case of line 7 is executed. Let $\mathbf{m}$ and $\mathbf{m}'$ be respectively the values of $\mathbf{M}_{\mathsf{i}}[\ell]$ and $\mathbf{M}_{\mathsf{f}}[\ell]$ when *traverse*$(X^{[\ell]})$ is invoked. Now, let $\mathbf{m}_3, \mathbf{m}_\Delta$ be such that $\mathbf{m}' = \mathbf{m}_3 \oplus \mathbf{m}_\Delta$ and such that upon completion of the call to *transfer_from_to* at line 8 we have that $\mathbf{M}_{\mathsf{f}}[\ell] = \mathbf{m}_\Delta$ and $\mathbf{M}_{\mathsf{f}}[\ell-1] = \mathbf{m}_3$. Moreover, let $\mathbf{m}_2$ be the marking such that $\mathbf{M}_{\mathsf{i}}[\ell-1] = \mathbf{m}_2$ upon completion of the call to *add_∗_to* at line 9. Therefore we find that $\mathbf{M}_{\mathsf{f}}[\ell]$ is updated to $\mathbf{m}_\Delta \oplus \mathbf{m}_2$. Next consider the successful proper call *traverse*$(C^{[\ell-1]})$ of line 10 with $\mathbf{M}_{\mathsf{i}}[\ell-1] = \mathbf{m}_2$, $\mathbf{M}_{\mathsf{f}}[\ell-1] = \mathbf{m}_3$. Observe that because the execution of *traverse*$(X^{[\ell]})$ yields the calls *traverse*$(C^{[\ell-1]})$ and *traverse*$(B^{[\ell]})$, we find that the number of times line 2 is executed in *traverse*$(C^{[\ell-1]})$ and *traverse*$(B^{[\ell]})$ is strictly less than n. Therefore, the induction hypothesis shows that there exists w_2 such that $w_2 \in L(C^{[\ell-1]})$ and $\mathbf{m}_2\, [w_2\rangle\, \mathbf{m}_3$. Then comes the successful assert of line 11 followed by the successful proper call *traverse*$(B^{[\ell]})$ of line 12 with $\mathbf{M}_{\mathsf{i}}[\ell] = \mathbf{m}$ and $\mathbf{M}_{\mathsf{f}}[\ell] = \mathbf{m}_\Delta \oplus \mathbf{m}_2$. Again by induction hypothesis, there exists w_1 such that $w_1 \in L(B^{[\ell]})$ and $\mathbf{m}\, [w_1\rangle\, (\mathbf{m}_\Delta \oplus \mathbf{m}_2)$.

Next we conclude from the monotonicity property of PN that since $\mathbf{m}_2\, [w_2\rangle\, \mathbf{m}_3$ then $(\mathbf{m}_2 \oplus \mathbf{m}_\Delta)\, [w_2\rangle\, (\mathbf{m}_3 \oplus \mathbf{m}_\Delta)$, hence that $\mathbf{m}\, [w_1\rangle\, (\mathbf{m}_2 \oplus \mathbf{m}_\Delta)\, [w_2\rangle\, (\mathbf{m}_3 \oplus \mathbf{m}_\Delta)$ and finally that $\mathbf{m}\, [w_1\, w_2\rangle\, \mathbf{m}'$ because $\mathbf{m}' = \mathbf{m}_3 \oplus \mathbf{m}_\Delta$. Finally since $w_1 w_2 \in L(X^{[\ell]})$ we conclude that $\mathbf{m}' \in [\mathbf{m}\rangle^{L(X^{[\ell]})}$ and we are done.

The left case (i.e. $p = (X^{[\ell]}, B^{[\ell-1]}C^{[\ell]}) \in \mathcal{P}^{[k]}$) is treated similarly. ∎

Part 2. In this section, we show that it is possible to construct a PNI N' such that the problem asking if the call to *traverse*$(A^{[k]})$ successfully returns can be reduced, in polynomial time, to a reachability problem for N'. Incidentally, we show that N' is a PNW, hence that the reachability problem for PN along finite-index CFL is decidable.

To describe N' we use a generalization of the net program formalism introduced by Esparza in [3] which enrich the instruction set with the test for 0 of a variable.

A *net program* is a finite sequence of *labelled commands* separated by semicolons. Basic commands have the following form, where $\ell, \ell', \ell_1, \ldots, \ell_k$ are *labels* taken from some arbitrary set, and x is a variable over the natural numbers, also called a *counter*.

$\ell\colon x := x - 1$	$\ell\colon$ **if** $x = 0$ **then goto** ℓ'	$\ell\colon$ **return**
$\ell\colon x := x + 1$	$\ell\colon$ **goto** ℓ_1 **or** $\cdots$ **or goto** ℓ_k	$\ell\colon$ **halt**
$\ell\colon$ **goto** ℓ'	$\ell\colon$ **gosub** ℓ'	

A net program is *syntactically correct* if the labels of commands are pairwise different, and if the destinations of jumps corresponds to existing labels. Moreover we require the net program to be decomposable into a main program that only calls first-level *subroutines*, which in turn only call second level *subroutines*, etc and the jump commands in a subroutine can only have commands of the same subroutine as destinations.‡ Each subroutine has a unique *entry command* labelled with a subroutine name, and a unique *exit command* of the form $\ell\colon$ **return**. Entry and exit labelled commands are distinct.

A net program can only be executed once its variables have received initial values. In this paper we assume that the initial values are always 0. The semantics of net programs is that suggested by the syntax.

The compilation of a syntactically correct net program to a PNI is straightforward and omitted due to space constraints. See [3] for the compilation.

At Alg. 4 is the net program that implements Alg. 1. In what follows assume S, the set of places of the underlying Petri net, to be $\{1, \ldots, d\}$ for $d \geq 1$. The counter variables of the net program are given by $\{x^{[i]}\}_{0 \leq i \leq k, X \in \mathcal{X}}$ and $\mathbf{M}_\mathsf{f}[0..k][1..d]$ $\mathbf{M}_\mathsf{i}[0..k][1..d]$ which arranges counters into two matrices of dimension $(k+1) \times d$. For clarity, our net programs use some abbreviations whose semantics is clear from the syntax, e.g. $\mathbf{M}_\mathsf{i}[\ell] := \mathbf{M}_\mathsf{i}[\ell] \oplus \mathbf{m}$ stands for the sequence $\mathbf{M}_\mathsf{i}[\ell][1] := \mathbf{M}_\mathsf{i}[\ell][1] + \mathbf{m}(1); [\ldots]; \mathbf{M}_\mathsf{i}[\ell][d] := \mathbf{M}_\mathsf{i}[\ell][d] + \mathbf{m}(d)$.

Let us now make a few observations of Alg. 4:

• at the top level we have the subroutine **main** which first sets up $\mathbf{M}_\mathsf{i}[\ell]$ and $\mathbf{M}_\mathsf{f}[\ell]$, then simulates the call *traverse*$(X^{[\ell]})$ and finally checks that the postcondition holds (label $\mathbf{0}_1$) before halting (label **success**).

• the counter variables $\{x^{[i]}\}_{0 \leq i \leq k, X \in \mathcal{X}}$ defines the parameter of the calls to $\textbf{traverse}_j$. For instance, a call to *traverse*$(X^{[j]})$ is simulated in the net program by incrementing $x^{[j]}$ and then calling subroutine $\textbf{traverse}_j$.

‡Here we consider the main program as a zero-level subroutine, i.e. jump commands in the main program can only have commands of the main program as destinations.

Algorithm 4: **main** invoking $traverse(X^{[\ell]})$ with $\mathbf{m}$, $\mathbf{m}'$ and subroutines $\mathbf{traverse}_j$ where $0 < j \leq \ell$ implementing the calls $\left\{traverse(X^{[j]})\right\}_{X \in \mathcal{X}}$.

main: $\mathbf{M}_\mathsf{i}[\ell] := \mathbf{M}_\mathsf{i}[\ell] \oplus \mathbf{m}$;
$\mathbf{M}_\mathsf{f}[\ell] := \mathbf{M}_\mathsf{f}[\ell] \oplus \mathbf{m}'$;
$x^{[\ell]} := x^{[\ell]} + 1$;
gosub traverse$_\ell$;
$\mathbf{0}_1$ **if** $\mathbf{M}_\mathsf{i}[0..\ell] = \varnothing = \mathbf{M}_\mathsf{f}[0..\ell]$ **then**
  goto success;
traverse$_j$: **goto** $\mathbf{p}_1$ **or** $\cdots$ **or goto** $\mathbf{p}_n$;
[...];
$\mathbf{p}_{i_0}$: $x^{[j]} := x^{[j]} - 1$;
$\mathbf{M}_\mathsf{i}[j] := \mathbf{M}_\mathsf{i}[j] \ominus I(\sigma)$;
$\mathbf{M}_\mathsf{i}[j] := \mathbf{M}_\mathsf{i}[j] \oplus O(\sigma)$;
gosub sub_to$_j$;
goto exit;
[...];
$\mathbf{p}_{i_1}$: $x^{[j]} := x^{[j]} - 1$;
gosub tr_f$_j$**_f**$_{(j-1)}$;
gosub add_to_i$_{(j-1)}$**_f**$_j$;
$c^{[j-1]} := c^{[j-1]} + 1$;
gosub traverse$_{(j-1)}$;
$\mathbf{0}_2$ **if** $\mathbf{M}_\mathsf{i}[0..j-1] = \varnothing = \mathbf{M}_\mathsf{f}[0..j-1]$ **then**
  goto l1;
l1: $b^{[j]} := b^{[j]} + 1$;
goto traverse$_j$;
[...];
$\mathbf{p}_{i_2}$: $v^{[j]} := v^{[j]} - 1$;
gosub tr_i$_j$**_i**$_{(j-1)}$;
gosub add_to_i$_j$**_f**$_{(j-1)}$;
$y^{[j-1]} := y^{[j-1]} + 1$;
gosub traverse$_{(j-1)}$;
$\mathbf{0}_3$ **if** $\mathbf{M}_\mathsf{i}[0..j-1] = \varnothing = \mathbf{M}_\mathsf{f}[0..j-1]$ **then**
  goto l2;
l2: $z^{[j]} := z^{[j]} + 1$;
goto traverse$_j$;
[...];
exit: **return**;
[...];
success: **halt**;

• the non-deterministic jump at label $\mathbf{traverse}_j$ simulates the selection of a production rule $\mathbf{p}_{i_k} = (X^{[j]}, w)$ which will be fired next (if enabled else the program fails).
• the missing code for the subroutines **tr_f**$_j$**_f**$_{j-1}$, **add_to_i**$_j$**_f**$_{j-1}$, and **sub_to**$_j$ can be found in the appendix although it is pretty obvious to infer from Alg. 2 and Alg. 3. The code for **tr_i**$_j$**_i**$_{j-1}$, **add_to_f**$_j$**_i**$_{j-1}$ and $\mathbf{traverse}_0$ is also routine to write.
• the program is syntactically correct. First, the levels are assigned to subroutines as follows: the level of $\mathbf{traverse}_j$ is j, the level of **tr_f**$_j$**_f**$_{j-1}$, **tr_i**$_j$**_i**$_{j-1}$, **add_to_i**$_j$**_f**$_{j-1}$, **add_to_f**$_j$**_i**$_{j-1}$ and **sub_to**$_j$ is $j-1$. Given that level assignment, it is routine to check that subroutines of level i only call subroutines of level $i-1$. Moreover, thanks to the programming techniques that allow to implement the tail recursive call as a **goto** instead of **gosub** we find that the program is synctactically correct. (If we had used **gosub** everywhere, then the net program would be synctactically incorrect). Also observe that each jump commands does not leave the subroutine inside which it is invoked.
• the tests for 0 (labels $\mathbf{0}_1, \mathbf{0}_2, \mathbf{0}_3$) have a particular structure matching the level of the subroutines (level 0 for $\mathbf{0}_1$ and j for $\mathbf{0}_2$ and $\mathbf{0}_3$). So, after compilation of the net program into a PNI N', if we set a mapping f from the places of N' to $\mathbb{N}$ such that c is mapped to i if $c \in \{\mathbf{M}_\mathsf{i}[i][j] \mid j \in \{1, \ldots, d\}\} \cup \{\mathbf{M}_\mathsf{f}[i][j] \mid j \in \{1, \ldots, d\}\}$ and every other place is mapped to $\ell + 2$ then we find that N' is a PNW. Clearly, deciding whether Alg. 4 halts reduces to PNW reachability. Therefore, by Thm. 3, it is decidable whether Alg. 4 halts.

LEMMA 5. *Let $\ell \in \{0, \ldots, k\}$, $X^{[\ell]} \in \mathcal{X}^{[k]}$, and $\mathbf{m}, \mathbf{m}' \in \mathbb{M}[S]$. Then the proper call $traverse(X^{[\ell]})$ with $\mathbf{M}_\mathsf{i}[\ell] = \mathbf{m}$, $\mathbf{M}_\mathsf{f}[\ell] = \mathbf{m}'$ successfully returns iff Alg. 4 halts.*

Hence from Lem. 2, 4 and 5, we conclude the following.

COROLLARY 6. *The reachability problem for* PN *along finite-index* CFL *can be reduced to the reachability problem for* PNW.

4 From PNW reachability to PN reachability along fiCFL

In this section, we show that the reachability problem for PNW can be reduced to the reachability problem of PN along finite-index CFL. To this aim, let $N = (S, T, F = \langle Z, I, O\rangle, \mathbf{m}_\iota)$ be a PNW, $\mathbf{m}_f \in \mathbb{M}[S]$ a marking, and $f\colon S \mapsto \mathbb{N}$ an index function such that (1) holds.

Let $S = \{s_1, \ldots, s_{n+1}\}$ and $T = \{t_1, \ldots, t_m\}$. Because it simplifies the presentation we will make a few assumptions that yield no loss of generality. (*i*) For every $i \in \{1, \ldots, n\}$, we have $f(s_i) \leq f(s_{i+1})$, (*ii*) $\mathbf{m}_\iota = [\![s_{n+1}]\!]$, $\mathbf{m}_f = \varnothing$, (*iii*) $Z(t_1) \subseteq Z(t_2) \subseteq \cdots \subseteq Z(t_m) \subseteq \{s_1, \ldots, s_n\}$, and (*iv*) for every $t \in T$, if $s \in Z(t)$ then $O(t)(s) = 0$ (see [15], Lemma 2.1). Notice that the Petri net N can not test if the place s_{n+1} is empty or not.

In the following, we show that it is possible to construct a Petri net (without inhibitor arcs) N', a marking $\mathbf{m}'_f$, and a finite-index CFL L such that: $\mathbf{m}_f \in [\mathbf{m}_\iota\rangle_N^{T^*}$ iff $\mathbf{m}'_f \in [\mathbf{m}'_\iota\rangle_{N'}^{L}$.

Constructing the Petri net N'**:** Let $N' = (S', T', F' = \langle I', O'\rangle, \mathbf{m}'_\iota)$ be a PN which consists in $n+1$ unconnected PN widget: the widget N_0 given by N without tests for zero (i.e. $Z(t)$ is set to $\varnothing$ for every $t \in T$) and the widgets $N_1, \ldots, N_n$ where each $N_i = (\{r_i\}, \{p_i, c_i\}, F_i, \varnothing)$ where $F_i(p_i) = \langle \varnothing, [\![r_i]\!]\rangle$ and $F_i(c_i) = \langle [\![r_i]\!], \varnothing\rangle$. N_i is depicted as follows: $\overset{p_i}{\blacksquare} \rightarrow \overset{r_i}{\bigcirc} \rightarrow \overset{c_i}{\blacksquare}$. Finally, define $\mathbf{m}'_\iota \in \mathbb{M}[S']$ to be $\mathbf{m}'_\iota(s) = \mathbf{m}_\iota(s)$ for $s \in S$ and 0 elsewhere; and $\mathbf{m}'_f = \varnothing$.

Since we have the ability to restrict the possible sequences of transitions that fire in N', we can enforce the invariant that the sum of tokens in s_i and r_i stays constant. To do so it suffices to force that whenever a token produced in s_i then a token is consumed from r_i and vice versa. Call L the language enforcing that invariant. Then, let $\mathbf{m}$ be a marking such that $\mathbf{m}(s_i) = \mathbf{m}(r_i) = 0$, observe that by firing from $\mathbf{m}$ a sequence of the form: (*i*) p_i repeated n times, (*ii*) any sequence $w \in L$ and (*iii*) c_i repeated n times; the marking $\mathbf{m}'$ that is reached is such that $\mathbf{m}'(s_i) = \mathbf{m}'(r_i) = 0$. This suggests that to simulate faithfully a transition t_0 of N that does test s_i for 0 we allow the occurrence of the counterpart of t_0 in N_0 right before (*i*) or right after (*iii*) only. In what follows, we build upon the above idea the language L_n which, as we we will show, coincides with the finite-index approximation of some CFG.

We need the following notation. Given a word $v \in \Sigma^*$ and $\Theta \subseteq \Sigma$, we define $v|_\Theta$ to be the word obtained from v by erasing all the symbols that are not in Θ. We extend it to languages as follows: Let $L \subseteq \Sigma^*$. Then $L|_\Theta = \{u|_\Theta \mid u \in L\}$.

Constructing the language L_n**:** For every $j \in \{1, \ldots, m\}$, let $u_j = p_1^{i_1} p_2^{i_2} \cdots p_n^{i_n}$ and $v_j = c_1^{k_1} c_2^{k_2} \cdots c_n^{k_n}$ be two words over the alphabet T' such that $i_\ell = I(t_j)(s_\ell)$ and $k_\ell = O(t_j)(s_\ell)$ for all $\ell \in \{1, \ldots, n\}$. Observe that firing $v_j t_j u_j$ keeps unchanged the total number of tokens in $\{s_i, r_i\}$ for each $i \in \{1, \ldots, n\}$. Let $\ell \in \{0, \ldots, n\}$ define $T_\ell = \{v_j \cdot t_j \cdot u_j \mid Z(t_j) = \{s_1, \ldots, s_\ell\}\}$.[§] Also given $a, b \in \Sigma^*$ and $Z \subseteq \Sigma^*$, define $\langle a, b\rangle \star Z$ as the set $\{a^i \cdot z \cdot b^i \mid i \in \mathbb{N} \wedge z \in Z\}$.

Define the CFLs $L_0, \ldots, L_n$ inductively as follows: $L_0 = T_0^*$ and for $0 < \ell \leq n$ define $L_\ell = \big((\langle p_\ell, c_\ell\rangle \star L_{\ell-1}) \cup T_\ell\big)^*$. It is routine to check that $L_0 \subseteq L_1 \subseteq \cdots \subseteq L_n$ (since $L_{\ell-1} \subseteq \langle p_\ell, c_\ell\rangle \star L_{\ell-1})$) and $L_n|_T = T^*$ (since $L_n \supseteq \bigcup_{i=0}^n T_i$). Also, L_0 is a regular language and therefore there exists a CFG G_0 and a variable A_0 of G_0 such that $L^{(1)}(A_0) = L_0$. Now, let us assume that for L_i there exists a CFG G_i and a variable A_i such that $L^{(i+1)}(A_i) = L_i$. From

[§]Note that if $\ell = 0$ then $\{s_1, \ldots, s_\ell\} = \varnothing$.

the definition of L_{i+1} it is routine to check that there exists a CFG G_{i+1} and a variable A_{i+1} such that $L^{(i+2)}(A_{i+1}) = L_{i+1}$. Finally we find that L_n can be captured by the $n+1$-index approximation of a CFG.

LEMMA 7. *Let $\ell \in \{0,\dots,n\}$. If $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{M}[S']$ such that $\mathbf{m}_2 \in [\mathbf{m}_1\rangle_{N'}^{L_\ell}$, then $\mathbf{m}_2(s_j) + \mathbf{m}_2(r_j) = \mathbf{m}_1(s_j) + \mathbf{m}_1(r_j)$ for all $j \in \{1,\dots,n\}$.*

Let us make a few observations about the transitions of N' which were carrying out 0 test in N. In L_ℓ no transition t such that $s_{\ell+1} \in Z(t)$ is allowed, that is no test of place $s_{\ell+1}$ for 0 is allowed along any word of L_ℓ. The language L_ℓ imposes that the place s_ℓ can only be tested for 0 along T_ℓ. The intuition is that L_ℓ allows to test s_ℓ for 0 provided all places s_j and r_j for $j \le \ell$ are empty.

Let us introduce the following notations. Let $\mathbf{m} \in \mathbb{M}[S']$ and $Q \subseteq S'$, we write $Q(\mathbf{m})$ for the multiset of $\mathbb{M}[Q]$ such that $Q(\mathbf{m})(q) = \mathbf{m}(q)$ for all $q \in Q$. We define the following subsets of places of N': R_ℓ (resp. S_ℓ) is given by $\{r_1,\dots,r_\ell\}$ (resp. $\{s_1,\dots,s_\ell\}$). The proofs of lemmata that follow are done by induction and given in the appendix.

LEMMA 8. *Let $\ell \in \{0,\dots,n\}$, $w \in L_\ell$, and $\mathbf{m}_a, \mathbf{m}_b \in \mathbb{M}[S']$ such that $(S_\ell \cup R_\ell)(\mathbf{m}_a) = (S_\ell \cup R_\ell)(\mathbf{m}_b) = \varnothing$ and $\mathbf{m}_a\,[w\rangle_{N'}\,\mathbf{m}_b$. Then $S(\mathbf{m}_a)\,[w|_T\rangle_N\,S(\mathbf{m}_b)$.*

LEMMA 9. *Let $\ell \in \{0,\dots,n\}$, $\mu_1, \mu_2 \in \mathbb{M}[S]$ such that $S_\ell(\mu_1) = S_\ell(\mu_2) = \varnothing$ and $\mu_2 \in [\mu_1\rangle_N^{L_\ell|_T}$. Then there are $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{M}[S']$ such that $S(\mathbf{m}_1) = \mu_1$, $S(\mathbf{m}_2) = \mu_2$, $R_\ell(\mathbf{m}_1) = R_\ell(\mathbf{m}_2) = \varnothing$, and $\mathbf{m}_2 \in [\mathbf{m}_1\rangle_{N'}^{L_\ell}$.*

LEMMA 10. *$\mathbf{m}_f(=\varnothing) \in [\mathbf{m}_\imath\rangle_N$ if and only if $\mathbf{m}'_f(=\varnothing) \in [\mathbf{m}'_\imath\rangle_{N'}^{L_n}$.*

PROOF. ($\Rightarrow$) Assume that $\mathbf{m}_f \in [\mathbf{m}_\imath\rangle_N$. Since $L_n|_T = T^*$ and $S_n(\mathbf{m}_\imath) = S_n(\mathbf{m}_f) = \varnothing$, the result of Lem. 9 shows that there are $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{M}[S']$ such that $S(\mathbf{m}_1) = \mathbf{m}_\imath$, $S(\mathbf{m}_2) = \mathbf{m}_f$, $R_n(\mathbf{m}_1) = R_n(\mathbf{m}_2) = \varnothing$, and $\mathbf{m}_2 \in [\mathbf{m}_1\rangle_{N'}^{L_n}$. This implies that $\mathbf{m}'_f \in [\mathbf{m}'_\imath\rangle_{N'}^{L_n}$ since $\mathbf{m}'_f = \mathbf{m}_2$ and $\mathbf{m}'_\imath = \mathbf{m}_1$ by definition.

($\Leftarrow$) Assume that $\mathbf{m}'_f \in [\mathbf{m}'_\imath\rangle_{N'}^{L_n}$. The definition of $\mathbf{m}'_\imath$ and $\mathbf{m}'_f$ shows that $(S_n \cup R_n)(\mathbf{m}'_\imath) = (S_n \cup R_n)(\mathbf{m}'_f) = \varnothing$ and therefore, by Lem. 8, we find that $S(\mathbf{m}'_f) \in [S(\mathbf{m}'_\imath)\rangle_N^{L_n|T}$, hence that $\mathbf{m}_f \in [\mathbf{m}_\imath\rangle_N^{L_n|T}$ by definition of $\mathbf{m}_\imath, \mathbf{m}_f$, and finally that $\mathbf{m}_f \in [\mathbf{m}_\imath\rangle_N$ since $L_n|T = T^*$. ∎

As an immediate consequence of Lemma 10, we obtain the following result:

COROLLARY 11. *The reachability problem for* PNW *can be reduced to the reachability problem for* PN *along finite-index* CFL.

5 Conclusion

In this paper, we have defined the class finite-index context-free languages (which is an interesting sub-class of context-free languages). We have shown that the problem of checking whether the intersection of a finite-index context-free language and a Petri net language is empty is decidable. This result is obtained through a non-trivial reduction to the reachability problem for Petri nets with weak inhibitor arcs. On the other hand, we have proved that the reachability problem for Petri nets with weak inhibitor arcs can be reduced to the the emptiness problem of the language obtained from the intersection of a finite-index context-free language and a Petri net language, which implies by [13] that the latter is EXPSPACE-hard.

References

[1] R. Bonnet. The reachability problem for vector addition systems with one zero-test. In *MFCS '11: Proc. 36th Int. Symp. on Mathematical Foundations of Computer Science*, LNCS. Springer, 2011. To appear.

[2] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL '77*, pages 238–252. ACM Press, 1977.

[3] J. Esparza. Decidability and complexity of petri net problems – an introduction. In *Lectures on Petri Nets I: Basic Models*, volume 1491 of *LNCS*, pages 374–428. Springer, 1998.

[4] J. Esparza, P. Ganty, S. Kiefer, and M. Luttenberger. Parikh's theorem: A simple and direct automaton construction. *Information Processing Letters*, 111:614–619, 2011.

[5] J. Esparza, S. Kiefer, and M. Luttenberger. Newton's method for ω-continuous semirings. In *ICALP '08*, volume 5126 of *LNCS*, pages 14–26. Springer, 2008. Invited paper.

[6] J. Esparza, S. Kiefer, and M. Luttenberger. Newtonian program analysis. *Journal of the ACM*, 57(6):33:1–33:47, 2010.

[7] P. Ganty, B. Monmege, and R. Majumdar. Bounded underapproximations. In *CAV '10*, volume 6174 of *LNCS*, pages 600–614. Springer, 2010.

[8] S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In *CAV '97*, volume 1254 of *LNCS*, pages 72–83. Springer, 1997.

[9] M. H. T. Hack. Decidability questions for petri nets. Technical Report 161, MIT, 1976.

[10] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, third edition, July 2006.

[11] M. Lange and H. Leiß. To CNF or not to CNF ? An efficient yet presentable version of the CYK algorithm. *Informatica Didactica*, 8, 2008-2010.

[12] J. Leroux. Vector addition system reachability problem (a short self-contained proof). In *POPL '11*, pages 307–316. ACM, 2011.

[13] R. Lipton. The reachability problem is exponential-space hard. Technical Report 62, Department of Computer Science, Yale University, Jan. 1976.

[14] M. Luker. A family of languages having only finite-index grammars. *Information and Control*, 39(1):14–18, 1978.

[15] K. Reinhardt. Reachability in petri nets with inhibitor arcs. *Electr. Notes Theor. Comput. Sci*, 223:239 – 264, 2008. RP '08.

[16] A. Salomaa. On the index of a context-free grammar and language. *Information and Control*, 14(5):474 – 477, 1969.

A Missing Net programs

Alg. 5 gives the net program which implements the call $sub\_*\_to(\mathbf{M}_{\mathsf{i}}[\ell], \mathbf{M}_{\mathsf{f}}[\ell])$.

Algorithm 5:

$\mathbf{sub\_to}_\ell$ **goto exit or** s_1 **or** ... **or** s_d ;
s_1: $\mathbf{M}_{\mathsf{i}}[\ell][1] := \mathbf{M}_{\mathsf{i}}[\ell][1] - 1$;
$\mathbf{M}_{\mathsf{f}}[\ell][1] := \mathbf{M}_{\mathsf{f}}[\ell][1] - 1$;
goto $\mathbf{sub\_to}_\ell$;
[...];
s_d: $\mathbf{M}_{\mathsf{i}}[\ell][d] := \mathbf{M}_{\mathsf{i}}[\ell][d] - 1$;
$\mathbf{M}_{\mathsf{f}}[\ell][d] := \mathbf{M}_{\mathsf{f}}[\ell][d] - 1$;
goto $\mathbf{sub\_to}_\ell$;
exit: **return**;

Alg. 6 implements the call $add\_*\_to(\mathbf{M}_{\mathsf{i}}[\ell], \mathbf{M}_{\mathsf{f}}[\ell-1])$.

Algorithm 6:

$\mathbf{add\_to\_i}_\ell\mathbf{\_f}_{\ell-1}$ **goto exit or** s_1 **or** ... **or** s_d ;
s_1: $\mathbf{M}_{\mathsf{i}}[\ell][1] := \mathbf{M}_{\mathsf{i}}[\ell][1] + 1$;
$\mathbf{M}_{\mathsf{f}}[\ell-1][1] := \mathbf{M}_{\mathsf{f}}[\ell-1][1] + 1$;
goto $\mathbf{add\_to\_i}_\ell\mathbf{\_f}_{\ell-1}$;
[...];
s_d: $\mathbf{M}_{\mathsf{i}}[\ell][d] := \mathbf{M}_{\mathsf{i}}[\ell][d] + 1$;
$\mathbf{M}_{\mathsf{f}}[\ell-1][d] := \mathbf{M}_{\mathsf{f}}[\ell-1][d] + 1$;
goto $\mathbf{add\_to\_i}_\ell\mathbf{\_f}_{\ell-1}$;
exit: **return**;

Alg. 7 implements the call $transfer\_from\_to(\mathbf{M}_{\mathsf{f}}[\ell], \mathbf{M}_{\mathsf{f}}[\ell-1])$.

Algorithm 7:

$\mathbf{tr\_f}_\ell\mathbf{\_f}_{\ell-1}$ **goto exit or** s_1 **or** ... **or** s_d ;
s_1: $\mathbf{M}_{\mathsf{f}}[\ell][1] := \mathbf{M}_{\mathsf{f}}[\ell][1] - 1$;
$\mathbf{M}_{\mathsf{f}}[\ell-1][1] := \mathbf{M}_{\mathsf{f}}[\ell-1][1] + 1$;
goto $\mathbf{tr\_f}_\ell\mathbf{\_f}_{\ell-1}$;
[...];
s_d: $\mathbf{M}_{\mathsf{f}}[\ell][d] := \mathbf{M}_{\mathsf{f}}[\ell][d] - 1$;
$\mathbf{M}_{\mathsf{f}}[\ell-1][d] := \mathbf{M}_{\mathsf{f}}[\ell-1][d] + 1$;
goto $\mathbf{tr\_f}_\ell\mathbf{\_f}_{\ell-1}$;
exit: **return**;

B Missing Proofs

B.1 Proof of Lemma 2

PROOF. Let $w \in \Sigma^*$, we shall demonstrate that $A^{[k]} \Rightarrow^* w$ iff there exists a derivation $A \Rightarrow^* w$ that is $k+1$ index bounded.
Only if. We have $A^{[k]} \Rightarrow^\ell w$ for some $\ell \in \mathbb{N} \setminus \{0\}$. The proof is done by induction on ℓ. For the case $\ell = 1$, we have $A^{[k]} \Rightarrow w$, hence that $(A^{[k]}, w) \in \mathcal{P}^{[k]}$ and $(A, w) \in \mathcal{P}$ by definition of $G^{[k]}$ and finally that $A \Rightarrow w$ is $1 \leq k+1$ index bounded. For the case $\ell > 1$, the definition of $G^{[k]}$ shows that there exists a derivation of the form (1) $A^{[k]} \Rightarrow B^{[k-1]}C^{[k]} \Rightarrow^i w_1 C^{[k]} \Rightarrow^j w_1 w_2 = w$ where $i + j = \ell - 1$ or (2) $A^{[k]} \Rightarrow B^{[k]}C^{[k-1]} \Rightarrow^{j'} B^{[k]}w_2 \Rightarrow^{i'} w_1 w_2 = w$ where $i' + j' = \ell - 1$ which is treated similarly. Assume case (1) holds. Because $B^{[k-1]} \Rightarrow^i w_1$ where $i < \ell$ we find, by induction hypothesis, that there exists a derivation $B \Rightarrow^* w_1$ that is k index bounded. Also, since $C^{[k]} \Rightarrow^j w_2$ where $j < \ell$, the induction hypothesis shows that there exists a derivation $C \Rightarrow^* w_2$ that is $k+1$ index bounded. Finally, we conclude from $(A^{[k]}, B^{[k-1]}C^{[k]}) \in \mathcal{P}^{[k]}$, that $(A, BC) \in \mathcal{P}$, hence that there exists a derivation $A \Rightarrow BC \Rightarrow^* w_1 C \Rightarrow^* w_1 w_2 = w$ that is $k+1$ index bounded and we are done.
If. Let $A \Rightarrow^\ell w$ for some $\ell \in \mathbb{N} \setminus \{0\}$ be a $k+1$ index bounded derivation. The proof is done by induction on ℓ. For the case $\ell = 1$, we conclude from $A \Rightarrow w$ is $k+1$ index bounded that $(A, w) \in \mathcal{P}$ by definition of G, hence that $(A^{[k]}, w) \in \mathcal{P}^{[k]}$ by definition of $G^{[k]}$ and finally that $A^{[k]} \Rightarrow w$.

For the case $\ell > 1$, there is a $k+1$ index bounded derivation of the form $A \Rightarrow BC \Rightarrow^{\ell-1} w$ such that one of the following derivation is $k+1$ index bounded: $A \Rightarrow BC \Rightarrow^i w_1 C \Rightarrow^j w_1 w_2 = w$ or $A \Rightarrow BC \Rightarrow^j B w_2 \Rightarrow^i w_1 w_2 = w$ where $i + j = \ell - 1$.

Assume the former case holds (the other is handled similarly). Since the derivation is $k+1$ index bounded we find that $B \Rightarrow^i w_1$ is k index bounded and $C \Rightarrow^j w_2$ is $k+1$ bounded. Because $i < \ell$ and $j < \ell$ we find, by induction hypothesis, that $w_1 \in L(B^{[k-1]})$ and $w_2 \in L(C^{[k]})$. Finally, $A \Rightarrow BC$ shows that $(A, BC) \in \mathcal{P}$, hence we deduce that $\left\{(A^{[k]}, B^{[k-1]}C^{[k]}), (A^{[k]}, B^{[k]}C^{[k-1]})\right\} \subseteq \mathcal{P}^{[k]}$, and finally that $A^{[k]} \Rightarrow^* w$ holds. ∎

B.2 Proof of Lemma 7

PROOF. The proof is done by induction on ℓ.
Basis. $\ell = 0$. Let $w \in L_0$, that is $w \in T_0^k$ for some $k \in \mathbb{N}$. The proof is by induction on k. The case $k = 0$ ($w = \varepsilon$) is trivially solved. Let $k > 0$, then w can be decomposed in $w_1, \ldots, w_k$ where each $w_i \in T_0$ for $i \in \{1, \ldots, k\}$ and w_i is necesarily of the form $v_j \cdot t_j \cdot u_j$. Finally since the firing of $v_j \cdot t_j \cdot u_j \in T_0$ keeps unchanged the total number of tokens in $\{s_i, r_i\}$ for each $i \in \{1, \ldots, n\}$ then so does all $w \in T_0$ and we are done.
Step. $\ell > 0$. The definition of L_ℓ shows that $w \in ((\langle p_\ell, c_\ell \rangle \star L_{\ell-1}) \cup T_\ell)^k$ for some $k \in \mathbb{N}$. The proof is done by induction on k. The case $k = 0$ ($w = \epsilon$) is trivially solved. For $k > 0$ we have that $w = w_1 \cdots w_k$ where $w_i \in \langle p_\ell, c_\ell \rangle \star L_{\ell-1}$ or $w_i \in T_\ell$. If $w_1 \in T_\ell$, then using the above reasoning we find that the the firing of any $w \in T_\ell$ keeps unchanged the total number of tokens in $\{s_i, r_i\}$ for each $i \in \{1, \ldots, n\}$. If $w_1 \in \langle p_\ell, c_\ell \rangle \star L_{\ell-1}$ then $w_1 = p_\ell^i v c_\ell^i$ for some $i \in \mathbb{N}$, $v \in L_{\ell-1}$. Since the result holds for every $v \in L_{\ell-1}$ by induction hypothesis, we find

that it also holds for w_1 by definition of p_ℓ and c_ℓ and because they fire an equal number of times. Finally we use the induction hypothesis on $w_2 \cdots w_k$ (we can because $w_2 \cdots w_k \in L_\ell$) and we are done. ∎

B.3 Proof of Lemma 8

PROOF. The proof is done by induction on ℓ.

Basis. $\ell = 0$. $w \in L_0 = T_0^*$ and every transition t occurring in $w|_T$ is such that $Z(t) = \emptyset$, hence the def. of N' and $\mathbf{m}_a\, [w\rangle_{N'}\, \mathbf{m}_b$ show that $S(\mathbf{m}_a)\, [w|_T\rangle_N\, S(\mathbf{m}_b)$.

Step. $\ell > 0$. The definition of L_ℓ shows that $w \in ((\langle p_\ell, c_\ell\rangle \star L_{\ell-1}) \cup T_\ell)^k$ for some $k \in \mathbb{N}$. The proof is done by induction on k. The case $k = 0$ ($w = \epsilon$) is trivially solved. For $k > 0$ we have that $w = w_1 \cdots w_k$ where $w_i \in \langle p_\ell, c_\ell\rangle \star L_{\ell-1}$ or $w_i \in T_\ell$. If $w_1 \in \langle p_\ell, c_\ell\rangle \star L_{\ell-1}$ then $w_1 = p_\ell^i v c_\ell^i$ for some $i \in \mathbb{N}$, $v \in L_{\ell-1}$. Let $\mathbf{m}_0, \mathbf{m}_0', \mathbf{m}_1', \mathbf{m}_1$ such that $\mathbf{m}_a = \mathbf{m}_0\, [p_\ell^i\rangle\, \mathbf{m}_0'\, [v\rangle\, \mathbf{m}_1'\, [c_\ell^i\rangle\, \mathbf{m}_1$. We conclude from $(S_\ell \cup R_\ell)(\mathbf{m}_a) = \emptyset$ and p_ℓ^i that $(S_{\ell-1} \cup R_{\ell-1})(\mathbf{m}_0') = \emptyset$. Next Lem. 7 shows that $(S_{\ell-1} \cup R_{\ell-1})(\mathbf{m}_1') = \emptyset$. Hence, the induction hypothesis on $L_{\ell-1}$ shows that $S(\mathbf{m}_0')\, [v|_T\rangle_N\, S(\mathbf{m}_1')$. Finally the definition of w_1 shows that $w_1|_T = v|_T$, hence that $S(\mathbf{m}_0')\, [w_1|_T\rangle_N\, S(\mathbf{m}_1')$, and finally that $S(\mathbf{m}_0)\, [w_1|_T\rangle_N\, S(\mathbf{m}_1)$ since $S(\mathbf{m}_0) = S(\mathbf{m}_0')$ and $S(\mathbf{m}_1) = S(\mathbf{m}_1')$. Also from the assumption $(S_\ell \cup R_\ell)(\mathbf{m}_0) = \emptyset$, $w_1 \in L_\ell$ and Lem. 7 we conclude that $(S_\ell \cup R_\ell)(\mathbf{m}_1) = \emptyset$.

Let us now turn to the case $w_1 \in T_\ell$. Let $\mathbf{m}_1$ such that $\mathbf{m}_a\, [w_1\rangle\, \mathbf{m}_1$, we conclude from $(S_\ell \cup R_\ell)(\mathbf{m}_a) = \emptyset$, $w_1 \in L_\ell$ and Lem. 7 that $(S_\ell \cup R_\ell)(\mathbf{m}_1) = \emptyset$, hence that $S(\mathbf{m}_a)\, [w_1|_T\rangle_N\, S(\mathbf{m}_1)$ since $w_1|_T = t_j$, $Z(t_j) = S_\ell$ and $S_\ell(\mathbf{m}_a) = \emptyset$.

Finally we use the induction hypothesis on $w_2 \cdots w_k$ (we can because (1) $w_2 \cdots w_k \in L_\ell$ and (2) we have shown that $(S_\ell \cup R_\ell)(\mathbf{m}_1) = \emptyset$ in both cases) and we are done. ∎

B.4 Proof of Lemma 9

PROOF. The proof is done by induction on ℓ.

Basis. $\ell = 0$. First, let us observe that, since $\ell = 0$, the predicates $S_\ell(\mu_1) = S_\ell(\mu_2) = \emptyset$ and $R_\ell(\mathbf{m}_1) = R_\ell(\mathbf{m}_2) = \emptyset$ are vacuously true. Let $\mu_1\, [u\rangle_N\, \mu_2$ where $u \in L_0|_T$. Then, there is a word $w \in L_0$ such that $u = w|_T$. Let $\mathbf{m}_1 \in \mathbb{M}[S']$ defined as follows: $S(\mathbf{m}_1) = \mu_1$, and $\mathbf{m}_1(r_i) = |w|$ for all $i \in \{1, \ldots, n\}$. Then, we have $\mathbf{m}_1\, [w\rangle_{N'}$ which yields $\mathbf{m}_2$ since there are enough tokens in the places R_n. Moreover, we have $S(\mathbf{m}_2) = \mu_2$ since no transition in $\{p_1, c_1, \ldots, p_n, c_n\}$ has an arc to a place in S.

Step. $\ell > 0$. Since there is $u \in L_\ell|_T$ such that $\mu_1\, [u\rangle_N\, \mu_2$, then either case must hold:

- **Case 1:** $u \in L_{\ell-1}|_T$. Then, we can use the induction hypothesis to show that there are $\mathbf{m}_1', \mathbf{m}_2' \in \mathbb{M}[S']$ and $w' \in L_{\ell-1}$ such that $S(\mathbf{m}_1') = \mu_1$, $S(\mathbf{m}_2') = \mu_2$, $R_{\ell-1}(\mathbf{m}_1') = R_{\ell-1}(\mathbf{m}_2') = \emptyset$, and $\mathbf{m}_1'\, [w'\rangle_{N'}\, \mathbf{m}_2'$. Next, Lem. 7 shows that $\mathbf{m}_1'(s_\ell) + \mathbf{m}_1'(r_\ell) = \mathbf{m}_2'(s_\ell) + \mathbf{m}_2'(r_\ell)$, hence that $\mathbf{m}_1'(r_\ell) = \mathbf{m}_2'(r_\ell)$ since $S_\ell(\mu_i) = \emptyset$ and $S(\mathbf{m}_i') = \mu_i$ for $i \in \{1, 2\}$. Let $w = p_\ell^j\, w'\, c_\ell^j \in L_\ell$ where $j = \mathbf{m}_1'(r_\ell)$, and let $\mathbf{m}_1, \mathbf{m}_2 \in \mathbb{M}[S']$ such that $(S' \setminus \{r_\ell\})(\mathbf{m}_i) = (S' \setminus \{r_\ell\})(\mathbf{m}_i')$ and $\mathbf{m}_i(r_\ell) = 0$ for $i \in \{1, 2\}$. From the above we find that (i) $S(\mathbf{m}_i) = S(\mathbf{m}_i') = \mu_i$ for $i \in \{1, 2\}$, (ii) $R_\ell(\mathbf{m}_1) = R_\ell(\mathbf{m}_2) = \emptyset$

(since $R_{\ell-1}(\mathbf{m}'_1) = R_{\ell-1}(\mathbf{m}'_2) = \varnothing$ and by def. of $\mathbf{m}_1, \mathbf{m}_2$) and (iii) $\mathbf{m}_1 \left[w\right\rangle_{N'} \mathbf{m}_2$ (since $\mathbf{m}'_1(r_\ell) = \mathbf{m}'_2(r_\ell)$ we can show that $\mathbf{m}_1 \left[p^j_\ell\right\rangle \mathbf{m}'_1 \left[w'\right\rangle \mathbf{m}'_2 \left[c^j_\ell\right\rangle \mathbf{m}_2$) and we are done.

- **Case 2:** $u = w_0 t_{i_1} w_1 t_{i_2} w_2 \cdots t_{i_k} w_k$ for some $w_1, \ldots, w_k \in L_{\ell-1}|_T$ and $t_{i_1}, \ldots, t_{i_k} \in T_\ell|_T$ (also $Z(t_{i_1}) = \cdots = Z(t_{i_k}) = S_\ell$). To simplify the presentation, we assume that $k = 1$. (The general case can be handled in the same way.) Then, there are $\mu'_1, \mu'_2 \in \mathbb{M}[S]$ such that $\mu_1 \left[w_0\right\rangle \mu'_1 \left[t_{i_1}\right\rangle \mu'_2 \left[w_1\right\rangle \mu_2$. Since $\mu'_1 \left[t_{i_1}\right\rangle \mu'_2$, $Z(t_{i_1}) = S_\ell$ and $S_\ell\big(O(t_{i_1})\big) = \varnothing$, we have $S_\ell(\mu'_1) = S_\ell(\mu'_2) = \varnothing$. Hence, we can apply the first case to the runs $\mu_1 \left[w_0\right\rangle \mu'_1$ and $\mu'_2 \left[w_1\right\rangle \mu_2$, to show there are $\mathbf{m}_1, \mathbf{m}'_1, \mathbf{m}'_2, \mathbf{m}_2 \in \mathbb{M}[S']$ such that $S(\mathbf{m}_i) = \mu_i$, $S(\mathbf{m}'_i) = \mu'_i$, $R_\ell(\mathbf{m}'_i) = R_\ell(\mathbf{m}_i) = \varnothing$ for $i \in \{1, 2\}$, $\mathbf{m}'_1 \in \left[\mathbf{m}_1\right\rangle^{L_\ell}_{N'}$, and $\mathbf{m}_2 \in \left[\mathbf{m}'_2\right\rangle^{L_\ell}_{N'}$. Moreover $t_{i_1} \in T_\ell|_T$ shows that there exist $u_{i_1} \in \{p_{\ell+1}, \ldots, p_n\}^*$ and $v_{i_1} \in \{c_{\ell+1}, \ldots, c_n\}^*$ such that $u_{i_1} \cdot t_{i_1} \cdot v_{i_1} \in T_\ell$. Therefore we can pick $\mathbf{m}_1, \mathbf{m}'_1, \mathbf{m}'_2, \mathbf{m}_2$ such that in addition to the above constraints we have $\mathbf{m}'_1 \left[u_{i_1} t_{i_1} v_{i_1}\right\rangle_{N'} \mathbf{m}'_2$ which is possible since $\mu'_1 \left[t_{i_1}\right\rangle \mu'_2$ and $S(\mathbf{m}'_i) = \mu'_i$ for $i \in \{1, 2\}$. Finally the above reasoning shows that $\mathbf{m}'_1 \in \left[\mathbf{m}_1\right\rangle^{L_\ell}_{N'}$, $\mathbf{m}'_2 \in \left[\mathbf{m}'_1\right\rangle^{L_\ell}_{N'}$, $\mathbf{m}_2 \in \left[\mathbf{m}'_2\right\rangle^{L_\ell}_{N'}$, hence that $\mathbf{m}_2 \in \left[\mathbf{m}_1\right\rangle^{L_\ell}_{N'}$ by definition of L_ℓ and we are done since $S(\mathbf{m}_i) = \mu_i$, $R_\ell(\mathbf{m}_i) = \varnothing$ for $i \in \{1, 2\}$. ∎