The Complexity of Coverability in nu-Petri Nets

Ranko Lazić, Sylvain Schmitz

The Complexity of Coverability in ν-Petri Nets *

Ranko Lazić
DIMAP, Department of Computer Science
University of Warwick, UK
lazic@dcs.warwick.ac.uk

Sylvain Schmitz
LSV, ENS Cachan & CNRS & Inria
Université Paris-Saclay, France
schmitz@lsv.ens-cachan.fr

Abstract

We show that the coverability problem in ν-Petri nets is complete for 'double Ackermann' time, thus closing an open complexity gap between an Ackermann lower bound and a hyper-Ackermann upper bound. The coverability problem captures the verification of safety properties in this nominal extension of Petri nets with name management and fresh name creation. Our completeness result establishes ν-Petri nets as a model of intermediate power among the formalisms of nets enriched with data, and relies on new algorithmic insights brought by the use of well-quasi-order ideals.

Categories and Subject Descriptors F.2.2 [*Analysis of Algorithms and Problem Complexity*]: Nonnumerical Algorithms and Problems

Keywords Well-structured transition system, formal verification, well-quasi-order, order ideal, fast-growing complexity

1. Introduction

ν-Petri nets (νPN) generalise Petri nets by decorating tokens with data values taken from some infinite countable data domain $\mathbb{D}$. These values act as pure names: they can only be compared for equality or non-equality upon firing transitions; νPNs have furthermore the ability to create *fresh* data values, never encountered before in the history of the computation. Such systems were introduced to model distributed protocols where process identities need to be taken into account (Rosa-Velardo and de Frutos-Escrig 2008, 2011). As such, they can also be seen as a kind of 'monadic' restriction of the π-calculus: their *polyadic* extension, which allows to manipulate tuples of tokens, is indeed equivalent to the full π-calculus (Rosa-Velardo and Martos-Salgado 2012)—and Turing-complete.

In spite of their high expressiveness, νPNs fit in the large family of Petri net extensions among the *well-structured* ones (Abdulla et al. 2000; Finkel and Schnoebelen 2001). As such, they still enjoy decision procedures for several verification problems, prominently safety (through the *coverability* problem) and termination. They share these properties with the other extensions of Petri nets with data defined by Lazić, Newcomb, Ouaknine, Roscoe, and Worrell (2008), but are something of an intermediate model. Indeed, as shown in Figure 1, they extend *unordered Petri data nets* with the ability to create fresh data values, but in turn this ability can be simulated (as far as the coverability problem is concerned) by either *ordered data Petri nets*—where $\mathbb{D}$ is equipped with a dense linear ordering—or *unordered data nets*—where 'whole-place operations' allow to transfer, duplicate, or destroy the entire contents of places.

* Work funded in part by the ANR grant ANR-14-CE28-0005 PRODAQ, the EPSRC grant EP/M011801/1, and the Leverhulme Trust Visiting Professorship VP1-2014-041.

The Power of Well-Structured Systems. This work is part of a general program that aims to understand the expressive power and algorithmic complexity of well-structured transition systems (WSTS), for which the complexity of the coverability problem is a natural proxy. Besides the intellectual satisfaction one might find in classifying the worst-case complexity of this problem, we hope indeed to gain new insights into the algorithmics of the systems at hand, and into their relative 'power.' A difficulty is that the generic *backward coverability* algorithm developed by Abdulla, Čerāns, Jonsson, and Tsay (2000) and Finkel and Schnoebelen (2001) to solve coverability in WSTS relies on well-quasi-orders (wqos), for which complexity analysis techniques are not so widely known.

Nevertheless, in a series of recent papers, the exact complexity of coverability for several classes of WSTS has been established. These complexities are expressed using ordinal-indexed *fast-growing* complexity classes $(\mathbf{F}_\alpha)_\alpha$ (Schmitz 2016), e.g. 'Tower' complexity corresponds to the class $\mathbf{F}_3$ and is the first non elementary complexity class in this hierarchy, 'Ackermann' corresponds to $\mathbf{F}_\omega$ and is the first non primitive-recursive class, 'hyper-Ackermann' to $\mathbf{F}_{\omega^\omega}$ and is the first non multiply-recursive class, etc. (see Figure 4). To cite a few of these complexity results, coverability is $\mathbf{F}_\omega$-complete for reset Petri nets and affine nets (Schnoebelen 2010; Figueira et al. 2011), $\mathbf{F}_{\omega^\omega}$-complete for lossy channel systems (Chambart and Schnoebelen 2008; Schmitz and Schnoebelen 2011) and unordered data nets (Rosa-Velardo 2014), and even higher complexities appear for timed-arc Petri nets and ordered data Petri nets ($\mathbf{F}_{\omega^{\omega^\omega}}$-complete, see Haddad et al. 2012) and priority channel systems and nested counter systems ($\mathbf{F}_{\varepsilon_0}$-complete, see Haase et al. 2014; Decker and Thoma 2016); see the complexities in violet in Figure 1 for the Petri net extensions related to νPNs.

All those results rely on the same general template (see Schmitz and Schnoebelen 2013, for a gentle introduction):

1. for the upper bound, a *controlled bad sequence* can be extracted from any run of the backward coverability algorithm, and in turn the length of this sequence can be bounded using a *length function theorem* for the wqo at hand (e.g., Cichoń and Tahhan Bittar 1998; Figueira et al. 2011; Schmitz and Schnoebelen 2011; Rosa-Velardo 2014, for the mentioned results);
2. for the lower bound, *weak computers* for *Hardy functions* and their inverses are implemented in the formalism at hand, allowing to build a working space on which a Turing or Minsky machine can be simulated.

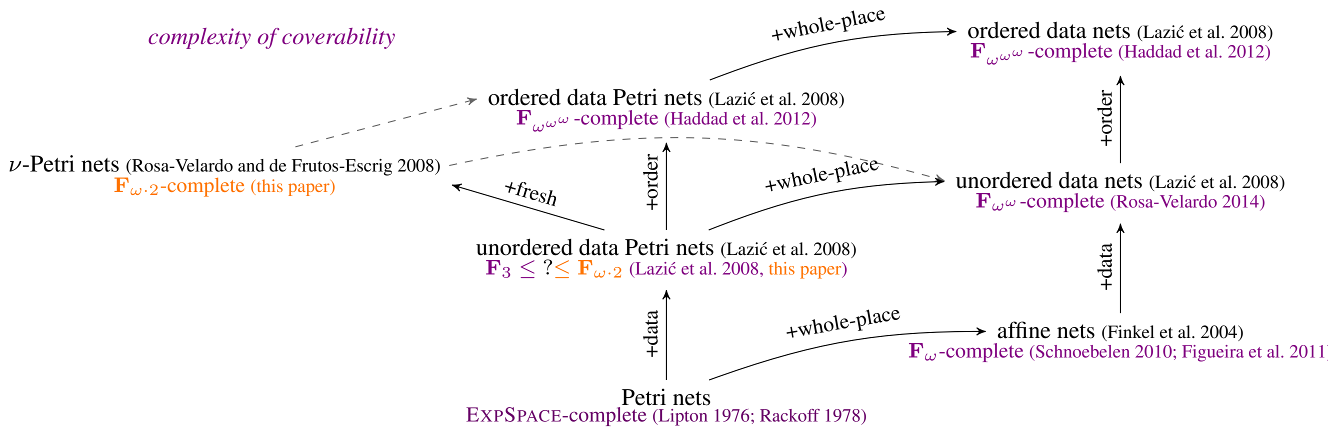

Figure 1. A short taxonomy of some data enrichements of Petri nets. Complexities in violet refer to the already known complexities for the coverability problem; the exact complexity in unordered data Petri nets is unknown at the moment. As indicated by the dashed arrows, freshness can be enforced using a dense linear order or whole-place operations.

Contributions. In this paper, we pinpoint the complexity of coverability in νPNs by showing that it is complete for $\mathbf{F}_{\omega\cdot 2}$, i.e. for 'double Ackermann' complexity. This solves an open problem: the best known lower bound was $\mathbf{F}_\omega$, from a reduction from coverability in reset Petri nets (Rosa-Velardo and de Frutos-Escrig 2011), while the best known upper bound was $\mathbf{F}_{\omega^\omega}$ from the more general case of unordered data nets (Rosa-Velardo 2014), leaving a considerable complexity gap.

We believe this $\mathbf{F}_{\omega\cdot 2}$-completeness is remarkable on two counts. First, this is the first instance of a 'natural' decision problem complete for an intermediate complexity class between Ackermann and hyper-Ackermann. Second, the usual template for such complexity results, summed up in points 1 and 2 above, *fails* for νPNs, in the sense that all it could prove are the aforementioned $\mathbf{F}_\omega$ lower bound and $\mathbf{F}_{\omega^\omega}$ upper bound. As a result, we had to design new techniques, which we think are of independent interest.

These new techniques are inspired by another case where the template in 1 and 2 fails, namely that of Petri nets. Indeed, coverability in Petri nets is ExpSpace-complete, as shown by Rackoff (1978) for the upper bound and by Lipton (1976) for the lower bound. These results however do not rely on wqos and are quite specific to Petri nets, and their generalisation to a formalism as rich as νPNs required new insights:

- For the upper bound, we analyse the complexity of the backward coverability algorithm when seen dually as computing a decreasing sequence of *downwards-closed* sets. Such sets can be represented as finite unions of *ideals* (Bonnet 1975; Finkel and Goubault-Larrecq 2009); see Section 3.

  We have recently shown that, for Petri nets, this dual view allows to exhibit an invariant on the ideals appearing during the course of the execution of the backward coverability algorithm, which in turn yields a dramatic improvement on its complexity analysis from $\mathbf{F}_\omega$ to 2ExpTime (Lazić and Schmitz 2015). The same bound had already been established by Bozzelli and Ganty (2011) using Rackoff's analysis, but this new viewpoint is applicable to any WSTS with effective ideal representations, and enables us to proceed along similar lines in Section 4 and to obtain the desired $\mathbf{F}_{\omega\cdot 2}$ upper bound.
- For the lower bound, we follow the pattern of Lipton's proof, in that we design an 'object-oriented' implementation of the double Ackermann function in νPNs. By this, we mean that the implementation provides an interface with increment, decrement, zero, and max operations on larger and larger *counters* up to a double Ackermannian value. This allows then the simulation of a Minsky machine working in double Ackermann space and establishes the matching $\mathbf{F}_{\omega\cdot 2}$ lower bound.

  The basic building blocks of this development are Ackermannian counters reminiscent of the construction of Schnoebelen (2010) for reset Petri nets. The catch is that we need to be able to mimick this construction for non-fixed dimensions and to combine it with an iteration operator—of the kind employed recently by Lazić et al. (2013) in the context of channel systems with insertion errors to show Ackermann-hardness—, which led us to develop delicate indexing mechanisms by data values; see Section 5.

We assume the reader is already familiar with the basics of Petri nets, and start with the formal definition of νPNs and of their semantics in the upcoming Section 2. Due to space constraints, some technical material and proofs will be found in the appendices.

2. ν-Petri Nets

We define the syntax of νPNs exactly like Rosa-Velardo and Martos-Salgado (2012). Their semantics can be stated in terms of finitely supported partial maps from an infinite data domain $\mathbb{D}$ to markings in $\mathbb{N}^P$, telling for each data value how many tokens with that value appear in each place. However, we find it easier to work with a slightly more abstract but equivalent *multiset semantics*, which accounts for the fact that the semantics is invariant under permutations of the data domain $\mathbb{D}$, and eschews any explicit reference to this data domain; see Section 2.2. Following Rosa-Velardo and Martos-Salgado (2012), we also illustrate the expressive power of νPNs in Section 2.3 by showing how they can implement reset Petri nets.

2.1 Finite Multisets

Let A be a set. Consider the *commutation* equivalence $\sim$ over finite sequences in A^*: this is the transitive reflexive closure $\sim \stackrel{\text{def}}{=} \sim_1^*$ of the relation $\sim_1$ defined by $uabv \sim_1 ubav$ for all $u, v \in A^*$ and $a, b \in A$. We define (finite) *multisets* as $\sim$-equivalence classes of A^*, and write $A^\circledast \stackrel{\text{def}}{=} A^*/\sim$ for the set of multisets over A.

We manipulate a multiset through any of its representatives in A^*, e.g. $[aab] = [aba] = [baa]$ are all equal to the $\sim$-equivalence class $\{aab, aba, baa\}$. We write accordingly '$[\,]$' for the empty multiset. Note that this viewpoint matches the definition of a finite

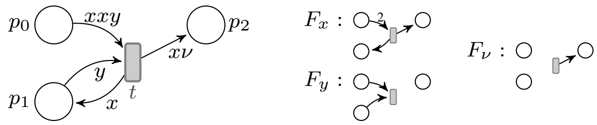

Figure 2. A νPN and the associated flows of x, y, and ν.

multiset as a *finitely supported* function $m\colon A \to \mathbb{N}$, i.e. such that its *support* $\mathit{Supp}(m) \stackrel{\text{def}}{=} \{a \in A \mid m(a) \neq 0\}$ is finite. For instance, $\mathit{Supp}([aab]) = \{a, b\}$ and $[aab](a) = 2$ and $[aab](b) = 1$. The *length* $|m|$ of a multiset m is the length of any representative and satisfies $|m| = \sum_{a\in A} m(a) = \sum_{a\in \mathit{Supp}(m)} m(a)$ in the functional view.

Sums. Given two multisets m and m' over a set A, their *sum* (also called their *union*) $m \oplus m'$ is represented by the concatenation of their representatives. From the functional viewpoint, $(m\oplus m')(a) = m(a) + m'(a)$ for all $a \in A$, with length $|m \oplus m'| = |m| + |m'|$ and support $\mathit{Supp}(m \oplus m') = \mathit{Supp}(m) \cup \mathit{Supp}(m')$.

Embeddings. Assume $(A, \leq_A)$ is a quasi-order (qo), i.e. that A is equipped with a reflexive transitive relation $\leq_A \subseteq A \times A$. An *embedding* from a multiset $m = [a_1 \cdots a_{|m|}]$ into a multiset $m' = [a'_1 \cdots a'_{|m'|}]$ is an injective function $e\colon \{1, \ldots , |m|\} \to \{1, \ldots , |m'|\}$ such that $a_i \leq_A a'_{e(i)}$ for all $1 \leq i \leq |m|$. In such a case we can decompose m' in a unique manner as $m'' \oplus [a'_{e(1)} \cdots a'_{e(|m|)}]$ for some m''. Note that in general $m \neq [a'_{e(1)} \cdots a'_{e(|m|)}]$, unless $\leq_A$ is the equality relation over A.

We say that m' *embeds* m and write $m \sqsubseteq m'$ if there exists an embedding e from m to m'; observe that $(A^{\circledast}, \sqsubseteq)$ is also a qo.

Markings. Let $(P, =)$ be a finite set ordered by equality. We call a vector $\boldsymbol{m} \in \mathbb{N}^P$ a *marking*. Markings can be added pointwise by $(\boldsymbol{m}+\boldsymbol{m}')(p) \stackrel{\text{def}}{=} \boldsymbol{m}(p)+\boldsymbol{m}'(p)$ for all $p \in P$, and compared using the *product ordering* $\boldsymbol{m} \leq \boldsymbol{m}'$, holding iff $\boldsymbol{m}(p) \leq \boldsymbol{m}'(p)$ for all $p \in P$. Note that $(\mathbb{N}^P, +, \leq)$ is isomorphic to $(P^{\circledast}, \oplus, \sqsubseteq)$, but we shall use the former to avoid confusion with other multisets.

2.2 Syntax and Semantics

Let $\mathcal{X}$ and Υ be two disjoint infinite countable sets of *non-fresh variables* and *fresh variables* respectively, and let $\mathit{Vars} \stackrel{\text{def}}{=} \mathcal{X} \uplus \Upsilon$.

Syntax. A *ν-Petri net* is a tuple $N = \langle P, T, F\rangle$ where P is a finite non-empty set of *places*, T is a finite set of *transitions* disjoint from P, and $F\colon (P \times T) \cup (T \times P) \to \mathit{Vars}^{\circledast}$ is a *flow* function.

For any transition $t \in T$, let $\mathit{InVars}(t) \stackrel{\text{def}}{=} \bigcup_{p\in P} \mathit{Supp}(F(p, t))$ and $\mathit{OutVars}(t) \stackrel{\text{def}}{=} \bigcup_{p\in P} \mathit{Supp}(F(t, p))$ denote its sets of input and output variables respectively, and $\mathit{Vars}(t) \stackrel{\text{def}}{=} \mathit{InVars}(t) \cup \mathit{OutVars}(t)$; we require that

1. fresh variables are never input variables: $\Upsilon \cap \mathit{InVars}(t) = \emptyset$,
2. all the non-fresh output variables are also input variables: $\mathit{OutVars}(t) \cap \mathcal{X} \subseteq \mathit{InVars}(t)$.

Writing $\mathcal{X}(t) \stackrel{\text{def}}{=} \mathit{Vars}(t)\cap\mathcal{X}$ and $\Upsilon(t) \stackrel{\text{def}}{=} \mathit{Vars}(t)\cap\Upsilon$, this entails $\mathcal{X}(t) = \mathit{InVars}(t)$ and $\Upsilon(t) = \mathit{OutVars}(t) \cap \Upsilon$.

For a variable $x \in \mathit{Vars}$, the *flow of* x is the function $F_x\colon (P \times T) \cup (T \times P) \to \mathbb{N}$ defined by $F_x(p, t) \stackrel{\text{def}}{=} F(p, t)(x)$ and $F_x(t, p) \stackrel{\text{def}}{=} F(t, p)(x)$. When we fix a transition $t \in T$, we see $F_x(P, t)$ and $F_x(t, P)$ as markings in $\mathbb{N}^P$. Intuitively, a νPN synchronises a potentially infinite number of Petri nets acting on the same places and transitions. See Figure 2 for a depiction; as usual with Petri nets, places are represented by circles, transitions by rectangles, and non-null flows by arrows labelled with their values.

We define the *size* of a νPN as $|N| \stackrel{\text{def}}{=} \max(|P|, |T|, \sum_{p,t} |F(p, t)|+ |F(t, p)|)$ (this corresponds to a unary encoding of the coefficients in the multisets defined by F).

Multiset Semantics. A νPN defines an infinite transition system $\langle \mathit{Confs}, \to\rangle$ where $\mathit{Confs} \stackrel{\text{def}}{=} (\mathbb{N}^P)^{\circledast}$ is the set of *configurations* and $\to\, \subseteq \mathit{Confs} \times \mathit{Confs}$ is called the *step* relation.

Let us associate with any transition $t \in T$ two multisets of markings, in $(\mathbb{N}^P)^{\circledast}$, of inputs and fresh outputs respectively:

$$\mathsf{in}(t) \stackrel{\text{def}}{=} \bigoplus_{x\in\mathcal{X}(t)} [F_x(P, t)]\,, \qquad \mathsf{out}_\Upsilon(t) \stackrel{\text{def}}{=} \bigoplus_{\nu\in\Upsilon(t)} [F_\nu(t, P)]\,. \tag{1}$$

Given a configuration $M = [\boldsymbol{m}_1 \cdots \boldsymbol{m}_{|M|}]$, we say that t is *fireable* from M if there exists an embedding from $\mathsf{in}(t)$ into M, which here can be seen as an injective function $e\colon \mathcal{X}(t) \to \{1, \ldots , |M|\}$ with $F_x(P, t) \leq \boldsymbol{m}_{e(x)}$ for all $x \in \mathcal{X}(t)$. We call such an e a *mode* for t and M; given t and M there are finitely many different modes.

A mode e for t and M defines a step: it uniquely determines two configurations M' and M'' such that

$$M = M'' \oplus \bigoplus_{x\in\mathcal{X}(t)} [\boldsymbol{m}_{e(x)}]\,, \tag{2}$$

$$M' = M'' \oplus \mathsf{out}_\Upsilon(t) \oplus \bigoplus_{x\in\mathcal{X}(t)} [\boldsymbol{m}'_{e(x)}]\,, \tag{3}$$

where for all $x \in \mathcal{X}(t)$, $\boldsymbol{m}'_{e(x)} \stackrel{\text{def}}{=} \boldsymbol{m}_{e(x)} - F_x(P, t) + F_x(t, P)$. We write $M \xrightarrow{e,t} M'$ in such a case. We write as usual $M \xrightarrow{t} M'$ if there exists e for t and M such that $M \xrightarrow{e,t} M'$, and $M \to M'$ if there exists $t \in T$ such that $M \xrightarrow{t} M'$. In other words, the transition t:

- applies F_x for each non-fresh variable $x \in \mathcal{X}(t)$ to a different individual marking $\boldsymbol{m}_{e(x)} \leq F_x(P, t)$ of M, replacing it with the marking $\boldsymbol{m}'_{e(x)}$,
- leaves the remaining markings in M'' untouched, and
- furthermore adds new markings $F_\nu(t, P)$ for each fresh variable $\nu \in \Upsilon(t)$ to the resulting configuration.

Example 1. Consider the transition t in Figure 2 acting on $P = \{p_0, p_1, p_2\}$ and a configuration $M = [\boldsymbol{m}_1\boldsymbol{m}_2\boldsymbol{m}_3]$ where $\boldsymbol{m}_1 = (2, 1, 1)$, $\boldsymbol{m}_2 = (2, 0, 0)$, and $\boldsymbol{m}_3 = (1, 1, 0)$.

We have $\mathsf{in}(t) = [(2, 0, 0)(1, 1, 0)]$ and $\mathsf{out}_\Upsilon(t) = [\boldsymbol{m}_4]$ with $\boldsymbol{m}_4 = (0, 0, 1)$, and three possible modes. We can have $e_1(x) = \boldsymbol{m}_1$, resulting in $\boldsymbol{m}'_1 = (0, 2, 2)$, and $e_1(y) = \boldsymbol{m}_3$, resulting in $\boldsymbol{m}'_3 = (0, 0, 0)$, hence $M \xrightarrow{e_1,t} [\boldsymbol{m}'_1\boldsymbol{m}_2\boldsymbol{m}'_3\boldsymbol{m}_4]$. Another possibility is to have $e_2(x) = \boldsymbol{m}_2$ yielding $\boldsymbol{m}'_2 = (0, 1, 1)$ and $e_2(y) = \boldsymbol{m}_1$ yielding $\boldsymbol{m}''_1 = (1, 0, 1)$, showing that $M \xrightarrow{e_2,t} [\boldsymbol{m}''_1\boldsymbol{m}'_2\boldsymbol{m}_3\boldsymbol{m}_4]$, and a last possibility is to have $e_3(x) = \boldsymbol{m}_2$ and $e_3(y) = \boldsymbol{m}_3$, resulting in a step $M \xrightarrow{e_3,t} [\boldsymbol{m}_1\boldsymbol{m}'_2\boldsymbol{m}'_3\boldsymbol{m}_4]$.

2.3 Example: Reset Petri Nets

Rosa-Velardo and Martos-Salgado (2012) show that νPNs are able to simulate *reset Petri nets*, an extension of Petri nets with special arcs that empty a place upon firing a transition. A remarkable aspect of the construction we are going to sketch here is that three places and a simple addressing mechanism are enough to simulate reset Petri nets with an arbitrary number of places—recall that the latter have an Ackermannian-hard coverability problem (Schnoebelen 2010). This explains why we will be able to push the lower bound beyond Ackermann-hardness in Section 5, where we design more involved addressing mechanisms.

Given any reset Petri net with places $P = \{p_0, \ldots , p_{n-1}\}$, we build a νPN with three places a, $\bar{a}$, and v. The intuition is for a

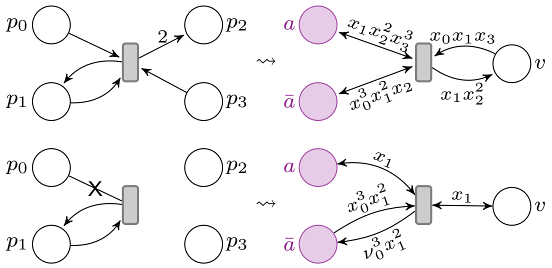

Figure 3. Examples of simulations of reset Petri net transitions (left) by a νPN (right).

and $\bar{a}$ to maintain an *addressing* mechanism for the original places in P, while v maintains the actual token counts of the original net. The places a and $\bar{a}$ use n different data values, each with distinct counts of tokens; more precisely, all the reachable configurations M are of the form $[\boldsymbol{m}_0 \cdots \boldsymbol{m}_{n-1}] \oplus M'$ where $\boldsymbol{m}_i(a) = i$ and $\boldsymbol{m}_i(\bar{a}) = n-1-i$ for all $0 \leq i < n$, and all the markings $\boldsymbol{m}$ in M' are *inactive*, i.e. with $\boldsymbol{m}(a) + \boldsymbol{m}(\bar{a}) < n-1$. Each active marking $\boldsymbol{m}_i$ simulates the place p_i of the original net by holding in $\boldsymbol{m}_i(v)$ the number of tokens in place p_i.

For instance, the top of Figure 3 shows how a transition of a Petri net with 4 places (on the left) can be simulated with this construction (on the right). The flows of each variable x_0, x_1, x_2, x_3 with places a and $\bar{a}$ identify uniquely the places p_0, p_1, p_2, p_3 of the original net, while the flows with place v update the token counts accordingly.

The interest of this addressing mechanism is that it allows to simulate reset transitions, like the one on the bottom left of Figure 3 that empties place p_0 upon firing. This is performed by creating a fresh marking $\boldsymbol{m}'_0$ with $\boldsymbol{m}'_0(a) = 0$, $\boldsymbol{m}'_0(\bar{a}) = 3$, and $\boldsymbol{m}'_0(v) = 0$; after the transition step, we will have $\boldsymbol{m}_0(a) = \boldsymbol{m}_0(\bar{a}) = 0$ and $\boldsymbol{m}_0$ might still have some leftover tokens in v, but it is inactive and will be ignored in the remainder of the computation.

3. Backward Coverability

The decision problem we are interested in is *coverability*:

input: a νPN, and two configurations M_0, M_1

question: does there exist $M \sqsupseteq M_1$ such that $M_0 \to^* M$?

We instantiate in this section the backward coverability algorithm from (Lazić and Schmitz 2015) for νPNs. This algorithm is a dual of the classical algorithm of Abdulla et al. (2000) and Finkel and Schnoebelen (2001): instead of building an increasing chain $U_0 \subsetneq U_1 \subsetneq \cdots$ of upwards-closed sets U_k of configurations that can cover the target M_1 in at most k steps, it constructs instead a decreasing chain $D_0 \supsetneq D_1 \supsetneq \cdots$ of downwards-closed sets D_k of configurations that *cannot* cover the target in k or fewer steps (see Section 3.3). Like the usual backward algorithm, the termination and correctness of this dual version hinges on the fact that $\langle \mathit{Confs}, \to, \sqsubseteq \rangle$ is a WSTS (see Section 3.1). We need however an additional ingredient, which is a means of effectively representing and computing our downwards-closed sets D_k of configurations. We rely for this on *ideals* of $(\mathit{Confs}, \sqsubseteq)$, which play the same role as finite bases in the classical algorithm; see Section 3.2.

3.1 ν-Petri Nets are Well-Structured

Well-Quasi-Orders. Let $(A, \leq_A)$ be a qo. Given a set $S \subseteq A$, its *downward-closure* is ${\downarrow}S \stackrel{\text{def}}{=} \{a \in A \mid \exists s \in S\,.\, a \leq_A s\}$; when S is a singleton $\{s\}$ we write more simply ${\downarrow}s$. A set $D \subseteq A$ is *downwards-closed* (also called *initial*) if ${\downarrow}D = D$. Upward-closures ${\uparrow}S$ and upwards-closed subsets ${\uparrow}U = U$ are defined similarly.

A *well-quasi-order* (wqo) is a qo $(A, \leq_A)$ where every *bad sequence* $a_0, a_1, \ldots$ of elements over A, i.e. with $a_i \not\leq_A a_j$ for all $i < j$, is finite (Higman 1952). Equivalently, it is a qo with the *descending chain property*: all the chains $D_0 \supsetneq D_1 \supsetneq \cdots$ of downwards-closed subsets $D_j \subseteq A$ are finite. Equivalently, it has the *finite basis property*: any non-empty subset $S \subseteq A$ has a finite number of minimal elements (and at least one minimal element) up to equivalence.

For instance, any finite set P equipped with equality forms a wqo $(P, =)$: its downwards-closed subsets are singletons $\{p\}$ for $p \in P$, and its chains of downwards-closed sets are of length at most one. Assuming $(A, \leq_A)$ is a wqo, then finite multisets over A provide another instance: $(A^\circledast, \sqsubseteq)$ is also a wqo as a consequence of Higman's Lemma. Hence both the sets of markings $(\mathbb{N}^P, \leq)$ and of configurations $(\mathit{Confs}, \sqsubseteq)$ of a νPN are wqos.

Compatibility. The transition system $\langle \mathit{Confs}, \to \rangle$ defined by a νPN further satisfies a *compatibility* condition with the embedding relation: if $M_1 \sqsubseteq M'_1$ and $M_1 \to M_2$, then there exists $M'_2 \sqsupseteq M_2$ with $M'_1 \to M'_2$. In other words, $\sqsubseteq$ is a simulation relation on the transition system $\langle \mathit{Confs}, \to \rangle$. Since $(\mathit{Confs}, \sqsubseteq)$ is a wqo, this transition system is therefore *well-structured* (Abdulla et al. 2000; Finkel and Schnoebelen 2001).

3.2 Effective Ideal Representations

Ideals. Let $(A, \leq_A)$ be a wqo. An *ideal* I of A is a non-empty, downwards-closed, and (up-) *directed* subset of A; this last condition enforces that, if a, a' are in I, then there exists $b \in I$ that dominates both: $a \leq_A b$ and $a' \leq_A b$. For example, looking again at the case of finite sets $(P, =)$, we can see that singletons $\{p\}$ are ideals. In fact, more generally ${\downarrow}a$ for $a \in A$ is always an ideal of A. But there can be other ideals, e.g. $I^\circledast$ is an ideal of $A^\circledast$ if I is an ideal of A.

The key property of wqo ideals is that any downwards-closed set D over a wqo has a unique *decomposition* as a finite union $D = I_1 \cup \cdots \cup I_n$, where the I_j's are incomparable for inclusion—this was shown e.g. by Bonnet (1975), and by Finkel and Goubault-Larrecq (2009) in the context of complete WSTS (and generalised to Noetherian topologies). Ideals are also *irreducible*: if $I \subseteq D_1 \cup D_2$ for two downwards-closed sets D_1 and D_2, then $I \subseteq D_1$ or $I \subseteq D_2$.

Effective Representations. Although ideals provide finite decompositions for downwards-closed sets, they are themselves usually infinite, and some additional effectiveness assumptions are necessary to employ them in algorithms. In this paper, we will say that a wqo $(A, \leq_A)$ has *effective* ideal representations (see Finkel and Goubault-Larrecq 2009; Goubault-Larrecq et al. 2016, for more stringent requisites) if every ideal can be represented, and there are algorithms on those representations:

(CI) to check $I \subseteq I'$ for two ideals I and I',

(II) to compute the ideal decomposition of $I \cap I'$ for two ideals I and I',

(CU') to compute the ideal decomposition of the residual $A \setminus {\uparrow}a = \{a' \in A \mid a \not\leq_A a'\}$ for any a in A.

All these effectiveness assumptions are true of the representations for $(\mathbb{N}^P, \leq)$ and $(\mathit{Confs}, \sqsubseteq)$ described by Goubault-Larrecq et al. (2016), which we recall next.

Extended Markings. Let $\mathbb{N}_\omega \stackrel{\text{def}}{=} \mathbb{N} \uplus \{\omega\}$, where '$\omega$' denotes a new top element with $\omega + n = \omega - n = \omega > n$ for all $n \in \mathbb{N}$. An *extended marking* is a vector $\boldsymbol{e} \in \mathbb{N}_\omega^P$. The product ordering and pointwise sum operations are lifted accordingly. Then the ideals of $(\mathbb{N}^P, \leq)$ are exactly the sets

$$[\![\boldsymbol{e}]\!] \stackrel{\text{def}}{=} \{\boldsymbol{m} \in \mathbb{N}^P \mid \boldsymbol{m} \leq \boldsymbol{e}\} \tag{4}$$

defined by extended markings $\boldsymbol{e} \in \mathbb{N}_\omega^P$. Note that $\mathbb{N}_\omega^P$ contains $\mathbb{N}^P$ as a substructure. Regarding effectiveness assumptions, let us just mention that, for (CI), $[\![\boldsymbol{e}]\!] \subseteq [\![\boldsymbol{e}']\!]$ iff $\boldsymbol{e} \sqsubseteq \boldsymbol{e}'$. See (Goubault-Larrecq et al. 2016) for more details.

Extended Configurations. Note that, since $(\mathbb{N}_\omega^P, \leq)$ is a qo, $((\mathbb{N}_\omega^P)^\circledast, \sqsubseteq)$ is also a qo—they are in fact both wqos. An *extended configuration* is a pair (B, S) comprising a finite *base* multiset $B \in (\mathbb{N}_\omega^P)^\circledast$ and a finite *star* set $S \subseteq \mathbb{N}_\omega^P$. Then the ideals of $(\mathit{Confs}, \sqsubseteq)$ are exactly the sets

$$[\![B, S]\!] \stackrel{\text{def}}{=} \{M \in \mathit{Confs} \mid \exists E \in S^\circledast \,.\, M \sqsubseteq B \oplus E\} \tag{5}$$

defined by extended configurations.

This representation is however not canonical, in the sense that there can be $(B, S) \neq (B', S')$ with $[\![B, S]\!] = [\![B', S']\!]$. For instance, if $\boldsymbol{e} \geq \boldsymbol{e}'$, then for all extended configurations (B, S),

$$[\![B, S \cup \{\boldsymbol{e}, \boldsymbol{e}'\}]\!] = [\![B, S \cup \{\boldsymbol{e}\}]\!] \;, \tag{6}$$

$$[\![B \oplus [\boldsymbol{e}'], S \cup \{\boldsymbol{e}\}]\!] = [\![B, S \cup \{\boldsymbol{e}\}]\!] \;. \tag{7}$$

In fact, those are the only two situations, and reading equations (6) and (7) left-to-right as *reduction* rules—which are furthermore confluent—we can associate to any extended configuration (B, S) a unique *reduced* extended configuration. Such an extended configuration (B, S) is such that S is an antichain and, for all extended markings $\boldsymbol{e} \in S$ and $\boldsymbol{e}' \in \mathit{Supp}(B)$, $\boldsymbol{e} \not\geq \boldsymbol{e}'$. Reduced extended configurations provide canonical representatives for the ideals of $(\mathit{Confs}, \sqsubseteq)$; we write XConfs for the set of all reduced extended configurations. In the following, for an ideal I of $(Confs, \sqsubseteq)$ we write $(B(I), S(I))$ for its canonical representative in XConfs.

Observe that XConfs also embeds Confs as an isomorphic substructure: any configuration M can be associated to the extended configuration $(M, \emptyset)$.

Regarding effectiveness assumptions, we shall only comment on (CI) and refer the reader to (Goubault-Larrecq et al. 2016) for details. Given two reduced extended configurations (B, S) and (B', S') in XConfs, $[\![B, S]\!] \subseteq [\![B', S']\!]$ iff $\exists E \in S'^\circledast$ such that $B \sqsubseteq B' \oplus E$, and $S \subseteq_H S'$, where '$\subseteq_H$' denotes the *Hoare* ordering: $S \subseteq_H S'$ iff for all $\boldsymbol{e} \in S$ there exists $\boldsymbol{e}' \in S'$ such that $\boldsymbol{e} \leq \boldsymbol{e}'$.

3.3 Backward Coverability Algorithm

Consider a νPN and a target configuration M_1. Define

$$D_* \stackrel{\text{def}}{=} \{M \in \mathit{Confs} \mid \forall M' \sqsupseteq M_1 \,.\, M \not\rightarrow^* M'\} \tag{8}$$

as the set of configurations that do not cover M_1. The purpose of the backward coverability algorithm is to compute D_*; solving a coverability instance with source configuration M_0 then amounts to checking whether M_0 belongs to D_*.

Let us define the reachability relation in at most $k \in \mathbb{N}$ steps by $\rightarrow^{\leq 0} \stackrel{\text{def}}{=} \{(M, M) \mid M \in \mathit{Confs}\}$ and $\rightarrow^{\leq k+1} \stackrel{\text{def}}{=} \rightarrow^{\leq k} \cup \{(M, M'') \mid \exists M' \in \mathit{Confs} \,.\, M \rightarrow M' \rightarrow^{\leq k} M''\}$. The idea of the algorithm is to compute successively for every k the set D_k of configurations that do *not* cover M_1 in k or fewer steps:

$$D_k \stackrel{\text{def}}{=} \{M \in \mathit{Confs} \mid \forall M' \sqsupseteq M_1 \,.\, M \not\rightarrow^{\leq k} M'\} \;. \tag{9}$$

As shown in (Lazić and Schmitz 2015, Claim 3.2) these over-approximations D_k can be computed inductively on k:

$$D_0 = \mathit{Confs} \setminus \uparrow M_1 \;, \quad D_{k+1} = D_k \cap \mathrm{Pre}_\forall(D_k) \;, \tag{10}$$

where for any set $S \subseteq \mathit{Confs}$ its set of *universal predecessors* is

$$\mathrm{Pre}_\forall(S) \stackrel{\text{def}}{=} \{M \in \mathit{Confs} \mid \forall M'.(M \rightarrow M' \Rightarrow M' \in S)\}. \tag{11}$$

This set is downwards-closed if S is downwards-closed (Lazić and Schmitz 2015, Claim 3.3). We need here to check an additional effectiveness assumption for νPNs (which holds, see Section A.2):

(Pre) the ideal decomposition of $\mathrm{Pre}_\forall(D)$ is computable for all downwards-closed D,

where D is given as a finite set of ideal representations. Then D_0 is computed using (CU'), and at each iteration the intersection of $\mathrm{Pre}_\forall(D_k)$ with D_k is also computable by (Pre) and (II).

The algorithm terminates as soon as $D_k \subseteq D_{k+1}$, and then $D_{k+j} = D_k = D_*$ for all j. This is guaranteed to arise eventually by the descending chain condition, since otherwise we would have an infinite descending chain of downwards-closed sets $D_0 \supsetneq D_1 \supsetneq D_2 \supsetneq \cdots$. The termination check $D_k \subseteq D_{k+1}$ is effective by (CI): by ideal irreducibility, $D_k = I_1 \cup \cdots \cup I_n \subseteq J_1 \cup \cdots \cup J_r = D_{k+1}$ for ideals $I_1, \ldots, I_n$ and ideals $J_1, \ldots, J_m$ if and only if for all $1 \leq i \leq n$ there exists $1 \leq j \leq r$ such that $I_i \subseteq J_j$.

4. Complexity Upper Bounds

We establish in this section a double Ackermann upper bound on the complexity of coverability in νPNs. The main ingredient to that end is a combinatorial statement on the length of *controlled* descending chains of downwards-closed sets, and we define in Section 4.2 control functions and exhibit a control on the descending chain $D_0 \supsetneq D_1 \supsetneq \cdots$ built by the backward coverability algorithm for a νPN. One can extract a controlled bad sequence from such a controlled descending chain (see Section 4.3), from which the *length function theorem* of Rosa-Velardo (2014) yields in turn an hyper-Ackermann upper bound. In order to obtain the desired double Ackermann upper bound, we need to refine this analysis. We observe in Section 4.4 that the descending chains for νPNs enjoy an additional *star monotonicity* property. This in turn allows to prove the upper bound by extracting Ackermann-controlled bad sequences of extended markings; see Theorem 9. The final step is to put this upper bound in the complexity class $\mathbf{F}_{\omega\cdot 2}$.

4.1 Fast-Growing Complexity Classes

In order to express the non-elementary functions required for our complexity statements, we employ a family of subrecursive functions $(h_\alpha)_\alpha$ indexed by ordinals α known as the *Cichoń* hierarchy (Cichoń and Tahhan Bittar 1998).

Ordinal Terms. We use ordinal terms α in *Cantor Normal Form* (CNF), which can be written as terms $\alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_n}$ where $\alpha_1 \geq \cdots \geq \alpha_n$ are themselves written in CNF. Using such notations, we can express any ordinal below ε_0, the minimal fixpoint of $x = \omega^x$. The ordinal 0 is obtained when $n = 0$; otherwise if $\alpha_n = 0$ the ordinal α is a *successor* ordinal $\omega^{\alpha_1} + \cdots + \omega^{\alpha_{n-1}} + 1$, and if $\alpha_n > 0$ the ordinal α is a *limit* ordinal. We usually write 'λ' to denote limit ordinals.

Fundamental Sequences. For all x in $\mathbb{N}$ and limit ordinals λ, we use a standard assignment of *fundamental sequences* $\lambda(0) < \lambda(1) < \cdots < \lambda(x) < \cdots < \lambda$ with supremum λ. Fundamental sequences are defined by transfinite induction by:

$$(\gamma + \omega^{\beta+1})(x) \stackrel{\text{def}}{=} \gamma + \omega^\beta \cdot (x+1), \;\; (\gamma + \omega^\lambda)(x) \stackrel{\text{def}}{=} \gamma + \omega^{\lambda(x)}.$$

For instance, $\omega(x) = x + 1$, $\omega^2(x) = \omega \cdot (x+1)$, $\omega^\omega(x) = \omega^{x+1}$, etc.

The Cichoń Hierarchy. Let $h{:}\, \mathbb{N} \rightarrow \mathbb{N}$ be a strictly increasing function. The *Cichoń* functions $(h_\alpha{:}\, \mathbb{N} \rightarrow \mathbb{N})_\alpha$ are defined by

$$h_0(x) \stackrel{\text{def}}{=} 0 \;, \quad h_{\alpha+1}(x) \stackrel{\text{def}}{=} 1 + h_\alpha(h(x)) \;, \quad h_\lambda(x) \stackrel{\text{def}}{=} h_{\lambda(x)}(x) \;.$$

For instance, $h_k(x) = k$ for all finite k, but for limit ordinals λ, $h_\lambda(x)$ performs a form of diagonalisation: for instance, setting $H(x) \stackrel{\text{def}}{=} x + 1$ the successor function, $H_\omega(x) = H_{x+1}(x) = x + 1$, $H_{\omega^2}(x) = (2^{x+1} - 1)(x + 1)$ is a function of exponential growth, while H_{ω^3} is a non elementary function akin to a tower of

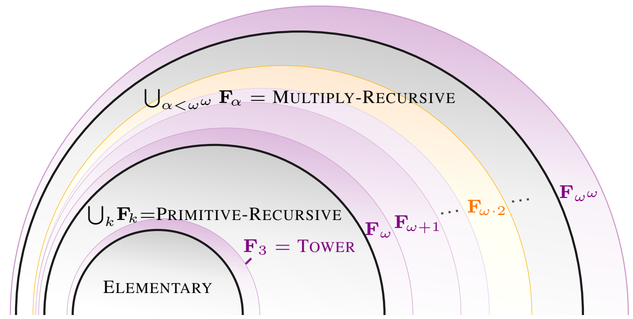

Figure 4. Pinpointing $\mathbf{F}_{\omega\cdot 2}$ among the complexity classes beyond ELEMENTARY.

exponentials of height x, H_{ω^ω} is a non primitive-recursive function with growth similar to the Ackermann function, and $H_{\omega^{\omega^\omega}}$ is a non multiply-recursive function characteristic of hyper-Ackermannian complexity.

The Cichoń functions are weakly increasing. If $g(x) \leq h(x)$ for all x, then also $g_\alpha(x) \leq h_\alpha(x)$ for all x. Finally, if $\alpha < \beta$, then h_α is eventually bounded by h_β: there exists x_0 such that for all $x \geq x_0$, $h_\alpha(x) \leq h_\beta(x)$.

Complexity Classes. Following (Schmitz 2016), we can define complexity classes for computations with time or space resources bounded by Cichoń functions of the size of the input. We concentrate in this paper on the double Ackermann complexity class. For $\alpha > 2$, let $\mathscr{F}_{<\alpha}$ denote the set of number-theoretic functions computable in deterministic time bounded by H_β for $\beta < \omega^\alpha$, which we can write as:

$$\mathscr{F}_{<\alpha} = \bigcup_{\beta<\omega^\alpha} \mathrm{FDTIME}(H_\beta(n)) \; . \tag{12}$$

This class coincides with $\bigcup_{\beta<\alpha} \mathscr{F}_\beta$ in the *extended Grzegorczyk hierarchy* $(\mathscr{F}_\alpha)_\alpha$ (Wainer 1970).

Let h be any primitive-recursive function, i.e. any function in $\mathscr{F}_{<\omega}$. Then we can define $\mathbf{F}_{\omega\cdot 2}$ by (see Schmitz 2016, Theorem 4.2):

$$\mathbf{F}_{\omega\cdot 2} = \bigcup_{p\in\mathscr{F}_{<\omega\cdot 2}} \mathrm{DTIME}(h_{\omega^{\omega\cdot 2}}(p(n))) \; . \tag{13}$$

This is the set of decision problems solvable with resources bounded by a doubly Ackermann function $h_{\omega^{\omega\cdot 2}}$ applied to some 'slower' function p of the size of the input. The definition is tailored to define completeness for $\mathbf{F}_{\omega\cdot 2}$ through many-one reductions in $\mathscr{F}_{<\omega\cdot 2}$. Although we know many examples of problems complete for the related classes $\mathbf{F}_\omega$ and $\mathbf{F}_{\omega^\omega}$ (see Figure 4 for a depiction), this is the first time we encounter the class $\mathbf{F}_{\omega\cdot 2}$.

4.2 Controlled Descending Sequences

Consider some set A with a norm $\|.\|\colon A \to \mathbb{N}$. Given a strictly increasing *control function* $g{:}\mathbb{N} \to \mathbb{N}$ and an *initial norm* $n \in \mathbb{N}$, we say that a sequence $a_0, a_1, \ldots$ of elements from A is *strongly* (g,n)*-controlled* if $\|a_0\| \leq n$ and $\|a_{i+1}\| \leq g(\|a_i\|)$ for all i. A less stringent, amortised requisite is to ask $\|a_i\| \leq g^i(n)$ for all i, where g^i is the ith iterate of g; we say in that case that the sequence is (g,n)*-controlled*.

These notions can be applied to sequences $D_0, D_1, \ldots$ of downwards-closed subsets of $(\mathit{Confs}, \sqsubseteq)$ seen as finite sets of reduced extended configurations in XConfs. Let us therefore equip extended configurations $(B,S) \in \mathit{XConfs}$ and extended markings $\boldsymbol{e} \in \mathbb{N}_\omega^P$ with the following *norm*: $\|B,S\| \stackrel{\text{def}}{=} \max(\|B\|, \|S\|)$, $\|B\| \stackrel{\text{def}}{=} \max_{\boldsymbol{e}\in \mathit{Supp}(B)}(|B|, \|\boldsymbol{e}\|)$, $\|S\| \stackrel{\text{def}}{=} \max_{\boldsymbol{e}\in S}(\|\boldsymbol{e}\|)$, and $\|\boldsymbol{e}\| \stackrel{\text{def}}{=} \max_{p\in P\,\mid\,\boldsymbol{e}(p)<\omega} \boldsymbol{e}(p)$. For a finite set D of extended configurations, we then set $\|D\| \stackrel{\text{def}}{=} \max_{(B,S)\in D} \|B,S\|$.

By controlling how big the extended configurations of $\mathrm{Pre}_\forall(D)$ can grow as a function of $\|D\|$, we show that the descending chain $D_0 \supsetneq D_1 \supsetneq \cdots$ computed by the backward coverability algorithm for νPNs is strongly controlled (see Section A.2):

Lemma 2 (Strong Control for νPNs)**.** *The descending chain computed by the backward coverability algorithm for a νPN N and target configuration M is strongly (g,n)-controlled for $g(x) \stackrel{\text{def}}{=} x + |N|$ and $n \stackrel{\text{def}}{=} \|M\|$.*

4.3 Length Functions Theorems

Length function theorems are combinatorial statements that provide upper bounds on the lengths ℓ of (g,n)-controlled sequences $a_0, a_1, \ldots, a_\ell$.

Bad Sequences of Extended Markings. A first example of a length function theorem is the following Ackermannian upper bound for bad sequences $\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots$ of extended markings in $\mathbb{N}_\omega^P$: combining Corollary 2.25 and Theorem 2.34 from (Schmitz and Schnoebelen 2012) (see also Appendix A of Lazić and Schmitz 2015):

Fact 3 (Length Function Theorem for Bad Sequences in $\mathbb{N}_\omega^P$)**.** *Let $n > 0$. Any (g,n)-controlled bad sequence $\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_\ell$ of extended markings in $(\mathbb{N}_\omega^P, \leq)$ has length bounded by $h_{\omega^{|P|+1}}(n \cdot |P|!)$, where $h(x) \stackrel{\text{def}}{=} |P| \cdot g(x)$.*

Proper Ideals in Descending Chains. When considering a descending chain $D_0 \supsetneq D_1 \supsetneq \cdots \supsetneq D_\ell$ of downwards-closed subsets of some wqo $(A, \leq_A)$, where each set D_k is represented as a finite set of ideals, observe that we can extract at each step $0 \leq k < \ell$ an ideal I_k from the decomposition of D_k that disappears in the next decomposition D_{k+1}. We call such an ideal *proper*; it satisfies $I_k \subseteq D_k$ but $I_k \not\subseteq D_{k+1}$, and as a consequence $I_k \not\subseteq I_{k'}$ for all $k' > k$ since $D_{k'} \subseteq D_{k+1}$ in this case. Hence we can extract a sequence $I_0, I_1, \ldots, I_{\ell-1}$ of ideals, which is a bad sequence for the inclusion ordering.

As an application, consider a (g,n)-controlled descending chain $S_0 \supsetneq_H S_1 \supsetneq_H \cdots \supsetneq_H S_\ell$ of antichains $S_k \subseteq \mathbb{N}_\omega^P$ for the Hoare ordering. Each antichain S_k is in fact an ideal representation for the downwards-closed set of markings $D_k = \bigcup_{\boldsymbol{e}\in S} [\![\boldsymbol{e}]\!] \subseteq \mathbb{N}^P$, i.e. this defines a descending chain $D_0 \supsetneq D_1 \supsetneq \cdots \supsetneq D_\ell$ (the reader can check that $S \supsetneq_H S'$ iff the associated downwards-closed sets are strictly included: $\bigcup_{\boldsymbol{e}\in S} [\![\boldsymbol{e}]\!] \supsetneq \bigcup_{\boldsymbol{e}'\in S'} [\![\boldsymbol{e}']\!]$). As pointed out just before, we can extract a bad sequence $\boldsymbol{e}_0, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{\ell-1}$ of extended markings in $\mathbb{N}_\omega^P$ representing proper ideals. Furthermore, this sequence is also (g,n)-controlled, thus Fact 3 can be applied:

Corollary 4 (Length Function Theorem for Hoare-Descending Chains over $\mathbb{N}_\omega^P$)**.** *Let $n > 0$. Any (g,n)-controlled descending chain $S_0 \supsetneq_H S_1 \supsetneq_H \cdots \supsetneq_H S_\ell$ of antichains of $(\mathbb{N}_\omega^P, \leq)$ has length at most $h_{\omega^{|P|+1}}(n \cdot |P|!) + 1$, where $h(x) \stackrel{\text{def}}{=} |P| \cdot g(x)$.*

4.4 Star-Monotone Descending Chains

Let us lift the step relation $\to$ to work over ideals. Define for any ideal $S \subseteq \mathit{Confs}$

$$\mathrm{Post}_\exists(S) \stackrel{\text{def}}{=} \{M' \in \mathit{Confs} \mid \exists M \in S \,.\, M \to M'\} \; . \tag{14}$$

Then for any ideal I of $(\mathit{Confs}, \sqsubseteq)$, $\downarrow\mathrm{Post}_\exists(I)$ is downwards-closed with a unique decomposition into maximal ideals. We follow Blondin et al. (2014) and write '$I \to J$' if J is an ideal from the decomposition of $\downarrow\mathrm{Post}_\exists(I)$. We will use the following fact proven in (Lazić and Schmitz 2015, Claim 4.2):

Fact 5 (Proper Transition Sequences)**.** *If I_{k+1} is a proper ideal of D_{k+1}, then there exist an ideal J and a proper ideal I_k of D_k such that $I_{k+1} \to J \subseteq I_k$.*

We can check that this step relation lifted to ideals is monotone in the star set (see Section A.1):

Lemma 6 (Ideal Steps are Star-Monotone)**.** *If I and J are two ideals and $I \to J$, then $S(I) \subseteq_H S(J)$.*

We say that a descending chain $D_0 \supsetneq D_1 \supsetneq \cdots \supsetneq D_\ell$ of downwards-closed subsets of *Confs* is *star-monotone* if for all $0 \le k < \ell - 1$ and all proper ideals I_{k+1} in the decomposition of D_{k+1}, there exists a proper ideal I_k in the decomposition of D_k such that $S(I_{k+1}) \subseteq_H S(I_k)$.

Lemma 7 (νPN Descending Chains are Star-Monotone)**.** *The descending chains computed by the backward coverability algorithm for νPNs are star monotone.*

Proof. Let $D_0 \supsetneq D_1 \supsetneq \cdots \supsetneq D_\ell$ be the descending chain computed for our νPN. Suppose $0 \le k < \ell - 1$ and I_{k+1} is a proper ideal in the decomposition of D_{k+1}. By Fact 5, there exists a proper ideal I_k in the decomposition of D_k and an ideal J such that $I_{k+1} \to J$ and $J \subseteq I_k$. By Lemma 6, $S(I_{k+1}) \subseteq_H S(J)$, and by (CI), $S(J) \subseteq_H S(I_k)$. □

The crux of our proof is the following theorem:

Theorem 8 (Length Function Theorem for Star-Monotone Descending Chains over $(\mathbb{N}_\omega^P)^\circledast$)**.** *Let $n > 0$. Any strongly (g, n)-controlled star-monotone descending chain $D_0 \supsetneq D_1 \supsetneq \cdots \supsetneq D_\ell$ of configurations in $(\mathbb{N}_\omega^P)^\circledast$ has length at most $h_{\omega^{\omega\cdot 2}}(|P| + n)$ for some h primitive-recursive in g.*

Proof idea. We prove the theorem in Section B.2, but provide here a quick overview of its proof.

Since the sequence $D_0 \supsetneq D_1 \supsetneq \cdots \supsetneq D_\ell$ is star-monotone, starting from some proper ideal $I_{\ell-1}$ in the decomposition of $D_{\ell-1}$, we can find a sequence of proper ideals $I_0, \ldots, I_{\ell-1}$ such that $S(I_k) \supseteq_H S(I_{k+1})$ for all $0 \le k < \ell - 1$. We can split the associated sequence of star sets as $S_{i_0} \stackrel{\text{def}}{=} S_0 \equiv_H S_1 \equiv_H \cdots \equiv_H S_{i_1-1} \supsetneq_H S_{i_1} \equiv_H \cdots \equiv_H S_{i_2-1} \supsetneq_H S_{i_2} \cdots S_{i_r-1} \supsetneq_H S_{i_r} \equiv_H \cdots \equiv_H S_{\ell-1}$, where two star sets are *Hoare-equivalent*, noted $S \equiv_H S'$, iff $S \subseteq_H S'$ and $S \supseteq_H S'$.

We analyse independently the length of any Hoare-equivalent segment $S_{i_j} \equiv_H S_{i_j+1} \equiv_H \cdots \equiv_H S_{i_{j+1}-1}$ and the length of the Hoare-descending chain $S_{i_0} \supsetneq_H S_{i_1} \supsetneq_H \cdots \supsetneq_H S_{i_r}$. For the former, we show that the associated sequence of bases $B_{i_j}, B_{i_j+1}, \ldots, B_{i_{j+1}-1}$ is a bad sequence controlled by $(g, \|D_{i_j}\|)$, where all the B_k can be treated as $(|P| \cdot \|D_{i_j}\|)$-dimensional vectors in $\mathbb{N}_\omega^{|P|\cdot\|D_{i_j}\|}$. We can therefore apply Fact 3 to this sequence and obtain an Ackermannian control (a, n) on the sequence $S_{i_0} \supsetneq_H S_{i_1} \supsetneq_H \cdots \supsetneq_H S_{i_r}$. In turn, this sequence is bounded thanks to Corollary 4 by $a_{\omega^\omega}(n \cdot |P|!)$, a function that nests an Ackermannian blowup at each of its Ackermannian-many steps. An analysis of this last function yields the result. □

Together with the primitive-recursive control (g, n) exhibited in Lemma 2 and the star-monotonicity of the descending chains computed by the backward algorithm shown in Lemma 7, Theorem 8 provides an upper bound in $\mathbf{F}_{\omega\cdot 2}$ as defined in Equation 13:

Theorem 9. *The coverability problem for νPNs is in $\mathbf{F}_{\omega\cdot 2}$.*

5. Complexity Lower Bounds

5.1 Ackermann Functions

When it comes to lower bounds, we find it more convenient to work with a variant of the functions from Section 4.1 called the *Ackermann hierarchy*. Here we shall only need the functions $(A_\alpha)_{\alpha<\omega\cdot 2}$ from this hierarchy, which can be defined as follows for all k and x in $\mathbb{N}$:

$$A_1(x) \stackrel{\text{def}}{=} 2x\,, \qquad A_{k+2}(x) \stackrel{\text{def}}{=} A_{k+1}^x(1)\,,$$
$$A_\omega(x) \stackrel{\text{def}}{=} A_{x+1}(x)\,, \qquad A_{\omega+k+1}(x) \stackrel{\text{def}}{=} A_{\omega+k}^x(1)\,.$$

The *double Ackermann* function is then defined as

$$A_{\omega\cdot 2}(x) \stackrel{\text{def}}{=} A_{\omega+x+1}(x)\,; \tag{15}$$

note that this is considerably larger than $A_\omega(A_\omega(x))$. We can employ the function $A_{\omega\cdot 2}$ instead of $H_{\omega^{\omega\cdot 2}}$ since, by (Schmitz 2016, Theorem 4.1),

$$\mathbf{F}_{\omega\cdot 2} = \bigcup_{p\in\mathscr{F}_{<\omega\cdot 2}} \mathrm{DT{\small IME}}(A_{\omega\cdot 2}(p(n)))\,. \tag{16}$$

5.2 Routines, Libraries, and Programs

To present our lower bound construction, we shall develop some simple and limited but convenient mechanisms for programming with νPNs.

Syntax of Routines and Libraries. Let a *library* mean a sequence of named routines

$$\ell_1 : R_1, \ldots, \ell_K : R_K,$$

where $\ell_1, \ldots, \ell_K$ are pairwise distinct labels. In turn, a *routine* is a sequence of commands $c_1, \ldots, c_{K'}$, where each c_i for $i < K'$ is one of the following:

- a νPN transition,
- a nondeterministic jump goto G for a nonempty subset G of $\{1, \ldots, K'\}$, or
- a subroutine invocation call ℓ';

and $c_{K'}$ is return.

Semantics of Programs. When a library contains no subroutine calls, we say it is a *program*. The denotation of a program L as above is a νPN $\mathcal{N}(L)$ constructed so that:

- The places of $\mathcal{N}(L)$ are all the places that occur in L, and four special places p, $\overline{p}$, p', $\overline{p}'$. Places $\langle p, \overline{p}\rangle$ are used to store the pair of numbers $\langle i, K - i\rangle$ where $\ell_i : R_i$ is the routine being executed, and then places $\langle p', \overline{p}'\rangle$ to store the pair of numbers $\langle i', K' - i'\rangle$ where i' is the current line number in routine R_i and K' is the maximum number of lines in any $R_1, \ldots, R_K$.
- Each transition of $\mathcal{N}(L)$ either executes a transition command $c_{i'}$ inside some R_i ensuring that $\langle p, \overline{p}\rangle$ contains $\langle i, K - i\rangle$ and modifying the contents of $\langle p', \overline{p}'\rangle$ from $\langle i', K' - i'\rangle$ to $\langle i' + 1, K' - (i' + 1)\rangle$, or similarly executes a nondeterministic jump command.

Initial and Final Tape Contents. We shall refer to the special p, $\overline{p}$, p', $\overline{p}'$ as *control places*, to the rest as *tape places*, and to markings of the latter places as *tape contents*. For two tape contents M and M', we say that a routine $\ell_i : R_i$ can *compute* M' from M if and only if $\mathcal{N}(L)$ can reach M' with the control at the last line of R_i from M with the control at the first line of R_i; when no M' is computable from M by $\ell_i : R_i$, we say that the routine *cannot terminate* from M.

Interfaces and Compositions of Libraries. For a library L, let us write $\Lambda_{\text{in}}(L)$ (resp., $\Lambda_{\text{out}}(L)$) for the set of all routine labels that are invoked (resp., provided) in L. We say that libraries L_0 and L_1 are *compatible* if and only if $\Lambda_{\mathrm{in}}(L_0)$ is contained in $\Lambda_{\mathrm{out}}(L_1)$. In that case, we can compose them to produce a library $L_0 \circ L_1$ in which tape contents of L_1 persist between successive invocations of its routines, as follows:

- $\Lambda_{\mathrm{in}}(L_0 \circ L_1) = \Lambda_{\mathrm{in}}(L_1)$ and $\Lambda_{\mathrm{out}}(L_0 \circ L_1) = \Lambda_{\mathrm{out}}(L_0)$.
- $L_0 \circ L_1$ has an additional place w used to store the name space of L_0 (i.e., for each name manipulated by L_0, one token labelled by it) and an additional place $\overline{w}$ for the same purpose for L_1.
- For each routine $\ell \ : \ R$ of L_0, the corresponding routine $\ell \ : \ R \circ L_1$ of $L_0 \circ L_1$ is obtained by ensuring that the transition commands in R (resp., L_1) maintain the name space stored on place w (resp., $\overline{w}$), and then inlining the subroutine calls in R.

5.3 Counter Libraries

Our main technical objective, after which it will be easy to arrive at the claimed lower bound for νPN coverability, is to construct libraries that implement increments, decrements and zero tests on a pair of counters whose values range up to a bound which is doubly-Ackermannian in the sizes of the libraries.

To begin, we define the general notion of libraries that provide the operations we need on a pair of bounded counters, as well as what it means for a stand-alone such library to be correct up to a specific bound. A key step is then to consider counter libraries that may not be programs, i.e. may invoke operations on another pair of counters (which we call *auxiliary*). We define correctness of such libraries also, where the bounds of the provided counters may depend on the bounds of the auxiliary counters.

As illustrations of both notions of correctness, we provide examples that will moreover be used in the sequel.

Interfaces of Counter Libraries. Letting Γ denote the set of labels of operations on pairs of bounded counters

$$\Gamma \stackrel{\text{def}}{=} \{\mathit{init}, \mathit{eq}, i.\mathit{inc}, i.\mathit{dec}, i.\mathit{iszero}, i.\mathit{ismax} \ : \ i \in \{1,2\}\} \ ,$$

we regard L to be a *counter* library if and only if $\Lambda_{\mathrm{out}}(L) = \Gamma$ and $\Lambda_{\mathrm{in}}(L) \subseteq \Gamma$.

Correct Counter Programs. When L is also a program, and N is a positive integer, we say that L is N*-correct* if and only if, after initialisation, the routines behave as expected with respect to the bound N. Namely, for every tape contents M which can be computed from the empty tape contents by a sequence σ of operations from Γ, provided *init* occurs only as the first element of σ and letting n_i be the difference between the numbers of occurrences in σ of $i.\mathit{inc}$ and $i.\mathit{dec}$, we must have for both $i \in \{1,2\}$:

- eq can terminate from M if and only if $n_1 = n_2$;
- $i.\mathit{inc}$ can terminate if and only if $n_i < N-1$;
- $i.\mathit{dec}$ can terminate if and only if $n_i > 0$;
- $i.\mathit{iszero}$ can terminate if and only if $n_i = 0$;
- $i.\mathit{ismax}$ can terminate if and only if $n_i = N-1$.

Example: Enumerated Counter Program. For any positive integer N, it is trivial to implement a pair of N-bounded counters by manipulating the values and their complements directly. Let $\mathit{Enum}(N)$ be a counter program which uses four places $e_1, \overline{e}_1, e_2, \overline{e}_2$ and such that for both $i \in \{1,2\}$:

- routine *init* puts $N-1$ tokens onto $\overline{e}_1$ and $N-1$ tokens onto $\overline{e}_2$, all carrying a fresh name d;
- routine eq guesses $n \in \{0,\ldots,N-1\}$, takes $\langle n, N-1-n, n, N-1-n\rangle$ tokens from places $\langle e_1, \overline{e}_1, e_2, \overline{e}_2\rangle$, and then puts them back;
- routine $i.\mathit{inc}$ moves a token from $\overline{e}_i$ to e_i;
- routine $i.\mathit{dec}$ moves a token from e_i to $\overline{e}_i$;
- routine $i.\mathit{iszero}$ takes $N-1$ tokens from place $\overline{e}_i$ and then puts them back;
- routine $i.\mathit{ismax}$ takes $N-1$ tokens from place e_i and then puts them back.

It is simple to verify that $\mathit{Enum}(N)$ is computable in space logarithmic in N, and that:

Lemma 10. *For every N, the counter program $\mathit{Enum}(N)$ is N-correct.*

Note that the size of $\mathit{Enum}(N)$ is at least polynomial in N, whereas our technical aim is to build correct counter programs whose bounds are doubly-Ackermannly larger than their sizes.

Correct Counter Libraries. Given a counter library L, and given a function $F\colon \mathbb{N}^+ \to \mathbb{N}^+$, we say that L is F*-correct* if and only if, for all N-correct counter programs C, $L \circ C$ is $F(N)$-correct.

We employ Acker, a counter library such that the bound of the provided counters equals the Ackermann function $A_\omega(N)$ applied to the bound N of the auxiliary counters. This is an adaptation of the construction by Schnoebelen (2010) of Ackermannian values in reset Petri nets, using the addressing mechanism described in Section 2.3 to simulate an N-dimensional reset Petri net; see Appendix C for details.

Lemma 11. *The counter library Acker is A_ω-correct.*

5.4 A Star Operator

The most complex part of our construction is an operator $-^*$ whose input is any counter library L. Its output L^* is also a counter library, which essentially consists of an arbitrary number of copies of L composed in sequence. Namely, for any N-correct counter program C, the counter operations provided by $L^* \circ C$ behave in the same way as those provided by

$$\overbrace{L \circ \cdots \circ L}^{N} \circ \mathit{Enum}(1).$$

Hence, when L is F-correct, we have that L^* is F'-correct, where $F'(x) = F^x(1)$. Recall that $\mathit{Enum}(1)$ provides trivial counters, i.e. with only one possible value, so its testing operations are essentially no-ops whereas its increments and decrements cannot terminate successfully.

The main idea for the definition of L^* is to combine a distinguishing of name spaces as in the composition of libraries with an arbitrarily wide indexing mechanism like the one employed in Section 2.3. The key insight here is that a whole collection of 'addressing places' $\langle a_i, \bar{a}_i\rangle_i$ as used in Section 2.3 can be simulated by adding one layer of addressing.

More precisely, numbering the copies of L by $0,\ldots,N-1$, writing $\ell_1 \ : \ R_1, \ldots, \ell_K \ : \ R_K$ for the routines of L where $\ell_1 = \mathit{init}$,[1] and writing K' for the maximum number of lines in any $R_1,\ldots,R_K$, L^* can maintain the control and the tape of each copy of L in the implicit composition as follows:

- To record that the program counter of the ith copy of L is currently in routine $\ell_j \ : \ R_j$ at line j', $\langle i, N-1-i, j, K-j, j', K'-j', 1\rangle$ tokens carrying a separate name d_i are kept on special places $\langle w, \overline{w}, p, \overline{p}, p', \overline{p}', \overline{t}\rangle$.
- The current height i of the stack of subroutine calls is kept in one of the auxiliary counters, and we have that:

[1] Since L is a counter library, $K = 10$.

- for all $i' < i$, the program counter of the i'th copy of L is at some subroutine invocation call ℓ' such that the program counter of the $(i'+1)$th copy of L is in the routine named ℓ';
- for all $i' > i$, there are $\langle i, N-1-i, 0, 0, 0, 0, 1\rangle$ tokens carrying d_i on places $\langle w, \overline{w}, p, \overline{p}, p', \overline{p}', \overline{t}\rangle$.

- For every name manipulated by the ith copy of L, $\langle i, N-1-i, 1\rangle$ tokens carrying it are kept on special places $\langle w, \overline{w}, t\rangle$.

To define L^*, its places are all the places that occur in L, plus nine special places w, $\overline{w}$, p, $\overline{p}$, p', $\overline{p}'$, $\overline{t}$, t and f. Writing N for the bound of the auxiliary counters and I, I' for the two auxiliary counters, routine $\ell_j : R^*_j$ of L^* is defined to execute:

- If $\ell_j = init$, initialise the auxiliary counters, and then using the auxiliary counters and place f, for each $i \in \{0, \ldots, N-1\}$, put $\langle i, N-1-i, 1\rangle$ tokens carrying a fresh name d_i onto places $\langle w, \overline{w}, \overline{t}\rangle$.
- Put $\langle j, K-j, 1, K'-1\rangle$ tokens carrying d_0 onto places $\langle p, \overline{p}, p', \overline{p}'\rangle$. ($I$ will always be 0 at this point.)
- Repeatedly, using I' and place f, identify j and j' such that there are $\langle I, N-1-I, j, K-j, j', K'-j', 1\rangle$ tokens carrying d_I on places $\langle w, \overline{w}, p, \overline{p}, p', \overline{p}', \overline{t}\rangle$, and advance the Ith copy of L by performing the command c at line j' in routine $\ell_j : R_j$ of L as follows:
  - If c is a νPN transition, use I' and place f to maintain the Ith name space, i.e. to ensure that all names manipulated by c have $\langle I, N-1-I, 1\rangle$ tokens on places $\langle w, \overline{w}, t\rangle$.
  - If c is a nondeterministic jump goto G, choose $j^\ddagger \in G$ and ensure that there are $\langle j^\ddagger, K'-j^\ddagger\rangle$ tokens carrying d_I on places $\langle p', \overline{p}'\rangle$.
  - If c is a subroutine invocation call $\ell_{j^\dagger}$ and $I < N-1$, put $\langle j^\dagger, K-j^\dagger, 1, K'-1\rangle$ tokens carrying d_{I+1} on places $\langle p, \overline{p}, p', \overline{p}'\rangle$, and increment I.
  - If c is a subroutine invocation call ℓ', $I = N-1$ and ℓ' is not an increment or a decrement (of the trivial counter program $Enum(1)$), simply increment the program counter by moving a token carrying d_I from place $\overline{p}'$ to place p'. When ℓ' is an increment or a decrement, L^* blocks at this point.
  - In the remaining case, c is return. Remove the tokens carrying d_I from places $\langle p, \overline{p}, p', \overline{p}'\rangle$. If $I > 0$, move a token carrying d_{I-1} from $\overline{p}'$ to place p' and decrement I. Otherwise, exit the loop.

We observe that L^* is computable from L in logarithmic space.

Lemma 12. *For every F-correct counter library L, we have that L^* is $\lambda x.F^x(1)$-correct.*

Proof. We argue by induction on N that, for every N-correct counter program C, $L^* \circ C$ is $F^N(1)$-correct.

The base case $N = 1$ is straightforward. Suppose C is 1-correct, i.e. provides counters with only one possible value. By the definition of L^*, when the bound of the auxiliary counters is 1, only one copy of L is simulated. Hence $L^* \circ C$ as a counter program is indistinguishable from $L \circ Enum(1)$. The latter is $F^N(1)$-correct, i.e. $F(1)$-correct, because L is assumed F-correct and $Enum(1)$ is 1-correct by Lemma 10.

For the inductive step, suppose C is $(N+1)$-correct. For any tape contents M of $L^* \circ C$ and $i \in \{0, \ldots, N\}$, let M_i denote the subcontents belonging to the ith copy of L, i.e. the restriction of M to the names that label $\langle i, N-i, 1\rangle$ tokens on places $\langle w, \overline{w}, t\rangle$ and to the places of L.

By the inductive hypothesis and Lemma 10, $L^* \circ Enum(N)$ is $F^N(1)$-correct, and so $L \circ (L^* \circ Enum(N))$ is $F^{N+1}(1)$-correct. For any tape contents M' of the latter counter program and $i \in \{0, \ldots, N\}$, let M'_i denote: if $i = 0$, the subcontents of the left-hand L; otherwise, the subcontents belonging to the $(i-1)$th copy of L in $L^* \circ Enum(N)$.

The required conclusion that $L^* \circ C$ is $F^{N+1}(1)$-correct is implied by the next claim:

Claim 12.1. For every tape contents M and M' which $L^* \circ C$ and $L \circ (L^* \circ Enum(N))$ (respectively) can compute from the empty tape contents by a sequence σ of counter operations where $init$ occurs only as the first element, we have that:

1. $M_i = M'_i$ for all $i \in \{0, \ldots, N\}$;
2. for every counter operation $op \neq init$, $L^* \circ C$ can complete op from M if and only if $L \circ (L^* \circ Enum(N))$ can complete op from M'.

Proof of Claim 12.1. To establish the claim, we first define a correspondence between

- tape contents M of $L^* \circ C$ that occur at the main loop of a routine op (observe that the latter is then determined by M) and
- triples $\langle j, j', M'\rangle$ where M' is a tape contents of $L \circ (L^* \circ Enum(N))$ that occurs in routine ℓ_j, either at line j' of the left-hand L which is not a subroutine call, or at the main loop of a routine of $L^* \circ Enum(N)$ that is inlined for the invocation at line j' in routine ℓ_j of L.

To that end, define $M \cong \langle j, j', M'\rangle$ if and only if

(a) the control of the 0th copy of L in M is in routine $\ell_j = op$ at line j',
(b) if the height I of the stack in M (i.e. the value of the first counter of C) is at least 1, then the stack of $L^* \circ Enum(N)$ in M' has height $I-1$ and the control of any $(i+1)$th copy of L in M is the same as the control of the ith right-hand copy of L in M', and
(c) $M_i = M'_i$ for all $i \in \{0, \ldots, N\}$.

The claim now follows by induction on the length of the sequence σ, where both the base case and the inductive step are completed by checking that the correspondence $\cong$ is a bisimulation. ☐

5.5 Doubly-Ackermannian Minsky Machines

We are now equipped to reduce from the following $\mathbf{F}_{\omega\cdot 2}$-complete problem (cf. Schmitz 2016, Section 2.3.2):

> Given a deterministic Minsky machine $\mathcal{M}$, does it halt while the sum of counters is less than $A_{\omega\cdot 2}(|\mathcal{M}|)$?

and thereby establish our lower bound. The idea here is classical: simulate $\mathcal{M}$ by a reset Petri net on a doubly-Ackermannian budget, and if it halts then check that the simulation was accurate.

Theorem 13. *The coverability problem for νPNs is $\mathbf{F}_{\omega\cdot 2}$-hard.*

Proof. Suppose $\mathcal{M}$ a deterministic Minsky machine.

Let $L_{|\mathcal{M}|}$ be the counter library $\overbrace{(\cdots(Acker^*)^*\cdots)^*}^{|\mathcal{M}|+1}$. By Lemmas 10, 11 and 12, we have that $L_{|\mathcal{M}|}$ is $A_{\omega+|\mathcal{M}|+1}$-correct and that $L_{|\mathcal{M}|} \circ Enum(|\mathcal{M}|)$ is $A_{\omega\cdot 2}(|\mathcal{M}|)$-correct.

Finally, let $Sim(\mathcal{M})$ be a one-routine library that uses one of the pair of counters provided by the counter program $L_{|\mathcal{M}|} \circ Enum(|\mathcal{M}|)$ as follows:

- Initialise $L_{|\mathcal{M}|} \circ Enum(|\mathcal{M}|)$.

- Simulate $\mathcal{M}$ where zero tests are performed as resets (cf. Section 2.3) and where the difference between the total number of increments and the total number of decrements is maintained in a counter T of $L_{|\mathcal{M}|} \circ \mathit{Enum}(|\mathcal{M}|)$. Any attempt to increment T beyond its maximum value blocks the simulation.
- If $\mathcal{M}$ halts, check that the sum of its counters is at least T, i.e. decrease T to zero while at each step decrementing some counter of $\mathcal{M}$.

Observe that the latter check succeeds if and only if there was no reset of a non-zero counter, i.e. all zero tests in the simulation were correct. Hence, $\mathcal{M}$ halts while the sum of its counters is less than $A_{\omega \cdot 2}(|\mathcal{M}|)$ if and only if the one-routine program

$$\mathit{Test}(\mathcal{M}) = \mathit{Sim}(\mathcal{M}) \circ (L_{|\mathcal{M}|} \circ \mathit{Enum}(|\mathcal{M}|))$$

can terminate, i.e. the νPN $\mathcal{N}(\mathit{Test}(\mathcal{M}))$ can cover the marking in which the control places point to the last line of $\mathit{Test}(\mathcal{M})$.

Since the star operator is computable in logarithmic space and increases the number of places by adding a constant, we have that the counter library $L_{|\mathcal{M}|}$ and thus also the νPN $\mathcal{N}(\mathit{Test}(\mathcal{M}))$ are computable in time elementary in $|\mathcal{M}|$, and that their numbers of places are linear in $|\mathcal{M}|$.

We conclude the $\mathbf{F}_{\omega \cdot 2}$-hardness by the closure under any sub-doubly-Ackermannian reduction (i.e. in $\mathscr{F}_{<\omega \cdot 2}$), and therefore certainly any elementary one (Schmitz 2016, Section 2.3.1). $\square$

6. Concluding Remarks

In this paper, we have shown that coverability in ν-Petri nets is complete for double Ackermann time, i.e. $\mathbf{F}_{\omega \cdot 2}$-complete. In order to solve this open problem, we have applied a new technique to analyse the complexity of the backward coverability algorithm using ideal representations—thereby demonstrating the versatility of this technique designed in (Lazić and Schmitz 2015)—, and pushed for the first time the 'object oriented' construction of Lipton (1976) beyond Ackermann-hardness. This is also the first known instance of a natural decision problem for double Ackermann time.

Our $\mathbf{F}_{\omega \cdot 2}$ upper bound furthermore improves the best known upper bound for coverability in unordered data Petri nets. In this case however, the currently best known lower bound is hardness for $\mathbf{F}_3$, which was proven by Lazić et al. already in 2008, leaving quite a large complexity gap.

Acknowledgements

The authors thank R. Meyer and F. Rosa-Velardo for their insights into the relationships between π-calculus and νPNs, and J. Goubault-Larrecq, P. Karandikar, K. Narayan Kumar, and Ph. Schnoebelen for sharing their draft paper.

References

P. A. Abdulla, K. Čerāns, B. Jonsson, and Y.-K. Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Inform. and Comput.*, 160 (1–2):109–127, 2000.

M. Blondin, A. Finkel, and P. McKenzie. Handling infinitely branching WSTS. In *Proc. ICALP 2014*, volume 8573 of *Lect. Notes in Comput. Sci.*, pages 13–25, 2014.

R. Bonnet. On the cardinality of the set of initial intervals of a partially ordered set. In *Infinite and finite sets: to Paul Erdős on his 60th birthday, Vol. 1*, Coll. Math. Soc. János Bolyai, pages 189–198. North-Holland, 1975.

L. Bozzelli and P. Ganty. Complexity analysis of the backward coverability algorithm for VASS. In *Proc. RP 2011*, volume 6945 of *Lect. Notes in Comput. Sci.*, pages 96–109. Springer, 2011.

P. Chambart and Ph. Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *Proc. LICS 2008*, pages 205–216. IEEE Press, 2008.

E. A. Cichoń and E. Tahhan Bittar. Ordinal recursive bounds for Higman's Theorem. *Theor. Comput. Sci.*, 201(1–2):63–84, 1998.

N. Decker and D. Thoma. On freeze LTL with ordered attributes. In *Proc. FoSSaCS 2016*, Lect. Notes in Comput. Sci. Springer, 2016. URL `http://arxiv.org/abs/1504.06355`. To appear.

D. Figueira, S. Figueira, S. Schmitz, and Ph. Schnoebelen. Ackermannian and primitive-recursive bounds with Dickson's Lemma. In *Proc. LICS 2011*, pages 269–278. IEEE Press, 2011.

A. Finkel and J. Goubault-Larrecq. Forward analysis for WSTS, part I: Completions. In *Proc. STACS 2009*, volume 3 of *Leibniz Int. Proc. Inf.*, pages 433–444. LZI, 2009.

A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1–2):63–92, 2001.

A. Finkel, P. McKenzie, and C. Picaronny. A well-structured framework for analysing Petri net extensions. *Inform. and Comput.*, 195(1–2):1–29, 2004.

J. Goubault-Larrecq, P. Karandikar, K. Narayan Kumar, and Ph. Schnoebelen. The ideal approach to computing closed subsets in well-quasi-orderings. In preparation, 2016. See also an earlier version in:
J. Goubault-Larrecq. On a generalization of a result by Valk and Jantzen. Research Report LSV-09-09, LSV, ENS Cachan, 2009. URL `http://www.lsv.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2009-09.pdf`.

C. Haase, S. Schmitz, and Ph. Schnoebelen. The power of priority channel systems. *Logic. Meth. in Comput. Sci.*, 10(4:4):1–39, 2014.

S. Haddad, S. Schmitz, and Ph. Schnoebelen. The ordinal recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *Proc. LICS 2012*, pages 355–364. IEEE Press, 2012.

G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Math. Soc.*, 3(2):326–336, 1952.

R. Lazić and S. Schmitz. The ideal view on Rackoff's coverability technique. In *Proc. RP 2015*, volume 9328 of *Lect. Notes in Comput. Sci.*, pages 1–13. Springer, 2015.

R. Lazić, T. Newcomb, J. Ouaknine, A. Roscoe, and J. Worrell. Nets with tokens which carry data. *Fund. Inform.*, 88(3):251–274, 2008.

R. Lazić, J. Ouaknine, and J. Worrell. Zeno, Hercules and the Hydra: Downward rational termination is Ackermannian. In *Proc. MFCS 2013*, volume 8087 of *Lect. Notes in Comput. Sci.*, pages 643–654. Springer, 2013. Revised version to appear in *ACM Trans. Comput. Logic*.

R. Lipton. The reachability problem requires exponential space. Technical Report 62, Yale University, 1976.

C. Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6(2):223–231, 1978.

F. Rosa-Velardo. Ordinal recursive complexity of unordered data nets. Technical Report TR-4-14, Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, 2014. URL `http://antares.sip.ucm.es/frosa/docs/complexityUDN.pdf`.

F. Rosa-Velardo and D. de Frutos-Escrig. Name creation vs. replication in Petri net systems. *Fund. Inform.*, 88(3):329–356, 2008.

F. Rosa-Velardo and D. de Frutos-Escrig. Decidability and complexity of Petri nets with unordered data. *Theor. Comput. Sci.*, 412(34):4439–4451, 2011.

F. Rosa-Velardo and M. Martos-Salgado. Multiset rewriting for the verification of depth-bounded processes with name binding. *Inform. and Comput.*, 215:68–87, 2012.

S. Schmitz. Complexity hierarchies beyond Elementary. *ACM Trans. Comput. Theory*, 2016. URL `http://arxiv.org/abs/1312.5686`. To appear.

S. Schmitz and Ph. Schnoebelen. Multiply-recursive upper bounds with Higman's Lemma. In *Proc. ICALP 2011*, volume 6756 of *Lect. Notes in Comput. Sci.*, pages 441–452. Springer, 2011.

S. Schmitz and Ph. Schnoebelen. Algorithmic aspects of WQO theory. Lecture notes, 2012. URL `http://cel.archives-ouvertes.fr/cel-00727025`.

S. Schmitz and Ph. Schnoebelen. The power of well-structured systems. In *Proc. Concur 2013*, volume 8052 of *Lect. Notes in Comput. Sci.*, pages 5–24. Springer, 2013.

Ph. Schnoebelen. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In *Proc. MFCS 2010*, volume 6281 of *Lect. Notes in Comput. Sci.*, pages 616–628. Springer, 2010.

S. S. Wainer. A classification of the ordinal recursive functions. *Arch. Math. Logic*, 13(3):136–153, 1970.

A. Ideal Representations

We gather in this appendix some technical material that deals with effective ideal representations.

A.1 Existential Successors

Lemma 14 (Existential Successors are Computable)**.** *The ideal decomposition of* $\downarrow\mathrm{Post}_\exists(I)$ *is computable for all ideals* I *in a* ν*PN.*

Proof sketch. Let us define for any $S \subseteq \mathit{Confs}$ and $t \in T$

$$\mathrm{Post}^t_\exists \stackrel{\text{def}}{=} \{M' \in \mathit{Confs} \mid \exists M \in S \,.\, M \xrightarrow{t} M'\} \;. \tag{17}$$

Then

$$\mathrm{Post}_\exists(S) = \bigcup_{t\in T} \mathrm{Post}^t_\exists(S) \;, \tag{18}$$

hence it suffices to show that $\downarrow\mathrm{Post}^t_\exists(I)$ is computable for any $t \in T$.

Let $I = [\![B, S]\!]$ for $(B, S) \in \mathit{XConfs}$. Observe that $\mathrm{Post}^t_\exists(I)$ is non-empty if and only if the set

$$\mathcal{E} \stackrel{\text{def}}{=} \{E \in S^\circledast \mid B \oplus E \sqsupseteq \mathsf{in}(t)\} \tag{19}$$

is non-empty. In such a case, for any $E \in \mathcal{E}$ we can find a mode e and construct (the uniquely determined) $B' \in (\mathbb{N}^P_\omega)^\circledast$ using equations (2) and (3). Thus $B \oplus E = B'' \oplus \bigoplus_{x\in\mathcal{X}(t)} \boldsymbol{e}_{e(x)}$ and $B' = B'' \oplus \mathsf{out}_\Upsilon(t) \oplus \bigoplus_{x\in\mathcal{X}(t)} \boldsymbol{e}'_{e(X)}$ with $\boldsymbol{e}'_{e(x)} = \boldsymbol{e}_{e(x)} - F_x(P, t) + F_x(t, P)$ for all $x \in \mathcal{X}(t)$. This yields

$$\downarrow\mathrm{Post}^t_\exists(I) = \bigcup_{E\in\mathcal{E}} \bigcup_{e} \{[\![B', S]\!] \mid B \oplus E \xrightarrow{e,t} B'\} \;. \tag{20}$$

We want to show that the infinite union in (20) can be replaced by a finite one:

$$\downarrow\mathrm{Post}^t_\exists(I) = \bigcup_{|E|\leq|\mathsf{in}(t)|} \bigcup_{e} \{[\![B', S]\!] \mid B \oplus E \xrightarrow{e,t} B'\} \;. \tag{21}$$

Indeed, note that the set $\mathcal{E}$ is infinite but upwards-closed, hence it has a finite set of minimal elements $\min_\sqsubseteq \mathcal{E}$. Also note that any E in $\min_\sqsubseteq \mathcal{E}$ has length at most $|\mathsf{in}(t)|$. Furthermore, if $E \sqsubseteq E'$ with $E \in \min_\sqsubseteq \mathcal{E}$, then since (B, S) is reduced, $E' = E \oplus E''$ for some $E'' \in S^\circledast$, hence if $B \oplus E \xrightarrow{e,t} B'$ then $B \oplus E' = B \oplus E \oplus E'' \xrightarrow{e,t} B' \oplus E''$ with $E'' \in S^\circledast$, and therefore $[\![B', S]\!] = [\![B' \oplus E'', S]\!]$.

Conversely, if $B \oplus E' \xrightarrow{e',t} B''$ for some mode e', then by splitting E' into $E \oplus E''$ and B'' into $B' \oplus E''$ where E is the part of E' ranged over by e' (hence $|E| \leq |\mathsf{in}(t)|$), we see that $B \oplus E \xrightarrow{e',t} B'$. Therefore $[\![B', S]\!] = [\![B'', S]\!]$ since $E'' \in S^\circledast$.

To conclude, there are finitely many modes e for t and $B \oplus E$ to select from in (21), thus this is a finite union, and it is computable. (Note that the extended configurations thus constructed might have to be reduced.) ☐

Lemma 6 (Ideal Steps are Star-Monotone)**.** *If* I *and* J *are two ideals and* $I \to J$*, then* $S(I) \subseteq_H S(J)$*.*

Proof. Let $I = [\![B, S]\!]$. Simply note that $J = [\![B', S']\!]$ computed by (21) has $S = S'$. ☐

A.2 Universal Predecessors

We establish here both two results in a single stroke, by showing that, if (B, S) is a representative of an ideal in $\mathrm{Pre}_\forall(D)$ then $\|B, S\| \leq \|D\| + |N|$ (see Lemma 15). As a first consequence $\mathrm{Pre}_\forall(D)$ is effective by brute-force search, proving Corollary 18. As a second consequence, the descending chain $D_0 \supsetneq D_1 \supsetneq \cdots$ computed by the backward coverability algorithm is controlled, proving Lemma 2.

Lemma 15 (Universal Predecessors are Controlled)**.** *Let* $I = [\![B, S]\!]$ *be an ideal in the decomposition of* $\mathrm{Pre}_\forall(D)$ *for* $(B, S) \in \mathit{XConfs}$*. Then* $\|B, S\| \leq \|D\| + |N|$*.*

Proof. Let us write $n \stackrel{\text{def}}{=} \|D\| + |N|$. Since I is a maximal ideal of the decomposition of $\mathrm{Pre}_\forall(D)$, we can write $\mathrm{Pre}_\forall(D) = I \cup D'$ with $I \not\subseteq D'$. The following claim will be handy:

Claim 15.1. If $U \subseteq I$ is non-empty and upwards-closed *inside* I, i.e. if $M \in U$ and $M' \in I$ are such that $M \sqsubseteq M'$, then $M' \in U$. Then $U \not\subseteq D'$.

Proof of Claim 15.1. Since I is maximal, there exists some element $M_I \in I \setminus D'$. Furthermore, since D' is downwards-closed, $\uparrow M_I \cap D' = \emptyset$.

By hypothesis, there exists some $M \in U$. Because I is directed, there exists $M' \in I$ such that $M' \sqsupseteq M$ and $M' \sqsupseteq M_I$. As U is upwards-closed and by definition of M_I, $M' \in U \setminus D'$. [15.1]

Let us return to the main proof. Let

$$C \stackrel{\text{def}}{=} \max\{\boldsymbol{e}(p) \mid \boldsymbol{e} \in \mathit{Supp}(B) \cup S \wedge p \in P \wedge \boldsymbol{e}(p) < \omega\} \tag{22}$$

be the maximal constant that appears in an extended marking of (B, S). Assume that $\|B, S\| > n$ to derive a contradiction. By the definition of $\|B, S\|$, this can only happen in three different situations:

1. there is an extended marking $\boldsymbol{e} \in \mathit{Supp}(B)$ and a place $p \in P$ such that $\boldsymbol{e}(p) = C > n$, or
2. there is an extended marking $\boldsymbol{e} \in S$ and a place $p \in P$ such that $\boldsymbol{e}(p) = C > n$, or
3. $|B| > n$.

Let us show that we can reach a contradiction in all cases.

Case 1. For any $\boldsymbol{e}' \in \mathbb{N}_\omega^P$, define the marking $\boldsymbol{e}'_{\restriction\mathbb{N}}$ as the 'finite projection' of $\boldsymbol{e}'$: for all $p' \in P$,

$$\boldsymbol{e}'_{\restriction\mathbb{N}}(p') \stackrel{\text{def}}{=} \begin{cases} C+2 & \text{if } \boldsymbol{e}'(p') = \omega\,, \\ \boldsymbol{e}'(p') & \text{otherwise.} \end{cases} \tag{23}$$

Let $B = [\boldsymbol{e}_1 \cdots \boldsymbol{e}_k]$; then $B_{\restriction\mathbb{N}} \stackrel{\text{def}}{=} [\boldsymbol{e}_{1\,\restriction\mathbb{N}} \cdots \boldsymbol{e}_{k\,\restriction\mathbb{N}}]$. Since (B, S) is reduced, for all $\boldsymbol{e}' \in \mathit{Supp}(S)$ and all $1 \leq j \leq k$, $\boldsymbol{e}_j \not\leq \boldsymbol{e}'$, from which we deduce that $\boldsymbol{e}_{j\,\restriction\mathbb{N}} \not\leq \boldsymbol{e}'$.

Let us define

$$\mathcal{M} \stackrel{\text{def}}{=} I \cap \uparrow B_{\restriction\mathbb{N}}\ . \tag{24}$$

By maximality of I, $\mathcal{M}' \stackrel{\text{def}}{=} \mathcal{M} \setminus D'$ is non-empty, since $\mathcal{M}$ is non-empty and upwards-closed in I and Claim 15.1 applies.

We distinguish two subcases, depending on whether there exists $t \in T$ such that $\mathcal{M}' \cap \uparrow \mathsf{in}(t) \neq \emptyset$. We shall treat the subcase where there does exist such a t; the other subcase is similar and we will treat it for Case 3. Let thus t be such that there exists $M \in \mathcal{M}'$ with $M \sqsupseteq \mathsf{in}(t)$. We decompose M into $[\boldsymbol{m}] \oplus M''$ where $\boldsymbol{m} \stackrel{\text{def}}{=} \boldsymbol{e}_{\restriction\mathbb{N}}$, thus $\boldsymbol{m}(p) = \boldsymbol{e}(p) = C$.

By definition of $\mathrm{Pre}_\forall(D)$, for all $t \in T$ fireable from M, for all modes e of t and M, the configuration M' defined by $M \xrightarrow{e,t} M'$ in Equation 3 must belong to D. More precisely, M' must belong to some ideal $I' = [\![B', S']\!]$ from the decomposition of D, i.e. there exists $E' \in S'^{\circledast}$ and an embedding f from M' into $B \oplus E'$. Call $\boldsymbol{m}'$ the marking associated to $\boldsymbol{m}$ by (3) in M'. The embedding f then maps $\boldsymbol{m}'$ to some $\boldsymbol{e}' \in \mathit{Supp}(B') \cup S'$. Because $\boldsymbol{m}(p) > n$, $\boldsymbol{m}'(p) > \|D\|$, and thus $\boldsymbol{e}'(p) = \omega$.

Let us define the configuration $M_{+1} \stackrel{\text{def}}{=} [\boldsymbol{m}_{+1}] \oplus M''$ where $\boldsymbol{m}_{+1}(p') = \boldsymbol{m}(p)$ for all $p' \neq p$ and $\boldsymbol{m}_{+1}(p) = C + 1$.

- For all $t \in T$ fireable from M_{+1}, we could also have fired from M since the only difference between the two configurations lies in the values of $\boldsymbol{m}(p)$ and $\boldsymbol{m}_{+1}(p)$, but both are greater than $|N| \geq \|\mathsf{in}(t)\|$. Therefore, for each such $t \in T$, we can choose the same mode e and construct the configuration M'_{+1} containing a marking $\boldsymbol{m}'_{+1}$. By picking the same E' and the same f, we map this $\boldsymbol{m}'_{+1}$ to $\boldsymbol{e}'$ with $\boldsymbol{e}'(p) = \omega$. Hence M'_{+1} belongs to $I' \subseteq D$. All this holds for all t, hence $M_{+1} \in \mathrm{Pre}_\forall(D)$.
- $M_{+1} \sqsupseteq M$ thus $M_{+1} \notin D'$,
- Observe that $M_{+1} \sqsupseteq B_{\restriction\mathbb{N}}$, which forces any putative embedding from M_{+1} into $B \oplus E$ for some $E \in S^{\circledast}$ to map $\boldsymbol{m}_{+1}$ into $\boldsymbol{e}$. But $\boldsymbol{m}_{+1}(p) > \boldsymbol{e}(p)$ and no such embedding is possible, therefore $M_{+1} \notin I$.

This contradicts $\mathrm{Pre}_\forall(D) = I \cup D'$.

Case 2. Similar to case 1; define $\mathcal{M}$ as $I \cap \uparrow(B_{\restriction\mathbb{N}} \oplus [\boldsymbol{e}_{\restriction\mathbb{N}}])$ instead.

Case 3. Define as before a set of configurations

$$\mathcal{M} \stackrel{\text{def}}{=} \{(M_1 \oplus M_2) \in I \mid M_1, M_2 \in \mathit{Confs} \wedge |M_1| = |B| \wedge \mathit{Supp}(M_1) \cap (\bigcup_{\boldsymbol{e} \in S} [\![\boldsymbol{e}]\!]) = \emptyset\}\ . \tag{25}$$

This set is non-empty: (B, S) is reduced, hence for all $\boldsymbol{e} \in \mathit{Supp}(B)$ and all $\boldsymbol{e}' \in S$, $\boldsymbol{e} \not\leq \boldsymbol{e}'$, i.e. there exists some $\boldsymbol{m}$ in $[\![\boldsymbol{e}]\!] \setminus (\bigcup_{\boldsymbol{e}' \in S} [\![\boldsymbol{e}']\!])$. We can therefore build M_1 from those markings. Therefore, $\mathcal{M}' \stackrel{\text{def}}{=} \mathcal{M} \setminus D'$ is also non-empty because $\mathcal{M}$ is upwards-closed inside I and non-empty, hence we meet the conditions of Claim 15.1.

We distinguish again two subcases, depending on whether there exists $t \in T$ such that $\mathcal{M}' \cap \uparrow\mathsf{in}(t) \neq \emptyset$. Let us treat the subcase where no such t exists and leave the other one as an exercise for the reader. Then $\mathcal{M}' \subseteq \bigcap_{t\in T}(\mathit{Confs} \setminus \uparrow\mathsf{in}(t))$. Note that the latter set is included in $\mathrm{Pre}_\forall(D)$, is downwards-closed, and its ideal decomposition can be computed by (CU') and (II).

Since $\mathcal{M}' \neq \emptyset$, there exists $M = M_1 \oplus M_2 \in \mathcal{M}'$, and therefore there exists an ideal $I' = [\![B', S']\!]$ of $\bigcap_{t\in T}(\mathit{Confs} \setminus \uparrow\mathsf{in}(t))$ such that $M \in I'$. Hence there exists $E' \in S'^{\circledast}$ and an embedding f from M to $B' \oplus E'$. Note that (CU') and (II) yield a bound $\|B', S'\| \leq |N|$ (Goubault-Larrecq et al. 2016), hence $|B'| \leq n$ and there must be a marking $\boldsymbol{m} \in \mathit{Supp}(M_1)$ mapped by f to some extended marking $\boldsymbol{e} \in \mathit{Supp}(E') \subseteq S'$.

Define the marking $M_{+1} \stackrel{\text{def}}{=} M \oplus [\boldsymbol{m}]$ with one additional occurrence of $\boldsymbol{m}$.

- $M_{+1} \sqsupseteq B' \oplus E' \oplus [\boldsymbol{e}]$ by suitably extending f, hence $M_{+1} \in I'$, and therefore $M_{+1} \in \mathrm{Pre}_\forall(D)$,
- $M_{+1} \sqsupseteq M$ and therefore $M_{+1} \notin D'$,
- $\boldsymbol{m} \in \mathit{Supp}(M_1)$, thus $\boldsymbol{m} \notin \bigcup_{\boldsymbol{e}\in S}[\![\boldsymbol{e}]\!]$ and therefore $M_{+1} \notin I$.

This contradicts $\mathrm{Pre}_\forall(D) = I \cup D'$. □

Remark 16. One could also observe that $\mathrm{Pre}_\forall(D) = \mathit{Confs} \setminus (\mathrm{Pre}_\exists(\mathit{Confs} \setminus D))$. The effectiveness of $\mathrm{Pre}_\exists(U)$ for an upwards-closed U is standard for backward computations—and rather simpler to reason about than $\mathrm{Pre}_\forall(D)$—, while the construction of a finite basis for the upwards-closed $\mathit{Confs} \setminus D$ is effective according to Goubault-Larrecq et al. (2016). The proof of Lemma 15 might thus be replaced by one that bounds the effect on norms of these operations.

Effective Pre Computations. We deduce the effectivity of $\mathrm{Pre}_\forall$ computations using Lemma 15 and the following lemma:

Lemma 17. *For all ideals I and downwards-closed subsets D of $(\mathit{Confs}, \sqsubseteq)$, $\downarrow\mathrm{Post}_\exists(I) \subseteq D$ iff $I \subseteq \mathrm{Pre}_\forall(D)$.*

Proof of 'if'. Assume that, for all $M \in I$, $M \in \mathrm{Pre}_\forall(D)$, i.e. for all $M' \in \mathit{Confs}$, if $M \to M'$, then $M' \in D$. Let now $M'' \in \downarrow\mathrm{Post}_\exists(I)$: then by assumption there exists $M' \sqsupseteq M''$ and $M \in I$ such that $M \to M'$. Thus $M' \in D$ and since D is downwards-closed, $M'' \in D$ as well. □

Proof of 'only if'. Assume that, whenever $M'' \in \downarrow\mathrm{Post}_\exists(I)$, then $M'' \in D$. This means that, for all M'', if there exists $M' \sqsupseteq M''$ and $M \in I$ such that $M \to M'$, then $M'' \in D$. In particular, choosing $M'' = M'$, we see that for all M' and for all $M \in I$, $M \to M'$ implies $M' \in D$. Hence for all $M \in I$, $M \in \mathrm{Pre}_\forall(D)$. □

Corollary 18 (Universal Predecessors are Computable)**.** *The ideal decomposition of* $\mathrm{Pre}_\forall(D)$ *is computable for all downwards-closed D in a νPN.*

Proof. By Lemma 15, given an ideal representation of D, we know that the ideals of the decomposition of $\mathrm{Pre}_\forall(D)$ are of norm at most $\|D\| + |N|$, hence there are finitely many candidates. For each candidate I, we can effectively check whether $I \subseteq \mathrm{Pre}_\forall(D)$: by Lemma 17, it suffices to check whether $\mathrm{Post}_\exists(I) \subseteq D$, where the inclusion check is effective by (CI) and $\mathrm{Post}_\exists(I)$ is computable by Lemma 14. □

Strong Control. The second corollary of Lemma 15 is a strong control on the norm of the ideals appearing during the execution of the backward coverability algorithm.

Lemma 2 (Strong Control for νPNs)**.** *The descending chain computed by the backward coverability algorithm for a νPN N and target configuration M is strongly (g, n)-controlled for $g(x) \stackrel{\text{def}}{=} x + |N|$ and $n \stackrel{\text{def}}{=} \|M\|$.*

Proof. The initial decomposition for the set $D_0 = \mathit{Confs} \setminus \uparrow M$ is computed via (CU'); by inspecting the construction of Goubault-Larrecq et al. (2016), we can see that this implies $\|D_0\| \leq \|M\|$. Then, $\mathrm{Pre}_\forall(D_k)$ is of norm at most $\|D_k\| + |N|$ by Lemma 15, and therefore $D_{k+1} = D_k \cap \mathrm{Pre}_\forall(D_k)$ computed by (II) is also of norm at most $\|D_k\| + |N|$ (again by inspection of Goubault-Larrecq et al. 2016). □

B. Complexity Upper Bound

We gather in this section the details needed in order to prove Theorem 8.

B.1 Subrecursive Hierarchies

The Hardy Hierarchy. We shall use another subrecursive hierarchy related to the Cichoń functions. Let $h:\mathbb{N}\to\mathbb{N}$ be strictly increasing. The *Hardy* functions $(h^\alpha:\mathbb{N}\to\mathbb{N})_\alpha$ allow to denote large iterates of the function h; see Equation 26 below. The Hardy functions are defined by transfinite induction on their ordinal superscripts by (Cichoń and Tahhan Bittar 1998):

$$h^0(x)\stackrel{\text{def}}{=}x\,,\qquad h^{\alpha+1}(x)\stackrel{\text{def}}{=}h^\alpha(h(x))\,,\qquad h^\lambda(x)\stackrel{\text{def}}{=}h^{\lambda(x)}(x)\,.$$

Observe that $h^k(x)$ for a finite k is simply the kth iterate of h. More generally, the Hardy function at rank α computes the effect of iterating $h_\alpha(x)$ times the function h on x (Cichoń and Tahhan Bittar 1998, Equation 4):

$$h^\alpha(x)=h^{h_\alpha(x)}(x)\;. \tag{26}$$

The Hardy functions grow slightly faster than the corresponding Cichoń functions: using the successor function $H(x)\stackrel{\text{def}}{=}x+1$, $H^\alpha(x)=H_\alpha(x)+x$ (justifying (12)), and for h strictly increasing, $h^\alpha(x)\geq h_\alpha(x)+x$ (this justifies (13)). The Hardy functions h^α are strictly increasing. If $g(x)\leq h(x)$ for all x, then also $g^\alpha(x)\leq h^\alpha(x)$ for all x.

Relativisation. A feature that makes Hardy functions especially attractive is that they *relativise* well: provided $\alpha\cdot\beta$ is 'structured',[2] the βth Hardy function with base function h^α is $(h^\alpha)^\beta=h^{\alpha\cdot\beta}$ (see Cichoń and Tahhan Bittar 1998, Equation 3).

Things are not as clear-cut for Cichoń functions, but we have the following equations when $\alpha+\beta$ and $\alpha\cdot\beta$ are structured:

$$h_{\alpha+\beta}(x)=h_\alpha(h^\beta(x))+h_\beta(x) \tag{27}$$
$$h_{\alpha\cdot\beta}(x)=h_{\alpha\cdot(g_\beta(x))}(x)\qquad\text{where } g(x)\stackrel{\text{def}}{=}h^\alpha(x)\;. \tag{28}$$

The first equation (27) is also stated by Cichoń and Tahhan Bittar (1998, Eq. 5). The second equation seems to be original and shows that $h_{\alpha\cdot\beta}(x)=h_{\alpha\cdot\ell}(x)$, where the coefficient $\ell=g_\beta(x)$ is computed by the βth Cichoń function based on the Hardy function $g(x)=h^\alpha(x)$.

Let us prove (28) by transfinite induction on β. For the base case 0,

$$h_{\alpha\cdot[g_0(x)]}(x)=h_{\alpha\cdot 0}(x)=0\;.$$

For the successor case $\beta+1$,

$$\begin{aligned}
h_{\alpha\cdot[g_{\beta+1}(x)]}(x)&=h_{\alpha\cdot[g_\beta(g(x)+1)]}(x)\\
&=h_{\alpha\cdot[g_\beta(g(x))]+\alpha}(x)\\
&=h_{\alpha\cdot[g_\beta(g(x))]}\big(h^\alpha(x)\big)+h_\alpha(x) &&\text{by (27)}\\
&=h_{\alpha\cdot[g_\beta(h^\alpha(x))]}\big(h^\alpha(x)\big)+h_\alpha(x) &&\text{by definition of } g(x)\stackrel{\text{def}}{=}h^\alpha(x)\\
&=h_{\alpha\cdot\beta}(h^\alpha(x))+h_\alpha(x) &&\text{by ind. hyp.}\\
&=h_{\alpha\cdot\beta+\alpha}(x) &&\text{by (27)}\\
&=h_{\alpha\cdot(\beta+1)}(x)\;.
\end{aligned}$$

Finally, for a limit ordinal λ,

$$\begin{aligned}
h_{\alpha\cdot[g_\lambda(x))}(x)&=h_{\alpha\cdot[g_{\lambda(x)}]}(x)\\
&=h_{\alpha\cdot(\lambda(x))}(x) &&\text{by ind. hyp.}\\
&=h_{(\alpha\cdot\lambda)(x)}(x) &&\text{using Def. 3.1 of Cichoń and Tahhan Bittar (1998)}\\
&=h_{\alpha\cdot\lambda}(x)\;.
\end{aligned}$$

B.2 Proof of Theorem 8

Theorem 8 (Length Function Theorem for Star-Monotone Descending Chains over $(\mathbb{N}^P_\omega)^\circledast$)**.** *Let $n>0$. Any strongly (g,n)-controlled star-monotone descending chain $D_0\supsetneq D_1\supsetneq\cdots\supsetneq D_\ell$ of configurations in $(\mathbb{N}^P_\omega)^\circledast$ has length at most $h_{\omega^{\omega\cdot 2}}(|P|+n)$ for some h primitive-recursive in g.*

Proof. Note that $D_{\ell-1}$ has a proper ideal $I_{\ell-1}$ in its decomposition, otherwise we would have $D_{\ell-1}\subseteq D_\ell$. Since the chain is star-monotone, we can therefore proper ideals I_k from the decomposition of D_k with $S(I_k)\supseteq_H S(I_{k+1})$ for all $0\leq k<\ell-1$.

Let $S_k=S(I_k)$ for all $0\leq k<\ell$ and consider the sequence $S_0\supseteq_H S_1\supseteq_H\cdots\supseteq_H S_{\ell-1}$. We extract a subsequence $S_{i_0}\supsetneq_H S_{i_1}\supsetneq_H\cdots\supsetneq_H S_{i_r}$ such that $S_{i_0}\stackrel{\text{def}}{=}S_0\equiv_H S_1\equiv_H\cdots\equiv_H$

[2] See Cichoń and Tahhan Bittar (1998) for a definition; all the ordinal terms we manipulate in this paper are structured.

$S_{i_1-1} \supsetneq_H S_{i_1} \equiv_H \cdots \equiv_H S_{i_2-1} \supsetneq_H S_{i_2} \cdots S_{i_r-1} \supsetneq_H S_{i_r} \equiv_H \cdots \equiv_H S_{\ell-1}$, where $S \equiv_H S'$ iff $S \subseteq_H S'$ and $S \supseteq_H S'$. In other words, letting $i_{r+1} \stackrel{\text{def}}{=} \ell$, then for all $0 \leq j < r$,

$$S_{i_j} \supsetneq_H S_{i_{j+1}} , \tag{29}$$

and for all $0 \leq j \leq r$ and $i_j \leq k < i_{j+1} - 1$,

$$S_k \equiv_H S_{k+1} . \tag{30}$$

Consider any segment $S_{i_j} \equiv_H \cdots \equiv_H S_{i_{j+1}-1}$ for $0 \leq j \leq r$. Since the descending chain is strongly (g, n)-controlled, this segment is $(g, \|D_{i_j}\|)$-controlled, and we apply the following Claim 18.1 to bound its length $i_{j+1} - i_j$:

Claim 18.1. The length of any segment $D_{i_j} \supsetneq D_{i_j+1} \supsetneq \cdots \supsetneq D_{i_{j+1}-1}$ is bounded by $C_{\omega^\omega}(\|D_{i_j}\|)$ for a strictly increasing function $C(x) \geq g(x)$ primitive recursive in g and $|P|$.

Proof of Claim 18.1. Consider the sequence of multisets of extended markings defined by $B_k \stackrel{\text{def}}{=} B(I_k)$ for all $i_j \leq k < i_{j+1}$:

$$B_{i_j}, B_{i_j+1}, \ldots, B_{i_{j+1}-1} . \tag{31}$$

Let us show that (31) is a bad sequence: let $i_j \leq k < k' < i_{j+1}$: assume for contradiction that $B_k \sqsubseteq B_{k'}$, then since $S_k \equiv_H S_{k'}$, $I_k \subseteq I_{k'}$. But this would violate the fact that I_k is proper and thus $I_k \not\subseteq D_{k'}$. Hence (31) is a $(g, \|D_{i_j}\|)$-controlled bad sequence over $(\mathbb{N}_\omega^P)^\circledast$.

Furthermore, because $I_k \subseteq D_k \subsetneq D_{i_j}$ for all $i_j \leq k < i_{j+1}$, by ideal irreducibility there exists an ideal I'_k of D_{i_j} such that $I_k \subseteq I'_k$. By (CI) for *reduced* extended configurations, that entails $B_k \sqsubseteq B(I'_k)$, which in turn means that, for all k,

$$|B_k| \leq |B(I'_k)| \leq \|B(I'_k)\| \leq \|D_{i_j}\| . \tag{32}$$

Thus, the multisets B_k have a bounded size, and can be seen instead as vectors in $\mathbb{N}_\omega^{|P|\cdot\|D_{i_j}\|}$; the multiset embedding relation being in such a case equivalent to the product ordering (strictly speaking we have to account for empty cells in those vectors, but this has no impact on the final statement). By Fact 3, we deduce a bound

$$i_{j+1} - i_j \leq c_{\omega^{|P|\cdot\|D_{i_j}\|+1}}(\|D_{i_j}\| \cdot |P|!) \tag{33}$$

for $c(x) \stackrel{\text{def}}{=} |P| \cdot \|D_{i_j}\| \cdot g(x)$. We can over-approximate and simplify this by defining a large enough $C(x) \stackrel{\text{def}}{=} (|P| + 1)! \cdot x \cdot g(x)$:

$$\begin{aligned} i_{j+1} - i_j &\leq C_{\omega^{\|D_{i_j}\|+1}}(\|D_{i_j}\|) \qquad (34)\\ &= C_{\omega^\omega}(\|D_{i_j}\|) . \qquad [18.1] \end{aligned}$$

A consequence of Claim 18.1 on the norm of $\|D_{i_{j+1}}\|$ is that:

$$\begin{aligned} \|D_{i_{j+1}}\| &\leq g^{i_{j+1}-i_j}(\|D_{i_j}\|) && \text{by the } (g, \|D_{i_j}\|)\text{-control}\\ &\leq C^{i_{j+1}-i_j}(\|D_{i_j}\|) && \text{since } \forall x. C(x) \geq g(x)\\ &\leq C^{C_{\omega^\omega}(\|D_{i_j}\|)}(\|D_{i_j}\|) && \text{by Claim 18.1 and since } C \text{ is strictly increasing}\\ &= C^{\omega^\omega}(\|D_{i_j}\|) . && \text{by (26)} \end{aligned}$$

This implies that the Hoare-descending chain $S_{i_0} \supsetneq_H S_{i_1} \supsetneq_H \cdots \supsetneq_H S_{i_r}$ is (C^{ω^ω}, n)-controlled (and even strongly so). We can therefore apply Corollary 4 to bound its length by

$$r + 1 \leq b_{\omega^{|P|+1}}(n \cdot |P|!) + 1 \tag{35}$$

for $b(x) \stackrel{\text{def}}{=} |P| \cdot C^{\omega^\omega}(x)$. Define $h(x) \stackrel{\text{def}}{=} C(x \cdot |P|)$; then the bounds on the length and norms of segments can be restated as

$$i_{j+1} - i_j \leq h_{\omega^\omega}(\|D_{i_j}\|) , \qquad \|D_{i_{j+1}}\| \leq h^{\omega^\omega}(\|D_{i_j}\|) . \tag{36}$$

Defining $a(x) \stackrel{\text{def}}{=} h^{\omega^\omega}(x)$, the bound on r in (35) can be restated as

$$r + 1 \leq a_{\omega^{|P|+1}}(n \cdot |P|!) + 1 . \tag{37}$$

Let us bound the indices i_j for all $0 \leq j \leq r + 1$:

Claim 18.2. For all $0 \leq j \leq r + 1$, $i_j \leq h_{\omega^\omega \cdot j}(n)$.

Proof of Claim 18.2. We proceed by induction on j. For $j = 0$ we have $i_0 = 0 = h_0(n) = h_{\omega^\omega \cdot 0}(n)$, and for the induction step,

$$
\begin{aligned}
i_{j+1} &= i_j + (i_{j+1} - i_j) \\
&\leq i_j + h_{\omega^\omega}(\|D_{i_j}\|) && \text{by (36)} \\
&\leq i_j + h_{\omega^\omega}(g^{i_j}(n)) && \text{since the chain is } (g, n)\text{-controlled} \\
&\leq i_j + h_{\omega^\omega}(h^{i_j}(n)) && \text{since } \forall x. g(x) \leq h(x) \\
&\leq h_{\omega^\omega \cdot j}(n) + h_{\omega^\omega}(h^{[h_{\omega^\omega \cdot j}(n)]}(n)) && \text{by ind. hyp.} \\
&= h_{\omega^\omega \cdot j}(n) + h_{\omega^\omega}(h^{\omega^\omega \cdot j}(n)) && \text{by (26)} \\
&= h_{\omega^\omega + \omega^\omega \cdot j}(n) && \text{by (27)} \\
&= h_{\omega^\omega \cdot (j+1)}(n) \;.
\end{aligned}
$$

[18.2]

Therefore, for $j = r + 1$, we have

$$
\begin{aligned}
\ell = i_{r+1} &\leq h_{\omega^\omega \cdot (r+1)}(n) && \text{by Claim 18.2} \\
&\leq h_{\omega^\omega \cdot [a_{\omega^{|P|+1}}(n \cdot |P|!) + 1]}(n) && \text{by (37)} \\
&\leq h_{\omega^\omega \cdot [a_{\omega^{|P|+1}}(n \cdot |P|!)]}(a(n)) + h_{\omega^\omega}(n) && \text{by (27)} \\
&\leq h_{\omega^\omega \cdot [a_{\omega^{|P|+1}}(a(n))]}(a(n)) + h_{\omega^\omega}(n) && \text{since } \forall x. a(x) \geq x \cdot |P|! \\
&= h_{\omega^\omega \cdot \omega^{|P|+1}}(a(n)) + h_{\omega^\omega}(n) && \text{by (28)} \\
&= h_{\omega^\omega \cdot (\omega^{|P|+1} + 1)}(n) && \text{by (27)} \\
&\leq h_{\omega^\omega \cdot \omega^\omega}(|P| + n) && \text{since } n > 0 \text{ and } |P| > 0 \\
&= h_{\omega^{\omega \cdot 2}}(|P| + n) \;.
\end{aligned}
$$

Note that $h(x) = (|P| + 1)! \cdot (x \cdot |P|) \cdot g(x \cdot |P|)$ depends on both $|P|$ and g. As a final step, observe that we can replace this by the function $(x + 1)! \cdot x^2 \cdot g(x^2)$ since we only apply it to arguments $x > |P|$, thereby yielding a new function primitive recursive in g solely. □

C. Ackermannian Counter Library

The $Acker$ library is a composition $Main \circ Schnoeb$, where library $Schnoeb$ operates on the auxiliary counters (let N denote their bound) and consists of the following routines working on places $\{a, \bar{a}, c, f\}$:

$getset$**:** Initialises the auxiliary counters.

$ascend$**:** First, the auxiliary counters are employed to put, for each $i \in \{0, \ldots, N + 1\}$, $\langle i, N + 1 - i \rangle$ tokens carrying a fresh name d_i onto places $\langle q, \overline{q} \rangle$, where a place f is used to keep track of which name is being worked with.

Let us write $S(i)$ for the number of tokens carrying d_i on place v. Observe that: initially, $S(i) = 0$ for all i; and, using places q, $\overline{q}$, f and the auxiliary counters, the routine is able to increment, decrement (if nonzero), and reset (i.e. set to zero) specific members of the sequence $S(0), \ldots, S(N + 1)$. E.g., to reset $S(i)$, the $\langle i, N + 1 - i \rangle$ tokens carrying d_i on $\langle a, \overline{a} \rangle$ are replaced by the same numbers of tokens carrying a fresh name, the latter thus becoming the new d_i; cf. also Section 2.3.

Now, following Schnoebelen (2010), the Ackermann function A_ω is computed weakly on the sequence $S(0), \ldots, S(N + 1)$ from argument N:

- $S(0)$ is set to N, and $S(N + 1)$ is set to 1;
- repeatedly:
  - either $S(1)$ is decremented and $S(0)$ is weakly doubled,
  - or for some $i \in \{2, \ldots, N + 1\}$, $S(i)$ is decremented, all $S(j)$ for $j \in \{1, \ldots, i - 1\}$ are reset, $S(0)$ is weakly transferred to $S(i - 1)$, and $S(0)$ is set to 1,
  - or all $S(i)$ for $i \in \{1, \ldots, N + 1\}$ are reset and the loop is exited.

Finally, put $\langle 1, N + 1 \rangle$ tokens carrying a fresh name d'_0 onto places $\langle a, \overline{a} \rangle$, and put d_0 and d'_0 onto place f (by one token each).

Let us write $S'(0)$ for the number of tokens carrying d'_0 on place v, which is zero at this point.

hop**:** Decrements $S(0)$ and increments $S'(0)$.

$descend$**:** First, replace $S(0)$ by $S'(0)$: remove the token carrying d'_0 from place a, remove the $N + 1$ tokens carrying d_0 from place $\overline{a}$, and remove d_0 and d'_0 from place f.

Following Schnoebelen (2010) again, the inverse of the Ackermann function A_ω is computed weakly on the sequence $S(0), \ldots, S(N+1)$, and the result is checked to be at least N:

- repeatedly:
  - either $S(1)$ is incremented and $S(0)$ is weakly halved,
  - or for some $i \in \{2, \ldots, N+1\}$, $S(0)$ is decremented and then reset, $S(i-1)$ is weakly transferred to $S(0)$, all $S(j)$ for $j \in \{1, \ldots, i-2\}$ are reset, and $S(i)$ is incremented,
  - or all $S(i)$ for $i \in \{1, \ldots, N\}$ are reset and the loop is exited;
- $S(N+1)$ is decremented, and $S(0)$ is decremented N times, thereby checking that the result is at least N.

Finally, employing place f and the auxiliary counters, for each $i \in \{0, \ldots, N+1\}$, remove the $\langle i, N+1-i \rangle$ tokens carrying d_i from places $\langle a, \overline{a} \rangle$.

Provided $Schnoeb$ is initialised by the routine $getset$, it ensures that any sequence of invocations of the routine hop between a call of $ascend$ and a call of $descend$ is necessarily of length exactly the Ackermann function applied to the bound of the auxiliary counters:

Lemma 19. *Consider any computation of the library Schnoeb where the auxiliary counters are assumed to be N-correct, the initial tape contents is empty, the sequence of routines invoked is of the form*

$$getset\ ascend\ hop^{H_1}\ descend\ \cdots\ ascend\ hop^{H_k}\ descend,$$

and M denotes the final tape contents. Then $H_1 = \cdots = H_k = A_\omega(N)$ and M contains no tokens.

Proof. Arguing inductively, it suffices to consider a sequence of invocations $ascend\ hop^{Y'}\ descend$ with auxiliary counters that have been initialised and from a tape contents with no tokens.

From Schnoebelen (2010, Lemma 3.2, Lemma 3.3 and Section 6), we infer:

- The final value Y of $S(0)$ in $ascend$ is at most $A_\omega(N)$.
- Since the final value Y' of $S'(0)$ in the last hop is at most Y and $descend$ has terminated, we have that the final values of $S(N+1)$ and $S(0)$ in $descend$ are zero, and that $Y' = Y = A_\omega(N)$.
- The computations of the Ackermann function in $ascend$ and of its inverse in $descend$ have not been weak, i.e. all resets are zero tests, and all doublings, halvings and transfers are full.
- All tokens introduced by $ascend$ have been removed in $descend$. ☐

It is now easy to implement the library $Main$ that consists of routines for the counters provided by $Acker$ and working on places $\{q, \overline{q}, q', \overline{q}', q_1, \overline{q}_1, q_2, \overline{q}_2\}$:

$init$**:** After calling $getset$, uses $ascend$, hop (many times) and $descend$ to put exactly $A_\omega(N)$ tokens onto both places $\overline{q}_1$, $\overline{q}_2$.

eq**:** Uses $ascend$, hop (many times) and $descend$ to perform the following exactly $A_\omega(N)$ times: either take a token from both q_1, q_2 and put a token onto q'; or take a token from both $\overline{q}_1$, $\overline{q}_2$ and put a token onto $\overline{q}'$. Then, similarly, move them back.

$i.inc$**:** Moves a token from $\overline{q}_i$ to q_i.

$i.dec$**:** Moves a token from q_i to $\overline{q}_i$.

$i.iszero$**:** Uses $ascend$, hop, $descend$ as before to move exactly $A_\omega(N)$ tokens from $\overline{q}_i$ to q_i, and then move them back.

$i.iszero$**:** Uses $ascend$, hop, $descend$ as before to move exactly $A_\omega(N)$ tokens from q_i to $\overline{q}_i$, and then move them back.

By the definition of $Main$ and by Lemma 19, we obtain:

Lemma 11. *The counter library Acker is A_ω-correct.*