THE POWER OF PRIORITY CHANNEL SYSTEMS

CHRISTOPH HAASE, SYLVAIN SCHMITZ, AND PHILIPPE SCHNOEBELEN

Abstract. We introduce Priority Channel Systems, a new natural class of channel systems where messages carry a numeric priority and where higher-priority messages can supersede lower-priority messages preceding them in the fifo communication buffers. The decidability of safety and inevitability properties is shown via the introduction of a *priority embedding*, a well-quasi-ordering that has not previously been used in well-structured systems. We then show how Priority Channel Systems can compute Fast-Growing functions and prove that the aforementioned verification problems are $\mathbf{F}_{\varepsilon_0}$-complete.

1. Introduction

Channel systems are a family of distributed models where concurrent agents communicate via usually unbounded fifo communication buffers called "channels." An agent of a channel system is modeled by a finite-state controller, and when taking a transition an agent can read messages from the channel or write into it. These models have turned out to be well-suited for the formal specification and algorithmic analysis of communication protocols and concurrent programs [33, 8, 11, 14, 30]. They are also a fundamental model of computation, closely related to Post's tag systems. In all generality, channel systems are a Turing powerful model, which implies that most of their decision problems are undecidable.

A particularly interesting decidable and widely studied class of channel systems are the so-called *lossy channel systems* (LCSs), where channels are unreliable and may lose messages, see *e.g.* [15, 1, 12]. For LCSs, several important behavioral properties such as safety or inevitability are decidable. This is because, due to the lossy behavior of their channels, these systems are *well-structured*: transitions are monotonic with respect to a decidable well-quasi-ordering of the configuration space [2, 20, 38]. Beyond their applications in verification, LCSs have turned out to be an important automata-theoretic tool for decidability or hardness in areas like Timed Automata, Metric Temporal Logic, modal logics, *e.g.* [3, 24, 32, 25]. Moreover, they are also a fundamental model of computation capturing the $\mathbf{F}_{\omega^\omega}$-complexity level in the fast-growing complexity hierarchy [35], see [16, 36].

Lossy channel systems do not provide an adequate way to model systems or protocols that treat messages discriminatingly according to some specified rule set. An example is the prioritization of messages, which is central to ensuring *quality of service* (QoS) properties in networking architectures, and is usually implemented by allowing for tagging messages with some relative priority. For instance, the Differentiated Services (DiffServ) architecture

Work partially funded by the ReacHard project ANR 11 BS02 001 01.

described in RFC 2475 [7], which enables QoS on modern IP networks, allows for a field specifying the relative priority of an IP packet with respect to a finite set of priorities, and network links may decide to arbitrarily drop IP packets of lower priority in favor of higher priority packets once the network congestion reaches a critical point. Another example of a similar priority-based policy arises in the context of ATM networks, where priorities are expressed via a single Cell Loss Priority bit in order to allow for giving preference (by dropping low-priority packages) to audio or video over less time-critical data [26].

Inspired by the aforementioned types of protocols, in this paper we introduce *priority channel systems* (PCSs), a family of channel systems where each message is equipped with a priority level, and where higher-priority messages can supersede lower-priority messages by dropping them. Priority channel systems rely on the *prioritized superseding ordering*, a novel ordering that generalizes Higman's subword ordering and has not been considered before in the area of well-structured systems. It is however closely related to the gap-embedding considered in [41]. Showing it to be a well-quasi-ordering entails, among others, showing the decidability of safety and termination for PCSs. We complement our decidability results by showing that these problems become undecidable for channel systems that build upon more restrictive priority mechanisms, supporting the design choices made for our model.

1.1. **Structure of this Paper.**

This paper can roughly be divided into two parts. In the first part, we define priority channel systems, explore this new model and analyze its power in complexity-theoretical terms. Beginning in Section 6, the second part relates priority channel systems in the broadest sense to related models or mathematical objects found in the literature.

In more detail, in Section 2 we provide an *at-a-glance* introduction to a simplified model of priority channel systems. This allows us to discuss on a high level the ideas behind our model, the main theorems, and the main algorithmic problems that we consider in this paper. We outline the decidability of fundamental decision problems via the framework of well-structured systems. Section 3 is then devoted to proving well-quasi-ordering properties of the prioritized superseding ordering which underlies priority channel systems. To this end, we characterize the superseding ordering via *priority embeddings*, which is an analogue and can in fact be seen as a generalization of Higman's subword embedding. Using techniques from [36, 41], we show in Section 4 an $\mathbf{F}_{\varepsilon_0}$ upper bound on the complexity of PCS verification, far higher than the $\mathbf{F}_{\omega^\omega}$-complete complexity known for LCSs. We then prove in Section 5 a matching lower bound and this is the main technical result for PCSs of this paper: building upon techniques developed for less powerful models [16, 39, 23], we show how PCSs can robustly simulate the computation of the fast growing functions F_α and their inverses for all ordinals α up to ε_0. This gives a precise measure of the expressive power of PCSs.

In the second part of the paper, we first show in Section 6 that other natural choices of models of channel systems with priority mechanisms different from ours lead to undecidability of these problems. We then show in Section 7 how *higher-order* models, which generalize the dynamic LCS from [4], naturally embed into our formalism. Applications of the priority embedding to other well-quasi-ordered data structures such as depth-bounded trees found in the literature are subsequently discussed in Section 8.

2. Priority Channel Systems

In this section, we formally introduce Priority Channel Systems and give an overview about the decision problems we consider in this paper.

Definition 2.1. For every $d \in \mathbb{N}$, the *level-d priority alphabet* is $\Sigma_d \overset{\text{def}}{=} \{0, 1, \ldots, d\}$. A *level-$d$ priority channel system* (d-PCS) is a tuple $S = (\Sigma_d, \mathtt{Ch}, Q, \Delta)$, where Σ_d is as above, $\mathtt{Ch} = \{\mathtt{c}_1, \ldots, \mathtt{c}_m\}$ is a set of m *channel names*, $Q = \{q_1, q_2, \ldots\}$ is a finite set of *control states*, and $\Delta \subseteq Q \times \mathtt{Ch} \times \{!, ?\} \times \Sigma_d \times Q$ is a set of *transition rules.*

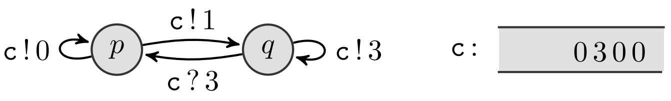


FIGURE 1. A simple single-channel 3-PCS.

For the sake of a simplified introduction to PCSs in this section, the alphabet of a PCS abstracts away from actual message contents and only consists of naturals that indicate the priority of a message, where d is a message of highest and 0 of lowest priority, respectively. A treatment of more general alphabets is deferred to Section 3. The simple alphabet introduced here is however sufficient in order to show the lower bounds in Section 5. Moreover, from our definition it follows that Priority Channel Systems consist of a single process, which is sufficient for our purposes in this paper, since systems made of several concurrent components can be represented by a single process obtained as an asynchronous product of the components.

Figure 1 depicts a 3-PCS with a single channel and control states p and q. (A possible configuration of the channel is depicted alongside.) Informally speaking, when in control state q the PCS in Figure 1 can non-deterministically loop while writing the alphabet symbol "3" to the channel (to its right end), or switch to control state q if "3" can be read from the channel (from its left end). The key feature of PCSs is that messages with higher priority can erase messages with lower priority, *cf.* the formal semantics given next.

2.1. **Semantics.** The operational semantics of a PCS S is given in terms of a transition system. We let $\mathit{Conf}_S \overset{\text{def}}{=} Q \times (\Sigma_d^*)^m$ be the set of all *configurations* of S, denoted $C, D, \ldots$ in the following. A configuration $C = (q, x_1, \ldots, x_m)$ records an instantaneous control state $q \in Q$ and the contents of the m channels, *i.e.*, sequences of messages from Σ_d. A sequence $x \in \Sigma_d^*$ has the form $x = a_1 \cdots a_\ell$ and we let $\ell = |x|$. Concatenation is denoted multiplicatively, with ε denoting the empty sequence.

The labeled transition relation between configurations, denoted $C \xrightarrow{\delta} C'$, is generated by the rules in $\Delta = \{\delta_1, \ldots, \delta_k\}$. From a technical perspective, it is convenient to define two such transition relations, denoted $\rightarrow_{\text{rel}}$ and $\rightarrow_{\#}$.

2.1.1. *Reliable Semantics.* We start with $\rightarrow_{\text{rel}}$ that corresponds to "reliable" steps, or more correctly steps with no superseding of lower-priority messages. As is standard, for a *reading rule* of the form $\delta = (q, \mathsf{c}_i?a, q') \in \Delta$, there is a step $C \xrightarrow{\delta}_{\text{rel}} C'$ if $C = (q, x_1, \ldots, x_m)$ and $C' = (q', y_1, \ldots, y_m)$ for some $x_1, y_1, \ldots, x_m, y_m$ such that $x_i = a\, y_i$ and $x_j = y_j$ for all $j \neq i$, while for a *writing rule* $\delta = (q, \mathsf{c}_i!a, q') \in \Delta$, there is a step $C \xrightarrow{\delta}_{\text{rel}} C'$ if $y_i = x_i\, a$ and $x_j = y_j$ for all $j \neq i$. These reliable steps correspond to the behavior of queue automata, or (reliable) channel systems, a Turing-powerful computation model [13].

2.1.2. *Internal-Superseding.* The actual behavior of PCSs is obtained by extending reliable steps with *internal superseding steps*, denoted $C \xrightarrow{\mathsf{c}_i\#k}_{\#} C'$, which can be performed at any time in an uncontrolled manner. Formally, for two words $x, y \in \Sigma_d^*$ and $k \in \mathbb{N}$, we write $x \xrightarrow{\#k}_{\#} y \overset{\text{def}}{\Leftrightarrow}$

(1) x can be decomposed as $a_1 \cdots a_\ell$;
(2) there is $1 \leq k < |x| = \ell$ such that $a_k \leq a_{k+1}$; and
(3) $y = a_1 \cdots a_{k-1}\, a_{k+1} \cdots a_\ell$.

In other words, the kth message in x is superseded by its immediate successor a_{k+1}, with the condition that a_k is not of higher priority. We write $x \rightarrow_{\#} y$ when $x \xrightarrow{\#k}_{\#} y$ for some k, and use $x \leftarrow_{\#} y$ when $y \rightarrow_{\#} x$. The transitive reflexive closure $\overset{*}{\leftarrow}_{\#}$ is called the *superseding ordering* and is denoted by $\leq_{\#}$. Put differently, $\rightarrow_{\#}$ is a rewrite relation over Σ_d^* defined by the following string rewriting system (see [10]):

$$\{a\, a' \rightarrow a' \mid 0 \leq a \leq a' \leq d\}\ . \tag{1}$$

This is extended to steps between configurations by $C \xrightarrow{\mathsf{c}_i\#k}_{\#} C' \overset{\text{def}}{\Leftrightarrow} C = (q, x_1, \ldots, x_m)$, $C' = (q', y_1, \ldots, y_m)$, $q = q'$, $x_i \xrightarrow{\#k}_{\#} y_i$, and $x_j = y_j$ for $j \neq i$. Furthermore, every reliable step is a valid step: for any rule δ, $C \xrightarrow{\delta}_{\#} C'$ if $C \xrightarrow{\delta}_{\text{rel}} C'$, giving rise to a second transition system associated with S: $\mathcal{S}_{\#} \overset{\text{def}}{=} (\mathit{Conf}_S, \rightarrow_{\#})$.

Example 2.2. The following is a valid path in the transition system induced by the PCS from Figure 1:

$$(p, 0\,3\,0\,0) \xrightarrow{!1}_{\#} (q, 0\,3\,\underline{0}\,0\,1) \xrightarrow{\#3}_{\#} (q, \underline{0}\,3\,0\,1) \xrightarrow{\#1}_{\#} (q, 3\,\underline{0}\,1) \xrightarrow{\#2}_{\#} (q, 3\,1)\ .$$

Here, underlining is used to show which symbol, if any, is superseded in the next step.

2.1.3. *Write-Superseding Semantics.* The internal-superseding semantics allows superseding to occur at any time and anywhere in the channel. Another possible scenario considers communications going through relays, network switches, or buffers, which handle incoming traffic with a so-called *write-superseding* policy, where writes immediately supersede (*i.e.*, erase) the congested messages in front of them. We develop this aspect here and prove the two semantics to be essentially equivalent.

Let $S = (\Sigma_d, \mathtt{Ch}, Q, \Delta)$ be a d-PCS. We define a new transition relation, denoted $\rightarrow_{\mathrm{w}}$, between the configurations of S, giving rise to a transition system $\mathcal{S}_{\mathrm{w}} \overset{\mathrm{def}}{=} (\mathit{Conf}_S, \rightarrow_{\mathrm{w}})$. The relation $\rightarrow_{\mathrm{w}}$ is a variant of $\rightarrow_{\#}$ obtained by modifying the semantics of writing rules. Formally, for $\delta = (q, \mathtt{c}_i!a, q') \in \Delta$, and for two configurations $C = (q, x_1, \ldots, x_m)$ and $C' = (q, y_1, \ldots, y_m)$, there is a step $C \xrightarrow{\delta}_{\mathrm{w}} C'$ if $x_j = y_j$ for all $j \neq i$ and $y_i = z\, a$ for a factorization $x_i = z\, z'$ of x_i where $z' \in \Sigma_a^*$, *i.e.*, where z' only contains messages from the level-a priority subalphabet. In other words, after $\mathtt{c}_i!a$, the channel will contain a sequence y_i obtained from x_i by appending a in a way that may drop (erase) any number of suffix messages with priority $\leq a$, hence the "$z' \in \Sigma_a^*$" requirement. The semantics of reading rules is unchanged so that $C \xrightarrow{\delta}_{\mathrm{rel}} C'$ implies $C \xrightarrow{\delta}_{\mathrm{w}} C'$.

Example 2.3. The PCS from Figure 1 has the following write-superseding run:

$$(p, 0\,3\,\underline{0\,0}) \xrightarrow{!1}_{\mathrm{w}} (q, 0\,\underline{3\,1}) \xrightarrow{!3}_{\mathrm{w}} (q, 0\,3) \xrightarrow{!3}_{\mathrm{w}} (q, \underline{0\,3\,3}) \xrightarrow{!3}_{\mathrm{w}} (q, 3) \xrightarrow{?3}_{\mathrm{w}} (p, \varepsilon)$$

where in every configuration we underline the messages that will be superseded in the next step (and where, for simplicity, we do not write the full rule δ on the steps). Observe that $(p, 0\,3\,0\,0) \not\xrightarrow{*}_{\mathrm{w}} (q, 3\,1)$, to be contrasted with the internal-superseding run $(p, 0\,3\,0\,0) \xrightarrow{*}_{\#} (q, 3\,1)$ in Example 2.2. Under write-superseding, the occurrence of 3 that is initially in the channel is not allowed to erase the 0 in front of it. □

Compared to our standard PCS semantics, the write-superseding semantics adopts a localized viewpoint where the protocol managing priority levels and handling congestions resides at the sender's end, and is not distributed all along the channels.

In the rest of this subsection, we show that the write-superseding is essentially equivalent to the standard semantics, see Proposition 2.4. A consequence is that one can freely choose to adopt either $\mathcal{S}_{\#}$ or $\mathcal{S}_{\mathrm{w}}$ as their favorite operational semantics for priority channel systems. In practice, we find it simpler to design and prove the correctness of some PCS—as we will in Sections 5 and 7—when assuming the write-superseding semantics since it is less liberal and easier to control. And we find it simpler to develop the formal theory of PCSs when assuming the internal-superseding semantics since it is finer-grained.

Proposition 2.4. *Let $C_0 = (q, \varepsilon, \ldots, \varepsilon)$ be a configuration with empty channels, and C_f be any configuration. Then $C_0 \xrightarrow{+}_{\mathrm{w}} C_f$ if, and only if, $C_0 \xrightarrow{+}_{\#} C_f$.*

The proof is organized in the three Lemmata 2.6–2.8 below.

Remark 2.5. Observe that the requirement of empty channels for C_0 in Proposition 2.4 cannot be lifted, as illustrated with Example 2.3. However, using standard coding tricks (*e.g.*, storing *initial* channel contents in control states), one can reduce a reachability or termination problem starting from an arbitrary initial configuration to the same question starting from an empty-channel C_0, and show its decidability by combining Proposition 2.4 and Theorem 2.12. □

Lemma 2.6 (From $\mathcal{S}_{\mathrm{w}}$ to $\mathcal{S}_{\#}$)**.** *If $\mathcal{S}_{\mathrm{w}}$ has a run $C \xrightarrow{+}_{\mathrm{w}} D$ then $\mathcal{S}_{\#}$ has a run $C \xrightarrow{+}_{\#} D$.*

Proof. We show that $\rightarrow_{\mathrm{w}}$ is contained in $\xrightarrow{+}_{\#}$, assuming for the sake of simplicity that S has only one channel.

A writing step $(p,x) \xrightarrow{!a}_{\mathrm{w}} (q,y)$ with $x = z\,b_1 \cdots b_j$ and $y = z\,a$ in $\mathcal{S}_{\mathrm{w}}$ can be simulated in $\mathcal{S}_{\#}$ with $(p,x) \xrightarrow{!a}_{\#} (q, z\,b_1 \cdots b_j\,a) \xrightarrow{\#\ell}_{\#} (q, z\,b_1\,b_{j-1}) \xrightarrow{\#\ell-1}_{\#} \cdots \xrightarrow{\#k+1}_{\#} (q, z\,a)$, where $\ell = |x|$ and $k = |z|$. Reading steps simply coincide in $\mathcal{S}_{\mathrm{w}}$ and $\mathcal{S}_{\#}$. ☐

In the other direction, one can translate runs in $\mathcal{S}_{\#}$ to runs in $\mathcal{S}_{\mathrm{w}}$ as stated by following lemma.

Lemma 2.7 (From $\mathcal{S}_{\#}$ to $\mathcal{S}_{\mathrm{w}}$)**.** *If $\mathcal{S}_{\#}$ has a run $C \xrightarrow{*}_{\#} D$ then $\mathcal{S}_{\mathrm{w}}$ has a run $C' \xrightarrow{*}_{\mathrm{w}} D$ for some $C' \leq_{\#} C$. In particular, if the channels are empty in C, then necessarily $C' = C$ and $C \xrightarrow{*}_{\mathrm{w}} D$.*

Proof. Again we assume that S has only one channel.

Write the run $C \xrightarrow{*}_{\#} D$ under the form $C_0 \rightarrow_{\#} C_1 \rightarrow_{\#} \cdots \rightarrow_{\#} C_n$ and rearrange its steps so that superseding occurs greedily. This relies on Lemma 2.8 stated next.

Repeatedly applying Lemma 2.8 to transform $C_0 \xrightarrow{*}_{\#} C_n$ as long as possible is bound to terminate (with each commutation, superseding steps are shifted to the left of reliable steps, or the sum $\sum_i k_i$ of superseding positions in steps $C_{i-1} \xrightarrow{\#k_i}_{\#} C_i$ increases strictly while being bounded by $O(n^2)$ for a length-n run). One eventually obtains a new run $C_0 \xrightarrow{*}_{\#} C_n$ with same starting and final configurations, and where all the superseding steps occur (at the beginning of the run or) just after a write in *normalized* sequences of the form

$$C = (q,x) \xrightarrow{!a}_{\#} \xrightarrow{\#\ell}_{\#} \xrightarrow{\#\ell-1}_{\#} \xrightarrow{\#\ell-2}_{\#} \cdots \xrightarrow{\#\ell-r}_{\#} C' \,, \tag{2}$$

where furthermore $\ell = |x|$. In this case, $\mathcal{S}_{\mathrm{w}}$ has a step $C \xrightarrow{!a}_{\mathrm{w}} C'$.

Greedily shifting superseding steps to the left may move some of them at the start of the run instead of after a write: these steps are translated into $C \geq_{\#} C'$ in Lemma 2.7. Finally, the steps that are not in normalized sequences are reading steps which exist unchanged in $\mathcal{S}_{\mathrm{w}}$. ☐

Lemma 2.8 (Commuting $\#$-steps)**.**

(1) *If $C_1 \xrightarrow{?a}_{\#} C_2 \xrightarrow{\#k}_{\#} C_3$ then there is a configuration C_2' s.t. $C_1 \xrightarrow{\#k+1}_{\#} C_2' \xrightarrow{?a}_{\#} C''$.*

(2) *$C_1 = (q,x) \xrightarrow{!a}_{\#} C_2 \xrightarrow{\#k}_{\#} C_3$ with $k < |x|$, then there is a configuration C_2' s.t. $C_1 \xrightarrow{\#k}_{\#} C_2' \xrightarrow{!a}_{\#} C_3$.*

(3) *If $C_1 = (q,x) \xrightarrow{\#k_1}_{\#} C_2 \xrightarrow{\#k_2}_{\#} C_3$ with $k_1 \leq k_2$ then there is a configuration C_2' s.t. $C_1 \xrightarrow{\#k_2+1}_{\#} C_2' \xrightarrow{\#k_1}_{\#} C''$.*

2.2. **Priority Channel Systems are Well-Structured.**

Our main result regarding the verification of PCSs is that they are *well-structured* systems, which entails the decidability of standard decision problems via the generic decidability results from [2, 20, 38]. Let us first recall the definitions of well-quasi-orders and well-structured systems.

Definition 2.9 (wqo)**.** Let $(A, \leq_A)$ be a quasi order. Then $(A, \leq_A)$ is a *well-quasi-order* (wqo) if for any infinite sequence $x_0, x_1, x_2, \ldots$ over A there exists two indices $i < j$ such that $x_i \leq_A x_j$.

A simple example of a wqo is any finite set Σ with equality $(\Sigma, =)$, thanks to the pigeonhole principle. More generally, complex wqos can be build from simpler one by algebraic operations [37]. Let $(A_1, \leq_{A_1})$ and $(A_2, \leq_{A_2})$ be wqos:

- Their *disjoint sum* $A_1 + A_2 \overset{\text{def}}{=} \{\langle x, i\rangle \mid i \in \{1,2\} \text{ and } x \in A_i\}$ is well-quasi-ordered by the *sum ordering* $\leq_+$ defined by $\langle x, i\rangle \leq_+ \langle y, j\rangle \overset{\text{def}}{\Leftrightarrow} i = j$ and $x \leq_{A_i} y$.
- Their *Cartesian product* $A_1 \times A_2 \overset{\text{def}}{=} \{\langle x, y\rangle \mid x \in A_1 \text{ and } y \in A_2\}$ is well-quasi-ordered by the *product ordering* $\leq_\times$ defined by $\langle x, y\rangle \leq_\times \langle x', y'\rangle \overset{\text{def}}{\Leftrightarrow} x \leq_{A_1} x'$ and $y \leq_{A_2} y'$. This is also known as Dickson's Lemma.
- The set A_1^* of finite sequences over A_1 is well-quasi-ordered by the *substring embedding* relation $\leq_*$ defined by $x \leq_* y \overset{\text{def}}{\Leftrightarrow} x = a_1 \cdots a_\ell$, $y = y_0\, b_1\, y_1 \cdots y_\ell\, b_\ell\, y_{\ell+1}$ for some a_i, b_i in A and y_i in A^*, and $a_i \leq_{A_1} b_i$ for every $1 \leq i \leq \ell$. This is known as Higman's Lemma, and is instrumental in the study of lossy channel systems (*cf.* Section 6).

Definition 2.10 (WSTS)**.** A *well-structured (transition) system* (WSTS) is a tuple $\mathcal{S} = (A, \to, \leq_A)$ with $\to \subseteq A \times A$ such that

(1) $(A, \leq_A)$ is a wqo; and
(2) $\to$ is compatible with respect to $\leq_A$, *i.e.*, if $x \to y$ and $x \leq_A x'$ then there is some y' such that $x' \xrightarrow{*} y'$ and $y \leq_A y'$.

A WSTS enjoys a stronger *stuttering compatibility* if the second condition is altered to require $x' \xrightarrow{+} y'$. Let $S = (\Sigma_d, \mathtt{Ch}, Q, \Delta)$ be a PCS, we define the following order on configurations of S: $C \leq_\# D \overset{\text{def}}{\Leftrightarrow} C$ is some $(p, y_1, \ldots, y_m)$ and D is $(p, x_1, \ldots, x_m)$ with $x_i \leq_\# y_i$ for all $i = 1, \ldots, m$. Equivalently, $C \leq_\# D$ if C can be obtained from D by internal superseding steps.

Theorem 2.11 (PCSs are WSTSs)**.** *For any PCS S, $\mathcal{S}_\# = (Conf_S, \to_\#, \leq_\#)$, i.e., the transition system $\mathcal{S}_\#$ with configurations ordered by $\leq_\#$, is a well-structured system with stuttering compatibility.*

Proof. We have to show that the two conditions required in Definition 2.10 hold. Proving that $(Conf_S, \leq_\#)$ is a well-quasi-ordering is the topic of Section 3 and will be established in a more general setting in Theorem 3.6.

Checking stuttering compatibility is trivial with the $\leq_\#$ ordering. Indeed, assume that $C \leq_\# C'$ and that $C \to_\# D$ is a step from the "smaller" configuration. Then in particular $C' \xrightarrow{*}_\# C$ by definition of $\to_\#$, so that clearly $C' \xrightarrow{+}_\# D$ and C' can simulate any step from C. □

A consequence of the well-structuredness of PCSs is the decidability of several natural verification problems. In this paper we focus on "Reachability," aka "Safety" when we want to check that a configuration is *not* reachable: given a PCS, an initial configuration C_0, and a recursive set of configurations $G \subseteq \mathit{Conf}_S$, does $C_0 \xrightarrow{*}_{\#} D$ for some $D \in G$? Another decision problem is "Inevitability," *i.e.* to decide whether all maximal runs from C_0 eventually visit G, which includes "Termination" as a special case.

Theorem 2.12. *Reachability and Inevitability are decidable for PCSs.*

Proof (Sketch). In order to apply the generic WSTS algorithms from [20], we have to prove that the order $\leq_{\#}$ is decidable, and that the set of immediate successors of a configuration and the minimal immediate predecessors of an upward-closed set are computable.

Deciding the ordering $\leq_{\#}$ between configurations is in NLogSpace; the proof of this fact is the objective of Remark 3.7. Moreover, the operational semantics is finitely branching and effective, *i.e.*, one can compute the immediate successors of a configuration and the minimal immediate predecessors of an upward-closed set.

We note that Reachability and Coverability coincide (even for zero-length runs when C_0 has empty channels) since $\xrightarrow{+}_{\#}$ coincides with $\geq_{\#} \circ \xrightarrow{+}_{\#}$, and that the answer to a Reachability question only depends on the (finitely many) minimal elements of G. One can even compute $\mathit{Pre}^*(G)$ for G given, *e.g.*, as a regular subset of Conf_S.

For Inevitability, the algorithms in [2, 20] assume that G is downward-closed but, in our case where $\xrightarrow{+}_{\#}$ and $\geq_{\#} \circ \xrightarrow{+}_{\#}$ coincide, decidability can be shown for arbitrary (recursive) G, as in [40, Theorem 4.4]. ☐

3. Priority Embedding

In this section we establish that the superseding ordering $\leq_{\#}$ on words enjoys the well-quasi-ordering properties we require for reasoning about PCSs. In order to keep our results generic, as already stated at the beginning of Section 2, we establish those properties over an alphabet that is more general than the one introduced in Definition 2.1. Instead of allowing for messages consisting merely of priorities, we allow for messages over an arbitrary well-quasi-ordering to be tagged with priority numbers. This is in line with the algebraic operations on wqos presented at the beginning of Section 2.2.

Definition 3.1 (Generalized Priority Alphabet)**.** Let $d \in \mathbb{N}$ be a *priority level* and let $(\Gamma, \leq_\Gamma)$ be a well-quasi-order, a *generalized level-d priority alphabet over* Γ is $\Sigma_{d,\Gamma} \stackrel{\text{def}}{=} \{(a,w) \mid 0 \leq a \leq d, w \in \Gamma\}$.

Subsequently, we call $\Sigma_{d,\Gamma}$ a *generalized priority alphabet* for brevity. In analogy to the internal superseding steps in Section 2.1, we define the *generalized priority relation* $\rightarrow_{\#,\Gamma}$ over finite strings in $\Sigma^*_{d,\Gamma}$ via a string rewriting system with the following two families of rule schemata:

$$\{(a,w)(a',w') \rightarrow_{\#,\Gamma} (a',w') \mid a \leq a', w \in \Gamma\}\ , \tag{3}$$

$$\{(a,w) \rightarrow_{\#,\Gamma} (a,w') \mid w' \leq_\Gamma w\}\ . \tag{4}$$

Informally speaking, the first line states that a string can be rewritten if some higher-priority message supersedes a lower priority message, and the second that any message can be rewritten to a message that is below in the wqo $(\Gamma, \leq_\Gamma)$.

We define $\leq_{\#,\Gamma} \stackrel{\text{def}}{=} \xleftarrow{*}_{\#}$, *i.e.*, $\leq_{\#,\Gamma}$ is the reflexive transitive closure of the inverse of $\to_{\#,\Gamma}$. The main purpose of this section is to prove that $(\Sigma^*_{d,\Gamma}, \leq_{\#,\Gamma})$ is a well-quasi-ordering, *cf.* Definition 2.10. To this end, we will first establish a characterization of $\leq_{\#,\Gamma}$ via an embedding relation and subsequently prove that the obtained priority embeddings yield a well-quasi-ordering.

Before we continue, let us remark that the priority alphabet in Definition 2.1 and $\to_\#$ can be obtained by considering $(\Gamma, =)$ for some singleton set Γ. Whenever we drop the index Γ, we implicitly refer to this well-quasi-ordering. Letting Γ be a finite set of messages represented as strings and $\leq_\Gamma$ the identity relation yields a generalized priority alphabet where a priority can be assigned to each message. Such an alphabet underlies for instance the well-quasi-ordering that we will later use for showing that planar planted trees are well-quasi-ordered under minors, *cf.* Section 8.2. Another example is $\Gamma = \Sigma^*$ for some finite alphabet Σ and where $\leq_\Gamma$ is the substring embedding, which allows for representing unbounded messages on a lossy channel which are tagged with a priority level. Finally, we wish to mention that for a generalized priority alphabet $\Sigma_{d,\Gamma}$, if we wish to apply $\Sigma_{d,\Gamma}$ in a PCSs, for Theorem 2.12 to hold, $(\Gamma, \leq_\Gamma)$ has to fulfill the same properties required from $(\Sigma^*_d, \leq_\#)$ in Theorem 2.12.

3.1. **Embedding with Priorities.**

Given $x, y \in \Sigma^*_{d,\Gamma}$, we define the *generalized priority embedding* $\sqsubseteq_{\text{p},\Gamma}$ by

$$\begin{aligned} x \sqsubseteq_{\text{p},\Gamma} y \quad \stackrel{\text{def}}{\Leftrightarrow} \quad & x = (a_1, v_1) \cdots (a_\ell, v_\ell) \\ & y = y_1\,(a_1, w_1)\, y_2\,(a_2, w_2) \cdots y_\ell\,(a_\ell, w_\ell) \\ & \forall 1 \leq i \leq \ell : y_i \in \Sigma^*_{a_i,\Gamma} \text{ and } v_i \leq_\Gamma w_i\ . \end{aligned}$$

For example, in the singleton case, $201 \sqsubseteq_{\text{p}} 22011$ but $120 \not\sqsubseteq_{\text{p}} 10210$, since factoring 10210 as $z_1 1 z_2 2 z_3 0$ would require $z_3 = 1 \notin \Sigma^*_0$. If $x \sqsubseteq_{\text{p}} y$ then x is a subword of y and x can be obtained from y by removing factors of messages with priority not above the first preserved message to the right of the factor. Observe that $\sqsubseteq_{\text{p},\Gamma}$ is similar (but not equivalent) to the Higman subword embedding for $d = 0$. From the definition above, we get the following properties which we will implicitly use subsequently:

$$\varepsilon \sqsubseteq_{\text{p},\Gamma} y \quad \text{iff} \quad y = \varepsilon\ , \tag{5}$$

$$x_1 \sqsubseteq_{\text{p},\Gamma} y_1,\ x_2 \sqsubseteq_{\text{p},\Gamma} y_2 \quad \text{imply} \quad x_1\, x_2 \sqsubseteq_{\text{p},\Gamma} y_1\, y_2\ , \tag{6}$$

$$x_1\, x_2 \sqsubseteq_{\text{p},\Gamma} y \quad \text{implies} \quad \exists y_1 \sqsupseteq_{\text{p},\Gamma} x_1 : \exists y_2 \sqsupseteq_{\text{p},\Gamma} x_2 : y = y_1\, y_2\ , \tag{7}$$

$$v \leq_\Gamma w \quad \text{implies} \quad \forall 0 \leq a \leq d : \forall z \in \Sigma^*_{a,\Gamma} : (a, v) \sqsubseteq_{\text{p},\Gamma} z(a, w)\ . \tag{8}$$

Lemma 3.2. *Let $\Sigma_{d,\Gamma}$ be a generalized priority alphabet. Then $(\Sigma^*_{d,\Gamma}, \sqsubseteq_{\text{p},\Gamma})$ is a quasi-ordering.*

Proof. We have to show that $(\Sigma^*_{d,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma})$ is reflexive and transitive. Reflexivity is obvious from the definition of $\sqsubseteq_{\mathrm{p},\Gamma}$. Regarding transitivity, let $x, y, z \in \Sigma^*_{d,\Gamma}$ be such that $x \sqsubseteq_{\mathrm{p},\Gamma} y \sqsubseteq_{\mathrm{p},\Gamma} z$ and write $x = (a_1, u_1) \cdots (a_\ell, u_\ell)$. Since $x \sqsubseteq_{\mathrm{p},\Gamma} y$, by definition we can write $y = y_1(a_1, v_1) \cdots y_\ell(a_\ell, v_\ell)$, where $u_i \leq_\Gamma v_i$ and each $y_i = (b_{1,i}, v_{1,i}) \cdots (b_{m_i,i}, v_{m_i,i}) \in \Sigma^*_{a_i,\Gamma}$ for all $1 \leq i \leq \ell$. Consequently, since $y \sqsubseteq_{\mathrm{p},\Gamma} z$, we can decompose z as $z = z_1(a_1, w_1) \cdots z_\ell(a_\ell, w_\ell)$, where each z_i is of the form

$$z_i = z_{1,i}(b_{1,i}, w_{1,i}) \cdots z_{m_i,i}(b_{m_i,i}, w_{m_i,i}) z'_i .$$

Since each $(b_{j,i}, w_{j,i}) \in \Sigma^*_{a_i,\Gamma}$, by definition of $\sqsubseteq_{\mathrm{p},\Gamma}$ we have $z_i \in \Sigma^*_{a_i,\Gamma}$, hence the above decomposition of z in particular yields $x \sqsubseteq_{\mathrm{p},\Gamma} z$. □

The generalized priority embedding acts as a relational counterpart to the more operational generalized superseding ordering. In fact $\leq_{\#,\Gamma}$ and $\sqsubseteq_{\mathrm{p},\Gamma}$ coincide, as shown by the next lemma.

Lemma 3.3. *For any $x, y \in \Sigma^*_{d,\Gamma}$, $x \leq_{\#,\Gamma} y$ if, and only if, $x \sqsubseteq_{\mathrm{p},\Gamma} y$.*

Proof. In the following, write x as $x = (a_1, v_1) \cdots (a_k, v_k)$.

Suppose $x \leq_{\#,\Gamma} y$, *i.e.*, $y \xrightarrow{*}_{\#,\Gamma} x$. We show $x \sqsubseteq_{\mathrm{p},\Gamma} y$ by induction on the number of superseding steps. The base case where no superseding occurs entails $x = y$ and we rely on the reflexivity of $\sqsubseteq_{\mathrm{p},\Gamma}$. For the induction step, let $y \rightarrow_{\#,\Gamma} z$ such that $z \xrightarrow{*}_{\#,\Gamma} x$. By the induction hypothesis, $x \sqsubseteq_{\mathrm{p},\Gamma} z$, *i.e.*, z can be factored as $z = z_1(a_1, w_1) \cdots z_k(a_k, w_k)$ such that $z_i \in \Sigma^*_{a_i,\Gamma}$ and $v_i \leq_\Gamma w_i$ for all $1 \leq i \leq k$. We do a case distinction on which rewriting rule is applied in order to obtain $y \rightarrow_{\#,\Gamma} z$:

- If $y \rightarrow_{\#,\Gamma} z$ via (3) then y is obtained from z by replacing some $z_j = z_{j,1} \cdots z_{j,\ell_j}$ with $z'_j = z_{j,1} \cdots z_{j,i-1}(b, w) z_{j,i} \cdots z_{j,\ell_j}$ for some $1 \leq i \leq \ell_j + 1$, $1 \leq j \leq k$ and (b, w) such that in particular $b \leq a_j$, and hence $z'_j \in \Sigma^*_{a_j,\Gamma}$. Thus y factors as $y = z_1(a_1, w_1) \cdots z'_j(a_j, w_j) \cdots z_k(a_k, w_k)$, which allows us to conclude that $x \sqsubseteq_{\mathrm{p},\Gamma} y$.
- If $y \rightarrow_{\#,\Gamma} z$ via (4), y is obtained by replacing some (a, w) occurring in z with (a, w') for some $w' \geq_\Gamma w$. By transitivity of $\leq_\Gamma$, $x \sqsubseteq_{\mathrm{p},\Gamma} y$ follows immediately.

Conversely, assume $x \sqsubseteq_{\mathrm{p},\Gamma} y$: then y factors as $y = y_1(a_1, w_1) \cdots y_k(a_k, w_k)$. Since for every (a, w) occurring in some y_i we have $a \leq a_i$, by repeatedly applying (3) we have $y \rightarrow^*_{\#,\Gamma} z = (a_1, w_1) \cdots (a_k, w_k)$. Moreover, $v_i \leq_\Gamma w_i$ for all $1 \leq i \leq k$, and thus by repeated application of (4) we get $z \rightarrow^*_{\#,\Gamma} x$, as required. □

3.2. **Priority Embedding is a Well-Quasi-Ordering.** The purpose of this section is to prove that $\sqsubseteq_{\mathrm{p},\Gamma}$ is a well-quasi-ordering. By application of Lemma 3.3, this entails that $\leq_{\#,\Gamma}$ is a well-quasi-ordering as well. We rely for this on the algebraic operations presented in Section 2.2, and on *order reflections*:

Definition 3.4 (Order Reflection)**.** Let $(A, \leq_A)$ and $(B, \leq_B)$ be two quasi-orders. An *order reflection* is a mapping $r \colon A \to B$ such that $r(x) \leq_B r(y)$ implies $x \leq_A y$.

The following is folklore (and easy to see):

Fact 3.5. *Let $(A, \leq_A)$ and $(B, \leq_B)$ be two quasi-orders and r be an order reflection $A \to B$. If $(B, \leq_B)$ is a wqo, then $(A, \leq_A)$ is a wqo.*

In the following, we define the *height* of a sequence $x \in \Sigma^*_{d,\Gamma}$, written $h(x)$, as being the highest priority occurring in x. By convention, we let $h(\varepsilon) \overset{\text{def}}{=} -1$. Thus, $x \in \Sigma^*_{h,\Gamma}$ if and only if $h \geq h(x)$, and we further let $\Sigma_{-1,\Gamma} \overset{\text{def}}{=} \emptyset$. Any $x \in \Sigma^*_{d,\Gamma}$ has a unique *canonical* factorization $x = x_0(h, v_1)x_1 \cdots x_{m-1}(h, v_m)x_m$ where m is the number of occurrences of $h = h(x)$ in x and where the $m + 1$ *residuals* $x_0, x_1, \ldots, x_m$ are in $\Sigma^*_{h-1,\Gamma}$.

Theorem 3.6. *Let $\Sigma_{d,\Gamma}$ be a generalized priority alphabet. Then $(\Sigma^*_{d,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma})$ is a well-quasi-ordering.*

Proof. We proceed by induction on d. For the base case $d = -1$, *i.e.* for the empty priority alphabet, $(\Sigma^*_{-1,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma}) = (\{\varepsilon\}, =)$ is a wqo.

For the induction step, any $x \in \Sigma^*_{d,\Gamma}$ the canonical height factoring gives

$$x = x_0(d, v_1)x_1 \cdots x_{m-1}(d, v_m)x_m \tag{9}$$

with residuals $x_i \in \Sigma^*_{d-1,\Gamma}$ for all $0 \leq i \leq m$. By the induction hypothesis, $(\Sigma^*_{d-1,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma})$ is a well-quasi-ordering. We exhibit an order reflection $r\colon \Sigma^*_{d,\Gamma} \to \Theta_{d,\Gamma}$, where

$$\Theta_{d,\Gamma} \overset{\text{def}}{=} \Sigma^*_{d-1,\Gamma} + \Sigma^*_{d-1,\Gamma} \times ((\{d\} \times \Gamma) \times \Sigma_{d-1,\Gamma})^* \times (\{d\} \times \Gamma) \times \Sigma^*_{d-1,\Gamma}. \tag{10}$$

Since $\Theta_{d,\Gamma}$ is obtained from the well-quasi-orders $(\Sigma^*_{d-1,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma})$, $(\Gamma, \leq_\Gamma$ $)$, and equality on $\{d\}$ by disjoint sum, Cartesian product, and substring embedding, Fact 3.5 will allow us to conclude that $(\Sigma^*_{d,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma})$ is a well-quasi-order. To this end, for x and m as above, if $m > 0$ we define

$$r(x) \overset{\text{def}}{=} (x_0, (((d, v_1), x_1) \cdots ((d, v_{m-1}), x_{m-1})), (d, v_m), x_m) \ , \tag{11}$$

and $r(x) \overset{\text{def}}{=} x_0 = x$ if $m = 0$. We need to verify that, whenever $r(x) \preccurlyeq r(y)$ with respect to the ordering $\preccurlyeq$ associated with $\Theta_{d,\Gamma}$ by the algebraic operations, then $x \sqsubseteq_{\mathrm{p},\Gamma} y$. This is obvious when both $r(x) = x$ and $r(y) = y$ are in $\Sigma^*_{d-1,\Gamma}$. Otherwise, let $r(x)$ be as in (11) and write

$$r(y) = (y_0, (((d, w_1), y_1) \cdots ((d, w_{n-1}), y_{n-1})), (d, w_n), y_n) \ .$$

Since $r(x) \preccurlyeq r(y)$, from the product ordering we obtain

$$x_0 \sqsubseteq_{\mathrm{p},\Gamma} y_0 \ , \tag{12}$$

$v_m \leq_\Gamma w_n$, and $x_m \sqsubseteq_{\mathrm{p},\Gamma} y_n$, while from the subword ordering we obtain the existence of indices $1 \leq i_1, \ldots, i_{m-1} < n$ such that $v_j \leq_\Gamma w_{i_j}$ and $x_j \sqsubseteq_{\mathrm{p},\Gamma} y_{i_j}$ for all $0 < j < m$. Setting $i_0 \overset{\text{def}}{=} 0$ and $i_m \overset{\text{def}}{=} n$, observe that by (6) and (8),

$$(d, v_j)x_j \sqsubseteq_{\mathrm{p},\Gamma} (d, w_{i_{j-1}+1})y_{i_{j-1}+1} \cdots y_{i_j-1}(d, w_{i_j})y_{i_j} \tag{13}$$

for all $0 < j \leq m$, which together with (12) and (6) implies $x \sqsubseteq_{\mathrm{p},\Gamma} y$ as desired. □

Remark 3.7. Theorem 3.6 and Lemma 3.3 prove that $\leq_\#$ is a wqo on configurations of PCSs, as we assumed in Section 2.2. There we also assumed that $\leq_\#$ is decidable. We can now see that it is in NLogSpace, since, in view of Lemma 3.3, one can check whether $x \leq_\# y$ by reading x and y simultaneously while guessing nondeterministically a factorization $z_1a_1 \cdots z_\ell a_\ell$ of y, and checking that $z_i \in \Sigma^*_{a_i}$. □

4. Fast-Growing Upper Bounds

The verification of infinite-state systems and WSTSs in particular turns out to require astronomic computational resources expressed as *subrecursive functions* [27, 19] of the input size. We show in this section how to bound the complexity of the algorithms presented in Section 2.2 and classify the Reachability and Inevitability problems using *fast-growing complexity classes* [35].

To this end, we first provide the necessary background on subrecursive function in Section 4.1. The heart of the upper bound proof is a specialized *Length Function Theorem* for $(\Sigma^*_{d,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma})$, obtained in Section 4.2 by instrumenting the proof of Theorem 3.6 and applying a generic Length Function Theorem from [36]. This allows us to derive $\mathbf{F}_{\varepsilon_0}$ upper bounds and new *combinatorial algorithms* for PCS verification in Section 4.3.

4.1. **Subrecursive Hierarchies.**

Throughout this paper, we use *ordinal terms* inductively defined by the following grammar

$$(\Omega \ni)\ \ \alpha, \beta, \gamma \ ::=\ 0 \mid \omega^\alpha \mid \alpha + \beta$$

where addition is associative, with 0 as the neutral element (the empty sum). Such a term $\alpha = \sum_{i=0}^{k} \omega^{\alpha_i}$ is 0 if $k = 0$, otherwise a *successor* if $\alpha_k = 0$ and a *limit* otherwise. We often write 1 as short-hand for ω^0, and ω for ω^1. The symbol λ is reserved for limit ordinal terms.

We can associate a set-theoretic ordinal $o(\alpha)$ to each term α by interpreting $+$ as the direct sum operator and ω as $\mathbb{N}$; this gives rise to a well-founded quasi-ordering $\alpha < \beta \overset{\text{def}}{\Leftrightarrow} o(\alpha) < o(\beta)$. A term $\alpha = \sum_{i=1}^{k} \omega^{\alpha_i}$ is in *Cantor normal form* (CNF) if $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k$ and each α_i is itself in CNF for $i = 1, \ldots, k$. Terms in CNF and set-theoretic ordinals below ε_0 are in bijection; it will however be convenient later in Section 5 to manipulate terms that are *not* in CNF.

With any limit term λ, we associate a *fundamental sequence* of terms $(\lambda_n)_{n\in\mathbb{N}}$:

$$(\gamma + \omega^{\beta+1})_n \overset{\text{def}}{=} \gamma + \omega^\beta \cdot n = \gamma + \overbrace{\omega^\beta + \cdots + \omega^\beta}^{n}\ , \quad (\gamma + \omega^{\lambda'})_n \overset{\text{def}}{=} \gamma + \omega^{\lambda'_n}\ . \tag{14}$$

This yields $\lambda_0 < \lambda_1 < \cdots < \lambda$ for any λ, with furthermore $\lambda = \lim_{n\in\mathbb{N}} \lambda_n$. For instance, $\omega_n = n$, $(\omega^\omega)_n = \omega^n$, *etc.* Note that λ_n is in CNF when λ is.

We need to add a term ε_0 to Ω to represent the set-theoretic ε_0, *i.e.*, the smallest solution of $x = \omega^x$. We take this term to be a limit term as well; we define the fundamental sequence for ε_0 by $(\varepsilon_0)_n \overset{\text{def}}{=} \Omega_n$, where for $n \in \mathbb{N}$, we use Ω_n as short-hand notation for the ordinal $\omega^{\omega^{\cdot^{\cdot^{\cdot^{\omega}}}}}\}n \text{ stacked } \omega\text{'s}$, *i.e.*, for $\Omega_0 \overset{\text{def}}{=} 1$ and $\Omega_{n+1} \overset{\text{def}}{=} \omega^{\Omega_n}$.

4.1.1. *Inner Recursion Hierarchies.* Our main subrecursive hierarchy is the *Hardy hierarchy.* Given a monotone expansive unary function $h\colon \mathbb{N} \to \mathbb{N}$, it is defined as an ordinal-indexed hierarchy of unary functions $(h^\alpha\colon \mathbb{N} \to \mathbb{N})_\alpha$ through

$$h^0(n) \stackrel{\text{def}}{=} n\,, \qquad h^{\alpha+1}(n) \stackrel{\text{def}}{=} h^\alpha\big(h(n)\big)\,, \qquad h^\lambda(n) \stackrel{\text{def}}{=} h^{\lambda_n}(n)\,. \tag{15}$$

Observe that h^1 is simply h, and more generally h^α is the αth iterate of h, using diagonalization to treat limit ordinals.

A case of particular interest is to choose the successor function $H(n) \stackrel{\text{def}}{=} n+1$ for h. Then the *fast growing hierarchy* $(F_\alpha)_\alpha$ can be defined by $F_\alpha \stackrel{\text{def}}{=} H^{\omega^\alpha}$, resulting in $F_0(n) = H^1(n) = n+1$, $F_1(n) = H^\omega(n) = H^n(n) = 2n$, $F_2(n) = H^{\omega^2}(n) = 2^n n$ being exponential, $F_3 = H^{\omega^3}$ being non-elementary, $F_\omega = H^{\omega^\omega}$ being an Ackermannian function, F_{ω^k} a k-Ackermannian function, and $F_{\varepsilon_0} = H^{\varepsilon_0} \circ H$ a function whose totality is not provable in Peano arithmetic [19].

4.1.2. *Fast-Growing Complexity Classes.* Our intention is to establish the "F_{ε_0} completeness" of verification problems on PCSs. In order to make this statement more precise, we define the class $\mathbf{F}_{\varepsilon_0}$ as a specific instance of the *fast-growing complexity classes* defined for $\alpha \geq 3$ by [35]

$$\mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p \in \bigcup_{\beta<\alpha} \mathscr{F}_\beta} \mathrm{DT{\small IME}}(F_\alpha(p(n)))\,, \qquad \mathscr{F}_\alpha = \bigcup_{c<\omega} \mathrm{FDT{\small IME}}(F_\alpha^c(n))\,, \tag{16}$$

where the class of functions $\mathscr{F}_\alpha$ as defined above is the αth level of the *extended Grzegorczyk hierarchy* [27] when $\alpha \geq 2$. In other words, $\mathscr{F}_\alpha$ is the set of functions computable in time F_α^c, a finite iterate of F_α. In particular, $\mathscr{F}_2$ is the set of elementary functions, $\bigcup_{\alpha<\omega} \mathscr{F}_\alpha$ the set of primitive-recursive functions, while $\bigcup_{\alpha<\varepsilon_0} \mathscr{F}_\alpha$ is exactly the set of ordinal-recursive (aka "provably recursive") functions [19]. Then $\mathbf{F}_\alpha$ is the set of decision problems that can be solved in time $F_\alpha \circ p$ for some p in $\bigcup_{\beta<\alpha} \mathscr{F}_\beta$.

The complexity classes $\mathbf{F}_\alpha$ are naturally equipped with $\bigcup_{\beta<\alpha} \mathscr{F}_\beta$ as classes of reductions. For instance, $\mathscr{F}_2$ is the set of elementary functions, and $\mathbf{F}_3$ the class of problems with a tower of exponents of height bounded by some elementary function of the input as an upper bound.[1]

4.2. **The Length of Controlled Bad Sequences.** A finite or infinite sequence $x_0, x_1, x_2, \ldots$ over a quasi-order $(A, \leq_A)$ is called *bad* if, for all indices $i < j$, $x_i \not\leq_A x_j$. Definition 2.9 can thus be restated by saying that $(A, \leq_A)$ is a wqo if and only if every bad sequence over A is finite. In order to bound the complexity of the algorithms from Theorem 2.12, we wish to bound the lengths of bad sequences over the wqo $(\mathit{Conf}_\mathcal{S}, \leq_\#)$. More precisely, the main issue here is to bound the length of bad sequences over $(\Sigma_d^*, \sqsubseteq_\mathrm{p})$; we actually work in the more general case of $(\Sigma_{d,\Gamma}^*, \sqsubseteq_{\mathrm{p},\Gamma})$.

[1]Note that, at such high complexities, the usual distinctions between deterministic vs. nondeterministic, or time-bounded vs. space-bounded computations become irrelevant.

4.2.1. *Controlled Sequences.* We employ to this end the framework and results of [36]. The first observation is that bad sequences over $(\Sigma_d^*, \sqsubseteq_{\mathrm{p}})$ can be of arbitrary length: for every $N > 0$, the sequence

$$1, 0^N, 0^{N-1}, \dots, 0$$

is indeed a bad sequence of length $N+1$ over $(\Sigma_1^*, \sqsubseteq_{\mathrm{p}})$. Thankfully, what we are looking for are not general bounds over all the bad sequences, but over the kind of sequences that arise in the algorithms of Theorem 2.12: in particular, the bounds can take into account how fast the lengths of the strings in the sequence can grow. Define a *normed* wqo as a wqo $(A, \leq_A)$ further equipped with a norm $|.|_A\colon A \to \mathbb{N}$. As a sanity condition, we ask for

$$A_{\leq n} \stackrel{\text{def}}{=} \{x \in A \mid |x|_A \leq n\} \tag{17}$$

to be finite for each n.

The wqos introduced in Section 2.2 can be normed for instance by

$$|a|_\Sigma \stackrel{\text{def}}{=} 0\;, \qquad |\langle x, i\rangle|_{A_1+A_2} \stackrel{\text{def}}{=} |x|_{A_i}\;,$$
$$|\langle x, y\rangle|_{A_1\times A_2} \stackrel{\text{def}}{=} \max(|x|_{A_1}, |y|_{A_2})\;, \qquad |x_1 \cdots x_\ell|_{A_1^*} \stackrel{\text{def}}{=} \max_{1\leq i\leq \ell}(\ell, |x_i|_{A_1})\;,$$

where $(\Sigma, =)$ denotes a finite set with equality. For $(\Sigma_{d,\Gamma}^*, \sqsubseteq_{\mathrm{p},\Gamma})$, we choose similarly

$$|(a_1, w_1)\cdots(a_\ell, w_\ell)|_{\Sigma_{d,\Gamma}^*} \stackrel{\text{def}}{=} \max_{1\leq i\leq \ell}(\ell, |w_i|_\Gamma)\;, \tag{18}$$

where we assume $(\Gamma, \leq_\Gamma)$ to be normed by $|.|_\Gamma$. By the definition above, this simplifies to

$$|(a_1, w_1)\cdots(a_\ell, w_\ell)|_{\Sigma_{d,\Gamma}^*} = \ell \tag{19}$$

when Γ is finite.

Let $g\colon \mathbb{N} \to \mathbb{N}$ be a strictly monotone function (hereafter called a *control function*), and n be a non-negative integer. A sequence $x_0, x_1, x_2, \dots$ over $(A, \leq_A, |.|_A)$ is (g, n)-*controlled* if, for all i, $|x_i|_A \leq g^i(n)$ the ith iterate of g. Note in particular that this entails $|x_0|_A \leq n$. Given an algorithm that relies on $(A, \leq_A)$ being a wqo for its termination, *i.e.*, on the fact that bad sequences over $(A, \leq_A)$ are finite, the intuition is that g should bound how fast the norm of the elements in our bad sequences can grow, and n should bound the norm of the initial element. As shown in [36], for a given g and n, bad (g, n)-controlled sequences over $(A, \leq_A, |.|_A)$ have a maximal length denoted $L_{A,g}(n)$.

4.2.2. *Normed Reflections.* The *Length Function Theorem* in [36] provides suitable subrecursive upper bounds on the function $L_{A,g}$ when A is constructed using the *elementary* wqo algebra that allows disjoint unions, Cartesian products and Kleene star to be used over finite sets. We are going to exploit these bounds together with the order reflection employed in the proof of Theorem 3.6 to obtain a bound on $L_{\Sigma_{d,\Gamma}^*,g}$.

A reflection $r\colon A \to B$ between two normed wqos $(A, \leq_A, |.|_A)$ and $(B, \leq_B, |.|_B)$ is *normed* if $|r(x)|_B \leq |x|_A$ for all x in A. We write "$A \hookrightarrow B$" if there exists such a normed reflection from A to B. Observe that, if $x_0, x_1, \dots$

is a (g,n)-controlled bad sequence over A, then $r(x_0), r(x_1), \ldots$ is also a (g,n)-controlled bad sequence, this time over B. Thus $A \hookrightarrow B$ implies $L_{A,g} \leq L_{B,g}$.

One can check that the reflection $r\colon \Sigma^*_{d,\Gamma} \to \Theta_{d,\Gamma}$ used in the proof of Theorem 3.6 is normed. If x is in $\Sigma^*_{d-1,\Gamma}$, then $r(x) = x$ itself with $|x|_{\Sigma^*_{d,\Gamma}} = |r(x)|_{\Theta_{d,\Gamma}}$. Otherwise, let x be factorized as in (9); $r(x)$ is given in (11). Then on the one hand

$$|x|_{\Sigma^*_{d,\Gamma}} = \max_{0\leq j<k\leq m} \bigl(m + \sum_{i=0}^{m} |x_i|, |v_k|_\Gamma, |x_j|_{\Sigma^*_{d-1,\Gamma}}\bigr) , \tag{20}$$

while on the other hand

$$|r(x)|_{\Theta_{d,\Gamma}} = \max\bigl(|x_0|_{\Sigma^*_{d-1,\Gamma}}, \max_{1\leq i\leq m-1}(m-1, |v_i|_\Gamma, |x_i|_{\Sigma^*_{d-1,\Gamma}}), |w_m|_\Gamma, |x_m|_{\Sigma^*_{d-1,\Gamma}}\bigr) , \tag{21}$$

which indeed satisfy

$$|x|_{\Sigma^*_{d,\Gamma}} \geq |r(x)|_{\Theta_{d,\Gamma}} , \tag{22}$$

and therefore $\Sigma^*_{d,\Gamma} \hookrightarrow \Theta_{d,\Gamma}$ for all d, Γ. Define now

$$\Theta'_{-1,\Gamma} \stackrel{\text{def}}{=} \mathbf{1} \quad \Theta'_{d,\Gamma} \stackrel{\text{def}}{=} \Theta'_{d-1,\Gamma} + \Theta'_{d-1,\Gamma} \times (\Gamma \times \Theta'_{d-1,\Gamma})^* \times \Gamma \times \Theta'_{d-1,\Gamma} , \tag{23}$$

where $\mathbf{1}$ denotes the singleton set. Because $\hookrightarrow$ is a precongruence for the elementary algebraic operations [36, Proposition 3.5], we deduce that $\Sigma^*_{d,\Gamma} \hookrightarrow \Theta'_{d,\Gamma}$ and thus

$$L_{\Sigma^*_{d,\Gamma},g} \leq L_{\Theta'_{d,\Gamma},g} \tag{24}$$

for all normed wqos Γ, all d, and all control functions g. Assuming $(\Gamma, \leq_\Gamma, |.|_\Gamma)$ to be elementary, then each $\Theta'_{d,\Gamma}$ is also elementary, *i.e.* we are going to be able to apply the Length Function Theorem to it and derive an upper bound for $L_{\Theta'_{d,\Gamma},g}$, and thereby for $L_{\Sigma^*_{d,\Gamma},g}$.

4.2.3. *Maximal Order Types.* The version of the Length Function Theorem we wish to apply requires the computation of the *maximal order type* of $\Theta'_{d,\Gamma}$. This is a measure of the complexity of a wqo $(A, \leq_A)$ defined in de Jongh and Parikh [17] as the maximal order type of its linearizations: a *linearization* $\prec$ of $\leq_A$ is a total linear ordering over A that contains $\leq_A \setminus \geq_A$ as a subrelation. Any such linearization of a wqo is well-founded and thus isomorphic to an ordinal, called its order type, and the maximal order type of $(A, \leq_A)$ is therefore the maximal such ordinal.

De Jongh and Parikh provide formulæ to compute the maximal order types of elementary wqos based on their algebraic decompositions as disjoint sums, Cartesian products, and Kleene star—using respectively the sum ordering, the product ordering, and the subword embedding ordering—:

$$\begin{aligned} o(A+B) &= o(A) \oplus o(B) , \\ o(A \times B) &= o(A) \otimes o(B) , \\ o(A^*) &= \begin{cases} \omega^{\omega^{o(A)-1}} & \text{if } A \text{ is finite,} \\ \omega^{\omega^{o(A)}} & \text{otherwise.} \end{cases} \end{aligned}$$

Here, the $\oplus$ and $\otimes$ operations are the *natural sum* and *natural product* on ordinals, defined for ordinals in CNF in ε_0 by

$$\sum_{i=1}^{m} \omega^{\beta_i} \oplus \sum_{j=1}^{n} \omega^{\beta'_j} \overset{\text{def}}{=} \sum_{k=1}^{m+n} \omega^{\gamma_k} \;, \quad \sum_{i=1}^{m} \omega^{\beta_i} \otimes \sum_{j=1}^{n} \omega^{\beta'_j} \overset{\text{def}}{=} \bigoplus_{i=1}^{m} \bigoplus_{j=1}^{n} \omega^{\beta_i \oplus \beta'_j} \;, \tag{25}$$

where $\gamma_1 \geq \cdots \geq \gamma_{m+n}$ is a reordering of $\beta_1, \dots, \beta_m, \beta'_1, \dots, \beta'_n$.

Schütte and Simpson [41] compute the exact maximal order type of a wqo related to $(\Sigma_d^*, \sqsubseteq_{\mathrm{p}})$. Here we are content with the maximal order type of $\Theta'_{d,\Gamma}$ (which also provides an upper bound on that of $\Sigma^*_{d,\Gamma}$). By (24) and [36, Proposition 5.2], we obtain:[2]

Proposition 4.1 (Length Function Theorem for $\Sigma^*_{d,\Gamma}$)**.** *Let $d \in \mathbb{N}$ and assume Γ is an elementary wqo and g is a control function. Then there exists a polynomial p independent of d, Γ, g such that $L_{\Sigma^*_{d,\Gamma}, g} \leq (p \circ g)^{o(\Theta'_{d,\Gamma})}$.*

4.2.4. *Finite Alphabets and Successor Control.* Proposition 4.1 is more general than useful for deriving upper bounds on PCSs verification. Even if we use a generalized priority alphabet in our PCSs, by allowing read and write rules to manipulate pairs in $\Sigma_{d,\Gamma}$ instead of only priorities in Σ_d, we can safely assume that the underlying alphabet Γ is finite and part of the input, with maximal order type $o(\Gamma) = |\Gamma|$. Similarly, the successor function $H(x) \overset{\text{def}}{=} x + 1$ can be chosen for the control function g, thus $p \circ g$ in Proposition 4.1 is simply a polynomial.

We can furthermore simplify the ordinal index in Proposition 4.1. First note that

$$o(\Theta'_{d,\Gamma}) < (\Omega_{2(d+1)+1})_{|\Gamma|} \tag{26}$$

for all d in $\mathbb{N}$, where $(\Omega_{2(d+1)+1})_{|\Gamma|} = \omega^{\cdot^{\cdot^{\cdot^{\omega^{|\Gamma|}}}}} \}2(d+1) \text{ stacked } \omega\text{'s}$. Second, an ordinal term α in CNF can be written as $\alpha = \omega^{\alpha_1} \cdot c_1 + \cdots + \omega^{\alpha_m} \cdot c_m$ for $\alpha > \alpha_1 > \cdots \alpha_m$ and $0 < c_1, \dots, c_m < \omega$. Define then its *maximum coefficient* $N(\alpha)$ as $\max_{1 \leq i \leq m}(N(\alpha_i), c_i)$. Observe that for all d in $\mathbb{N}$,

$$N(o(\Theta'_{d,\Gamma})) \leq |\Gamma| \;. \tag{27}$$

The following simplified statement then holds:

Corollary 4.2. *Let $d \in \mathbb{N}$ and Γ be a finite non-empty alphabet. Then there exists a polynomial h independent of d, Γ such that $L_{\Sigma^*_{d,\Gamma}, H}(n) \leq h^{\Omega_{2(d+1)+1}}(n)$ for all $n \geq |\Gamma|$.*

Proof. By Proposition 4.1 it suffices to show that $h^{o(\Theta'_{d,\Gamma})}(n) \leq h^{\Omega_{2(d+1)+1}}(n)$ for all $n \geq |\Gamma| > 0$. We show instead $h^{o(\Theta'_{d,\Gamma})}(n) \leq h^{(\Omega_{2(d+1)+1})_{|\Gamma|}}(n)$ since it allows to conclude. By monotonicity of the Hardy functions in the ordinal index for the *pointwise ordering*—see [19, Theorem 2.21.2] or [37, Lemma A.10]—it suffices to show that $o(\Theta'_{d,\Gamma}) \in (\Omega_{2(d+1)+1})_{|\Gamma|}[|\Gamma|]$, which is entailed by (26–27) and [37, Lemma A.5]. ☐

[2] To be precise, [36, Proposition 5.2] only provides bounds for *exponential* wqos—where there are no nested applications of the Kleene star operation—but it can be generalized to elementary wqos.

4.3. **Complexity Upper Bounds.** Now that we are armed with a Length Function Theorem for $(\Sigma^*_{d,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma})$, we can prove an upper bound for PCS verification:

Theorem 4.3 (Complexity of PCS Verification)**.** *Reachability and Inevitability of PCSs are in* $\mathbf{F}_{\varepsilon_0}$.

Let us explain the steps towards an upper bound for Termination in some detail; the results for Reachability and Inevitability are similar but more involved—see [37, 38] for generic complexity arguments for WSTSs.

A Finite Witness. Observe that, if an execution $C_0 \rightarrow_\# C_1 \rightarrow_\# C_2 \rightarrow_\# \cdots$ of the transition system $\mathcal{S}_\#$ verifies $C_i \leq_\# C_j$ for some indices $i < j$, then because $\mathcal{S}_\#$ is a WSTS, we can simulate the steps performed in this sequence after C_i but starting from C_j and build an infinite run. Conversely, if the system does not terminate, *i.e.* if there is an infinite execution $C_0 \rightarrow_\# C_1 \rightarrow_\# C_2 \rightarrow_\# \cdots$, then because of the wqo we will eventually find $i < j$ such that $C_i \leq_\# C_j$. Therefore, the system is non-terminating if and only if there is a finite witness of the form $C_0 \xrightarrow{*}_\# C_i \xrightarrow{+}_\# C_j$ with $C_i \leq_\# C_j$.

Controlled Witnesses. Another observation is that the size of successive configurations cannot grow arbitrarily along runs; in fact, the length of the channels contents can only grow by one symbol at a time using a write transition. This means that if we define $|C = (q, x_1, \ldots, x_m)| = \sum_{j=1}^m |x_j|$, then in an execution $C_0 \rightarrow_\# C_1 \rightarrow_\# C_2 \rightarrow_\# \cdots$, $|C_i| \leq |C_0| + i = H^i(|C_0|)$, *i.e.* any execution is *controlled* by the successor function H.

Applying the Length Function Theorem. Corollary 4.2 yields an $h^{\Omega_{2(d+1)+1}}(|C_0| + |\Gamma|)$ upper bound on the length of bad $(H, |C_0|)$-controlled sequences over $(\Sigma^*_{d,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma})$ for some polynomial h. It can be lifted to bound the maximal length of a witness in $\mathcal{S}_\#$, when considering instead the ordinal $o_S \overset{\text{def}}{=} (\Omega_{2(d+1)+1})^m \cdot |Q|$. Setting $|S| = |\Delta| + |Q| + d + m + |\Gamma|$, we see that this length is less than $H^{\varepsilon_0}(p(|S| + |C_0|)) < F_{\varepsilon_0}(p(|S| + |C_0|))$ for some fixed ordinal-recursive function p.

A Combinatorial Algorithm. Because the functions $(h^\alpha)_\alpha$ are elementary constructive [35, Theorem 5.1], the above discussion yields a non-deterministic algorithm in $\mathbf{F}_{\varepsilon_0}$ for Termination: compute $L = h^{o_S}(|C_0| + |\Gamma|)$ and look for an execution of length $L + 1$ in $\mathcal{S}_\#$. If one exists, it is necessarily a witness for nontermination; otherwise, the system is guaranteed to terminate from C_0.

We call this a *combinatorial* algorithm, as it relies on the combinatorial analysis provided by the Length Function Theorem to derive an upper bound on the size of a finite witness for the property at hand—here Termination, but the same kind of techniques can be used for Reachability and Inevitability.

5. Hardy Computations by PCSs

In this section we show how PCSs can weakly compute the Hardy functions H^α and their inverses for all ordinals α below Ω, which is the key ingredient for Theorem 5.8 below stating our hardness result. For this, we develop in Section 5.1 encodings $s(\alpha) \in \Sigma_d^*$ for ordinals $\alpha \in \Omega_d$ and show how PCSs can compute with these codes, *e.g.* build the code for λ_n from the code of a limit λ. This is used in Section 5.3 to design PCSs that *weakly compute* H^α and $(H^\alpha)^{-1}$ in the sense of Definition 5.5 below.

5.1. **Encoding Ordinals.** Our encoding of ordinal terms as strings in Σ_d^* employs strings of a particular form. For $0 \leq a \leq d$, we use the following equation to define the language $C_a \subseteq \Sigma_d^*$ of *codes*:

$$C_a \overset{\text{def}}{=} \varepsilon + C_a C_{a-1} a\,, \qquad\qquad C_{-1} \overset{\text{def}}{=} \varepsilon\,. \tag{28}$$

Let $C = C_{-1} + C_0 + \cdots + C_d$. Each C_a (and then C itself) is a regular language, with $C_a = (C_{a-1}a)^*$; for instance, $C_0 = 0^*$.

5.1.1. *Decompositions.* A code x is either the empty word ε, or belongs to a unique C_a. If $x \in C_a$ is not empty, it has a unique factorization $x = yza$ according to (28) with $y \in C_a$ and $z \in C_{a-1}$. Recall that $h(x)$ denotes the height function, thus a non-empty $x = a_1 \cdots a_\ell$ is a code if and only if $a_\ell = h(x)$ and $a_{i+1} - a_i \leq 1$ for all $i < \ell$ (we say that x *has no jumps*: priorities only increase smoothly along codes, but they can decrease sharply). For instance, 02 is not a code (it has a jump), but 001122 and 01223400123334 are codes.

The factor $z \in C_{a-1}$ in $x = yza$ can be developed further, as long as $z \neq \varepsilon$: a non-empty code $x \in C_d$ has a unique factorization as $x = y_d\, y_{d-1} \ldots y_a\, a^\frown d$ with $y_i \in C_i$ for $i = a, \ldots, d$, and where for $0 \leq a \leq b$, we write $a^\frown b$ for the *staircase* word $a(a+1)\cdots(b-1)b$, letting $a^\frown b = \varepsilon$ when $a > b$. We call this the *decomposition* of x. Note that the value of a is obtained by looking for the maximal suffix of x that is a staircase word. For example, $x = 23312340121234 \in C_4$ is a code and decomposes as

$$x = \overbrace{2331234}^{y_4}\overbrace{\varepsilon}^{y_3}\overbrace{012}^{y_2}\overbrace{\varepsilon}^{y_1}\overbrace{1234}^{1^\frown 4}.$$

5.1.2. *Ordinal Encoding.* With a code $x \in C$, we associate an ordinal term $\eta(x)$ given by

$$\eta(\varepsilon) \overset{\text{def}}{=} 0\,, \qquad\qquad \eta(yza) \overset{\text{def}}{=} \eta(y) + \omega^{\eta(z)}\,, \tag{29}$$

where $x = yza$ is the factorization according to (28) of $x \in C_a \setminus \{\varepsilon\}$. For example, $\eta(a) = \omega^0 = 1$ for all $a \in \Sigma_d$, $\eta(012) = \eta(234) = \omega^\omega$, and more generally $\eta(a^\frown b) = \Omega_{b-a}$. One sees that $\eta(x) < \Omega_{a+1}$ when $x \in C_a$.

The *decoding* function $\eta\colon C \to \Omega_{d+1}$ is onto (or surjective) but it is not bijective. However, it is a bijection between C_a and Ω_{a+1} for any $a \leq d$. Its converse is the level-a *encoding* function $s_a\colon \Omega_{a+1} \to C_a$, defined with

$$s_a\Big(\sum_{i=1}^{p} \gamma_i\Big) \overset{\text{def}}{=} s_a(\gamma_1)\cdots s_a(\gamma_p)\,, \qquad s_a(\omega^\alpha) \overset{\text{def}}{=} s_{a-1}(\alpha)\, a\,. \tag{30}$$

Thus $s_a(0) = \varepsilon$ and, for example,

$$s_5(1) = 5\,, \qquad s_5(3) = 555\,, \qquad s_5(\omega) = 45\,,$$
$$s_5(\omega^3) = 4445\,, \qquad s_5(\omega^\omega) = 345\,, \qquad s_5(\omega^{\omega^\omega}) = 2345\,,$$

$$s_5(\omega^3+\omega^2) = 4445445\,, \qquad\qquad s_5(\omega\cdot 3) = 454545\,.$$

We may omit the subscript when $a = d$, *e.g.* writing $s(1) = d$.

5.1.3. *Successors and Limits.* Let $x = y_d\, y_{d-1} \dots y_a\, a^\frown d$ be the decomposition of $x \in C_d \setminus \varepsilon$. By (29), x encodes a successor ordinal $\eta(x) = \beta + 1$ if and only if $a = d$, *i.e.*, if x ends with two d's (or has length 1). Since then $\beta = \eta(y_d \dots y_a)$, one obtains the "predecessor of x" by removing the final d.

If $a < d$, x encodes a limit λ. Combining (14) and (29), one obtains the encoding $(x)_n$ of λ_n with

$$(x)_n = y_d\, y_{d-1} \cdots y_{a+1} \big(y_a (a+1)\big)^n (a+2)^\frown d\,. \tag{31}$$

For instance, with $d = 5$, decomposing $x = 333345 = s(\omega^{\omega^4})$ gives $a = 3$, $x = y_5 y_4 y_3 3^\frown 5$, with $y_3 = 333$ and $y_5 = y_4 = \varepsilon$. Then $(x)_n = (3334)^n 5$, agreeing with, *e.g.* $s(\omega^{\omega^{3\cdot 2}}) = 333433345$.

5.2. **Robustness.** A crucial property of our ordinal encoding is *robustness*, *i.e.* that $x \sqsubseteq_{\mathrm{p}} x'$ should reflect the corresponding relation $H^{\eta(x)}(n) \leq H^{\eta(x')}(n)$ on Hardy computations.

Proposition 5.1 (Robustness)**.** *Let $a \geq 0$ and $x \sqsubseteq_{\mathrm{p}} x'$ be two strings in C_a. Then, $H^{\eta(x)}(n) \leq H^{\eta(x')}(n')$ for all $n \leq n'$ in $\mathbb{N}$.*

The proof of Proposition 5.1 requires delving in some of the theory of subrecursive functions, and is postponed until §5.2.3.

5.2.1. *Properties of the Hardy Hierarchy.* We first list some useful properties of Hardy computations (see [19] or [37, App. A] for details). The first fact is that each Hardy function is expansive and monotone in its argument n:

Fact 5.2 (Expansiveness and Monotonicity)**.** *For all α, α' in Ω and $n > 0, m$ in $\mathbb{N}$,*

$$n \leq H^\alpha(n)\,, \tag{32}$$

$$n \leq m \textit{ implies } H^\alpha(n) \leq H^\alpha(m)\,. \tag{33}$$

However, the Hardy functions are not monotone in the ordinal parameter: $H^{n+1}(n) = 2n + 1 > 2n = H^n(n) = H^\omega(n)$, though $n + 1 < \omega$. We will introduce an ordering on ordinal terms in §5.2.2 that ensures monotonicity of the Hardy functions.

Another handful fact is that we can decompose Hardy computations:

Fact 5.3. *For all α, γ in Ω, and n in $\mathbb{N}$,*

$$H^{\gamma+\alpha}(n) = H^\gamma(H^\alpha(n))\,. \tag{34}$$

Note that (34) holds for all ordinal terms, and not only for those α, γ such that $\gamma + \alpha$ is in CNF—this is a virtue of working with terms rather than set-theoretic ordinals.

5.2.2. *Ordinal Embedding.* We introduce a partial ordering $\sqsubseteq_o$ on ordinal terms, called *embedding*, and which corresponds to a strict tree embedding on the structure of ordinal terms. Formally, it is defined by $\alpha \sqsubseteq_o \beta \overset{\text{def}}{\Leftrightarrow} \alpha = \omega^{\alpha_1} + \cdots + \omega^{\alpha_p}$, $\beta = \omega^{\beta_1} + \cdots + \omega^{\beta_m}$, and there exist $1 \leq i_1 < i_2 < \ldots < i_p \leq m$ such that $\alpha_1 \sqsubseteq_o \beta_{i_1} \wedge \cdots \wedge \alpha_p \sqsubseteq_o \beta_{i_p}$. Note that $0 \sqsubseteq_o \alpha$ for all α, that $1 \sqsubseteq_o \alpha$ for all $\alpha > 0$. In general, $\alpha \not\sqsubseteq_o \omega^\alpha$ and $\lambda_n \not\sqsubseteq_o \lambda$. This ordering is congruent for addition and ω-exponentiation of terms:

$$\alpha \sqsubseteq_o \alpha' \text{ and } \beta \sqsubseteq_o \beta' \text{ imply } \alpha + \beta \sqsubseteq_o \alpha' + \beta' \,, \tag{35}$$

$$\alpha \sqsubseteq_o \alpha' \text{ implies } \omega^\alpha \sqsubseteq_o \omega^{\alpha'} \,, \tag{36}$$

and could in fact be defined alternatively by the axiom $0 \sqsubseteq_o \alpha$ and the two deduction rules (35) and (36).

We list a few useful consequences of the definition of $\sqsubseteq_o$:

$$\alpha \sqsubseteq_o \gamma + \omega^\beta \text{ implies } \alpha \sqsubseteq_o \gamma, \text{ or } \alpha = \gamma' + \omega^{\beta'} \text{ with } \gamma' \sqsubseteq_o \gamma \text{ and } \beta' \sqsubseteq_o \beta \,, \tag{37}$$

$$n \leq m \text{ implies } \lambda_n \sqsubseteq_o \lambda_m \,, \tag{38}$$

$$\alpha \sqsubseteq_o \lambda \text{ implies } \alpha \sqsubseteq_o \lambda_n, \text{ or } \alpha \text{ is a limit and } \alpha_n \sqsubseteq_o \lambda_n \,. \tag{39}$$

Proof of (37). Intuitively, there are two cases when we consider $\alpha \sqsubseteq_o \alpha' = \gamma + \omega^\beta$: either the ω^β summand of α' is in the range of the embedding or not. If it is not, then already $\alpha \sqsubseteq_o \gamma$. If it is, then α must be some $\gamma' + \omega^{\beta'}$ and $\omega^{\beta'} \sqsubseteq_o \omega^\beta$, which implies in turn $\beta' \sqsubseteq_o \beta$. □

Proof of (38). By induction on λ: indeed if $\lambda = \gamma + \omega^{\beta+1}$ then $\lambda_m = \gamma + \omega^\beta \cdot m$, which is $\lambda_n + \omega^\beta \cdot (m-n)$. If $\lambda = \gamma + \omega^{\lambda'}$, the ind. hyp. gives $\lambda'_n \sqsubseteq_o \lambda'_m$, hence $\lambda_n = \gamma + \omega^{\lambda'_n} \sqsubseteq_o \gamma + \omega^{\lambda'_m} = \lambda_m$. □

Proof of (39). By induction on λ. We can write λ as some $\gamma + \omega^\beta$ with $\beta > 0$ so that $\lambda_n = \gamma + (\omega^\beta)_n$. If $\alpha \sqsubseteq_o \gamma$, then $\alpha \sqsubseteq_o \lambda_n$ trivially. If $\alpha = \gamma' + 1$ is a successor, $1 \sqsubseteq_o (\omega^\beta)_n$ and again $\alpha \sqsubseteq_o \lambda_n$. There remains the case where $\alpha = \gamma' + \omega^{\beta'}$ is a limit (*i.e.* $\beta' > 0$) with $\gamma' \sqsubseteq_o \gamma$ and $\beta' \sqsubseteq_o \beta$. If β is a limit, then by ind. hyp. either $\beta' \sqsubseteq_o \beta_n$ and hence $\alpha \sqsubseteq_o \lambda_n$, or β' is a limit and $\beta'_n \sqsubseteq_o \beta_n$, hence $\alpha_n \sqsubseteq_o \lambda_n$. Finally, if $\beta = \delta + 1$ is a successor, then either $\beta' \sqsubseteq_o \delta$ so that $\alpha \sqsubseteq_o \gamma + \omega^\delta \sqsubseteq_o \gamma + \omega^\delta \cdot n = \lambda_n$, otherwise by (37), β' is a successor $\delta' + 1$ with $\delta' \sqsubseteq_o \delta$, and then $(\omega^{\beta'})_n = \omega^{\delta'} \cdot n \sqsubseteq_o \omega^\delta \cdot n = (\omega^\beta)_n$, hence $\alpha_n \sqsubseteq_o \lambda_n$. □

Proposition 5.4 (Monotonicity). *For all α, α' in Ω and n in $\mathbb{N}$,*

$$\alpha \sqsubseteq_o \alpha' \textit{ implies } H^\alpha(n) \leq H^{\alpha'}(n) \,.$$

Proof. Let us proceed by induction on a proof of $\alpha \sqsubseteq_o \alpha'$, based on the deduction rules (35) and (36). For the base case, $0 \sqsubseteq_o \alpha'$ implies $H^0(n) = n \leq H^{\alpha'}(n)$ by expansiveness.

For the inductive step with (35), if $\alpha \sqsubseteq_o \alpha'$ and $\beta \sqsubseteq_o \beta'$, then

$$\begin{aligned} H^{\alpha+\beta}(n) &= H^{\alpha}\big(H^{\beta}(n)\big) && \text{by (34)}\\ &\leq H^{\alpha}\big(H^{\beta'}(n)\big) && \text{by ind. hyp. and (33)}\\ &\leq H^{\alpha'}\big(H^{\beta'}(n)\big) && \text{by ind. hyp.}\\ &= H^{\alpha'+\beta'}(n)\;. && \text{by (34)} \end{aligned}$$

For the inductive step with (36), if $\alpha \sqsubseteq_o \alpha'$, then we show $H^{\omega^{\alpha}}(n) \leq H^{\omega^{\alpha'}}(n)$ by induction on α':

- If $\alpha' = 0$, then $\alpha = 0$ and we are done.
- If $\alpha' = \beta'+1$ is a successor, then by (37) either $\alpha \sqsubseteq_o \beta'$, or $\alpha = \beta+1$ with $\beta \sqsubseteq_o \beta'$. In the first case, $H^{\omega^{\alpha}}(n) \leq H^{\omega^{\beta'}}(n) \leq H^{\omega^{\beta'}}(H(n)) = H^{\omega^{\alpha'}}(n)$ by ind. hyp. and expansiveness. In the second case, we see by induction on $i \in \mathbb{N}$ that
$$\left(H^{\omega^{\beta}}\right)^{i}(n) \leq \left(H^{\omega^{\beta'}}\right)^{i}(n) \tag{40}$$
for all i and n thanks to the ind. hyp. Thus $H^{\omega^{\beta+1}}(n) = \left(H^{\omega^{\beta}}\right)^{n}(n) \leq \left(H^{\omega^{\beta'}}\right)^{n}(n) = H^{\omega^{\beta'+1}}(n)$ for all n, and we are done.
- If $\alpha' = \lambda'$ is a limit, then by (39) either $\alpha \sqsubseteq_o \lambda'_n$ or α is a limit λ and $\lambda_n \sqsubseteq_o \lambda'_n$. In the first case $H^{\omega^{\alpha}}(n) \leq H^{\omega^{\lambda_n}}(n)$ by ind. hyp.; in the second case $H^{\omega^{\lambda}}(n) = H^{\omega^{\lambda_n}}(n) \leq H^{\omega^{\lambda'_n}}(n) = H^{\omega^{\lambda'}}(n)$ using the ind. hyp. □

5.2.3. *Robustness.* We are now in position to prove Proposition 5.1:

Proof of Proposition 5.1. We prove that $\eta(x) \sqsubseteq_o \eta(x')$ by induction on x and conclude using Proposition 5.4 and Eq. (33). If $x = \varepsilon$, $\eta(x) = 0 \sqsubseteq_o \eta(x')$. Otherwise we can decompose x as yza according to (28) with $y \in C_a$ and $z \in C_{a-1}$. By (7), $x' = y'z'a$ with $y \sqsubseteq_{\mathrm{p}} y'$ and $za \sqsubseteq_{\mathrm{p}} z'a$. Observe that y' and z' are in C_a, and writing $z'a = z'_1 a \cdots z'_m a$ for the canonical decomposition of z'—where necessarily each z'_j is in C_{a-1}—, then $z \sqsubseteq_{\mathrm{p}} z'_1$ as there is no other way of disposing of the other occurrences of a in z'.

By ind. hyp., $\eta(y) \sqsubseteq_o \eta(y')$ and $\eta(z) \sqsubseteq_o \eta(z'_1)$. Then, because $\eta(x) = \eta(y) + \omega^{\eta(z)}$ and $\eta(x') = \eta(y') + \omega^{\eta(z'_1)} + \cdots + \omega^{\eta(z'_m)}$, we see by (35) and (36) that $\eta(x) \sqsubseteq_o \eta(x')$. □

5.3. **Robust Hardy Computations in PCSs.** Our goal is to implement in a PCS the *canonical* Hardy steps, denoted with $\xrightarrow{H}$, and specified by the following two rewrite rules on pairs in $\Omega \times \mathbb{N}$:

$$(\alpha+1, n) \xrightarrow{H} (\alpha, n+1) \qquad \text{for successors,} \tag{41}$$

$$(\lambda, n) \xrightarrow{H} (\lambda_n, n) \qquad \text{for limits.} \tag{42}$$

A *Hardy computation* for $H^{\alpha}(n)$ is a sequence of rewrites $(\alpha, n) = (\alpha_0, n_0) \xrightarrow{H} (\alpha_1, n_1) \xrightarrow{H} \cdots \xrightarrow{H} (\alpha_\ell, n_\ell)$. Note that (41–42) provide a rewriting view of the definition of Hardy functions in (15). Thus $H^{\alpha_i}(n_i)$ remains constant

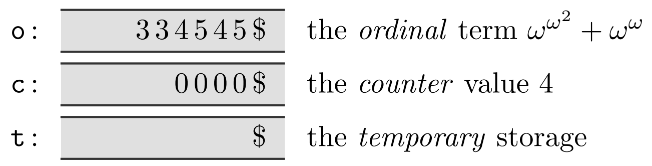


FIGURE 2. Channels for Hardy computations.

throughout the computation, and if $\alpha_\ell = 0$ then $n_\ell = H^\alpha(n)$, in which case we call the computation *complete*.

We do not implement canonical Hardy steps as PCSs, but construct instead *weak* computers, which might return lower values. Our PCSs for weak Hardy computations use three channels (see Figure 2), storing (codes for) a pair α, n on channels o (for "ordinal") and c (for "counter"), and employ an extra channel, t, for "temporary" storage. Instead of Σ_d, we use Σ_{d+1} with $d+1$ used as a position marker and written \$ for clarity: each channel always contains a single occurrence of \$.

Definition 5.5. A *weak Hardy computer* for Ω_{d+1} is a $(d+1)$-PCS S with channels $\mathtt{Ch} = \{\mathtt{o}, \mathtt{c}, \mathtt{t}\}$ and two distinguished states p_{beg} and p_{end} such that:

$$\text{if } (p_{\mathrm{beg}}, x\$, y\$, z\$) \xrightarrow{*}_{\#} (p_{\mathrm{end}}, u, v, w) \qquad \text{then } x \in C_d, y \in 0^+, z = \varepsilon, u, v, w \in \Sigma_d^*\$\,, \tag{safety}$$

$$\text{if } (p_{\mathrm{beg}}, s(\alpha)\$, 0^n\$, \$) \xrightarrow{*}_{\#} (p_{\mathrm{end}}, s(\beta)\$, 0^m\$, \$) \quad \text{then } H^\alpha(n) \geq H^\beta(m)\,. \tag{robustness}$$

Furthermore S is *complete* if for any $\alpha < \Omega_{d+1}$ and $n > 0$,

$$(p_{\mathrm{beg}}, s(\alpha)\$, 0^n\$, \$) \xrightarrow{*}_{\#} (p_{\mathrm{end}}, \$, 0^m\$, \$) \tag{complete}$$

for $m = H^\alpha(n)$, and it is *inv-complete* if

$$(p_{\mathrm{beg}}, \$, 0^m\$, \$) \xrightarrow{*}_{\#} (p_{\mathrm{end}}, s(\alpha)\$, 0^n\$, \$)\,. \tag{inv-complete}$$

Lemma 5.6 (PCSs weakly compute Hardy functions)**.** *For every $d \in \mathbb{N}$, there exists a weak Hardy computer S_d for Ω_{d+1} that is complete, and a weak S_d^{-1} that is inv-complete. Furthermore S_d and S_d^{-1} can be generated in* LOGSPACE *from d.*

We design a complete weak Hardy computer for Lemma 5.6 by assembling several components. The weak Hardy computer S_d is actually composed of two components $S_{d,+1}$ and $S_{d,\lambda}$ in charge respectively of applying the successor (41) and limit (42) steps, and similarly S_d^{-1} is composed of two components $S_{d,+1}^{-1}$ and $S_{d,\lambda}^{-1}$ in charge of reversing those steps.

5.3.1. *Successor Steps.* We start with "canonical successor steps", as *per* (41). They are implemented by $S_{d,+1}$, the PCS depicted in Figure 3. When working on codes, replacing $s(\alpha+1)$ by $s(\alpha)$ simply means removing the final d (see §5.1.3), but when the strings are in fifo channels this requires reading the whole contents of a channel and writing them back, relying on the \$ end-marker.

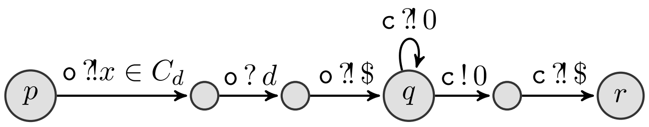


FIGURE 3. $S_{d,+1}$, a PCS for Hardy steps $(\alpha+1,n) \xrightarrow{H} (\alpha,n+1)$.

Remark 5.7 (Notational/graphical conventions). The label edge "$q \xrightarrow{\mathsf{c}?!0} q$" in Figure 3, with $\mathsf{c}?!0$ as label, is shorthand notation for "$q \xrightarrow{\mathsf{c}?0} \circ \xrightarrow{\mathsf{c}!0} q$", letting the intermediary state remain implicit. We also use meta-rules like "$p \xrightarrow{\mathsf{o}\,?!\,x \in C_d} \circ$" above to denote a subsystem tasked with reading and writing back a string x over o while checking that it belongs to C_d; since C_d is a regular language, such subsystems are trivial to implement. □

We first analyze the behavior of $S_{d,+1}$ when superseding of low-priority messages does not occur, *i.e.*, we first consider its "reliable" semantics. In this case, starting $S_{d,+1}$ in state p performs the step given in (41) for successor ordinals. More precisely, $S_{d,+1}$ guarantees

$$(p,\ s(\alpha+1)\$,\ 0^n\$,\ \$) \xrightarrow{*}_{\text{rel}} (r,u,v,w) \text{ iff } u = s(\alpha)\$ \ \wedge\ v = 0^{n+1}\$ \ \wedge\ w = \$\,. \tag{43}$$

Note that (43) refers to "$\xrightarrow{*}_{\text{rel}}$", with no superseding.

Observe that $S_{d,+1}$ has to non-deterministically guess where the end of $s(\alpha)$ occurs before reading $d\$$ in channel o, and will deadlock if it guesses incorrectly. We often rely on this kind of non-deterministic programming to reduce the size of the PCSs we build. Finally, we observe that if x does not end with dd (and is not just d), *i.e.*, if $\eta(x)$ is not a successor ordinal, then $S_{d,+1}$ will certainly deadlock.

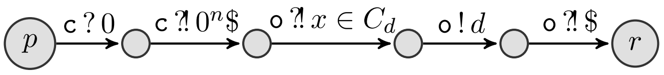


FIGURE 4. $S^{-1}_{d,+1}$, a PCS for inverse Hardy steps $(\alpha, n+1) \xrightarrow{H^{-1}} (\alpha+1,n)$.

We now consider $S^{-1}_{d,+1}$, the PCS depicted in Figure 4 that implements the inverse canonical steps $(\alpha,n+1) \xrightarrow{H^{-1}} (\alpha+1,n)$. Implementing such steps on codes is an easy string-rewriting task since $s(\alpha+1) = s(\alpha)d$, however our PCS must again read the whole contents of its channels, write them back with only minor modifications while fulfilling the safety requirement of Definition 5.5. When considering the reliable behavior, $S^{-1}_{d,+1}$ guarantees

$$(p, x\$, y\$, \$) \xrightarrow{*}_{\text{rel}} (r,u,v,w) \text{ iff } x \in C_d, \exists n : y = 0^{n+1}, u = s(\eta(x){+}1)\$, v = 0^n\$, \text{ and } w = \$. \tag{44}$$

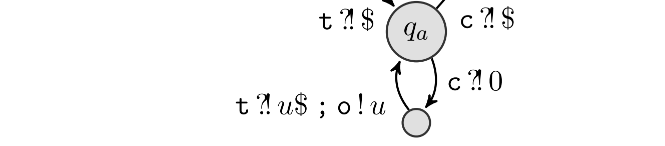
Consider now the behavior of $S_{d,+1}$ *when superseding may occur.* Note that a run $(p, x\$, y\$, z\$) \xrightarrow{*}_{\text{w}} (r, \ldots)$ from p to r is a *single-pass* run: it reads the whole contents of channels o and c once, and writes some new contents. This feature assumes that we start with a single \$ at the end of each channel, as expected by $S_{d,+1}$. For such single-pass runs, the PCS behavior with superseding semantics can be derived from the reliable behavior: for single-pass runs, $C \xrightarrow{*}_{\mathrm{w}} D$ if and only if $C \xrightarrow{*}_{\mathrm{rel}} D' \geq_{\#} D$ for some D'.

Combined with (43), the above remark entails robustness for $S_{d,+1}$: $(p, s(\alpha+1)\$, 0^n\$, \$) \xrightarrow{*}_{\mathrm{w}} (r, s(\beta)\$, 0^{n'}\$, \$)$ if and only if $s(\beta) \leq_{\#} s(\alpha)$ and $0^{n'}\$ \leq_{\#} 0^{n+1}\$$, *i.e.*, $n' \leq n+1$. With Proposition 5.1, we deduce $H^{\beta}(n') \leq H^{\alpha}(n)$.

The same reasoning applies to $S^{-1}_{d,+1}$ since this PCS also performs single-pass runs from p to r, hence $(p, s(\alpha)\$, 0^n\$, \$) \xrightarrow{*}_{\mathrm{w}} (r, s(\beta)\$, 0^{n'}\$, \$)$ if and only if $s(\beta) \leq_{\#} s(\alpha+1)$ and $n' \leq n-1$. Thus $H^{\beta}(n') \leq H^{\alpha}(n)$.

5.3.2. *Limit Steps.* Our next component is $S_{d,\lambda}$, see Figure 5, which implements the canonical Hardy steps for limits from (42). The construction follows (31): $S_{d,\lambda,a}$ reads (and writes back) the contents of channel `o`; guessing non-deterministically the decomposition $y_d \ldots y_{a+1} y_a a (a+1)^\frown d$ of $s(\lambda)$, it writes back $y_d \ldots y_{a+1}$ and copies y_a on the temporary `t` with $a+1$ appended. Then, a loop around state q_a copies 0^n from and back to `c`. Every time one 0 is transferred, the whole contents of `t`, initialized with $y_a(a+1)$, is copied to `o`. When the loop has been visited n times, $S_{d,\lambda,a}$ empties `t` and resumes the transfer of $s(\lambda)$ by copying the final $(a+2)^\frown d$.

For clarity, $S_{d,\lambda,a}$ as given in Figure 5 assumes that a is fixed. The actual $S_{d,\lambda}$ component guesses non-deterministically what is the value of a for the $s(\lambda)$ code on `o` and gives the control to $S_{d,\lambda,a}$ accordingly.



p_a
r_a
o ?! $y_d \cdots y_{a+1} \in C_d \cdots C_{a+1}$
o ?! $(a+2)^\frown d\$$
o ? $y_a \in C_a$; t ! y_a
t ?! \$
o ? $a(a+1)$; t ! $(a+1)$
t ? u
t ?! \$
q_a
c ?! \$
c ?! 0
t ?! $u\$$; o ! u

FIGURE 5. $S_{d,\lambda,a}$, a PCS for Hardy steps $(\lambda, n) \xrightarrow{H} (\lambda_n, n)$.

As far as reliable steps are considered, $S_{d,\lambda}$ guarantees

$$(p, s(\alpha)\$, 0^n\$, \$) \xrightarrow{*}_{\mathrm{rel}} (r, u, v, w) \text{ iff } \alpha \in \mathit{Lim}, u = s(\alpha_n)\$, v = 0^n\$, \text{ and } w = \$. \tag{45}$$

If superseding is allowed, a run $(p_a, s(\alpha)\$, 0^n\$, \$) \xrightarrow{*}_{\text{w}} (r_a, u, v, w)$ has the form

$$\begin{aligned}
(p_a, s(\alpha)\$, 0^n\$, \$) &\xrightarrow{*}_{\text{w}} C_0 = (q_a, (a+2)^\frown d\$x_0, 0^n\$, z_0\$)\\
&\xrightarrow{*}_{\text{w}} C_1 = (q_a, (a+2)^\frown d\$x_1, 0^{n-1}\$v_1, z_1\$)\\
&\quad\vdots\\
&\xrightarrow{*}_{\text{w}} C_n = (q_a, (a+2)^\frown d\$x_n, \$v_n, z_n\$)\\
&\xrightarrow{*}_{\text{w}} (r_a, x'_n\$, v'_n\$, \$)
\end{aligned}$$

where $C_i = (q_a, (a+2)^\frown d\$x_i, 0^{n-i}\$v_i, z_i\$)$ occurs when state q_a is visited for the ith time. Since the run is single-pass on c, we know that $v_i \leq_\# 0^i$ for all $i = 0, \dots, n$. Since it is single-pass on o, we deduce that $x_0 \leq_\# y_d \dots y_{a+1}$, then $x_{i+1} \leq_\# x_i z_i$ for all i, and finally $x'_n \leq_\# x_n (a+2)^\frown d$, with also $z_0 \leq_\# y_a(a+1)$. Finally, $z_{i+1} \leq_\# z_i$ since each subrun $C_i \xrightarrow{*}_{\text{w}} C_{i+1}$ is single-pass on t.

All this yields $x'_n \leq_\# s(\lambda_n)$ and $v'_n \leq_\# 0^n$. Hence $S_{d,\lambda}$ is safe and robust: $(p, s(\alpha)\$, 0^n\$, \$) \xrightarrow{*}_{\text{w}} (r, s(\beta), 0^{n'}\$, \$)$ if and only if $\alpha \in \mathit{Lim}$, $s(\beta) \leq_\# s(\alpha_n)$ and $n' \leq n$, entailing $H^\beta(n') \leq H^\alpha(n)$.

There remains to consider $S^{-1}_{d,\lambda}$, the PCS component that implements inverse Hardy steps for limits, see Figure 6. For given $a < d$, $S^{-1}_{d,\lambda,a}$ assumes that channel o contains $s(\lambda_n) = y_d \dots y_{a+1}[y_a(a+1)]^n(a+2)^\frown d$, guesses the position of the first $y_a(a+1)$ factor, and checks that it indeed occurs n times if c contains 0^n. This check uses copies $z_1, z_2, \dots$ of $y_a(a+1)$ temporarily stored on t. Then $S^{-1}_{d,\lambda}$ writes back $s(\lambda) = y_d \dots y_{a+1} z a^\frown d$ on o, where $z(a+1) = z_n$. The reader should be easily convinced that, as far as one considers reliable steps, $S^{-1}_{d,\lambda}$ guarantees

$$(p, s(\alpha)\$, 0^n\$, \$) \xrightarrow{*}_{\text{rel}} (r, u, v, w) \text{ iff } \exists\lambda \in \mathit{Lim} : \alpha = \lambda_n, u = s(\lambda)\$, v = 0^n\$, \text{ and } w = \$. \tag{46}$$

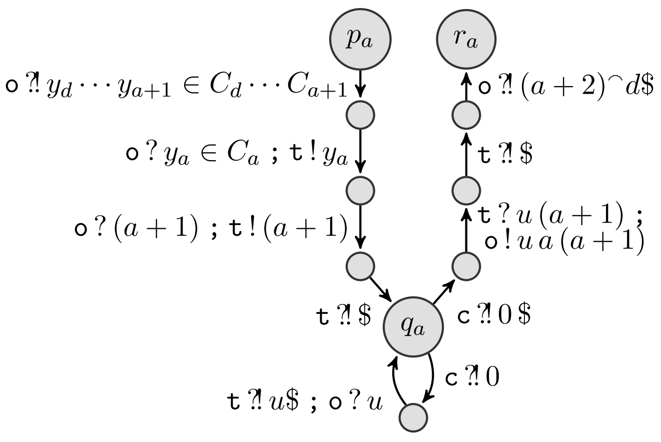


FIGURE 6. $S^{-1}_{d,\lambda,a}$, a PCS for inverse Hardy steps $(\lambda_n, n) \xrightarrow{H^{-1}} (\lambda, n)$.

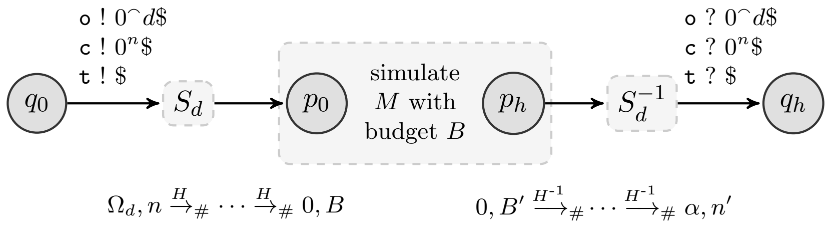


FIGURE 7. Schematics for Theorem 5.8.

When superseding is taken into account, a run from p to r in $S^{-1}_{d,\lambda}$ has the form $(p, s(\alpha)\$, 0^n\$, \$) \xrightarrow{*}_{\mathrm{w}} C_1 \xrightarrow{*}_{\mathrm{w}} C_2 \xrightarrow{*}_{\mathrm{w}} \cdots C_n \xrightarrow{*}_{\mathrm{w}} (r, u, v, w)$ where, for $i = 1, \ldots, n$, C_i is the ith configuration that visits state q_a. Necessarily, C_i is some $(q_a, x_i\$x, 0^{n-i}0\$v_i, z_i\$)$. The first visit to q_a has $x \leq_\# y_d \ldots y_{a+1}$, $z_1 \leq_\# y_a(a{+}1)$ and $v_1 = \varepsilon$, the following ones ensure $x_i = z_i x_{i+1}$, $z_{i+1} \leq_\# z_i$ and $v_{i+1} \leq_\# v_i 0$. Concluding the run requires $x_n = (a+2)^\frown d$. Finally $v \leq_\# 0^n\$$, $s(\beta) = y_d \ldots y_{a+1}(a+1)z_1 \ldots z_{n-1}(a+2)^\frown d$ and $u \leq_\# y_d \ldots y_{a+1} z a^\frown d$ for $z(a+1) = z_n \leq_\# z_{n-1} \leq_\# \cdots z_2 \leq_\# z_1 \leq_\# y_a(a+1)$. Thus $u = s(\beta)\$$ and $v = 0^{n'}\$$ imply $s(\beta) \leq_\# s(\lambda)$ for some λ with $s(\lambda_n) \leq_\# s(\alpha)$, yielding $H^\beta(n') \leq H^\lambda(n) = H^{\lambda_n}(n) \leq H^\alpha(n)$.

5.4. **Wrapping It Up.**

With the above weak Hardy computers, we have the essential gadgets required for our reductions. The wrapping-up is exactly as in [23, 39] (with a different encoding and a different machine model) and will only be sketched.

Theorem 5.8 (Verifying PCSs is Hard)**.** *Reachability and Termination of PCSs are $\mathbf{F}_{\varepsilon_0}$-hard.*

Proof. We exhibit a LOGSPACE reduction from the halting problem of a Turing machine M working in F_{ε_0} space to the Reachability problem in a PCS. We assume wlog. M to start in a state p_0 with an empty tape and to have a single halting state p_h that can only be reached after clearing the tape.

Figure 7 depicts the PCS S we construct for the reduction. Let $n \overset{\text{def}}{=} |M|$ and $d \overset{\text{def}}{=} n+1$. A run in S from the initial configuration to the final one goes through three stages:

(1) The first stage robustly computes $F_{\varepsilon_0}(|M|) = H^{\Omega_d}(n)$ by first writing $s(\Omega_d)\$$, *i.e.* $0^\frown d\$$, on o, $0^n\$$ on c, and \$ on t, then by using S_d to perform forward Hardy steps; thus upon reaching state p_0, o and t contain \$ and c encodes a budget $B \leq F_{\varepsilon_0}(|M|)$.

(2) The central component simulates M over c where the symbols 0 act as blanks—this is easily done by cycling through the channel contents to simulate the moves of the head of M on its tape. Due to superseding steps, the outcome upon reaching p_h is that c contains $B' \leq B$ symbols 0.

(3) The last stage robustly computes $(F_{\varepsilon_0})^{-1}(B')$ by running S^{-1}_d to perform backward Hardy steps. This leads to o containing the encoding

of some ordinal α and $\mathtt{c}$ of some n', but we empty these channels and check that $\alpha = \Omega_d$ and $n' = n$ before entering state q_h.

Because $H^{\Omega_d}(n) \geq B \geq B' \geq H^{\alpha}(n') = H^{\Omega_d}(n)$, all the inequalities are actually equalities, and the simulation of M in stage 2 has necessarily employed reliable steps. Hence, M halts if and only if $(q_h, \varepsilon, \varepsilon, \varepsilon)$ is reachable from $(q_0, \varepsilon, \varepsilon, \varepsilon)$ in S.

The case of (non-)Termination is similar, but employs a *time* budget in a separate channel in addition to the space budget, in order to make sure that the simulation of M terminates in all cases, and leads to a state q_h that is the only one from which an infinite run can start in S. □

6. Alternative Semantics

In this section we consider variant models for channel systems with priorities or losses and compare them with our PLCS model. The aim is to better understand the consequences, or lack thereof, of our choices.

We first consider *strict-superseding* systems, where messages may only supersede messages of strictly lower priority, and *overtaking* systems, where higher priority messages may move ahead of lower priority messages instead of erasing them. For completeness, we also discuss systems based on *priority queues*, where overtaking of lower priority messages is mandatory.

6.1. Strict-Superseding and Overtaking.

In this section, we discuss two alternative operational semantics for PCSs that may seem more natural than our standard $\mathcal{S}_{\#}$.

Strict-Superseding Semantics: Here, a high-priority message may only supersede messages having *strictly* lower priority. Formally, we replace the internal-superseding relation $C \xrightarrow{\#k}_{\#} C'$ with a new superseding relation, denoted $C \xrightarrow{\#k}_{\succ} C'$, and based on

$$x \xrightarrow{\#k}_{\succ} y \overset{\text{def}}{\Leftrightarrow} x = a_1 \cdots a_\ell \wedge y = a_1 \cdots a_{k-1} \cdot a_{k+1} \cdots a_\ell \wedge a_k < a_{k+1} .$$

Equivalently, one replaces the rewrite rules from Eq. (1) with $\big\{a\, a' \to a' \bigm| 0 \leq a < a' \leq d\big\}$.

Overtaking semantics: Here, a high-priority message may move ahead of a low-priority message but this does not erase the low-priority message. Formally, we replace $C \xrightarrow{\#k}_{\#} C'$ with $C \xrightarrow{\#k}_{\text{ot}} C'$, based on

$$x \xrightarrow{\#k}_{\text{ot}} y \overset{\text{def}}{\Leftrightarrow} x = a_1 \cdots a_k\, a_{k+1} \cdots a_\ell \wedge y = a_1 \cdots a_{k+1}\, a_k \cdots a_\ell \wedge a_k < a_{k+1} .$$

In rewriting terms, $\to_{\text{ot}}$ is defined by the rules $\big\{aa' \to a'\, a \bigm| 0 \leq a < a' \leq d\big\}$.

These two mechanisms drop fewer messages than our internal-superseding semantics. They may however be inadequate in case of network congestion, for instance they offer no solutions if all the messages in the congested buffers have the same priority. In any case, we show below that verification is undecidable for these two variant semantics (one can simulate Turing-powerful reliable channel systems with PCS under strict-superseding or overtaking semantics), which explains our choice of internal-superseding semantics.

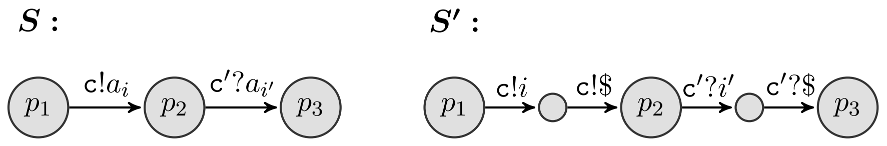


FIGURE 8. Simulating reliable channels (left) with "strict-superseding" or "overtaking" PCSs (right).

Theorem 6.1. *Reachability and Termination are undecidable for PCSs under both the strict-superseding and overtaking semantics.*

Proof. We reduce from the reachability problem for channel systems with reliable channels which is known to be undecidable [13]. A system S with reliable channels uses a finite (un-prioritized) alphabet $\Sigma = \{a_0, \ldots, a_{p-1}\}$ and is equipped with m channels $\mathsf{c}_1, \ldots, \mathsf{c}_m$. We simulate S with a PCS S' with strict superseding semantics having the same m channels and using the Σ_d priority alphabet with $d = p$. We use $d \in \Sigma_d$ as a separator, denoted \$ for clarity, while the other priorities $i \in \{0, \ldots, p-1\}$ represent the original messages a_i. A string $w = a_{i_1} \cdots a_{i_n} \in \Sigma^*$ is encoded as $\widetilde{w} \stackrel{\text{def}}{=} i_1 \,\$ \cdots i_n \,\$ \in \Sigma_d^*$. The actual reduction is obtained by equipping S' with transition rules that simulate the rules of S as illustrated in Figure 8. In essence, where S would write a_i, S' will write i followed by \$, *i.e.*, $\widetilde{a_i}$, and S' will read $i' \cdot \$$ where S would read $a_{i'}$.

With the *strict superseding* policy, the only superseding that can occur is to have \$ erase a preceding $i < \$$. This results in a channel containing two or possibly more consecutive \$ symbols, a pattern that will never vanish in this simulation and that eventually forbids reading on the involved channel. In particular, any run of S' that reaches a final configuration $C_{\text{end}} = (q_{\text{end}}, \varepsilon, \ldots, \varepsilon)$ has not used any strict superseding and thus corresponds to a run of the reliable channel system S. The same reduction works for Termination.

With the overtaking semantics, only \$ can overtake "original" messages of S in $\widetilde{w}$. However, such an overtake results in having two consecutive \$ symbols on the channel, a pattern that can never disappear. Behind the \$\$ block, two lower messages $0 \leq i, j < d$ may occur consecutively and be open to overtaking but this will not derail the simulation since S' cannot read beyond \$\$. □

6.2. **Channels as Priority Queues.** For the sake of completeness, let us mention channel systems where channels behave as priority queues. Here, reading from a channel will always read a message having highest priority among the contents of the channel. This can be seen as an extreme version of the overtaking semantics, where overtaking is mandatory. Such a model is not relevant for our purposes since it is not meant to handle congested communication links: instead, it uses priorities as a way of choosing in which order messages should be processed.

Let us mention that finite-state communicating systems with priority queues can easily simulate Minsky machines using a queue for each counter and two priorities: high-priority messages encode the counter value in unary, while a low-priority message can only be read in case of a zero-valued counter. They are hence Turing-powerful.

The most relevant case however is that of a single communication bus where many processes can read and write. Because only one queue is available—and still assuming a singleton alphabet for message contents—it is easy to see that priority queues systems are equivalent to Minsky counter machines restricted to *nested zero-tests.* Recall that for a machine with m counters $\mathtt{c}_1, \dots, \mathtt{c}_m$, nested zero-tests are tests of the form "$(\mathtt{c}_1 = 0 \wedge \mathtt{c}_2 = 0 \wedge \cdots \wedge \mathtt{c}_i = 0)$?", *i.e.*, one can only test the ith counter for emptiness when already the previous counters are empty. While Minsky counter machines with arbitrary zero-tests are Turing-powerful, they are equivalent to Petri nets when zero-tests are forbidden. Minsky machines with nested zero-tests are an intermediary model for which reachability is known to be decidable, see [34] and [9, Chapter 5].

7. Higher-Order Lossy Channel Systems

In this section, we introduce higher-order lossy channel systems, aka HOLCSs, a family of models that extend lossy channel systems. While a higher-order pushdown automaton has a stack of stacks of ... of stacks [5], a HOLCS has lossy channels inside lossy channels inside ... inside lossy channels. (In this setting, the "dynamic" lossy channel systems from [4] are a special case of 2LCSs, or second-order LCSs.) HOLCSs are well-structured, see Theorem 7.1, hence enjoy the usual decidability results from well-structured systems theory.

Our main result is that PCSs can simulate HOLCSs, see Theorem 7.7. On the one hand, this underlines the expressive power and naturalness of PCSs, in particular since the reductions we provide are quite straight-forward. On the other hand, we immediately obtain undecidability results: problems like boundedness or repeated control-state reachability are already undecidable for LCSs, *i.e.* first-order LCSs (see [28, 40]).

7.1. Syntax.

Formally, and given $k \in \mathbb{N}$, a *kth-order LCS* $S = (k, \Sigma, \mathtt{Ch}, Q, \Delta)$ has first-order, second-order, ..., up to kth-order, channels. We assume for simplicity that S has, for all $n = 1, \dots, k$, the same number m of nth-order channels, denoted $\mathtt{c}_{n,1}, \dots, \mathtt{c}_{n,m}$. Standardly, S uses a finite (un-prioritized) alphabet $\Sigma = \{a_1, \dots, a_p\}$. Write Σ^{*1} for the set of finite sequences of messages (usually written just Σ^*), and $\Sigma^{*(n+1)}$ for the set of finite sequences of elements from Σ^{*n}. We further order each $\Sigma^{*(n+1)}$ with $\leq_{*(n+1)}$, *i.e.*, the sequence extension of $(\Sigma^{*n}, \leq_{*n})$, equating $(\Sigma^{*0}, \leq_{*0})$ with $(\Sigma, =)$. Precisely, given two sequences $x = x_1 \dots x_\ell$ and $y = y_1 \dots y_m$ in $\Sigma^{*(n+1)}$, we let

$$x_1 \dots x_\ell \leq_{*(n+1)} y_1 \dots y_m \overset{\text{def}}{\Leftrightarrow} \exists 1 \leq i_1 < i_2 < \cdots < i_\ell \leq m : x_1 \leq_{*n} y_{i_1} \wedge \cdots \wedge x_\ell \leq_{*n} y_{i_\ell}$$

Using Higman's Lemma and induction over n, one sees that $(\Sigma^{*n}, \leq_{*n})$ is a well-quasi-order for any n.

At any given time, the contents of a nth-order channel is a sequence $w \in \Sigma^{*n}$, so that a configuration of S has the form $C = (q, x_{1,1}, \dots, x_{1,m}, \dots, x_{k,1}, \dots, x_{k,m})$ with q a control state and $x_{n,i} \in \Sigma^{*n}$ for all $1 \leq n \leq k$ and $1 \leq i \leq m$.

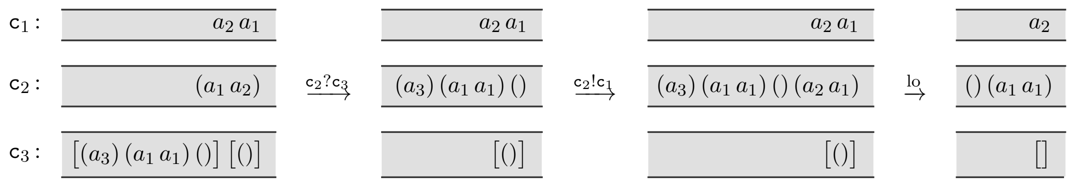


FIGURE 9. Reading, writing, and losing in HOLCSs.

These configurations are ordered by

$$(q, x_{1,1}, \ldots, x_{k,m}) \preccurlyeq_{\text{ho}} (q', y_{1,1}, \ldots, y_{k,m}) \overset{\text{def}}{\Leftrightarrow} q = q' \ \wedge\ \forall n, i : x_{n,i} \leq_{*n} y_{n,i} \,.$$

This ordering of configurations is a wqo since, for each $n = 1, \ldots, k$, $(\Sigma^{*n}, \leq_{*n})$ is.

7.2. **Semantics.** A HOLCS S as above has a *reliable* transition $C \xrightarrow{\delta} C'$ for $C = (q, \ldots, x_{i,n} \ldots)$ and $C' = (q', \ldots, y_{i,n}, \ldots)$ if $\delta = (q, op, q')$ is a rule moving control from q to q' and if the channel contents are modified according to the operation carried by δ, as we now define. There are four cases:

- ***op*** $= \mathsf{c}_{1,i}!a$**:** for some $a \in \Sigma$ and $1 \leq i \leq m$: then $y_{1,i} = x_{1,i} \cdot a$ while $y_{j,n} = x_{j,n}$ when $n > 1$ or $j \neq i$, *i.e.*, one writes a message to a 1st-order channel as in standard channel systems;
- ***op*** $= \mathsf{c}_{1,i}?a$**:** then one reads a message from a 1st-order channel, *i.e.*, $x_{1,i} = a \cdot y_{1,i}$ while the other channels are untouched;
- ***op*** $= \mathsf{c}_{n+1,i}!\mathsf{c}_{n,j}$**:** for some $1 \leq n < k$: then one appends a copy of the whole contents of $\mathsf{c}_{n,j}$ (a nth-order channel) to $\mathsf{c}_{n+1,i}$, where it becomes the last element of the higher-order sequence. Formally, $y_{n+1,i} = x_{n+1,i} \cdot x_{n,j}$ and the other channels are untouched;
- ***op*** $= \mathsf{c}_{n,i}?\mathsf{c}_{n+1,j}$**:** then one moves the first element of the higher-order sequence currently in $\mathsf{c}_{n+1,j}$ to $\mathsf{c}_{n,i}$ where it becomes the whole contents (the previous contents is erased). Formally, if $u \in \Sigma^{*n}$ is the first element of $x_{n+1,j}$, then $y_{n,i} = u$, $u \cdot y_{n+1,j} = x_{n+1,j}$, and the other channels are untouched.

In addition, all steps $C \to C'$ for $C' \prec_{\text{ho}} C$, called *losing steps*, are allowed. This states that at any time the system may lose individual messages, sequences of messages, sequences of sequences of ... of messages, anywhere inside the channels.

Figure 9 illustrates the behavior of higher-order channels under reads, writes and losses (control states omitted).

Theorem 7.1. *HOLCSs equipped with $\preccurlyeq_{ho}$ are well-structured.*

Proof. The ordering $C \preccurlyeq_{\text{ho}} C'$ entails $C' \xrightarrow{*} C$ (via losing steps). ☐

7.3. **Simulation by PCSs.** Let us consider a kth order LCS S with $\Sigma = \{a_1, \ldots, a_p\}$ and $k \cdot m$ channels. We assume that $m = 1$ in order to simplify our constructions and proofs but they extend directly to the more expressive cases where $m > 1$.

We simulate S with a d-PCS $\widetilde{S}$ having k channels and $d \overset{\text{def}}{=} p+k-1$. In Σ_d, the lower priorities $0, \ldots, p-1$ will be denoted $a_1, \ldots, a_p$ since they directly

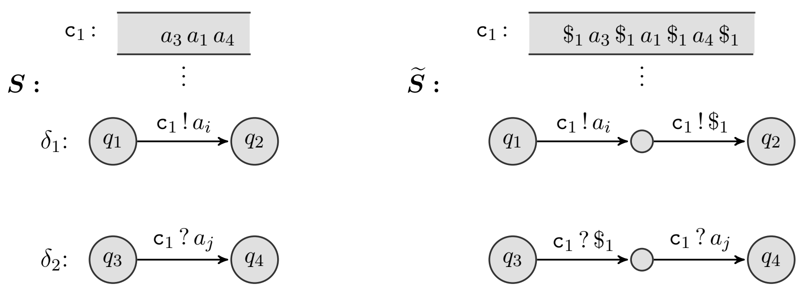


FIGURE 10. Simulation of HOLCS S with PCS $\widetilde{S}$: first-order rules.

encode the messages from Σ, while the higher priorities $p, \ldots, p+k-1$ will be denoted $\$_1, \ldots, \$_k$ since they are used as separators.

7.3.1. *Encoding Configurations.* A configuration $C = (q, x_1, \ldots, x_k)$ of S is encoded as $\widetilde{C} \stackrel{\text{def}}{=} (q, \lfloor x_1 \rfloor_1, \ldots, \lfloor x_k \rfloor_k)$, where, for $n = 1, \ldots, k$, $\lfloor x \rfloor_n$ denotes the *nth-level encoding* of a nth-level sequence $x = u_1 u_2 \cdots u_\ell \in \Sigma^{*n}$. Encodings are defined with

$$\lfloor a_{i_1} a_{i_2} \cdots a_{i_\ell} \rfloor_1 \stackrel{\text{def}}{=} \$_1 a_{i_1} \$_1 a_{i_2} \$_1 \cdots a_{i_\ell} \$_1 \,,$$

$$\lfloor u_1 u_2 \cdots u_\ell \rfloor_{n+1} \stackrel{\text{def}}{=} \$_{n+1} \lfloor u_1 \rfloor_n \$_{n+1} \lfloor u_2 \rfloor_n \$_{n+1} \cdots \lfloor u_\ell \rfloor_n \$_{n+1} \,.$$

Note that $\lfloor \varepsilon \rfloor_n = \$_n$ and differs from $\varepsilon \in \Sigma_d^*$. For $n = 1, \ldots, k$, we let $E_n \stackrel{\text{def}}{=} \{\lfloor x \rfloor_n \mid x \in \Sigma^{*n}\}$ denote the set of all n-level encodings. These are regular languages that are captured by the following regular expressions:

$$E_1 \stackrel{\text{def}}{=} \$_1 \cdot \big((a_1 + \cdots + a_p) \cdot \$_1\big)^* \,, \qquad E_{n+1} \stackrel{\text{def}}{=} \$_{n+1} \cdot \big(E_n \cdot \$_{n+1}\big)^* \,.$$

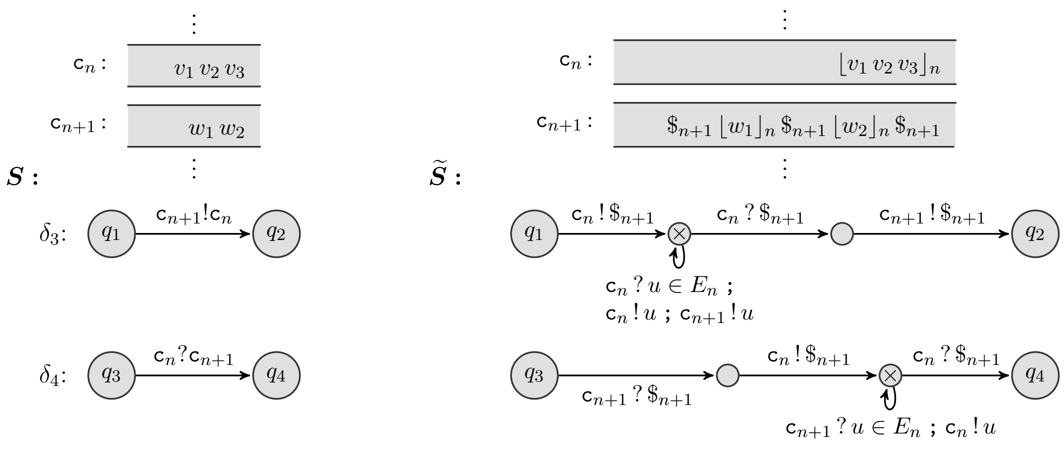
With this encoding, $(\Sigma^{*n}, \leq_{*n})$ and $(E_n, \leq_\#)$ are isomorphic:

Lemma 7.2. *For any $x, y \in \Sigma^{*n}$, $x \leq_{*n} y$ if, and only if, $\lfloor x \rfloor_n \leq_\# \lfloor y \rfloor_n$.*

Proof Idea. By induction on n. For the "$\Leftarrow$" direction it is easier to rely on priority embedding, *i.e.*, prove that $\lfloor x \rfloor_n \sqsubseteq_{\mathrm{p}} \lfloor y \rfloor_n$ implies $x \leq_{*n} y$ and apply Lemma 3.3. □

7.3.2. *Encoding First-Order Rules.* We may now complete the definition of $\widetilde{S}$ by describing its rules, with the goal of simulating the operational semantics of S while working on $\widetilde{C}$ and on encodings of sequences from some Σ^{*n}. This is easy for 1st-order rules that operate on c_1 only: where S writes a_i, $\widetilde{S}$ writes $a_i \cdot \$_1$. Where it reads a_j, $\widetilde{S}$ reads $\$_1 \cdot a_j$. This is illustrated in Figure 10.

7.3.3. *Encoding Higher-Order Rules.* Higher-order rules of the form $q_1 \xrightarrow{\mathsf{c}_{n+1}!\mathsf{c}_n} q_2$ are simulated in $\widetilde{S}$ as we illustrate in case δ_3 of Figure 11. Here $\widetilde{S}$ uses a loop, abbreviated as $\times \xrightarrow{\mathsf{c}_n?u;\mathsf{c}_n!u;\mathsf{c}_{n+1}!u} \times$, to append a copy of c_n's contents to c_{n+1}. This uses a high-priority $\$_{n+1}$ to mark the end of c_n's contents and ensure that all of it has been read (and written back). Note that the loop checks that u is in E_n, *i.e.*, is a well-formed encoding, which is done by following a DFA for E_n. When the transfer is completed, a $\$_{n+1}$ must be appended to c_{n+1} to ensure consistency.



c_n :
$v_1\,v_2\,v_3$
c_{n+1} :
$w_1\,w_2$
$\boldsymbol{S}$:
δ_3:
q_1
$\mathsf{c}_{n+1}!\mathsf{c}_n$
q_2
δ_4:
q_3
$\mathsf{c}_n?\mathsf{c}_{n+1}$
q_4
c_n :
$\lfloor v_1\,v_2\,v_3\rfloor_n$
c_{n+1} :
$\$_{n+1}\,\lfloor w_1\rfloor_n\,\$_{n+1}\,\lfloor w_2\rfloor_n\,\$_{n+1}$
$\widetilde{\boldsymbol{S}}$:
$\mathsf{c}_n\,!\,\$_{n+1}$
$\mathsf{c}_n\,?\,\$_{n+1}$
$\mathsf{c}_{n+1}\,!\,\$_{n+1}$
$\mathsf{c}_n\,?\,u \in E_n$;
$\mathsf{c}_n\,!\,u$; $\mathsf{c}_{n+1}\,!\,u$
$\mathsf{c}_{n+1}\,?\,\$_{n+1}$
$\mathsf{c}_n\,!\,\$_{n+1}$
$\mathsf{c}_n\,?\,\$_{n+1}$
$\mathsf{c}_{n+1}\,?\,u \in E_n$; $\mathsf{c}_n\,!\,u$

FIGURE 11. Simulation of HOLCS S with PCS $\widetilde{S}$: higher-order rules.

Simulating rules of the form $q_3 \xrightarrow{\mathsf{c}_n?\mathsf{c}_{n+1}} q_4$ follows the same logic (see Figure 11): $\widetilde{S}$ reads the first n-level encoding in c_{n+1}, checking that is is well-formed (with "$u \in E_n$") and writes it to c_n. Simultaneously, the previous contents of c_n is emptied by writing a $\$_{n+1}$ and reading it.

7.3.4. *Correctness.* The correctness of this simulation is captured by the next two propositions.

Proposition 7.3. *If S has a run $C \xrightarrow{*} C'$, then $\widetilde{S}$ has a run $\widetilde{C} \xrightarrow{*} \widetilde{C'}$.*

Proof Idea. On the one hand, $\widetilde{S}$ has been designed so that its behavior *without any superseding* directly mimics on encodings the effect of the reliable steps $C \to C'$. Then lossy steps in S are simulated by superseding since, thanks to Lemma 7.2, $C \succcurlyeq_{\text{ho}} C'$ entails $\widetilde{C} \geq_{\#} \widetilde{C'}$. □

There is an exact reciprocal to Proposition 7.3:

Proposition 7.4. *If $\widetilde{S}$ has a run $\widetilde{C} \xrightarrow{*} \widetilde{C'}$ then S has a run $C \xrightarrow{*} C'$.*

The correctness proof is harder in this direction since the steps of $\widetilde{S}$ are finer-grained than the steps of S. Note that a configuration $D = (q, w_1, \ldots, w_k)$ of $\widetilde{S}$ is not necessarily the encoding $\widetilde{C}$ of a configuration of S: if q is not an original state of S (*i.e.*, is one of the unnamed states depicted on the right-hand side of Figure 10 or Figure 11) then D is not a $\widetilde{C}$. Furthermore, if q is an original state, it is possible that some w_n does not belong to E_n.

With these difficulties in mind, we say that $D \in \mathit{Conf}_{\widetilde{S}}$ is *safe* if every w_n ends with a $\$_n$ (and contains no $\$_{n'}$ for $n' > n$), and that D is a *stable* configuration if q is an original state. We rely on the write-superseding semantics (see §2.1.3) for a better control on the form of the runs. We say that a write-superseding run $D_0 \to_{\text{w}} D_1 \to_{\text{w}} \cdots \to_{\text{w}} D_r$ in $\widetilde{S}$ is a *macro-step* if D_0 and D_r are the only stable configurations it visits (in essence, a macro-step just follows the rules introduced in $\widetilde{S}$ to simulate a single rule of

S using write-superseding semantics). We are now ready for the following lemmata.

Lemma 7.5. *Let $D \xrightarrow{+}_{\mathrm{w}} D'$ be any macro-step in $\widetilde{S}$. If D is safe then D' is safe too.*

Proof. Since the last message in a channel cannot be superseded, a $\mathtt{c}_n$ whose contents is safe remains safe if one does not read its final $\$_n$, or one appends some safe contents. We now consider all four cases for the macro-step $D \xrightarrow{+}_{\mathrm{w}} D'$.

- **δ_1:** using rules of the form $q_1 \xrightarrow{\mathtt{c}_1!a_i}\xrightarrow{\mathtt{c}_1!\$_1} q_2$, it writes a safe $a_i \cdot \$_1$ in $\mathtt{c}_1$ (and does not read from the other channels).
- **δ_2:** reading with some $q_3 \xrightarrow{\mathtt{c}_1?\$_1}\xrightarrow{\mathtt{c}_1?a_j} q_4$, the read $\$_1$ cannot be the final one in $\mathtt{c}_1$.
- **δ_3:** using a macro-step that simulates $q_1 \xrightarrow{\mathtt{c}_{n+1}!\mathtt{c}_n} q_2$, the last write to $\mathtt{c}_n$ is a safe $u \in E_n$, and the last write to $\mathtt{c}_{n+1}$ is $\$_{n+1}$.
- **δ_4:** using a macro-step that simulates $q_3 \xrightarrow{\mathtt{c}_n?\mathtt{c}_{n+1}} q_4$, the last write to $\mathtt{c}_n$ is a safe $u \in E_n$, which is also the last read from $\mathtt{c}_{n+1}$, implying that the final $\$_{n+1}$ in $\mathtt{c}_{n+1}$ cannot have been read. ☐

Lemma 7.6. *If D is safe and $D \xrightarrow{+}_{\mathrm{w}} \widetilde{C'}$ is a write-superseding macro-step in $\widetilde{S}$, then there is a $C \in Conf_S$ such that $D \geq_{\#} \widetilde{C}$ and $C \xrightarrow{+} C'$ in S.*

Proof. Write $D = (q, w_1, \ldots, w_k)$ and $\widetilde{C'} = (q', v_1, \ldots, v_k)$. For each $n = 1, \ldots, k$, we know that w_n is some $w'_n\$_n$ (since D is safe) and that v_n is some $\lfloor y_n \rfloor_n$. We now consider four cases for the macro-step $D \xrightarrow{+}_{\mathrm{w}} \widetilde{C}$:

- **δ_1:** it uses some $q = q_1 \xrightarrow{\mathtt{c}_1!a_i}\xrightarrow{\mathtt{c}_1!\$_1} q_2 = q'$. From the definition of $\rightarrow_{\mathrm{w}}$ (see §2.1.3) we deduce that $v_1 = x \cdot \$_1$ where x is a prefix of $w'_1 \cdot \$_1 \cdot a_i$ while $w_n = v_n$ for $n > 1$. Since $v_1 \in E_1$, either $v_1 = w_1 \cdot a_i \cdot \$_1$ and we take $\widetilde{C} = D$, or v_1 is a safe prefix of w_1, in which case we take $C = (q, y_1, \ldots, y_n)$.
- **δ_2:** it uses some $q = q_3 \xrightarrow{\mathtt{c}_1?\$_1}\xrightarrow{\mathtt{c}_1?a_j} q_4 = q'$ with no writing. The write-superseding semantics entails $w_1 = \$_1 \cdot a_j \cdot v_1$. Here $w_1 = \lfloor a_j \cdot y_1 \rfloor_1$ and we take $\widetilde{C} = D$.
- **δ_3:** the macro-step writes a $\$_{n+1}$ to $\mathtt{c}_n$ and reads it back, so that $u = w_n$ and we deduce that $w_n \in E_n$ and is some $\lfloor x \rfloor_n$. Furthermore v_n is u perhaps after some superseding and one obtains $x \geq_{*n} y_n$ from $u \geq_{\#} v_n$.

  On $\mathtt{c}_{n+1}$ the macro-step writes $u \cdot \$_{n+1}$ and we reason as in case δ_1: if write-superseding erases the $\$_{n+1}$ that closes w_{n+1} then $v_{n+1} \leq_{\#} w_{n+1}$ and we may take $C = (q, \ldots, y_{n-1}, x, y_{n+1}, \ldots)$, otherwise $v_{n+1} = w_{n+1} \cdot \lfloor x' \rfloor_n \$_{n+1}$ with $\lfloor x' \rfloor_n \leq_{\#} u$, we know that w_{n+1} is some $\lfloor x_{n+1} \rfloor_{n+1}$ and we may take $C = (q, \ldots, y_{n-1}, x, x_{n+1}, \ldots)$.
- **δ_4:** On $\mathtt{c}_{n+1}$ the macro-step just reads $\$_{n+1} \cdot u$, where $u \in E_n$ is some $\lfloor x \rfloor_n$. Necessarily w_n is $\lfloor x \cdot y_{n+1} \rfloor_{n+1}$. On $\mathtt{c}_n$, one writes $\$_{n+1} \cdot u$ and reads $\$_{n+1}$. Necessarily $y_n \leq_{\#} u$, hence $y_n \preccurlyeq_{\mathrm{ho}} x$. Taking $C = (q, \ldots, y_{n-1}, \varepsilon, x \cdot y_{n+1}, y_{n+2}, \ldots)$ works. ☐

We can now conclude our correctness proof.

Proof of Proposition 7.4. Assume $\widetilde{C} \xrightarrow{*} \widetilde{C'}$. We first apply Lemma 2.7 and deduce the existence of a write-superseding run $D_0 \xrightarrow{+}_{\rm w} \widetilde{C'}$ for some $D_0 \leq_\# \widetilde{C}$. Let us single out the stable configurations along this run and write it under the form $D_0 \xrightarrow{+}_{\rm w} D_1 \xrightarrow{+}_{\rm w} \cdots D_{r-1} \xrightarrow{+}_{\rm w} D_r = \widetilde{C'}$, i.e., as a sequence of r macro-steps.

We now reason by induction on r. If $r = 0$, then $D_0 = \widetilde{C'}$ and $\widetilde{C} \geq_\# \widetilde{C'}$, implying $C \succcurlyeq_{\rm ho} C'$ by Lemma 7.2 so that S has $C \xrightarrow{*} C'$ via lossy steps.

If $r > 0$, we first observe that D_0 is safe (since $\widetilde{C}$ is) hence $D_1, \ldots, D_r$ too by Lemma 7.5. Using Lemma 7.6 on $D_{r-1} \xrightarrow{+}_{\rm w} \widetilde{C'}$ implies that there is some $C'' \in \mathit{Conf}_S$ with $D_{r-1} \geq_\# \widetilde{C''}$ and $C'' \xrightarrow{+} C'$. On the other hand, the run $\widetilde{C} \xrightarrow{*} D_0 \xrightarrow{+}_{\rm w} D_{r-1} \xrightarrow{*} \widetilde{C''}$ can be transformed into $r - 1$ macro-steps. We can thus apply the ind. hyp. and deduce that $C \xrightarrow{*} C''$ in S, □

We may now state formally the main theorem of this section.

Theorem 7.7. *There is a logspace reduction that transforms reachability problems on kth-order HOLCSs to reachability problems on PCSs of level* $d = k + 1$.

Proof Sketch. The above ideas use $p+k$ priority levels. One can tighten these simulations to use fewer priorities and complete the proof of Theorem 7.7 by encoding the messages $a_0, \ldots, a_{p-1}$ as *fixed length* binary strings over $\{0, 1\}$ followed by a $\$_1$ separator. Then the prioritized alphabet $\{0, 1, \$_1, \ldots, \$_k\}$ with $k + 2$ priority levels suffices.

In particular $\{0, 1, \$\}$ is enough for LCSs. In the case of *weak* LCSs where the set of messages is linearly ordered (say $a_0 < a_1 < \cdots < a_{p-1}$) and where, in addition to message losses, any message can be replaced by a lower message inside the channels, we can further tighten this to $\{0, \$\}$ with a unary encoding of message a_i as $0^i\$$. □

Remark 7.8. The simulation of HOLCSs by PCSs is quite straightforward. We believe that a reduction from PCS reachability to HOLCS reachability must exist (on complexity-theoretical grounds) but we do not have at the moment any suggestion for a simple encoding of PCSs in HOLCSs. □

8. Applications of the Priority Embedding to Trees

In this section we show how tree orderings can be reflected into sequences over a priority alphabet. This illustrates the "power" of priority embeddings, and yields a proof that strong tree embeddings form a wqo as a byproduct. The consequence will be that PCSs can perform operations on encodings of trees in a *robust* way, *i.e.*, such that superseding steps respect the tree ordering. This was already the key insight

- in Section 5 when we encoded ordinals, seen as terms, into sequences over Σ_d^*, and
- in Section 7 when we encoded nested sequences over Σ^{n*}, which can be seen as terms of bounded depth, also into sequences over Σ_d^*.

We generalize these ideas here, and allow in particular for labeled trees.

8.1. **Reflecting Bounded Depth Trees.** One can provide a formal meaning to the notion of a wqo $(B, \leq_B)$ being more powerful than another one $(A, \leq_A)$ through *order reflections*: by Fact 3.5, if B reflects A, $(A, \leq_A)$ is a qo, and $(B, \leq_B)$ is a wqo, then $(A, \leq_A)$ is necessarily a wqo. Our goal is to show that $(\Sigma_{d,\Gamma}, \sqsubseteq_{\mathrm{p},\Gamma})$ reflects trees of depth at most $d+1$ endowed with the *strong tree-embedding* relation.

Given an alphabet Γ, we note $T(\Gamma)$ for the set of finite, ordered, unranked labeled trees (aka variadic terms) over Γ. Let d be a depth in $\mathbb{N}$. We work here with the set $T_d(\Gamma)$ of *trees of depth at most* d over Γ. Formally, $T_d(\Gamma)$ is defined by induction over d by $T_0(\Gamma) \stackrel{\text{def}}{=} \emptyset$, and for $d > 0$, $T_d(\Gamma)$ is the smallest set containing $T_{d-1}(\Gamma)$ and such that, if $t_1, \ldots, t_n$ are trees in $T_{d-1}(\Gamma)$ and f is in Γ, then the tree $f(t_1 \cdots t_n)$ obtained by adding an f-labeled root over them is in $T_d(\Gamma)$. When $n = 0$ we write "f" rather than $f()$. Figure 12 presents two labeled trees of depth 3.

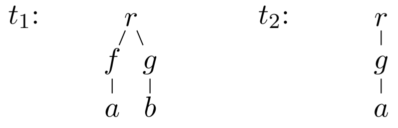


FIGURE 12. Two trees in $T_3(\{a, b, f, g, r\})$.

It will be convenient in the following to use the *extension* operation "@" on trees, which is defined for $n \geq 0$ by

$$f(t_1 \cdots t_n) \mathbin{@} t \stackrel{\text{def}}{=} f(t_1 \cdots t_n t) \; ; \tag{47}$$

in particular, $f \mathbin{@} t = f(t)$. For instance, t_1 in Figure 12 can be decomposed as $r \mathbin{@} (f \mathbin{@} a) \mathbin{@} (g \mathbin{@} b)$.

In case of a singleton Γ, we denote by "$\bullet$" its only element and write T_d for $T_d(\{\bullet\})$. For instance, $T_1 = \{\bullet\}$ contains a single tree.

8.1.1. *Strong Tree Embeddings.* Assume that $(\Gamma, \leq_\Gamma)$ is a wqo, and that we have already defined a well-quasi-ordering $\sqsubseteq_T$ on trees of maximal depth d—note that as a base case, since $T_0(\Gamma)$ is empty, it is vacuously well-quasi-ordered by the empty relation. We can lift it into a wqo $(T_{d+1}(\Gamma), \sqsubseteq_T)$ on trees of maximal depth $d+1$ by $f(t_1 \cdots t_n) \sqsubseteq_T f'(t'_1 \cdots t'_m) \stackrel{\text{def}}{\Leftrightarrow} f \leq_\Gamma f'$ and $t_1 \cdots t_n \sqsubseteq_{T*} t'_1 \cdots t'_m$, *i.e.*, by considering a product between $(\Gamma, \leq_\Gamma)$ and the sequence embedding ordering on tree sequences $(T_d(\Gamma)^*, \sqsubseteq_{T*})$: by Dickson's Lemma and Higman's Lemma, this defines a wqo on trees for every finite d, which we call the *strong tree embedding.* Put differently $t \sqsubseteq_T t'$ if t it can be obtained from t' by deleting whole subtrees or by decrementing node labels.

Strong tree embeddings refine the *homeomorphic tree embeddings* used in Kruskal's Tree Theorem; in general they do not give rise to a wqo, but in the case of bounded depth trees they do. The two trees in Figure 12 are *not* related by any homeomorphic tree embedding, and thus neither by strong tree embedding.

Observe that $\sqsubseteq_T$ is a precongruence for @:

$$t_1 \sqsubseteq_T t'_1 \text{ and } t_2 \sqsubseteq_T t'_2 \text{ imply } t_1 \mathbin{@} t_2 \sqsubseteq_T t'_1 \mathbin{@} t'_2 \, , \tag{48}$$

$$t \sqsubseteq_T t \mathbin{@} t' \, . \tag{49}$$

8.1.2. *Encoding Trees as Strings.* It is easy to encode trees into finite sequences. For instance, drawing inspiration from the ordinal encodings employed in Section 5, one might be tempted to encode the two trees in Figure 12 by

$$s_2(t_1) = (0,a)(1,f)(0,b)(1,g)(2,r) \tag{50}$$

$$s_2(t_2) = (0,a)(1,g)(2,r) \tag{51}$$

as the result of an inductive encoding $s_d(w(t_1 \cdots t_n)) \overset{\text{def}}{=} s_{d-1}(t_1) \cdots s_{d-1}(t_n) \cdot (d,w)$. Observe however that we wish our encoding to be an order reflection (*cf.* Section 3.2), which is not the case with s_d: we see that $s_2(t_2) \sqsubseteq_{\mathrm{p},\Gamma} s_2(t_1)$, although $t_2 \not\sqsubseteq_T t_1$. Over a singleton alphabet however, s_d *is* an order reflection from T_{d+1} to Σ_d^*.

Here we present a more redundant encoding, apt to handle arbitrary alphabets. We encode trees of bounded depth using the mapping $\lceil . \rceil_d \colon T_{d+1}(\Gamma) \to \Sigma_{d,\Gamma}^*$ defined by induction on d by

$$\lceil w(t_1 \cdots t_n) \rceil_d \overset{\text{def}}{=} \begin{cases} (d,w) & \text{if } n = 0\ , \\ \lceil t_1 \rceil_{d-1} \cdot (d,w) \cdots \lceil t_n \rceil_{d-1} \cdot (d,w) & \text{otherwise.} \end{cases} \tag{52}$$

For instance, if we fix $d = 2$, the trees in Figure 12 are encoded as

$$\lceil t_1 \rceil_2 = (0,a)(1,f)(2,r)(0,b)(1,g)(2,r)\ , \tag{53}$$

$$\lceil t_2 \rceil_2 = (0,a)(1,g)(2,r)\ . \tag{54}$$

This satisfies $\lceil t_2 \rceil_2 \not\sqsubseteq_{\mathrm{p},\Gamma} \lceil t_1 \rceil_2$ as desired.

8.1.3. *Proper Words.* Not every string in $\Sigma_{d,\Gamma}^*$ is the encoding of a tree according to $\lceil . \rceil_d$: for $0 \le a \le d$, we let

$$P_{-1,\Gamma} \overset{\text{def}}{=} \emptyset\ , \quad P_{a,\Gamma} \overset{\text{def}}{=} \bigcup_{w \in \Gamma} \left(P_{a-1,\Gamma} \cdot \{(a,w)\}\right)^* \cdot (P_{a-1,\Gamma} \cup \{\varepsilon\}) \cdot \{(a,w)\} \tag{55}$$

be the set of *proper encodings of height a*. Then $P \overset{\text{def}}{=} \bigcup_{a \le d} P_{a,\Gamma}$ is the set of *proper* words in $\Sigma_{d,\Gamma}^*$. A proper word x belongs to a unique $P_{a,\Gamma}$ with $a = h(x)$, where $h(x)$ is the height of x, and has then a canonical factorization of the form $x = x_1(a,w) \cdots x_m(a,w)$ with every x_j in $P_{a-1,\Gamma}$ and w in Γ.

Given a depth d, we see that $\lceil . \rceil_d$ is a bijection between $T_{d+1}(\Gamma)$ and $P_{d,\Gamma}$, with inverse $\tau \colon P_{d,\Gamma} \to T_{d+1}(\Gamma)^*$ defined using canonical decompositions by

$$\tau(x = x_1(h(x),w) \cdots x_m(h(x),w)) \overset{\text{def}}{=} w(\tau(x_1) \cdots \tau(x_m)) \tag{56}$$

$$= w \mathbin{@} \tau(x_1) \mathbin{@} \cdots \mathbin{@} \tau(x_m)\ . \tag{57}$$

Proposition 8.1. *The map $\lceil . \rceil_d$ is an order reflection from $(T_{d+1}^*, \sqsubseteq_{T*})$ to $(\Sigma_d^*, \sqsubseteq_{\mathrm{p}})$.*

Proof. Let x and x' be two proper words in $P_{d,\Gamma}$ with $x \sqsubseteq_{\mathrm{p},\Gamma} x'$; we show by induction on x that $\tau(x) \sqsubseteq_T \tau(x')$. We consider the canonical factorizations $x = x_1(d,w) \cdots x_m(d,w)$ and $x' = x'_1(d,w') \cdots x'_n(d,w')$ for $m,n \ge 0$, x_j, x'_j in $P_{d-1,\Gamma}$ for all j, and w, w' in Γ.

By definition of the generalized priority embedding, the m pairs (d, w) occurring in x must be mapped to some pairs (d, w') occurring in x' with $w \leq_\Gamma w'$. Hence there exist $1 \leq i_1, \ldots, i_m \leq n$ such that $x_j \sqsubseteq_{\mathrm{p},\Gamma} x'_{i_j}$. Therefore,

$$\begin{aligned}
\tau(x) &= w \mathbin{@} \tau(x_1) \mathbin{@} \cdots \mathbin{@} \tau(x_m) && \text{by (57)}\\
&\sqsubseteq_T w' \mathbin{@} \tau(x_1) \mathbin{@} \cdots \mathbin{@} \tau(x_m) && \text{by (48) since } w \leq_\Gamma w'\\
&\sqsubseteq_T w' \mathbin{@} \tau(x'_{i_1}) \mathbin{@} \cdots \mathbin{@} \tau(x'_{i_m}) && \text{by (48) and ind. hyp. on } x_j \sqsubseteq_{\mathrm{p},\Gamma} x'_{i_j}\\
&\sqsubseteq_T w' \mathbin{@} \tau(x'_1) \mathbin{@} \cdots \mathbin{@} \tau(x'_n) && \text{by (49)}\\
&= \tau(x') && \text{by (57)}\,.
\end{aligned}$$

□

Note that Proposition 8.1 provides an alternative proof of the fact that $(T_d(\Gamma), \sqsubseteq_T)$ is a wqo, thanks to Theorem 3.6 and Fact 3.5.

8.2. **Relationship to Tree Minors.** As a further application of the priority embedding, we demonstrate that we also subsume another wqo on trees, the *tree minor* ordering, by using the techniques of Gupta [22] to encode trees into generalized prioritized alphabets. The tree minor ordering is coarser than the homeomorphic embedding, but the upside is that trees of *unbounded* depth can be encoded into strings.

The trees considered in [22] are unlabeled finite rooted trees with an ordering on the children of every internal vertex, called *planar planted trees* therein, *i.e.* trees from $T \overset{\text{def}}{=} T(\{\bullet\})$. Figure 13 illustrates two such trees. The ordering on the children in particular implies that, for instance, the tree $\bullet(\bullet(\bullet), \bullet)$ is not equivalent to the tree $\bullet(\bullet, \bullet(\bullet))$. Gupta gives in [22] a constructive proof that planar planted trees are well-quasi-ordered under minors. Recall that t_1 is a *minor* of t_2 if t_1 can be obtained from t_2 by a series of edge contractions, *e.g.* in Figure 13, the left tree is a minor of the right one. Note that, however, the two trees are incomparable for the previously considered homeomorphic embeddings.

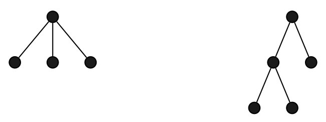


FIGURE 13. Two trees in T_2.

Gupta provides in [22] an effective linearization $\mathit{lin}\colon T \to \bigcup_{d\geq 0} \Sigma^*_{d,\Gamma}$ which associates with every tree t a word $\mathit{lin}(t)$ over the generalized prioritized alphabet $\Sigma_{d,\Gamma}$, where d is dubbed the *width* of t—which is at most its number of vertices—and $(\Gamma, =)$ is a finite alphabet with $\Gamma = \{0, 1, 2, 3\}$. Gupta continues by defining a so-called *immersion ordering* $\sqsubseteq_I$ on $\Sigma^*_{d,\Gamma}$ for any fixed $d \in \mathbb{N}$ as follows: given $x = (a_1, w_1)(a_2, w_2) \cdots (a_k, w_k) \in \Sigma^*_{d,\Gamma}$ and $y \in \Sigma^*_{d,\Gamma}$, $x \sqsubseteq_I y$ if y can be factored as $y = y_0 y_1 \cdots y_k y_{k+1}$ such that

$$y_i \in (\Sigma_{d,\Gamma} \setminus \Sigma_{d-a_i-1,\Gamma})^* \cdot (a_i, w_i) \cdot (\Sigma_{d,\Gamma} \setminus \Sigma_{d-a_i-1,\Gamma})^*,\ 1 \leq i \leq k.$$

The crucial relationship between trees in T, *lin* and $\sqsubseteq_I$ is established in [22, Theorem 4.1]: given planar planted trees $t_1, t_2 \in T$, whenever $lin(t_1) \sqsubseteq_I lin(t_2)$ then t_1 is a minor of t_2. By showing that $\sqsubseteq_I$ is a well-quasi-ordering, Gupta concludes that planar planted trees are well-quasi-ordered under minors.

The immersion ordering $\sqsubseteq_I$ is closely related to our generalized priority ordering. In fact, it is easily seen that $\sqsubseteq_{\mathrm{p},\Gamma}$ can be viewed as a sub-structure of $\sqsubseteq_I$. Define an automorphism $\kappa : \Sigma_{d,\Gamma} \to \Sigma_{d,\Gamma}$ as

$$\kappa(a, w) \overset{\text{def}}{=} (d - a, w)$$

which canonically extends to words over $\Sigma^*_{d,\Gamma}$. Now $x \sqsubseteq_{\mathrm{p},\Gamma} y$ in particular implies $\kappa(x) \sqsubseteq_I \kappa(y)$. Thus, our Theorem 3.6 yields as a corollary another proof that the immersion ordering $\sqsubseteq_I$ is a wqo.

Corollary 8.2. *The immersion ordering $\sqsubseteq_I$ is a well-quasi-ordering.*

8.3. **Further Applications.** As stated in the introduction to this section, our main interest in strong tree embeddings is in connection with structural orderings of ordinals; see Section 5. Bounded depth trees are also used in the verification of infinite-state systems as a means to obtain decidability results, in particular for tree pattern rewriting systems [21] in XML processing, and, using elimination trees [31], for bounded-depth graphs used e.g. in the verification of *ad-hoc* networks [18], the π-calculus [29], and programs [6]. These applications consider *labeled* trees, which motivate the generalized priority alphabets and embedding defined in Section 3.

The exact complexity of verification problems in the aforementioned models is currently unknown [21, 18, 29, 6]. Our encoding suggests they might be $\mathbf{F}_{\varepsilon_0}$-complete. We hope to see PCS Reachability employed as a "master" problem for $\mathbf{F}_{\varepsilon_0}$, like LCS Reachability for $\mathbf{F}_{\omega^\omega}$, which is used in reductions instead of more difficult proofs based on Turing machines and Hardy computations.

9. Concluding Remarks

We introduced Priority Channel Systems, a natural model for protocols and programs with differentiated, prioritized asynchronous communications, and showed how they give rise to well-structured systems with decidable model-checking problems.

We showed that Reachability and Termination for PCSs are $\mathbf{F}_{\varepsilon_0}$-complete, and we expect our techniques to be transferable to other models, *e.g.* models based on wqos on bounded-depth trees or graphs, whose complexity has not been analyzed [21, 18, 29, 6]. This is part of our current research agenda on complexity for well-structured systems [37].

In spite of their enormous worst-case complexity, we expect PCSs to be amenable to regular model checking techniques *à la* [1, 8]. This requires investigating the algorithmics of upward- and downward-closed sets of configurations wrt. the priority ordering. These sets, which are always regular, seem promising since $\sqsubseteq_{\mathrm{p}}$ shares some good properties with the better-known subword ordering, *e.g.* the upward- or downward-closure of a sequence $x \in \Sigma^*_d$ can be represented by a deterministic finite automaton with $|x|$ states.

Acknowledgments

We thank Lev Beklemishev who drew our attention to [41].

References

[1] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996. doi:10.1006/inco.1996.0053.

[2] P. A. Abdulla, K. Čerāns, B. Jonsson, and Y.-K. Tsay. Algorithmic analysis of programs with well quasi-ordered domains. *Information and Computation*, 160(1–2): 109–127, 2000. doi:10.1006/inco.1999.2843.

[3] P. A. Abdulla, J. Deneux, J. Ouaknine, and J. Worrell. Decidability and complexity results for timed automata via channel machines. In *ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 1089–1101. Springer, 2005. doi:10.1007/11523468_88.

[4] P. A. Abdulla, M. F. Atig, and J. Cederberg. Timed lossy channel systems. In *FST&TCS 2012*, volume 18 of *Leibniz International Proceedings in Informatics*, pages 374–386. Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.374.

[5] A. V. Aho. Nested stack automata. *J. ACM*, 16(3):383–406, 1969. doi:10.1145/321526.321529.

[6] K. Bansal, E. Koskinen, T. Wies, and D. Zufferey. Structural counter abstraction. In *TACAS 2013*, volume 7795 of *Lecture Notes in Computer Science*, pages 62–77. Springer, 2013. doi:10.1007/978-3-642-36742-7_5.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, December 1998. URL http://rfc.net/rfc2475.html.

[8] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using QDDs. *Formal Methods in System Design*, 14(3):237–255, 1999. doi:10.1023/A:1008719024240.

[9] R. Bonnet. *Theory of Well-Structured Transition Systems and Extended Vector-Addition Systems.* Thèse de doctorat, ENS Cachan, France, 2013. URL http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/bonnet-phd13.pdf.

[10] R. V. Book, M. Jantzen, and C. Wrathall. Monadic Thue systems. *Theoretical Comput. Sci.*, 19:231–251, 1982. doi:10.1016/0304-3975(82)90036-6.

[11] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. *Theoretical Comput. Sci.*, 221(1–2): 211–250, 1999. doi:10.1016/S0304-3975(99)00033-X.

[12] P. Bouyer, N. Markey, J. Ouaknine, Ph. Schnoebelen, and J. Worrell. On termination and invariance for faulty channel machines. *Formal Aspects of Computing*, 24(4–6): 595–607, 2012. doi:10.1007/s00165-012-0234-7.

[13] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30 (2):323–342, 1983. doi:10.1145/322374.322380.

[14] G. Cécé and A. Finkel. Verification of programs with half-duplex communication. *Information and Computation*, 202(2):166–190, 2005. doi:10.1016/j.ic.2005.05.006.

[15] G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996. doi:10.1006/inco.1996.0003.

[16] P. Chambart and Ph. Schnoebelen. The ordinal recursive complexity of lossy channel systems. In *LICS 2008*, pages 205–216. IEEE Press, 2008. doi:10.1109/LICS.2008.47.

[17] D. H. J. de Jongh and R. Parikh. Well-partial orderings and hierarchies. *Indagationes Mathematicae*, 39(3):195–207, 1977. doi:10.1016/1385-7258(77)90067-1.

[18] G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *Concur 2010*, volume 6269 of *Lecture Notes in Computer Science*, pages 313–327. Springer, 2010. doi:10.1007/978-3-642-15375-4_22.

[19] M. Fairtlough and S. S. Wainer. Hierarchies of provably recursive functions. In S. Buss, editor, *Handbook of Proof Theory*, chapter III, pages 149–207. Elsevier, 1998. doi:10.1016/S0049-237X(98)80018-9.

[20] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Comput. Sci.*, 256(1–2):63–92, 2001. doi:10.1016/S0304-3975(00)00102-X.

[21] B. Genest, A. Muscholl, O. Serre, and M. Zeitoun. Tree pattern rewriting systems. In *ATVA 2008*, volume 5311 of *Lecture Notes in Computer Science*, pages 332–346. Springer, 2008. doi:10.1007/978-3-540-88387-6_29.

[22] A. Gupta. A constructive proof that trees are well-quasi-ordered under minors. In *LFCS 1992*, volume 620 of *Lecture Notes in Computer Science*, pages 174–185. Springer, 1992. doi:10.1007/BFb0023872.

[23] S. Haddad, S. Schmitz, and Ph. Schnoebelen. The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. In *LICS 2012*, pages 355–364. IEEE Press, 2012. doi:10.1109/LICS.2012.46.

[24] A. Kurucz. Combining modal logics. In *Handbook of Modal Logics*, chapter 15, pages 869–926. Elsevier, 2006. doi:10.1016/S1570-2464(07)80018-8.

[25] S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Transactions on Computational Logic*, 9(2), 2008. doi:10.1145/1342991.1342994.

[26] J.-Y. Le Boudec. The Asynchronous Transfer Mode: a tutorial. *Computer Networks and ISDN Systems*, 24(4):279–309, 1992. doi:10.1016/0169-7552(92)90114-6.

[27] M. Löb and S. Wainer. Hierarchies of number theoretic functions, I. *Archiv für Mathematische Logik und Grundlagenforschung*, 13:39–51, 1970. doi:10.1007/BF01967649.

[28] R. Mayr. Undecidable problems in unreliable computations. *Theoretical Comput. Sci.*, 297(1–3):337–354, 2003. doi:10.1016/S0304-3975(02)00646-1.

[29] R. Meyer. On boundedness in depth in the π-calculus. In *IFIP TCS 2008*, volume 273 of *IFIP*, pages 477–489. Springer, 2008. doi:10.1007/978-0-387-09680-3_32.

[30] A. Muscholl. Analysis of communicating automata. In *LATA 2010*, volume 6031 of *Lecture Notes in Computer Science*, pages 50–57. Springer, 2010. doi:10.1007/978-3-642-13089-2_4.

[31] P. Ossona de Mendez and J. Nešetřil. *Sparsity*, chapter 6: Bounded height trees and tree-depth, pages 115–144. Springer, 2012. doi:10.1007/978-3-642-27875-4_6.

[32] J. Ouaknine and J. Worrell. On the decidability and complexity of Metric Temporal Logic over finite words. *Logical Methods in Computer Science*, 3(1):1–27, 2007. doi:10.2168/LMCS-3(1:8)2007.

[33] J. K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *PSTV '87*, pages 207–219. North-Holland, 1987.

[34] K. Reinhardt. Reachability in Petri nets with inhibitor arcs. In *RP 2008*, volume 223 of *Electronic Notes in Theoretical Computer Science*, pages 239–264. Elsevier, 2008. doi:10.1016/j.entcs.2008.12.042.

[35] S. Schmitz. Complexity hierarchies beyond elementary. Manuscript, Dec. 2013. URL http://arxiv.org/abs/1312.5686. arXiv:1312.5686 [cs.CC].

[36] S. Schmitz and Ph. Schnoebelen. Multiply-recursive upper bounds with Higman's Lemma. In *ICALP 2011*, volume 6756 of *Lecture Notes in Computer Science*, pages 441–452. Springer, 2011. doi:10.1007/978-3-642-22012-8_35.

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

[38] S. Schmitz and Ph. Schnoebelen. The power of well-structured systems. In *Concur 2013*, volume 8052 of *Lecture Notes in Computer Science*, pages 5–24. Springer, 2013. doi:10.1007/978-3-642-40184-8_2. Invited talk.

[39] Ph. Schnoebelen. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. In *MFCS 2010*, volume 6281 of *Lecture Notes in Computer Science*, pages 616–628. Springer, 2010. doi:10.1007/978-3-642-15155-2_54.

[40] Ph. Schnoebelen. Lossy counter machines decidability cheat sheet. In *RP 2010*, volume 6227 of *Lecture Notes in Computer Science*, pages 51–75. Springer, 2010. doi:10.1007/978-3-642-15349-5_4.

[41] K. Schütte and S. G. Simpson. Ein in der reinen Zahlentheorie unbeweisbarer Satz über endliche Folgen von natürlichen Zahlen. *Archiv für Mathematische Logik und Grundlagenforschung*, 25(1):75–89, 1985. doi:10.1007/BF02007558.

LSV, ENS Cachan & CNRS & INRIA, France
E-mail address: {haase,schmitz,phs}@lsv.ens-cachan.fr