The Power of Priority Channel Systems*

Christoph Haase, Sylvain Schmitz, and Philippe Schnoebelen

LSV, ENS Cachan & CNRS, France

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". These models are well-suited for the formal specification and algorithmic analysis of communication protocols and concurrent programs [6, 7, 9]. They are also a fundamental model of computation, closely related to Post's tag systems.

A particularly interesting class of channel systems are the so-called *lossy channel systems (LCSs)*, where channels are unreliable and may lose messages [10, 4, 8]. For LCSs, several important behavioral properties, like safety or inevitability, are decidable. This is because these systems are *well-structured*: transitions are monotonic wrt. a (decidable) well-quasi-ordering of the configuration space [2, 14]. 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, etc. [3, 18, 23, 19]. They are also a fundamental model of computation capturing the $\mathbf{F}_{\omega^\omega}$-complexity level in Wainer *et al.*'s Fast-Growing Hierarchy, see [11, 24, 25].

Lossy channel systems do not provide a natural 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) in networking architectures, and is usually implemented by allowing for tagging messages with some relative priority. For instance, the Differentiated Services (DiffServ) architecture described in RFC 2475, 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.

* Work supported by the ReacHard project, ANR grant 11-BS02-001-01.

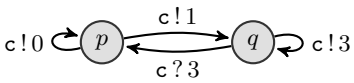

Fig. 1. A simple single-channel 3-PCS.

Our contributions. 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. For technical simplicity, our model abstracts from the contents of messages by just considering the priority levels (but see the full version of this work at http://arxiv.org/abs/1301.5500 for a general setting allowing arbitrary message contents and priorities).

Priority channel systems rely on the *(prioritized) superseding ordering*, a new ordering that has not been considered before in well-structured systems (though it is related to the gap-embedding of [28]). Showing that it is a well-quasi-ordering entails the decidability of safety and termination (among others) for PCSs. We also show the aforementioned problems to become undecidable for channel systems that build upon more restrictive priority mechanisms.

Using techniques from [24, 28], we show in Sec. 5 an $\mathbf{F}_{\varepsilon_0}$ upper bound on the complexity of PCS verification, far higher than the $\mathbf{F}_{\omega^\omega}$-complete complexity of LCSs. We then prove in Sec. 6 a matching lower bound and this is the main technical result of the paper: building upon techniques developed for less powerful models [11, 27, 17], 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.

Along the way, we show how other well-quasi-ordered data structures from the literature, e.g. trees with (strong) embedding, can be reflected in the superseding ordering (Sec. 4). This paves the way to new $\mathbf{F}_{\varepsilon_0}$ upper bounds for problems in other areas of algorithmic verification, whose complexity is wide open.

2 Priority Channel Systems

We define Priority Channel Systems as consisting of a single process since this is sufficient for our purposes in this paper.[1] For every $d \in \mathbb{N}$, the *level-d priority alphabet* is $\Sigma_d \stackrel{\text{def}}{=} \{0, 1, \ldots, d\}$. A *level-d priority channel system* (a "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* (see below).

2.1 Semantics

The operational semantics of a PCS S is given in the form of a transition system. We let $Conf_S \stackrel{\text{def}}{=} Q \times (\Sigma_d^*)^m$ be the set of all configurations of S, denoted $C, D, \ldots$

[1] Systems made of several concurrent components can be represented by a single process obtained as an asynchronous product of the components.

A configuration $C = (q, x_1, \ldots, x_m)$ records an instantaneous control point (a state 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 $x = a_1 \ldots a_\ell$ 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 $\rightarrow_{\mathrm{rel}}$ and $\rightarrow_{\#}$.

Reliable Semantics. We start with $\rightarrow_{\mathrm{rel}}$ 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 $C \xrightarrow{\delta}_{\mathrm{rel}} C'$ 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 $C \xrightarrow{\delta}_{\mathrm{rel}} C'$ 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.

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 $x \xrightarrow{\#k}_{\#} y \overset{\text{def}}{\Leftrightarrow} x$ is some $a_1 \ldots a_\ell$, $1 \leq k < |x| = \ell$, $a_k \leq a_{k+1}$ and $y = a_1 \ldots a_{k-1} a_{k+1} \ldots a_\ell$. In other words, the k-th 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^* according to the rules $\{aa' \rightarrow a' \mid 0 \leq a \leq a' \leq d\}$.

This is extended to steps between configurations by $C = (q, x_1, \ldots, x_m) \xrightarrow{\mathsf{c}_i \# k}_{\#} C' = (q', y_1, \ldots, y_m) \overset{\text{def}}{\Leftrightarrow} q = q'$ and $x_i \xrightarrow{\#k}_{\#} y_i$ (and $x_j = y_j$ for $j \neq i$). Furthermore, every reliable step is a valid step: for any rule δ, $C \xrightarrow{\delta}_{\#} C'$ iff $C \xrightarrow{\delta}_{\mathrm{rel}} C'$, giving rise to a second transition system associated with S: $\mathcal{S}_{\#} \overset{\text{def}}{=} (Conf_S, \rightarrow_{\#})$. E.g., the PCS from Fig. 1 can perform

$$p, 0200 \xrightarrow{!1}_{\#} q, 02\underline{0}01 \xrightarrow{\#3}_{\#} q, \underline{0}201 \xrightarrow{\#1}_{\#} q, 2\underline{0}1 \xrightarrow{\#2}_{\#} q, 21 \; .$$

The internal-superseding semantics allows superseding to occur at any time and anywhere in the channel. It is appropriate when abstracting from situations where end-to-end communication actually goes through a series of consecutive relays, network switches, and buffers, each of them possibly handling the incoming traffic with a so-called *write-superseding* policy, where writes immediately "consume" the congested messages in front of them in the buffer. We develop this aspect in the full version, where we also prove the two semantics to be essentially equivalent.

2.2 Related Models

It is possible to consider a stricter policy for priorities where a higher-priority message may only supersede messages with *strictly* lower priority. Another priority mechanism one could envision sees higher-priority messages *overtake* those of lower priority without dropping them. These two mechanisms are more restrictive, i.e., drop fewer messages, but they may be powerless in case of network congestion: for instance, they offer no solutions if all the messages in the congested buffers have the same priority. From a more theoretical standpoint, both semantics also yield Turing-powerful models; details are provided in the full version.

Theorem 1. *Reachability in PCSs is undecidable both for strict superseding and overtaking semantics.*

We conclude by observing that PCSs can simulate lossy channel systems. In fact they can simulate the dynamic lossy channel systems and the timed lossy channel systems from [1], see the full version. Hence reachability and termination (see Thm. 3) are at least $\mathbf{F}_{\omega^\omega}$-hard for PCSs, and problems like boundedness or repeated control-state reachability (see [26] for more) are undecidable for them.

2.3 Priority Channel Systems are Well-Structured

Our main result regarding the verification of PCSs is that they are *well-structured* systems. Recall that $C \leq_\# D \overset{\text{def}}{\Leftrightarrow} C$ is some $(p, y_1, \ldots, y_m)$ and D is $(p, x_1, \ldots, x_m)$ with $x_i \leq_\# y_i$ for $i = 1, \ldots, m$, or equivalently, C can be obtained from D by internal superseding steps.

Theorem 2 (PCSs are WSTSs). *For any PCS S, the transition system $\mathcal{S}_\#$ with configurations ordered by $\leq_\#$ is a well-structured transition system (with stuttering compatibility).*

Proof. There are two conditions to check:

1. **wqo:** $(\mathit{Conf}_S, \leq_\#)$ is a well-quasi-ordering as will be shown next (see Thm. 7 in Sec. 3).
2. **monotonicity:** Checking stuttering compatibility (see [14, def. 4.4]) is trivial with the $\leq_\#$ ordering. Indeed, assume that $C \leq_\# D$ and that $C \to_\# C'$ is a step from the "smaller" configuration. Then in particular $D \overset{*}{\to}_\# C$ by definition of $\to_\#$, so that clearly $D \overset{+}{\to}_\# C'$ and D 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"[2] (given a PCS, an initial configuration C_0, and a set of configurations $G \subseteq \mathit{Conf}_S$, does $C_0 \overset{*}{\to}_\# D$ for some $D \in G$?), and "Inevitability" (do all maximal runs from C_0 eventually visit G?) which includes "Termination" as a special case.

[2] Also called "Safety" when we want to check that G is *not* reachable.

Theorem 3. *Reachability and Inevitability are decidable for PCSs.*

Proof (Sketch). The generic WSTS algorithms [14] apply after we check the minimal effectiveness requirements: the ordering $\leq_\#$ between configurations is decidable (in NLogSpace, see Sec. 3.2) and the operational semantics is finitely branching and effective (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 $Pre^*(G)$ for G given, e.g., as a regular subset of $Conf_S$.

For Inevitability, the algorithms in [2, 14] 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 [26, Thm. 4.4]. □

3 Priority Embedding

This section focuses on the superseding ordering $\leq_\#$ on words and establishes the fundamental properties we use for reasoning about PCSs. Recall that $\leq_\# \stackrel{\text{def}}{=} \xleftarrow{*}_\#$, the reflexive transitive closure of the inverse of $\rightarrow_\#$; we prove that $(\Sigma_p^*, \leq_\#)$ is a *well-quasi-ordering* (a wqo). Recall that a quasi-ordering $(X, \preccurlyeq)$ is a wqo if any infinite sequence $x_0, x_1, x_2, \ldots$ over X contains an infinite increasing subsequence $x_{i_0} \preccurlyeq x_{i_1} \preccurlyeq x_{i_2} \preccurlyeq \cdots$

3.1 Embedding with Priorities

For two words $x, y \in \Sigma_d^*$, we let $x \sqsubseteq_\mathrm{p} y \stackrel{\text{def}}{\Leftrightarrow} x = a_1 \cdots a_\ell$ and y can be factored as $y = z_1 a_1 z_2 a_2 \cdots z_\ell a_\ell$ with $z_i \in \Sigma_{a_i}^*$ for $i = 1, \ldots, \ell$. For example, $201 \sqsubseteq_\mathrm{p} 22011$ but $120 \not\sqsubseteq_\mathrm{p} 10210$ (factoring 10210 as $z_1 1 z_2 2 z_3 0$ needs $z_3 = 1 \notin \Sigma_0^*$). If $x \sqsubseteq_\mathrm{p} y$ 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. In particular, $x \sqsubseteq_\mathrm{p} y$ implies $y \xrightarrow{*}_\# x$, i.e., $x \leq_\# y$. This immediately yields:

$$\varepsilon \sqsubseteq_\mathrm{p} y \text{ iff } y = \varepsilon\,, \tag{1}$$

$$x_1 \sqsubseteq_\mathrm{p} y_1 \text{ and } x_2 \sqsubseteq_\mathrm{p} y_2 \text{ imply } x_1 x_2 \sqsubseteq_\mathrm{p} y_1 y_2\,, \tag{2}$$

$$x_1 x_2 \sqsubseteq_\mathrm{p} y \text{ imply } \exists y_1 \sqsupseteq_\mathrm{p} x_1 : \exists y_2 \sqsupseteq_\mathrm{p} x_2 : y = y_1 y_2\,. \tag{3}$$

Lemma 4. $(\Sigma_d^*, \sqsubseteq_p)$ *is a quasi-ordering (i.e., is reflexive and transitive).*

Proof. Reflexivity is obvious from the definition. For transitivity, consider $x' \sqsubseteq_\mathrm{p} x \sqsubseteq_\mathrm{p} y$ with $x = a_1 \cdots a_\ell$ and $y = z_1 a_1 \cdots z_\ell a_\ell$. In view of Eqs. (1–3) it is enough to show $x' \sqsubseteq_\mathrm{p} y$ in the case where $|x'| = 1$. Consider then $x' = a$. Now $x' \sqsubseteq_\mathrm{p} x$ implies $a = a_\ell$ and $a \geq a_i$, hence $\Sigma_{a_i}^* \subseteq \Sigma_a^*$, for all $i = 1, \ldots, \ell$. Letting $z \stackrel{\text{def}}{=} z_1 a_1 \cdots z_{\ell-1} a_{\ell-1} z_\ell$ yields $y = za$ for $z \in \Sigma_a^*$. Hence $x' \sqsubseteq_\mathrm{p} z$. □

We can now relate superseding and priority orderings with:

Proposition 5. *For all $x, y \in \Sigma_d^*$, $x \sqsubseteq_p y$ iff $x \leq_\# y$.*

Proof. Obviously, $y \xrightarrow{\#k}_\# x$ allows $x \sqsubseteq_\mathrm{p} y$ with z_k being the superseded message (and $z_i = \varepsilon$ for $i \neq k$), so that $\leq_\#$ is included in $\sqsubseteq_\mathrm{p}$ by Lem. 4. In the other direction $x \sqsubseteq_\mathrm{p} y$ entails $x \leq_\# y$ as noted earlier. □

3.2 Canonical Factorizations and Well-quasi-ordering

For our next development, we define the *height*, written $h(x)$, of a sequence $x \in \Sigma_d^*$ as being the highest priority occurring in x (by convention, we let $h(\varepsilon) \stackrel{\text{def}}{=} -1$). Thus, $x \in \Sigma_h^*$ iff $h \geq h(x)$. (We further let $\Sigma_{-1} \stackrel{\text{def}}{=} \emptyset$.) Any $x \in \Sigma_d^*$ has a unique *canonical* factorization $x = x_0 h x_1 h \cdots x_{k-1} h x_k$ where k is the number of occurrences of $h = h(x)$ in x and where the $k+1$ *residuals* x_0, $x_1, \ldots, x_k$ are in Σ_{h-1}^*. The point of this decomposition is the following sufficient condition for $x \sqsubseteq_\mathrm{p} y$.

Lemma 6. *Let $x = x_0 h \cdots h x_k$ and $y = y_0 h \cdots h y_m$ be canonical factorizations with $h = h(x) = h(y)$. If there is a sequence $0 = j_0 < j_1 < j_2 < \cdots < j_{k-1} < j_k = m$ of indexes s.t. $x_i \sqsubseteq_p y_{j_i}$ for all $i = 0, \ldots, k$ then $x \sqsubseteq_p y$.*

Proof. We show $x \leq_\# y$. Note that $h y_i h \xrightarrow{*}_\# h$ for all $i = 1, \ldots, m$, so $y \xrightarrow{*}_\# y' \stackrel{\text{def}}{=} y_{j_0} h y_{j_1} h y_{j_2} \cdots h y_{j_k}$ (recall that $0 = j_0$ and $m = j_k$). From $x_i \sqsubseteq_\mathrm{p} y_{j_i}$ we deduce $y_{j_i} \xrightarrow{*}_\# x_i$ for all $i = 0, \ldots, k$, hence $y' \xrightarrow{*}_\# x_0 h \cdots h x_k = x$. □

The condition in the statement of Lemma 6 is usually written $\langle x_0, \ldots, x_k \rangle \preceq_* \langle y_0, \ldots, y_m \rangle$, using the *sequence extension* of $\sqsubseteq_\mathrm{p}$ on sequences of residuals.

Theorem 7. *$(\Sigma_d^*, \sqsubseteq_p)$ is a well-quasi-ordering (a wqo).*

Proof. By induction on d. The base case $d = -1$ is trivial since Σ_{-1}^* is $\emptyset^* = \{\varepsilon\}$, a singleton. For the induction step, consider an infinite sequence $x_0, x_1, \ldots$ over Σ_d^*. We can extract an infinite subsequence, where all x_i's have the same height h (since $h(x_i)$ is in a finite set) and, since the residuals are in Σ_{d-1}^*, a wqo by ind. hyp., further extract an infinite subsequence where the first and the last residuals are increasing, i.e., $x_{i_0,0} \sqsubseteq_\mathrm{p} x_{i_1,0} \sqsubseteq_\mathrm{p} x_{i_2,0} \sqsubseteq_\mathrm{p} \cdots$ and $x_{i_0,k_0} \sqsubseteq_\mathrm{p} x_{i_1,k_1} \sqsubseteq_\mathrm{p} x_{i_2,k_2} \sqsubseteq_\mathrm{p} \cdots$. Now recall that, by Higman's Lemma, the sequence extension $((\Sigma_{d-1}^*)^*, \preceq_*)$ is a wqo since, by ind. hyp., $(\Sigma_{d-1}^*, \sqsubseteq_\mathrm{p})$ is a wqo. We may thus further extract an infinite subsequence that is increasing for $\preceq_*$ on the residuals, i.e., with $\langle x_{i_0,0}, x_{i_0,1}, \ldots, x_{i_0,k_0} \rangle \preceq_* \langle x_{i_1,0}, x_{i_1,1}, \ldots, x_{i_1,k_1} \rangle \preceq_* \langle x_{i_2,0}, x_{i_2,1}, \ldots, x_{i_2,k_2} \rangle \preceq_* \cdots$ With Lemma 6 we deduce $x_{i_0} \sqsubseteq_\mathrm{p} x_{i_1} \sqsubseteq_\mathrm{p} x_{i_2} \sqsubseteq_\mathrm{p} \cdots$. Hence $(\Sigma_d^*, \sqsubseteq_\mathrm{p})$ is a wqo. □

Remark 8. Thm. 7 and Prop. 5 prove that $\leq_\#$ is a wqo on configurations of PCSs, as we assumed in Sec. 2.3. There we also assumed that $\leq_\#$ is decidable. We can now see that it is in NLogSpace, since, in view of Prop. 5, one can check whether $x \leq_\# y$ by reading x and y simultaneously while guessing nondeterministically a factorization $z_1 a_1 \cdots z_\ell a_\ell$ of y, and checking that $z_i \in \Sigma_{a_i}^*$.

Fig. 2. Two trees in T_2.

4 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 serves two purposes. First, it illustrates the "power" of priority embeddings, giving a simple proof that strong tree embeddings form a wqo as a byproduct. Second, the reflection defined will subsequently be used in Sec. 6 to provide an encoding of ordinals that PCSs can manipulate "robustly."

4.1 Encoding Bounded Depth Trees

Given an alphabet Γ, the set of finite, ordered, unranked labeled trees (aka variadic terms) over Γ, noted $T(\Gamma)$, is the smallest set such that, if f is in Γ and $t_1, \dots, t_n$ are $n \geq 0$ trees in $T(\Gamma)$, then the tree $f(t_1 \cdots t_n)$ is in $T(\Gamma)$. A *context* C is defined as usual as a tree with a single occurrence of a leaf labeled by a distinguished variable x. Given a context C and a tree t, we can form a tree $C[t]$ by plugging t instead of that x-labeled leaf.

Let d be a depth in $\mathbb{N}$ and $\bullet$ be a node label. We consider the set $T_d = T_d(\{\bullet\})$ of trees of depth at most d with $\bullet$ as single possible label; for instance, $T_0 = \{\bullet()\}$ contains a single tree, and the two trees shown in Fig. 2 are in T_2:

It is a folklore result that one can encode bounded depth trees into finite sequences using canonical factorizations. Here we present a natural variant that is rather well-suited for our constructions in Sec. 6. We encode trees of bounded depth using the mapping $s_d : T_{d+1} \to \Sigma_d^*$ defined by induction on d as

$$s_d(\bullet(t_1 \cdots t_n)) \stackrel{\text{def}}{=} \begin{cases} \varepsilon & \text{if } n = 0, \\ s_{d-1}(t_1)d \cdots s_{d-1}(t_n)d & \text{otherwise.} \end{cases} \tag{4}$$

For instance, if we fix $d = 1$, the left tree in Fig. 2 is encoded as "111" and the right one as "0011". Note that the encoding depends on the choice of d: for $d = 2$ we would have encoded the trees in Fig. 2 as "222" and "1122", respectively.

Not every string in Σ_d^* is the encoding of a tree according to s_d: for $-1 \leq a \leq d$, we let $P_a \stackrel{\text{def}}{=} (P_{a-1}\{a\})^*$ be the set of *proper encodings of height* a, with further $P_{-1} \stackrel{\text{def}}{=} \{\varepsilon\}$. Then $P \stackrel{\text{def}}{=} \bigcup_{a \leq d} P_a$ is the set of *proper* words in Σ_d^*. A proper word x is either empty or belongs to a unique P_a with $a = h(x)$, and has then a canonical factorization of the form $x = x_1 a \cdots x_m a$ with every x_j in P_{a-1}. Put differently, a non-empty $x = a_1 \cdots a_\ell$ is in P_a 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*: along proper words, priorities only increase smoothly, but can decrease sharply). For example, 02 is not proper (it has a jump) while 012 is proper; 233123401234 is proper too.

Given a depth a, we see that s_a is a bijection between T_{a+1} and P_a, with the inverse defined by

$$\tau(\varepsilon) \stackrel{\text{def}}{=} \bullet() \,, \qquad \tau(x = x_1 h(x) \cdots x_m h(x)) \stackrel{\text{def}}{=} \bullet(\tau(x_1) \cdots \tau(x_m)) \,. \tag{5}$$

4.2 Strong Tree Embeddings

One can provide a formal meaning to the notion of a wqo $(B, \preccurlyeq_B)$ being more powerful than another one $(A, \preccurlyeq_A)$ through *order reflections*, i.e. through the existence of a mapping $r\colon A \to B$ such that $r(x) \preccurlyeq_B r(y)$ implies $x \preccurlyeq_A y$ for all x, y in A. Observe that if B reflects A and $(B, \preccurlyeq_B)$ is a wqo, then $(A, \preccurlyeq_A)$ is necessarily a wqo. We show here that $(\Sigma_d^*, \sqsubseteq_{\mathrm{p}})$ reflects bounded-depth trees endowed with the strong tree-embedding relation.

Let t and t' be two trees in T_d. We say that t *strongly embeds* into t', written $t \sqsubseteq_T t'$, if it can be obtained from t' by deleting whole subtrees, i.e. $\sqsubseteq_T$ is the reflexive transitive closure of the relation $t \sqsubset_T^1 t' \stackrel{\text{def}}{\Leftrightarrow} t = C[\bullet(t_1 \cdots t_{i-1} t_{i+1} \cdots t_n)]$ and $t' = C[\bullet(t_1 \cdots t_{i-1} t_i t_{i+1} \cdots t_n)]$ for some context C and subtrees $t_1, \ldots, t_n$. 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 Fig. 2 are *not* related by any homeomorphic tree embedding, and thus neither by strong tree embedding. See the full version for the proofs of the following results:

Proposition 9. *The map s_d is an order reflection from $(T_{d+1}, \sqsubseteq_T)$ to $(\Sigma_d^*, \sqsubseteq_p)$.*

Corollary 10. *For each d, $(T_d, \sqsubseteq_T)$ is a wqo.*

4.3 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 Sec. 6. 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 [15] in XML processing, and, using elimination trees [see 21], for bounded-depth graphs used e.g. in the verification of ad-hoc networks [12], the π-calculus [22], and programs [5]. These applications consider *labeled* trees, which are dealt with thanks to a generalization of $\sqsubseteq_{\mathrm{p}}$ to pairs (a, w) where a is a priority and w a symbol from some wqo $(\Gamma, \leq)$; see the full version.

This generalization of $\sqsubseteq_{\mathrm{p}}$ also allows to treat another wqo on trees, the *tree minor* ordering, using the techniques of Gupta [16] to encode them in prioritized alphabets. The tree minor ordering is coarser than the homeomorphic embedding (e.g. in Fig. 2, the left tree is a minor of the right tree), but the upside is that trees of unbounded depth can be encoded into strings.

The exact complexity of verification problems in the aforementioned models is currently unknown [15, 12, 22, 5]. Our encoding suggests them to 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.

5 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* [20, 13] of the input size. We show in this section how to bound the complexity of the algorithms presented in Sec. 2.3 and classify the Reachability and Inevitability problems using *fast-growing complexity classes* [25].

5.1 Subrecursive Hierarchies

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

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

where addition is associative, with 0 as the neutral element (the empty sum). Equivalently, we can then see a term other than 0 as a tree over the alphabet $\{+\}$; for instance the two trees in Fig. 2 represent 3 and $\omega^2 + 1$ respectively, when putting the ordinal terms under the form $\alpha = \sum_{i=1}^{k} \omega^{\alpha_i}$. Such a term 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 limits.

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 $\alpha < \beta \stackrel{\text{def}}{\Leftrightarrow} o(\alpha) < o(\beta)$. 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 Sec. 6 to manipulate terms that are *not* in CNF.

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

$$\begin{aligned} (\gamma + \omega^{\beta+1})_n &\stackrel{\text{def}}{=} \gamma + \omega^\beta \cdot n = \gamma + \overbrace{\omega^\beta + \cdots + \omega^\beta}^{n}\,, \\ (\gamma + \omega^{\lambda'})_n &\stackrel{\text{def}}{=} \gamma + \omega^{\lambda'_n}\,. \end{aligned} \tag{6}$$

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 $(\varepsilon_0)_n \stackrel{\text{def}}{=} \Omega_n$, where for $n \in \mathbb{N}$, we use Ω_n as short-hand notation for the ordinal $\omega^{\omega^{\cdots^\omega}}\}n$ stacked ω's, i.e., for $\Omega_0 \stackrel{\text{def}}{=} 1$ and $\Omega_{n+1} \stackrel{\text{def}}{=} \omega^{\Omega_n}$.

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

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

Observe that h^1 is simply h, and more generally h^α is the αth iterate of h, using diagonalisation 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 [13].

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 [see 25, App. B]

$$\mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p \in \bigcup_{\beta<\alpha} \mathscr{F}_\beta} \text{DTime}(F_\alpha(p(n))) \,, \qquad \mathscr{F}_\alpha = \bigcup_{c<\omega} \text{FDTime}(F_\alpha^c(n)) \,, \tag{7}$$

where the class of functions $\mathscr{F}_\alpha$ as defined above is the αth level of the *extended Grzegorczyk hierarchy* [20] when $\alpha \geq 2$; in particular, $\bigcup_{\alpha<\varepsilon_0} \mathscr{F}_\alpha$ is exactly the set of ordinal-recursive (aka "provably recursive") functions [13].

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.[3]

5.2 Complexity Upper Bounds

Recall that an alternative characterization of a wqo $(X, \preccurlyeq)$ is that any sequence $x_0, x_1, x_2, \ldots$ over X verifying $x_i \not\preccurlyeq x_j$ for all $i < j$ is necessarily finite. Such sequences are called *bad*, and in order to bound the complexity of the algorithms from Thm. 3, we can bound the lengths of bad sequences over the wqo $(Conf_{\mathcal{S}}, \leq_\#)$ using the *Length Function Theorem* of [24]; see the full version for details:

Theorem 11 (Complexity of PCS Verification). *Reachability and Inevitability of PCSs are in* $\mathbf{F}_{\varepsilon_0}$.

6 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 Thm. 15. For this, we develop (Sec. 6.1) encodings $s(\alpha) \in \Sigma_d^*$ for ordinals

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

$\alpha \in \Omega_d$ 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 (Sec. 6.2) to design PCSs that "*weakly compute*" H^α and $(H^\alpha)^{-1}$ in the sense of Def. 13 below.

6.1 Encoding Ordinals

Our encoding of ordinal terms as strings in Σ_d^* is exactly the encoding of trees presented in Sec. 4. For $0 \le a \le d$, we use the following equation to define the language $P_a \subseteq \Sigma_d^*$ of *proper encodings*, or just *codes*:

$$P_a \stackrel{\text{def}}{=} \varepsilon + P_a P_{a-1} a\,, \qquad\qquad P_{-1} \stackrel{\text{def}}{=} \varepsilon\,. \tag{8}$$

Let $P = P_{-1} + P_0 + \cdots + P_d$. Each P_a (and then P itself) is a regular language, with $P_a = (P_{a-1}a)^*$ as in Sec. 4; for instance, $P_0 = 0^*$.

Decompositions A code x is either the empty word ε, or belongs to a unique P_a. If $x \in P_a$ is not empty, it has a unique factorization $x = yza$ according to (8) with $y \in P_a$ and $z \in P_{a-1}$. The factor $z \in P_{a-1}$ in $x = yza$ can be developed further, as long as $z \neq \varepsilon$: a non-empty code $x \in P_d$ has a unique factorization as $x = y_d\, y_{d-1} \dots y_a\, a^\frown d$ with $y_i \in P_i$ for $i = a, \dots, d$, and where for $0 \le a \le b$, 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, $x = 23312340121234 \in P_4$ is a code and decomposes as

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

Ordinal Encoding Following the tree encoding of Sec. 4, with a code $x \in P$, we associate an ordinal term $\eta(x)$ given by

$$\eta(\varepsilon) \stackrel{\text{def}}{=} 0\,, \qquad\qquad \eta(yza) \stackrel{\text{def}}{=} \eta(y) + \omega^{\eta(z)}\,, \tag{9}$$

where $x = yza$ is the factorization according to (8) of $x \in P_a \setminus \{\varepsilon\}$. 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 $x \in P_a$.

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

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

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

$$\begin{gathered} s_5(1) = 5\,, \qquad s_5(3) = 555\,, \qquad s_5(\omega) = 45\,, \\ s_5(\omega^3) = 4445\,, \qquad s_5(\omega^\omega) = 345\,, \qquad s_5(\omega^{\omega^\omega}) = 2345\,, \\ s_5(\omega^3 + \omega^2) = 4445445\,, \qquad s_5(\omega \cdot 3) = 454545\,. \end{gathered}$$

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

o:	3 3 4 5 4 5 $	the *ordinal* term $\omega^{\omega^2} + \omega^\omega$
c:	0 0 0 0 $	the *counter* value 4
t:	$	the *temporary* storage

Fig. 3. Channels for Hardy computations.

Successors and Limits Let $x = y_d\, y_{d-1} \dots y_a\, a^\frown d$ be the decomposition of $x \in P_d \setminus \varepsilon$. By (9), x encodes a successor ordinal $\eta(x) = \beta + 1$ iff $a = d$, 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 (6) and (9), one obtains the encoding $(x)_n$ of λ_n with

$$(x)_n = y_d\, y_{d-1} \dots y_{a+1} \big(y_a(a+1)\big)^n (a+2)^\frown d\,. \tag{10}$$

E.g., 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. $s(\omega^{\omega^3 \cdot 2}) = 333433345$.

Robustness Translated to ordinals, Prop. 9 means that, whenever $x \leq_\# x'$ for $x, x' \in P_a$, then the corresponding ordinal $\eta(x)$ will be "structurally" smaller than $\eta(x')$. This in turn yields that the corresponding Hardy function $H^{\eta(x)}$ grows at most as fast as $H^{\eta(x')}$; see the full version for details:

Proposition 12 (Robustness). *Let $a \geq 0$ and $x, x' \in P_a$. If $x \leq_\# x'$ then $H^{\eta(x)}(n) \leq H^{\eta(x')}(n')$ for all $n \leq n'$ in $\mathbb{N}$.*

6.2 Robust Hardy Computations in PCSs

Our PCSs for robust Hardy computations use three channels (see Fig. 3), 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 13. *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:*

$$\begin{aligned} &\textit{if } (p_{beg}, x\$, y\$, z\$) \xrightarrow{*}_\# (p_{end}, u, v, w) \\ &\quad \textit{then } x \in P_d, y \in 0^+, z = \varepsilon \textit{ and } u, v, w \in \Sigma_d^*\$\,, \end{aligned} \tag{safety}$$

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

Furthermore S is complete *if for any $\alpha < \Omega_{d+1}$ and $n > 0$, $(p_{beg}, s(\alpha)\$, 0^n\$, \$) \xrightarrow{*}_\# (p_{end}, \$, 0^m\$, \$)$ for $m = H^\alpha(n)$, and it is* inv-complete *if $(p_{beg}, \$, 0^m\$, \$) \xrightarrow{*}_\# (p_{end}, s(\alpha)\$, 0^n\$, \$)$.*

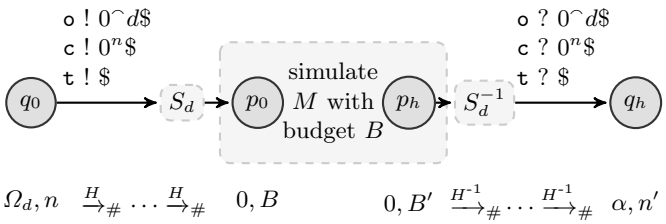

Fig. 4. Schematics for Thm. 15.

In the full version we prove the following:

Lemma 14 (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.*

6.3 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 [17, 27] (with a different encoding and a different machine model) and will only be sketched.

Theorem 15 (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 4 depicts the PCS S we construct for the reduction. Let $n \stackrel{\text{def}}{=} |M|$ and $d \stackrel{\text{def}}{=} n+1$. 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_d^{-1} to perform backward Hardy steps. This leads to `o` containing the encoding of some ordinal α and `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. □

7 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 [15, 12, 22, 5]. This is part of our current research agenda on complexity for well-structured systems [25].

In spite of their enormous worst-case complexity, we expect PCSs to be amenable to regular model checking techniques *à la* [4, 6]. 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 DFA with $|x|$ states.

References

1. Abdulla, P.A., Atig, M.F., Cederberg, J.: Timed lossy channel systems. FST&TCS 2012. LIPIcs, vol. 18, pp. 374–386. Leibniz-Zentrum für Informatik (2012)
2. Abdulla, P.A., Čerāns, K., Jonsson, B., Tsay, Y.K.: Algorithmic analysis of programs with well quasi-ordered domains. Inform. and Comput. 160(1–2), 109–127 (2000)
3. Abdulla, P.A., Deneux, J., Ouaknine, J., Worrell, J.: Decidability and complexity results for timed automata via channel machines. ICALP 2005. LNCS, vol. 3580, pp. 1089–1101. Springer (2005)
4. Abdulla, P.A., Jonsson, B.: Verifying programs with unreliable channels. Inform. and Comput. 127(2), 91–101 (1996)
5. Bansal, K., Koskinen, E., Wies, T., Zufferey, D.: Structural counter abstraction. TACAS 2013. LNCS, vol. 7795, pp. 62–77. Springer (2013)
6. Boigelot, B., Godefroid, P.: Symbolic verification of communication protocols with infinite state spaces using QDDs. Form. Methods in Syst. Des. 14(3), 237–255 (1999)
7. Bouajjani, A., Habermehl, P.: Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. Theor. Comput. Sci. 221(1–2), 211–250 (1999)

8. Bouyer, P., Markey, N., Ouaknine, J., Schnoebelen, Ph., Worrell, J.: On termination and invariance for faulty channel machines. Form. Asp. Comput. 24(4–6), 595–607 (2012)
9. Cécé, G., Finkel, A.: Verification of programs with half-duplex communication. Inform. and Comput. 202(2), 166–190 (2005)
10. Cécé, G., Finkel, A., Purushothaman Iyer, S.: Unreliable channels are easier to verify than perfect channels. Inform. and Comput. 124(1), 20–31 (1996)
11. Chambart, P., Schnoebelen, Ph.: The ordinal recursive complexity of lossy channel systems. LICS 2008. pp. 205–216. IEEE Press (2008)
12. Delzanno, G., Sangnier, A., Zavattaro, G.: Parameterized verification of ad hoc networks. Concur 2010. LNCS, vol. 6269, pp. 313–327. Springer (2010)
13. Fairtlough, M., Wainer, S.S.: Hierarchies of provably recursive functions. Handbook of Proof Theory, chap. III, pp. 149–207. Elsevier (1998)
14. Finkel, A., Schnoebelen, Ph.: Well-structured transition systems everywhere! Theor. Comput. Sci. 256(1–2), 63–92 (2001)
15. Genest, B., Muscholl, A., Serre, O., Zeitoun, M.: Tree pattern rewriting systems. ATVA 2008. LNCS, vol. 5311, pp. 332–346. Springer (2008)
16. Gupta, A.: A constructive proof that trees are well-quasi-ordered under minors. LFCS 1992. LNCS, vol. 620, pp. 174–185. Springer (1992)
17. Haddad, S., Schmitz, S., Schnoebelen, Ph.: The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. LICS 2012. pp. 355–364. IEEE Press (2012)
18. Kurucz, A.: Combining modal logics. Handbook of Modal Logics, chap. 15, pp. 869–926. Elsevier (2006)
19. Lasota, S., Walukiewicz, I.: Alternating timed automata. ACM Trans. Comput. Logic 9(2) (2008)
20. Löb, M., Wainer, S.: Hierarchies of number theoretic functions, I. Arch. Math. Logic 13, 39–51 (1970)
21. Ossona de Mendez, P., Nešetřil, J.: Sparsity, chap. 6: Bounded height trees and tree-depth, pp. 115–144. Springer (2012)
22. Meyer, R.: On boundedness in depth in the π-calculus. IFIP TCS 2008. IFIP, vol. 273, pp. 477–489. Springer (2008)
23. Ouaknine, J., Worrell, J.: On the decidability and complexity of Metric Temporal Logic over finite words. Logic. Meth. in Comput. Sci. 3(1), 1–27 (2007)
24. Schmitz, S., Schnoebelen, Ph.: Multiply-recursive upper bounds with Higman's lemma. ICALP 2011. LNCS, vol. 6756, pp. 441–452. Springer (2011)
25. Schmitz, S., Schnoebelen, Ph.: Algorithmic aspects of WQO theory. Lecture notes (2012), http://cel.archives-ouvertes.fr/cel-00727025
26. Schnoebelen, Ph.: Lossy counter machines decidability cheat sheet. RP 2010. LNCS, vol. 6227, pp. 51–75. Springer (2010)
27. Schnoebelen, Ph.: Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. MFCS 2010. LNCS, vol. 6281, pp. 616–628. Springer (2010)
28. Schütte, K., Simpson, S.G.: Ein in der reinen Zahlentheorie unbeweisbarer Satz über endliche Folgen von natürlichen Zahlen. Arch. Math. Logic 25(1), 75–89 (1985)