The Power of Well-Structured Systems*

S. Schmitz and Ph. Schnoebelen

LSV, ENS Cachan & CNRS, Cachan, France
{schmitz,phs}@lsv.ens-cachan.fr

Abstract. Well-structured systems, aka WSTS, are computational models where the set of possible configurations is equipped with a well-quasi-ordering which is compatible with the transition relation between configurations. This structure supports generic decidability results that are important in verification and several other fields.
This paper recalls the basic theory underlying well-structured systems and shows how two classic decision algorithms can be formulated as an exhaustive search for some "bad" sequences. This lets us describe new powerful techniques for the complexity analysis of WSTS algorithms. Recently, these techniques have been successful in precisely characterizing the power, in a complexity-theoretical sense, of several important WSTS models like unreliable channel systems, monotonic counter machines, or networks of timed systems.

Introduction

Well-Structured (Transition) Systems, aka WSTS, are a family of computational models where the usually infinite set of states is equipped with a well-quasi-ordering that is "compatible" with the computation steps. The existence of this well-quasi-ordering allows for the decidability of some important behavioural properties like Termination or Coverability.

Historically, the idea can be traced back to Finkel [21] who gave a first definition for WSTS abstracting from Petri nets and fifo nets, and who showed the decidability of Termination and Finiteness (aka Boundedness). Then Finkel [22] applied the WSTS idea to Termination of lossy channel systems, while Abdulla and Jonsson [4] introduced the backward-chaining algorithm for Coverability. One will find a good survey of these early results, and a score of WSTS examples, in [2, 24, 1, 8].

The basic theory saw several important developments in recent years, like the study of comparative expressiveness for WSTS [3], or the completion technique for forward-chaining in WSTS [23]. Simultaneously, many new WSTS models have been introduced (in distributed computing, software verification, or other fields), using well-quasi-orderings based on trees, sequences of vectors, or graphs (see references in [41]), rather than the more traditional vectors of natural numbers or words with the subword relation.

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

Another recent development is the complexity analysis of WSTS models and algorithms. New techniques, borrowing notions from proof theory and ordinal analysis, can now precisely characterize the complexity of some of the most widely used WSTS [13, 42, 27]. The difficulty here is that one needs complexity functions, complexity classes, and hard problems, with which computer scientists are not familiar.

The aim of this paper is to provide a gentle introduction to the main ideas behind the complexity analysis of WSTS algorithms. These complexity questions are gaining added relevance: more and more recent papers rely on reductions to (or from) a known WSTS problem to show the decidability (or the hardness) of problems in unrelated fields, from modal and temporal logic [31, 38] to XPath-like queries [29, 7].

Outline of the paper. Section 1 recalls the definition of WSTS, Sec. 2 illustrates it with a simple example, while Sec. 3 presents the two main verification algorithms for WSTS. Section 4 bounds the running time of these algorithms by studying the length of bad sequences using fast-growing functions. Section 5 explains how lower bounds matching these enormous upper bounds have been established in a few recent works, including the $\mathbf{F}_{\varepsilon_0}$-completeness result in this volume [26].

1 What are WSTS?

A simple, informal way to define WSTS is to say that they are transition systems *whose behaviour is monotonic w.r.t. a well-ordering*. Here, monotonicity of behaviour means that the states of the transition system are ordered in a way such that larger states have more available steps than smaller states. Requiring that the ordering of states is a well-ordering (more generally, a well-quasi-ordering) ensures that monotonicity translates into decidability for some behavioural properties like Termination or Coverability.

Let us start with monotonicity. In its simplest form, a *transition system* (a TS) is a structure $\mathcal{S} = (S, \to)$ where S is the set of *states* (typical elements $s_1, s_2, \ldots$) and $\to \subseteq S \times S$ is the *transition relation*. As usual, we write "$s_1 \to s_2$" rather than "$(s_1, s_2) \in \to$" to denote steps. A TS is *ordered* when it is further equipped with a quasi-ordering of its states, i.e., a reflexive and transitive relation $\leq \subseteq S \times S$.

Definition 1.1 (Monotonicity). *An ordered transition system $\mathcal{S} = (S, \to, \leq)$ is* monotonic $\overset{\text{def}}{\Leftrightarrow}$ *for all $s_1, s_2, t_1 \in S$*

$$\big(s_1 \to s_2 \text{ and } s_1 \leq t_1\big) \text{ implies } \exists t_2 \in S : \big(t_1 \to t_2 \text{ and } s_2 \leq t_2\big)\,.$$

This property is also called "*compatibility*" (of the ordering with the transitions) [24]. Formally, is just means that $\leq$ is a *simulation* relation for $\mathcal{S}$, in precisely the classical sense of Milner [37]. The point of Def. 1.1 is to ensure that a "larger state" can do "more" than a smaller state. For example, it entails the following Fact that plays a crucial role in Sec. 3.

Given two finite runs $\boldsymbol{s} = (s_0 \to s_1 \to \cdots \to s_n)$ and $\boldsymbol{t} = (t_0 \to t_1 \to \cdots \to t_m)$, we say that $\boldsymbol{s}$ simulates $\boldsymbol{t}$ *from below* (and $\boldsymbol{t}$ simulates $\boldsymbol{s}$ *from above*) if $n = m$ and $s_i \leq t_i$ for all $i = 0, \ldots, n$.

Fact 1.2. *Any run $\boldsymbol{s} = (s_0 \to s_1 \to \cdots \to s_n)$ in a WSTS $\mathcal{S}$ can be simulated from above, starting from any $t \geq s_0$.*

Remark 1.3. Definition 1.1 comes in many variants. For example, Finkel and Schnoebelen [24] consider *strict compatibility* (when $<$, the strict ordering underlying $\leq$, is a simulation), *transitive compatibility* (when $\leq$ is a weak simulation), and the definition can further extend to *labeled* transition systems. These are all inessential variations of the main idea. □

Now to the wqo ingredient.

Definition 1.4 (Wqo). *A quasi-order $(S, \leq)$ is* well *("is a* wqo*") if every infinite sequence $s_0, s_1, s_2, \ldots$ over S contains an increasing pair $s_i \leq s_j$ for some $i < j$. Equivalently, $(S, \leq)$ is a wqo if, and only if, every infinite sequence $s_0, s_1, s_2, \ldots$ over S contains an infinite increasing subsequence $s_{i_0} \leq s_{i_1} \leq s_{i_2} \leq \cdots$, where $i_0 < i_1 < i_2 < \cdots$*

We call *good* a sequence that contains an increasing pair, otherwise it is *bad*. Thus in a wqo all infinite sequences are good, all bad sequences are finite.

Definition 1.4 offers two equivalent definitions. Many other characterisations exist [30], and it is an enlightening exercise to prove their equivalence [see 41, Chap. 1]. Let us illustrate the usefulness of the alternative definition: for a dimension $k \in \mathbb{N}$, write $\mathbb{N}^k$ for the set of k-tuples, or vectors, of natural numbers. For two vectors $\boldsymbol{a} = (a_1, \ldots, a_k)$ and $\boldsymbol{b} = (b_1, \ldots, b_k)$ in $\mathbb{N}^k$, we let $\boldsymbol{a} \leq_\times \boldsymbol{b} \stackrel{\text{def}}{\Leftrightarrow} a_1 \leq b_1 \wedge \cdots \wedge a_k \leq b_k$.

Example 1.5 (Dickson's Lemma). $(\mathbb{N}^k, \leq_\times)$ is a wqo.

Proof (of Dickson's Lemma). Consider an infinite sequence $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3, \ldots$ over $\mathbb{N}^k$ and write $\boldsymbol{a}_i = (a_{i,1}, \ldots, a_{i,k})$. One can extract an infinite subsequence $\boldsymbol{a}_{i_1}, \boldsymbol{a}_{i_2}, \boldsymbol{a}_{i_3}, \ldots$ that is increasing over the first components, i.e., with $a_{i_1,1} \leq a_{i_2,1} \leq a_{i_3,1} \leq \cdots$, since $(\mathbb{N}, \leq)$ is a wqo (easy to prove, here the first definition suffices). From this infinite subsequence, one can further extract an infinite subsequence that is also increasing on the second components (again, using that $\mathbb{N}$ is wqo). After k extractions, one has an infinite subsequence that is increasing on all components, i.e., that is increasing for $\leq_\times$ as required. □

We can now give the central definition of this paper:

Definition 1.6 (WSTS). *An ordered transition system $\mathcal{S} = (S, \to, \leq)$ is a WSTS $\stackrel{\text{def}}{\Leftrightarrow}$ $\mathcal{S}$ is monotonic and $(S, \leq)$ is a wqo.*

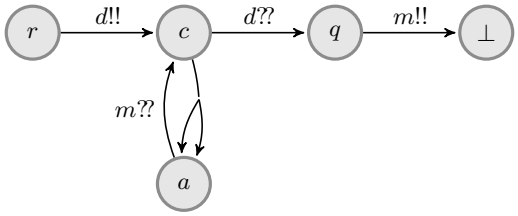

Fig. 1: A broadcast protocol.

2 A Running Example: Broadcast Protocols

As a concrete illustration of the principles behind WSTS, let us consider distributed systems known as *broadcast protocols* [16, 17]. Such systems gather an unbounded number of identical finite-state processes running concurrently, able to spawn new processes, and communicating either via *rendez-vous*—where two processes exchange a message—or via *broadcast*—where one process sends the same message to every other process. While this may seem at first sight rather restricted for modeling distributed algorithms, broadcast protocols have been employed for instance to verify the correction of cache coherence protocols without fixing a number of participating processes.[1]

Formally, a broadcast protocol is defined as a triple $\mathtt{B} = (Q, M, R)$ where Q is a finite set of *locations*, M a finite set of *messages*, and R is a set of *rules*, that is, tuples (q, op, q') in $Q \times Op \times Q$, each describing an operation op available in the location q and leading to a new location q', where op can be a sending (denoted $m!$) or a receiving ($m?$) operation of a rendez-vous message m from M, or a sending ($m!!$) or receiving ($m??$) operation of a broadcast message m from M, or a spawning ($\mathrm{sp}(p)$) of a new process that will start executing from location p. As usual, we write $q \xrightarrow{op}_{\mathtt{B}} q'$ if (q, op, q') is in R.

Figure 1 displays a toy example where $Q = \{r, c, a, q, \bot\}$ and $M = \{d, m\}$: processes in location c can spawn new "active" processes in location a, while also moving to location a (a rule depicted as a double arrow in Fig. 1). These active processes are flushed upon receiving a broadcast of either m (emitted by a process in location q) or d (emitted by a process in location r); location $\bot$ is a sink location modeling process destruction.

The operational semantics of a broadcast protocol is expressed as a transition system $\mathcal{S}_{\mathtt{B}} = (S, \to)$, where states, here called configurations, are (finite) multisets of locations in Q, hence $S = \mathbb{N}^Q$. Informally, the intended semantics for a configuration s in $\mathbb{N}^Q$ is to record for each location q in Q the number of processes $s(q)$ currently in this location. We use a "sets with duplicates" notation, like $s = \{q_1, \ldots, q_n\}$ where some q_i's might be identical, and feel free to write, e.g., $\{q^3, q'^4\}$ instead of $\{q, q, q, q', q', q', q'\}$. A natural ordering for $\mathbb{N}^Q$ is the *inclusion* ordering defined by $s \subseteq s' \stackrel{\text{def}}{\Leftrightarrow} \forall q \in Q,\ s(q) \leq s'(q)$. For instance,

[1] See also the up-to-date survey of parameterized verification problems in [18].

$\{q, q, q'\} \subseteq \{q, q, q, q'\}$ but $\{q, q', q'\} \not\subseteq \{q, q, q'\}$ if $q \neq q'$. We further write $s = s_1 + s_2$ for the union (not the lub) of two multisets, in which case $s - s_1$ denotes s_2.

It remains to define how the operations of $\mathtt{B}$ update such a configuration through transitions $s \to s'$ of $\mathcal{S}_\mathtt{B}$:

rendez-vous step: if $q_1 \xrightarrow{m!}_\mathtt{B} q_1'$ and $q_2 \xrightarrow{m?}_\mathtt{B} q_2'$ for some $m \in M$, then $s + \{q_1, q_2\} \to s + \{q_1', q_2'\}$ for all s in $\mathbb{N}^Q$,

spawn step: if $q \xrightarrow{\mathrm{sp}(p)}_\mathtt{B} q'$, then $s + \{q\} \to s + \{q', p\}$ for all s in $\mathbb{N}^Q$,

broadcast step: if $q_0 \xrightarrow{m!!}_\mathtt{B} q_0'$ and $q_i \xrightarrow{m??}_\mathtt{B} q_i'$ for all $1 \leq i \leq k$ (and some m), then $s + \{q_0, q_1, \ldots, q_k\} \to s + \{q_0', q_1', \ldots, q_k'\}$ for all s in $\mathbb{N}^Q$ that do not contain a potential receiver for the broadcast, i.e., such that $s(q) = 0$ for all rules of the form $q \xrightarrow{m??}_\mathtt{B} q'$.

With the protocol of Fig. 1, the following steps are possible (with the spawned location or exchanged messages indicated on the arrows):

$$\{c^2, q, r\} \xrightarrow{a} \{a^2, c, q, r\} \xrightarrow{a} \{a^4, q, r\} \xrightarrow{m} \{c^4, r, \bot\} \xrightarrow{d} \{c, q^4, \bot\}.$$

We have just associated an ordered transition system $\mathcal{S}_\mathtt{B} = (\mathbb{N}^Q, \to, \subseteq)$ with every broadcast protocol $\mathtt{B}$ and are now ready to prove the following fact.

Fact 2.1. *Broadcast protocols are WSTS.*

Proof. First, $(\mathbb{N}^Q, \subseteq)$ is a wqo: since Q is finite, this is just another instance of Dickson's Lemma.

There remains to check that $\mathcal{S}_\mathtt{B}$ is monotonic. Formally, this is done by considering an arbitrary step $s_1 \to s_2$ (there are three cases) and an arbitrary pair $s_1 \subseteq t_1$. It is enough to assume that $t_1 = s_1 + \{q\}$, i.e., t_1 is just one location bigger that s_1, and to rely on transitivity. If $s_1 \to s_2$ is a rendez-vous step with $s_2 = s_1 - \{q_1, q_2\} + \{q_1', q_2'\}$, then $t_1 = s_1 + \{q\}$ also has a rendez-vous step $t_1 \to t_2 = t_1 - \{q_1, q_2\} + \{q_1', q_2'\}$ and one sees that $s_2 \subseteq t_2$ as required. If now if $s_1 \to s_2$ is a spawn step, a similar reasoning proves that $s_1 + \{q\} \to s_2 + \{q\}$. Finally, when $s_1 = s + \{q_1, \ldots\} \to s_2 = s + \{q_1', \ldots\}$ is a broadcast step, one proves that $s_1 + \{q\} \to s_2 + \{q'\}$ when there is a rule $q \xrightarrow{m??}_\mathtt{B} q'$, or when q is not a potential receiver and $q' = q$. □

Remark 2.2. One can show that the protocol depicted in Fig. 1 always terminates, this from any initial configuration s_{init}. Recall that, for a TS $\mathcal{S}$, Termination is the question, given a state $s_{init} \in S$, whether all runs starting from s_{init} are finite, i.e., whether there are no infinite runs from s_{init}.

A first remark is that the processes in location $\bot$ can safely be ignored, since they have terminated. Then, consider any sequence of steps $s_0 \to s_1 \to \cdots \to s_i \to \cdots$ with each configuration of form $s_i = \{a^{n_{a,i}}, c^{n_{c,i}}, q^{n_{q,i}}, r^{n_{r,i}}\}$, and let us compare two configurations indexed by $i < j$:

– either only spawn steps occur between s_i and s_j, thus $n_{c,j} < n_{c,i}$,

- or at least one m has been broadcast but no d has been broadcast, thus $n_{q,j} < n_{q,i}$,
- or at least one d has been broadcast, and then $n_{r,j} < n_{r,i}$.

Thus in all cases, $s_i \not\subseteq s_j$, i.e. the sequence of successive configurations is bad. Since $(\mathbb{N}^Q, \subseteq)$ is a wqo, there is no infinite bad sequence, hence no infinite run. □

3 Verification of WSTS

In this section we present the two main generic decision algorithms for WSTS. We strive for a presentation that abstracts away from implementation details, and that can directly be linked to the complexity analysis we describe in the following sections. The general idea is that these algorithms can be seen as an exhaustive search for some kind of bad sequences.

3.1 Termination

There is a generic algorithm deciding Termination on WSTS. The algorithm has been adapted and extended to show decidability of Inevitability (of which Termination is a special case), Finiteness (aka Boundedness), or Regular Simulation, see [2, 24].

Lemma 3.1 (Finite Witnesses for Infinite Runs). *A WSTS $\mathcal{S}$ has an infinite run from s_{init} if, and only if, it has a finite run from s_{init} that is a good sequence.*

Proof. Obviously, any infinite run $s_{init} = s_0 \to s_1 \to s_2 \to \cdots$ is a good sequence by property of $\leq$ being a wqo (see Def. 1.4). Once a pair $s_i \leq s_j$ is identified, the finite prefix that stops at s_j is both a finite run and a good sequence.

Reciprocally, given a finite run $s_0 \to s_1 \to \cdots \to s_i \to \cdots \to s_j$ with $i < j$ and $s_i \leq s_j$, Fact 1.2 entails the existence of a run $s_j \to s_{j+1} \to \cdots \to s_{2j-i}$ that simulates $s_i \to \cdots \to s_j$ from above. Hence the finite run can be extended to some $s_0 \to \cdots \to s_i \to \cdots \to s_j \to \cdots \to s_{2j-i}$ with $s_i \leq s_{2j-i}$. Repeating this extending process ad infinitum, one obtains an infinite run. □

Very little is needed to turn Lemma 3.1 into a decidability proof for Termination. We shall make some minimal *effectiveness assumptions*: (EA1) the set of states S is recursive; (EA2) the function $s \mapsto \mathit{Post}(s)$, that associates with any state its image by the relation $\to$, is computable (and image-finite, aka finitely branching); and (EA3) the wqo $\leq$ is decidable. We say that $\mathcal{S}$ is an *effective* WSTS when all three assumptions are fulfilled. Note that (EA1–2) hold of most computational models, starting with Turing machines and broadcast protocols, but we have to spell out these assumptions at some point since Def. 1.6 is abstract and does not provide any algorithmic foothold.

We can now prove the decidability of Termination for effective WSTS. Assume $\mathcal{S} = (S, \to, \leq)$ is effective. We are given some starting state $s_{init} \in S$. The

existence of an infinite run is semi-decidable since infinite runs admit finite witnesses by Lemma 3.1. (Note that we rely on all three effectiveness assumptions to guess a finite sequence $(s_{init} =)s_0, \ldots, s_i, \ldots, s_j$ of states, check that it is indeed a run of $\mathcal{S}$, and that it is indeed a good sequence.) Conversely, if all runs from s_{init} are finite, then there are only finitely many of them (by Kőnig's Lemma, since $\mathcal{S}$ is finitely branching), and it is possible to enumerate all these runs by exhaustive simulation, thanks to (EA2). Thus Termination is semi-decidable as well. Finally, since the Termination problem and its complement are both semi-decidable, they are decidable.

3.2 Coverability

After Termination, we turn to the decidability of Coverability, a slightly more involved result that is also more useful for practical purposes: the decidability of Coverability opens the way to the verification of safety properties and many other properties defined by fixpoints, see [8].

Recall that, for an ordered TS $\mathcal{S}$, Coverability is the question, given a starting state $s_{init} \in S$ and a target state $t \in S$, whether there is a run from s_{init} that eventually covers t, i.e., whether there is some s reachable from s_{init} with $s \geq t$. We call any finite run $s_0 \to s_1 \to \cdots \to s_n$ s.t. $s_n \geq t$ a *covering run (for t)*.

Rather than using covering runs to witness Coverability, we shall use "pseudoruns". Formally, a *pseudorun* is a sequence $s_0, \ldots, s_n$ such that, for all $i = 1, \ldots, n$, s_{i-1} can cover s_i *in one step*, i.e., $s_{i-1} \to t_i$ for some $t_i \geq s_i$. In particular, any run is also a pseudorun. And the existence of a pseudorun $s_0, \ldots, s_n$ with $s_n \geq t$ witnesses the existence of covering runs from any $s_{init} \geq s_0$ (proof: by repeated use of Fact 1.2).

A pseudorun $s_0, \ldots, s_n$ is *minimal* if, for all $i = 1, \ldots, n$, s_{i-1} is minimal among all the states from where s_i can be covered in one step (we say that s_{i-1} is a *minimal pseudopredecessor* of s_i).

Lemma 3.2 (Minimal Witnesses for Coverability). *If $\mathcal{S}$ has a covering run from s_{init}, it has in particular a minimal pseudorun $s_0, s_1, \ldots, s_n$ with $s_0 \leq s_{init}$, $s_n = t$, and such that the* reverse sequence *$s_n, s_{n-1}, \ldots, s_0$ is bad (we say that $s_0, \ldots, s_n$ is "revbad").*

Proof. Assume that $s_{init} = s_0 \to s_1 \to \cdots \to s_n$ is a covering run. Replacing s_n by t gives a pseudorun ending in t. We now show that if the pseudorun is not minimal or not revbad, then there is a "smaller" pseudorun.

First, assume that $s_n, s_{n-1}, \ldots, s_0$ is not bad. Then $s_i \geq s_j$ for some $0 \leq i < j \leq n$ and $s_0, s_1, \ldots, s_{i-1}, s_j, s_{j+1}, \ldots, s_n$ is again a pseudorun, shorter in length (note that $s_n = t$ is unchanged, while s_0 may have been replaced by a smaller s_j in the case where $i = 0$). If now $s_0, s_1, \ldots, s_n$ is not minimal, i.e., if some s_{i-1} is not a minimal pseudopredecessor of s_i, we may replace s_{i-1} by some other pseudopredecessor $s'_{i-1} \leq s_{i-1}$ that is minimal (since $(S, \leq)$ is wqo, hence well-founded, its non-empty subsets do have minimal elements) and $s_0, \ldots, s_{i-2}, s'_{i-1}, s_i, \ldots, s_n$ is again a pseudorun (where $s_n = t$ as before and

where s_0 may have been replaced by a smaller s'_0). Repeating such shortening and lowering replacements as long as possible is bound to terminate (after at most polynomially many replacements). The pseudorun we end up with is minimal, revbad, has $s_0 \leq s_{init}$ and $s_n = t$ as claimed. □

Turning Lemma 3.2 into a decidability proof is similar to what we did for Termination. This time we make the following effectiveness assumptions: (EA1) and (EA3) as above, with (EA2') the assumption that the function *MinPPre*—that associates with any state its finite set of minimal pseudopredecessors—is computable.[2] The set of all minimal revbad pseudoruns ending in t is finite (Kőnig's Lemma again: finite branching of the tree is ensured by minimality of the pseudoruns, while finite length of the branches is ensured by the restriction to revbad pseudoruns). This set of pseudoruns can be built effectively, starting from t and applying *MinPPre* repeatedly, but it is enough to collect the states that occur along them, using a standard backward-chaining scheme. We write $\mathit{MinPPre}^*(t)$ to denote the set of all these states: once they have been computed, it only remains to be checked whether s_{init} is larger than one of them, using (EA3) once more.

We conclude by observing that assumption (EA2'), though less natural-looking than (EA2), is satisfied in most computational models. As Ex. 4.1 shows for the case of broadcast protocols, computing $\mathit{MinPPre}(s)$ is often a simple case of finding the minimal solutions to a simple inverse problem on rewrite rules.

3.3 What is the Complexity of WSTS Verification?

One aspect of the algorithms given in this section we have swept under the rug is how expensive they can be. This will be the topic of the next two sections, but as an appetizer, let us consider how long the Termination algorithm can run on the broadcast protocol of Fig. 1.

Let us ignore the number of processes in the sink location $\bot$. The protocol from Fig. 1 allows the following steps when spawn steps are performed as long as possible before broadcasting m:

$$\{c^n, q\} \xrightarrow{a^n} \{a^{2n}, q\} \xrightarrow{m} \{c^{2n}\} .$$

Such a greedy sequence of message thus doubles the number of processes in c and removes one single process from q. Iterating such sequences as long as some process is in q before broadcasting d then leads to:

$$\begin{aligned}\{c^{2^0}, q^n, r\} &\xrightarrow{a^{2^0} m} \{c^{2^1}, q^{n-1}, r\} \xrightarrow{a^{2^1} m} \{c^{2^2}, q^{n-2}, r\} \\ &\cdots \to \{c^{2^{n-1}}, q, r\} \xrightarrow{a^{2^{n-1}} m} \{c^{2^n}, r\} \xrightarrow{d} \{c^{2^0}, q^{2^n}\} .\end{aligned}$$

[2] Recall that any subset of a wqo has only finitely many minimal elements up to the equivalence given by $s \equiv s' \overset{\text{def}}{\Leftrightarrow} s \leq s' \leq s$.

Such iterations thus implement an exponentiation of the number of processes in q in exchange for decrementing the number of processes in r by one. Repeating this kind of sequences therefore allows:

$$\{c, q, r^n\} \to^* \{c, q^{\mathrm{tower}(n)}\},$$

where $\mathrm{tower}(0) \stackrel{\text{def}}{=} 1$ and $\mathrm{tower}(n+1) \stackrel{\text{def}}{=} 2^{\mathrm{tower}(n)}$: although it always terminates, the broadcast protocol of Fig. 1 can exhibit sequences of steps of non-elementary length. This also entails a non-elementary lower bound on the Termination algorithm when run on this protocol: since the system terminates, all the runs need to be checked, including this particular non-elementary one. □

4 Upper Bounds on Complexity

Although the theory of well-structured systems provides generic algorithms for numerous verification problems, it might seem rather unclear, what the computational cost of running these algorithms could be—though we know their complexity can be considerable (recall Sec. 3.3). Inspecting the termination arguments in Sec. 3, we see that the critical point is the finiteness of bad sequences. Unfortunately, the wqo definition does not mention anything about the *length* of such sequences, but merely asserts that they are finite.

It turns out that very broadly applicable hypotheses suffice in order to define a maximal length for bad sequences (Sec. 4.1), which then gives rise to so-called *length function theorems* bounding such lengths using ordinal-indexed functions (Sec. 4.2). These upper bounds allow for a classification of the power of many WSTS models in complexity-theoretic terms (Sec. 4.3), and also lead to simplified WSTS algorithms that take advantage of the existence of computable upper bounds on the length of bad sequences (Sec. 4.4).

4.1 Controlled Sequences

The Length of Bad Sequences. If we look at a very simple quasi-order, namely $(Q, =)$ with a finite support Q and equality as ordering—which is a wqo by the pigeonhole principle—, we can only exhibit bad sequences with length up to $\#_Q$, the cardinality of Q. But things start going awry as soon as we consider infinite wqos; for instance

$$n, n-1, n-2, \ldots, 0 \tag{S1}$$

is a bad sequence over $(\mathbb{N}, \leq)$ for every n in $\mathbb{N}$, i.e. the length of a bad sequence over $(\mathbb{N}, \leq)$ can be arbitrary. Even if we restrict ourselves to bad sequences where the first element is not too large, we can still build arbitrarily long sequences: for instance, over $(\mathbb{N}^Q, \subseteq)$ with $Q = \{p, q\}$,

$$\{p\}, \{q^n\}, \{q^{n-1}\}, \ldots, \{q\}, \emptyset \tag{S2}$$

is a bad sequence of length $n + 2$.

Controlling Sequences. Here is however a glimpse of hope: the sequence (S2) cannot be the run of a broadcast protocol, due to the sudden "jump" from $\{p\}$ to an arbitrarily large configuration $\{q^n\}$. More generally, the key insight is that, in an algorithm that relies on a wqo for termination, successive states cannot jump to arbitrarily large sizes, because these states are constructed algorithmically: we call such sequences *controlled*.

Let $(A, \leq_A)$ be a wqo. Formally, we posit a *norm* $|.|_A \colon A \to \mathbb{N}$ on the elements of our wqo, which we require to be *proper*, in that only finitely many elements have norm n: put differently, $A_{\leq n} \stackrel{\text{def}}{=} \{x \in A \mid |x|_A \leq n\}$ must be finite for every n. For instance, $|s|_{\mathbb{N}^Q} \stackrel{\text{def}}{=} \max_{q \in Q} s(q)$ is a proper norm for $\mathbb{N}^Q$.

Given an increasing *control function* $g \colon \mathbb{N} \to \mathbb{N}$, we say that a sequence $x_0, x_1, \ldots$ over A is (g, n_0)-*controlled* if the norm of x_i is no larger that the ith iterate of g applied to n_0: $|x_i|_A \leq g^i(n_0)$ for all i. Thus g bounds the growth of the elements in the sequence, and n_0 is a bound on the initial norm $|x_0|_A$.

Example 4.1 (Controlled Successors in a Broadcast Protocol). Let us see how these definitions work on broadcast protocols. First, on the sequences of successors built in the Termination algorithm: If $s \to s'$ is a rendez-vous step, then $|s'|_{\mathbb{N}^Q} \leq 2 + |s|_{\mathbb{N}^Q}$, corresponding to the case where the two processes involved in the rendez-vous move to the same location. If $s \to s'$ is a broadcast step, then $|s'|_{\mathbb{N}^Q} \leq \#_Q \cdot |s|_{\mathbb{N}^Q}$, corresponding to the case where all the processes in s (of which there are at most $\#_Q \cdot |s|_{\mathbb{N}^Q}$) enter the same location. Finally, spawn steps only incur $|s'|_{\mathbb{N}^Q} \leq |s|_{\mathbb{N}^Q} + 2$. Thus $g(n) \stackrel{\text{def}}{=} \#_Q \cdot n$ defines a control function for any run in a broadcast protocol with $\#_Q \geq 2$ locations, provided the initial norm n_0 is chosen large enough. □

Example 4.2 (Controlled Minimal Pseudopredecessors in a Broadcast Protocol). Now for the minimal pseudopredecessors built in the course of the Coverability algorithm: Assume $|t|_{\mathbb{N}^Q} \leq n$ and $s \to s' \geq t$ is a step from some minimal s:

rendez-vous step: If $s' = (s - \{q_1, q_2\} + \{q_1', q_2'\})$ in a rendez-vous, then
- either $\{q_1', q_2'\} \subseteq t$ and thus $s = t - \{q_1', q_2'\} + \{q_1, q_2\}$ and $|s|_{\mathbb{N}^Q} \leq n$,
- or $q_i' \notin t$ for exactly one i among $\{1, 2\}$, hence $s = t - \{q_{1-i}\} + \{q_1, q_2\}$ and $|s|_{\mathbb{N}^Q} \leq n + 1$,
- or $q_i' \notin t$ for any $i \in \{1, 2\}$ and $|s|_{\mathbb{N}^Q} \leq n + 2$ (note however that $s \subseteq t$ in this case and the constructed sequence would not be bad).

broadcast step: Assume $s' = (s - \{q_0, q_1, \ldots, q_k\} + \{q_0', q_1', \ldots, q_k'\})$ with $q_0 \xrightarrow{m!!}_{\mathtt{B}} q_0'$ the corresponding broadcast send rule. Because s is minimal, $\{q_1', \ldots, q_k'\} \subseteq t$, as otherwise a smaller s could be used. Hence,
- either $q_0' \in t$, and then $s = t - \{q_0', q_1', \ldots, q_k'\} + \{q_0, q_1, \ldots, q_k\}$ and $|s|_{\mathbb{N}^Q} \leq n$,
- or $q_0' \notin t$, and then $s = t - \{q_1', \ldots, q_k'\} + \{q_0, q_1, \ldots, q_k\}$ and $|s|_{\mathbb{N}^Q} \leq n + 1$.

spawn step: similar to a rendez-vous step, $|s|_{\mathbb{N}^Q} \leq n + 1$.

Therefore, $g(n) \stackrel{\text{def}}{=} n + 2$ defines a control function for any sequence of minimal pseudopredecessor steps in any broadcast protocol. □

Length Functions. The upshot of these definitions is that, unlike in the uncontrolled case, there is a longest (g, n_0)-controlled bad sequence over any normed wqo $(A, \leq_A)$: indeed, we can organize such sequences in a tree by sharing common prefixes; this tree has

- finite branching, bounded by the cardinal of $A_{\leq g^i(n_0)}$ for a node at depth i, and
- no infinite branches thanks to the wqo property.

By Kőnig's Lemma, this tree of bad sequences is therefore finite, of some height $L_{g,n_0,A}$ representing the length of the maximal (g, n_0)-controlled bad sequence(s) over A. In the following, since we are mostly interested in this length as a function of the initial norm n_0, we will see this as a *length function* $L_{g,A}(n)$; our purpose will then be to obtain complexity bounds on $L_{g,A}$ depending on g and A.

4.2 Length Function Theorems

Now that we are empowered with a suitable definition for the maximal length $L_{g,A}(n)$ of (g, n)-controlled bad sequences over A, we can try our hand at proving *length function theorems*, which provide constructible functions bounding $L_{g,A}$ for various normed wqos $(A, \leq)$. Examples of length function theorems can be found

- in [36, 15, 20, 5, 41] for Dickson's Lemma, i.e. for $(\mathbb{N}^Q, \subseteq)$ for some finite Q, which is isomorphic to $(\mathbb{N}^{\#_Q}, \leq_\times)$,
- in [5] for $(\mathcal{P}_f(\mathbb{N}^d), \preceq)$ the set of finite subsets of $\mathbb{N}^d$ with the majoring ordering defined by $X \preceq Y \stackrel{\text{def}}{\Leftrightarrow} \forall x \in X, \exists y \in Y, x \leq_\times y$,
- in [45, 14, 40] for Higman's Lemma, i.e. for $(\Sigma^*, \leq_*)$ the set of finite sequences over a finite alphabet Σ with the subword embedding $\leq_*$,
- in [45] for Kruskal's Tree Theorem, i.e. for $(T, \leq_T)$ the set of finite unranked ordered trees with the homeomorphic embedding $\leq_T$.

These theorems often differ in the hypotheses they put on g, the tightness of the upper bounds they provide, and on the simplicity of their proofs (otherwise the results of Weiermann [45] for Kruskal's Tree Theorem would include all the others). We will try to convey the flavour of the theorems from [40, 41] here.

Starting again with the case of a finite wqo $(Q, =)$, and setting $|x|_Q \stackrel{\text{def}}{=} 0$ for all x in Q as the associated norm, we find immediately that, for all g and n,

$$L_{g,Q}(n) = \#_Q \tag{1}$$

by the pigeonhole principle. Another easy example is $(\mathbb{N}, \leq)$ with norm $|k|_\mathbb{N} \stackrel{\text{def}}{=} k$:

$$L_{g,\mathbb{N}}(n) = n + 1\ , \tag{2}$$

the bad sequence (S1) being maximal.

We know however from Sec. 3.3 that very long bad sequences can be constructed, so we should not hope to find such simple statements for $(\mathbb{N}^Q, \subseteq)$ and

more complex wqos. The truth is that the "tower" function we used in Sec. 3.3 is really benign compared to the kind of upper bounds provided by length function theorems. Such functions of enormous growth have mainly been studied in the context of subrecursive hierarchies and reverse mathematics (see e.g. [43, Chap. 4] for further reference); let us summarily present them.

Ordinal Indexed Functions. An idea in order to build functions of type $\mathbb{N} \to \mathbb{N}$ with faster and faster growths is to iterate smaller functions a number of times that depends on the argument—this is therefore a form of diagonalisation. In order to keep track of the diagonalisations, we can index the constructed functions with ordinals, so that diagonalisations occur at limit ordinals.

Ordinal Terms. First recall that ordinals α below ε_0 can be denoted as terms in *Cantor Normal Form*, aka CNF:

$$\alpha = \omega^{\beta_1} \cdot c_1 + \cdots + \omega^{\beta_n} \cdot c_n \text{ where } \alpha > \beta_1 > \cdots > \beta_n \text{ and } \omega > c_1, \ldots, c_n > 0 \,.$$

In this representation, $\alpha = 0$ if and only if $n = 0$. An ordinal with CNF of the form $\alpha' + 1$ (i.e. with $n > 0$ and $\beta_n = 0$) is called a *successor* ordinal, and otherwise if $\alpha > 0$ it is called a *limit* ordinal, and can be written as $\gamma + \omega^\beta$ by setting $\gamma = \omega^{\beta_1} \cdot c_1 + \cdots + \omega^{\beta_n} \cdot (c_n - 1)$ and $\beta = \beta_n$. We usually write "λ" to denote a limit ordinal.

A *fundamental sequence* for a limit ordinal λ is a sequence $(\lambda(x))_{x<\omega}$ of ordinals with supremum λ, with a standard assignment defined inductively by

$$(\gamma + \omega^{\beta+1})(x) \stackrel{\text{def}}{=} \gamma + \omega^\beta \cdot (x+1) \,, \qquad (\gamma + \omega^\lambda)(x) \stackrel{\text{def}}{=} \gamma + \omega^{\lambda(x)} \,. \tag{3}$$

This is one particular choice of a fundamental sequence, which verifies e.g. $0 < \lambda(x) < \lambda(y)$ for all $x < y$. For instance, $\omega(x) = x + 1$, $(\omega^{\omega^4} + \omega^{\omega^3+\omega^2})(x) = \omega^{\omega^4} + \omega^{\omega^3+\omega\cdot(x+1)}$.

Hardy Hierarchy. Let $h\colon \mathbb{N} \to \mathbb{N}$ be an increasing function. The *Hardy hierarchy* $(h^\alpha)_{\alpha<\varepsilon_0}$ controlled by h is defined inductively by

$$h^0(x) \stackrel{\text{def}}{=} x \,, \qquad h^{\alpha+1}(x) \stackrel{\text{def}}{=} h^\alpha(h(x)) \,, \qquad h^\lambda(x) \stackrel{\text{def}}{=} h^{\lambda(x)}(x) \,. \tag{4}$$

Observe that h^k for some finite k is the kth iterate of h (by using the first two equations solely). This intuition carries over: h^α is a transfinite iteration of the function h, using diagonalisation to handle limit ordinals. For instance, starting with the successor function $H(x) \stackrel{\text{def}}{=} x + 1$, we see that a first diagonalisation yields $H^\omega(x) = H^{x+1}(x) = 2x + 1$. The next diagonalisation occurs at $H^{\omega\cdot 2}(x) = H^{\omega+x+1}(x) = H^\omega(2x+1) = 4x + 3$. Fast-forwarding a bit, we get for instance a function of exponential growth $H^{\omega^2}(x) = 2^{x+1}(x+1) - 1$, and later a non elementary function H^{ω^3}, an "Ackermannian" non primitive-recursive function H^{ω^ω}, and an "hyper-Ackermannian" non multiply-recursive function $H^{\omega^{\omega^\omega}}$. Hardy functions are well-suited for expressing large iterates of a control function, and therefore for bounding the norms of elements in a controlled bad sequence.

Cichoń Hierarchy. A variant of the Hardy functions is the *Cichoń hierarchy* $(h_\alpha)_{\alpha<\varepsilon_0}$ controlled by h [14], defined by

$$h_0(x) \stackrel{\text{def}}{=} 0\,, \qquad h_{\alpha+1}(x) \stackrel{\text{def}}{=} 1 + h_\alpha\left(h(x)\right), \qquad h_\lambda(x) \stackrel{\text{def}}{=} h_{\lambda(x)}(x)\,. \tag{5}$$

For instance, $h_d(x) = d$ for all finite d, thus $h_\omega(x) = x+1$ regardless of the choice of the function h. One can check that $H^\alpha(x) = H_\alpha(x) + x$ when employing the successor function H; in general $h^\alpha(x) \geq h_\alpha(x) + x$ since h is assumed to be increasing.

This is the hierarchy we are going to use for our statements of length function theorems: a Hardy function h^α is used to bound the maximal norm of an element in a bad sequence, and the corresponding Cichoń function h_α bounds the length of the bad sequence itself, the two functions being related for all h, α, and x by

$$h^\alpha(x) = h^{h_\alpha(x)}(x)\,. \tag{6}$$

Length Functions for Dickson's Lemma. We can now provide an example of a length function theorem for a non-trivial wqo: Consider $(\mathbb{N}^Q, \subseteq)$ and some control function g. Here is one of the parametric bounds proved in [41, Chap. 2]:[3]

Theorem 4.3 (Parametric Bounds for Dickson's Lemma). *If $x_0, \ldots, x_L$ is a (g,n)-controlled bad sequence over $(\mathbb{N}^Q, \subseteq)$ for some finite set Q, then $L \leq L_{g,\mathbb{N}^Q}(n) \leq h_{\omega^{\#Q}}(n)$ for the function $h(x) \stackrel{\text{def}}{=} \#_Q \cdot g(x)$.*

By Equation (6), we also deduce that the norm of the elements x_i in this bad sequence cannot be larger than $h^{\omega^{\#Q}}(n)$.

A key property of such bounds expressed with Cichoń and Hardy functions is that they are *constructible* with negligible computational overhead (just apply their definition on a suitable encoding of the ordinals), which means that we can employ them in algorithms (see Sec. 4.4 for applications).

Given how enormous the Cichoń and Hardy functions can grow, it is reasonable at this point to ask how tight the bounds provided by Thm. 4.3 really are. At least in the case $\#_Q = 1$, we see these bounds match (2) since $h_\omega(n) = n+1$. We will show in Sec. 5 that similarly enormous complexity lower bounds can be proven for Termination or Coverability problems on WSTS, leaving only inessential gaps with the upper bounds like Thm. 4.3. In fact, we find such parametric bounds to be overly precise, because we would like to express simple *complexity* statements about decision problems.

[3] A more general version of Thm. 4.3 in [40] provides $h_{o(A)}(n)$ upper bounds for (g,n)-controlled bad sequences over wqos $(A, \leq)$ constructed through disjoint unions, cartesian products, and Kleene star operations, where $o(A)$ is the maximal order type of $(A, \leq)$, i.e. the order type of its maximal linearization, and h is a low-degree polynomial in g. This matches Thm. 4.3 because $o(\mathbb{N}^Q) = \omega^{\#Q}$.

4.3 Fast Growing Complexy Classes

As witnessed in Sec. 3.3 and the enormous upper bounds provided by Thm. 4.3, we need to deal with non-elementary complexities. The corresponding non-elementary complexity classes are arguably missing from most textbooks and references on complexity. For instance, the *Complexity Zoo*[4], an otherwise very richly populated place, features no intermediate steps between Elementary and the next class, namely Primitive-Recursive (aka PR), and a similar gap occurs between PR and Recursive (aka R). If we are to investigate the complexity of decision problems on WSTSs, much more fine-grained hierarchies are required.

Here, we present a hierarchy of ordinal-indexed *fast growing* complexity classes $(\mathbf{F}_\alpha)_\alpha$ tailored to *completeness* proofs for non-elementary problems [see 41, App. B]. When exploring larger complexities, the hierarchy includes non primitive-recursive classes, for which quite a few complete problems have arisen in the recent years, e.g. $\mathbf{F}_\omega$ in [35, 28, 44, 42, 19, 10, 33], $\mathbf{F}_{\omega^\omega}$ in [13, 38, 32, 6, 12, 7], $\mathbf{F}_{\omega^{\omega^\omega}}$ in [27], and $\mathbf{F}_{\varepsilon_0}$ in [26].

Let us define the classes of reductions and of problems we will consider:

$$\mathscr{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{c<\omega} \text{FDTime}\left(H^{\omega^\alpha \cdot c}(n)\right), \quad \mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{p \in \bigcup_{\beta<\alpha} \mathscr{F}_\beta} \text{DTime}\left(H^{\omega^\alpha}(p(n))\right). \quad (7)$$

The hierarchy of function classes $(\mathscr{F}_\alpha)_{\alpha \geq 2}$ is the *extended Grzegorczyk hierarchy* [34], and provides us with classes of non-elementary reductions: for instance $\mathscr{F}_2$ is the set of elementary functions, $\bigcup_{\beta<\omega} \mathscr{F}_\beta$ that of primitive-recursive functions, and $\bigcup_{\beta<\omega^\omega} \mathscr{F}_\beta$ that of multiply-recursive functions. The hierarchy of complexity classes $(\mathbf{F}_\alpha)_{\alpha \geq 3}$ features for instance a class $\mathbf{F}_\omega$ of Ackermannian problems closed under primitive-recursive reductions, and a class $\mathbf{F}_{\omega^\omega}$ of hyper-Ackermannian problems closed under multiply-recursive reductions. Intuitively, $\mathbf{F}_\omega$-complete problems are not primitive-recursive, but only *barely* so, and similarly for the other levels.

4.4 Combinatory Algorithms

Theorem 4.3 together with Ex. 4.1 (resp. 4.2) provides complexity upper bounds on the Termination (resp. Coverability) algorithm when applied to broadcast protocols. Indeed, a nondeterministic program can guess a witness of (non-) Termination (resp. Coverability), which is of length bounded by $L_{g,\mathbb{N}^Q}(n) + 1$, where g was computed in Ex. 4.1 (resp. 4.2) and n is the size of the initial configuration s_{init} (resp. target configuration t). By Thm. 4.3 such a witness has length bounded by $h_{\omega^{\#_Q}}(n)$ for $h(n) = \#_Q \cdot g(n)$, and by (6), the norm of the elements along this sequence is bounded by $h^{\omega^{\#_Q}}(n)$. Thus this non-deterministic program only needs space bounded by $\#_Q \cdot \log(h^{\omega^{\#_Q}}(n)) \leq H^{\omega^\omega}(p(n + \#_Q))$ for some primitive-recursive function p. Hence:

[4] https://complexityzoo.uwaterloo.ca

Fact 4.4. *Termination and Coverability of broadcast protocols are in* $\mathbf{F}_\omega$.

Thanks to the upper bounds on the length of bad sequences, the algorithms sketched above are really *combinatory* algorithms: they compute a maximal length for a witness and then nondeterministically check for its existence. In the case of Termination, we can even further simplify the algorithm: if a run starting from s_{init} has length $> L_{g,A}(n)$, this run is necessarily a good sequence and the WSTS does not terminate.

5 Lower Bounds on Complexity

When considering the mind-numbling complexity upper bounds that come with applications of the Length Function Theorems from Sec. 4 to the algorithms of Sec. 3, a natural question that arises is whether this is the complexity one gets when using the rather simplistic Coverability algorithm from Sec. 3.2, or whether it is the intrinsic complexity of the Coverability *problem* for a given WSTS model.

There is no single answer here: for instance, on Petri nets, a breadth-first backward search for a coverability witness actually works in 2ExpTime [9] thanks to bounds on the size of minimal witnesses due to Rackoff [39]; on the other hand, a depth-first search for a termination witness can require Ackermannian time [11], this although both problems are ExpSpace-complete.

Nevertheless, in many cases, the enormous complexity upper bounds provided by the Length Function Theorems are matched by similar lower bounds on the complexity of the Coverability and Termination *problems*, for instance for reset/transfer Petri nets [$\mathbf{F}_\omega$-complete, see 42], lossy channel systems [$\mathbf{F}_{\omega^\omega}$-complete, see 13], timed-arc Petri nets [$\mathbf{F}_{\omega^\omega\omega}$-complete, see 27], or priority channel systems [$\mathbf{F}_{\varepsilon_0}$-complete, see 26].

Our goal in this section is to present the common principles behind these $\mathbf{F}_\alpha$-hardness proofs. We will avoid most of the technical details, relying rather on simple examples to convey the main points: the interested readers will find all details in the references.

Let us consider a WSTS model like broacast protocols or lossy channel systems. Without a rich repertoire of $\mathbf{F}_\alpha$-complete problems, one proves $\mathbf{F}_\alpha$-hardness by reducing, e.g., from the acceptance problem for a H^{ω^α}-space bounded Minsky machine M. In order to simulate M in the WSTS model at hand, the essential part is to design a way to compute $H^{\omega^\alpha}(n_0)$ reliably and store this number as a WSTS state, where it can be used as a working space for the simulation of M. In all the cases we know, these computations cannot be performed directly (indeed, our WSTS are not Turing-powerful), but $\mathcal{S}_M$, the constructed WSTS, is able to *weakly* compute such values. This means that $\mathcal{S}_M$ may produce the correct value for $H^{\omega^\alpha}(n_0)$ but also (nondeterministically) some smaller values. However, the reduction is able to include a check that the computation was actually correct, either at the end of the simulation [42, 13, 27, 26]—by weakly computing the inverse of H^{ω^α} and testing through the coverability condition whether the final

configuration is the one we started with—, or continuously at every step of the simulation [33].

5.1 Hardy Computations

As an example, when $\alpha < \omega^k$, one may weakly compute H^α and its inverse using a broadcast protocol with $k + O(1)$ locations. In order to represent an ordinal $\alpha = \omega^{k-1} \cdot c_{k-1} + \cdots + \omega^0 \cdot c_0$ in CNF, one can employ a configuration $s_\alpha = \{p_0^{c_0}, \ldots, p_{k-1}^{c_{k-1}}\}$ in a broadcast protocol having locations $p_0, \ldots, p_{k-1}$ among others.

There remains other issues with ordinal representations in a WSTS state (see Sec. 5.2), but let us first turn to the question of computing some $H^\alpha(n)$. The definition of the Hardy functions is based on very fine-grained steps, and this usually simplifies their implementations. We can reformulate (4) as a rewrite system over pairs (α, x) of an ordinal and an argument:

$$(\alpha + 1, x) \to (\alpha, x + 1)\,, \qquad (\lambda, x) \to (\lambda(x), x)\,. \tag{4'}$$

A sequence $(\alpha_0, x_0) \to (\alpha_1, x_1) \to \cdots \to (\alpha_\ell, x_\ell)$ of such "Hardy steps" implements (4) and maintains $H^{\alpha_i}(x_i)$ invariant. It must terminates since $\alpha_0 > \alpha_1 > \cdots$ is decreasing. When eventually $\alpha_\ell = 0$, the computation is over and the result is $x_\ell = H^{\alpha_0}(x_0)$.

Example 5.1 (Hardy Computations in Broadcast Protocols). Implementing (4') in a broadcast protocol requires us to consider two cases. Recognizing whether α_i is a successor boils down to checking that $c_0 > 0$ in the CNF. In that case, and assuming the above representation, (4') is implemented by moving one process from location p_0 to a location x where the current value of x_i is stored. The broadcast protocol will need a rule like $p_0 \xrightarrow{\mathrm{sp}(x)} \bot$.

Alternatively, α_i is a limit $\gamma + \omega^b$ when $c_0 = c_1 = \ldots = c_{b-1} = 0 < c_b$. In that case, (4') is implemented by moving one process out of the p_b location, and adding to p_{b-1} as many processes as there are currently in x. This can be implemented by moving temporarily all processes in x to some auxiliary x_{tmp} location, then putting them back in x one by one, each time spawning a new process in p_{b-1}.

The difficulty with these steps (and with recognizing that α_i is a limit) is that one needs to test that some locations are empty, an operation not provided in broadcast protocols (adding emptiness tests would make broadcast protocols Turing-powerful, and would break the monotonicity of behaviour). Instead of being tested, these locations can be forcefully emptied through broadcast steps. This is where the computation of $H^{\alpha_i}(x_i)$ may err and end up with a smaller value, if the locations were not empty. We refer to [42] or [41, Chap. 3] for a detailed implementation of this scheme using lossy counters machines: the encoding therein can easily be reformulated as a broadcast protocol:

Fact 5.2. *Termination and Coverability of broadcast protocols are* $\mathbf{F}_\omega$*-hard.*

5.2 Robust Encodings

The above scheme for transforming pairs (α, x) according to Eq. (4') can be used with ordinals higher than ω^k. Ordinals up to ω^{ω^ω} have been encoded as configurations of lossy channel systems [13], of timed-arc nets (up to $\omega^{\omega^{\omega^\omega}}$, see [27]), and of priority channel systems (up to ε_0, see [26]). The operations one performs on these encodings are recognizing whether an ordinal is a successor or a limit, transforming an $\alpha + 1$ in α, and a λ in $\lambda(x)$. Such operations can be involved, depending on the encoding and the facilities offered by the WSTS: see [27] for an especially involved example.

It is usually not possible to perform Hardy steps exactly in the WSTS under consideration. Hence one is content with weak implementations that may err when realizing a step $(\alpha_i, x_i) \to (\alpha_{i+1}, x_{i+1})$. One important difficulty arises here: it is not enough to guarantee that any weak step $(\alpha_i, x_i) \to (\alpha', x')$ has $\alpha' \leq \alpha_{i+1}$ and $x \leq x_{i+1}$. One further needs $H^{\alpha'}(x') \leq H^{\alpha_{i+1}}(x_{i+1})$, a property called "robustness". Since Hardy functions are in general not monotone in the α exponent (see [41]), extra care is needed in order to control what kinds of errors are acceptable when ending up with (α', x') instead of (α_{i+1}, x_{i+1}). We invite the reader to have a look at the three above-mentioned papers for examples of how these issues can be solved in each specific case.

6 Concluding Remarks

As the claim *Well-Structured Transition Systems Everywhere!* made in the title of [24] has been further justified by twelve years of applications of WSTS in various fields, the need to better understand the computational power of these systems has also risen. This research program is still very new, but it has already contributed mathematical tools and methodological guidelines for

- proving upper bounds, based on *length functions theorems* that provide bounds on the length of controlled bad sequences. We illustrated this on two algorithms for Coverability and Termination in Sec. 4, but the same ideas are readily applicable to many algorithms that rely on a wqo for their termination—and thus not only in a WSTS context—: one merely has to find out how the bad sequences constructed by the algorithm are *controlled*.
- establishing matching lower bounds: here our hope is for the problems we have proven hard for some complexity class $\mathbf{F}_\alpha$ to be reused as convenient "master" problems in reductions. Failing that, such lower bound proofs can also rely on a reusable framework developed in Sec. 5: our reductions from Turing or Minsky machines with bounded resources construct the machine workspace as the result of a Hardy computation, thanks to a suitable robust encoding of ordinals.

There are still many open issues that need to be addressed to advance this program: to develop length function theorems for more wqos, to investigate different wqo algorithms (like the computation of upward-closed sets from oracles

for membership and vacuity by Goubault-Larrecq [25]), and to populate the catalog of master $\mathbf{F}_\alpha$-hard problems, so that hardness proofs do not have to proceed from first principles and can instead rely on simpler reductions.

We hope that this paper can be used as an enticing primer for researchers who have been using WSTS as a decidability tool only, and are now ready to use them for more precise complexity analyses.

Acknowledgments. We thank Christoph Haase and Prateek Karandikar for their helpful comments on an earlier version of this paper. The remaining errors are of course entirely ours.

References

1. Abdulla, P.A.: Well (and better) quasi-ordered transition systems. Bull. Symbolic Logic 16(4), 457–515 (2010)
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., Delzanno, G., Van Begin, L.: A classification of the expressive power of well-structured transition systems. Inform. and Comput. 209(3), 248–279 (2011)
4. Abdulla, P.A., Jonsson, B.: Verifying programs with unreliable channels. Inform. and Comput. 127(2), 91–101 (1996)
5. Abriola, S., Figueira, S., Senno, G.: Linearizing bad sequences: upper bounds for the product and majoring well quasi-orders. WoLLIC 2012. LNCS, vol. 7456, pp. 110–126. Springer (2012)
6. Atig, M.F., Bouajjani, A., Burckhardt, S., Musuvathi, M.: On the verification problem for weak memory models. POPL 2010. pp. 7–18. ACM (2010)
7. Barceló, P., Figueira, D., Libkin, L.: Graph logics with rational relations and the generalized intersection problem. LICS 2012. pp. 115–124. IEEE Press (2012)
8. Bertrand, N., Schnoebelen, Ph.: Computable fixpoints in well-structured symbolic model checking. Form. Methods in Syst. Des. (2013), to appear
9. Bozzelli, L., Ganty, P.: Complexity analysis of the backward coverability algorithm for VASS. RP 2011. LNCS, vol. 6945, pp. 96–109. Springer (2011)
10. Bresolin, D., Della Monica, D., Montanari, A., Sala, P., Sciavicco, G.: Interval temporal logics over finite linear orders: The complete picture. ECAI 2012. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 199–204. IOS (2012)
11. Cardoza, E., Lipton, R., Meyer, A.R.: Exponential space complete problems for Petri nets and commutative subgroups. STOC'76. pp. 50–54. ACM (1976)
12. Chambart, P., Schnoebelen, Ph.: Post embedding problem is not primitive recursive, with applications to channel systems. FSTTCS 2007. LNCS, vol. 4855, pp. 265–276. Springer (2007)
13. Chambart, P., Schnoebelen, Ph.: The ordinal recursive complexity of lossy channel systems. LICS 2008. pp. 205–216. IEEE Press (2008)
14. Cichoń, E.A., Tahhan Bittar, E.: Ordinal recursive bounds for Higman's Theorem. Theor. Comput. Sci. 201(1–2), 63–84 (1998)
15. Clote, P.: On the finite containment problem for Petri nets. Theor. Comput. Sci. 43, 99–105 (1986)

16. Emerson, E.A., Namjoshi, K.S.: On model checking for non-deterministic infinite-state systems. LICS '98. pp. 70–80. IEEE Press (1998)
17. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. LICS '99. pp. 352–359. IEEE Press (1999)
18. Esparza, J., Ganty, P., Majumdar, R.: Parameterized verification of asynchronous shared-memory systems. CAV 2013. LNCS, Springer (2013), to appear
19. Figueira, D.: Alternating register automata on finite words and trees. Logic. Meth. in Comput. Sci. 8(1), 22 (2012)
20. Figueira, D., Figueira, S., Schmitz, S., Schnoebelen, Ph.: Ackermannian and primitive-recursive bounds with Dickson's Lemma. LICS 2011. pp. 269–278. IEEE Press (2011)
21. Finkel, A.: A generalization of the procedure of Karp and Miller to well structured transition systems. ICALP '87. LNCS, vol. 267, pp. 499–508. Springer (1987)
22. Finkel, A.: Decidability of the termination problem for completely specificied protocols. Distributed Computing 7(3), 129–135 (1994)
23. Finkel, A., Goubault-Larrecq, J.: Forward analysis for WSTS, part II: Complete WSTS. Logic. Meth. in Comput. Sci. 8(4:28) (2012)
24. Finkel, A., Schnoebelen, Ph.: Well-structured transition systems everywhere! Theor. Comput. Sci. 256(1–2), 63–92 (2001)
25. Goubault-Larrecq, J.: On a generalization of a result by Valk and Jantzen. Research Report LSV-09-09, Laboratoire Spécification et Vérification, ENS Cachan, France (2009)
26. Haase, C., Schmitz, S., Schnoebelen, Ph.: The power of priority channel systems. CONCUR 2013. LNCS, Springer (2013), this volume
27. 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)
28. Jančar, P.: Nonprimitive recursive complexity and undecidability for Petri net equivalences. Theor. Comput. Sci. 256(1–2), 23–30 (2001)
29. Jurdziński, M., Lazić, R.: Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Logic 12(3) (2011)
30. Kruskal, J.B.: The theory of well-quasi-ordering: A frequently discovered concept. J. Comb. Theory A 13(3), 297–305 (1972)
31. Kurucz, A.: Combining modal logics. Handbook of Modal Logics, chap. 15, pp. 869–926. Elsevier Science (2006)
32. Lasota, S., Walukiewicz, I.: Alternating timed automata. ACM Trans. Comput. Logic 9(2), 10 (2008)
33. Lazić, R., Ouaknine, J., Worrell, J.: Zeno, Hercules and the Hydra: Downward rational termination is Ackermannian. MFCS 2013. LNCS, Springer (2013), to appear
34. Löb, M., Wainer, S.: Hierarchies of number theoretic functions, I. Arch. Math. Logic 13, 39–51 (1970)
35. Mayr, E.W., Meyer, A.R.: The complexity of the finite containment problem for Petri nets. J. ACM 28(3), 561–576 (1981)
36. McAloon, K.: Petri nets and large finite sets. Theor. Comput. Sci. 32(1–2), 173–183 (1984)
37. Milner, R.: Operational and algebraic semantics of concurrent processes. Handbook of Theoretical Computer Science, chap. 19, pp. 1201–1242. Elsevier Science (1990)
38. Ouaknine, J., Worrell, J.: On the decidability and complexity of Metric Temporal Logic over finite words. Logic. Meth. in Comput. Sci. 3(1), 8 (2007)

39. Rackoff, C.: The covering and boundedness problems for vector addition systems. Theor. Comput. Sci. 6(2), 223–231 (1978)
40. Schmitz, S., Schnoebelen, Ph.: Multiply-recursive upper bounds with Higman's Lemma. ICALP 2011. LNCS, vol. 6756, pp. 441–452. Springer (2011)
41. Schmitz, S., Schnoebelen, Ph.: Algorithmic aspects of wqo theory. Lecture notes (2012), http://cel.archives-ouvertes.fr/cel-00727025
42. Schnoebelen, Ph.: Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. MFCS 2010. LNCS, vol. 6281, pp. 616–628. Springer (2010)
43. Schwichtenberg, H., Wainer, S.S.: Proofs and Computation. Perspectives in Logic, Cambridge University Press (2012)
44. Urquhart, A.: The complexity of decision procedures in relevance logic II. J. Symb. Log. 64(4), 1774–1802 (1999)
45. Weiermann, A.: Complexity bounds for some finite forms of Kruskal's Theorem. J. Symb. Comput. 18(5), 463–488 (1994)