The Power of Well-Structured Transition Systems

Sylvain Schmitz & Philippe Schnoebelen

LSV, CNRS & ENS Cachan

CMI, Chennai, Feb. 19, 2014

Based on CONCUR 2013 invited paper, see my web page for pdf

The Problem with WSTS

- Well-structured transition systems (WSTS) are a family of infinite-state models supporting generic verification algorithms based on well-quasi-ordering (WQO) theory.
- WSTS invented in 1987, developed and popularized in 1996–2005 by Abdulla & Jonsson, Finkel & Schnoebelen, etc. First used with Petri nets (or VAS) extensions, channel systems, counter machines, integral automata, etc.
- Still thriving today, with several new WSTS models (based on wqos on graphs, etc.), or applications (deciding data logics, modal logics, etc.) appearing every year
- Main question not answered during all these developments: what is the complexity of WSTS verification? Related question: what is the expressive power of these WSTS models?

THE PROBLEM WITH WSTS

- Well-structured transition systems (WSTS) are a family of infinite-state models supporting generic verification algorithms based on well-quasi-ordering (WQO) theory.
- WSTS invented in 1987, developed and popularized in 1996–2005 by Abdulla & Jonsson, Finkel & Schnoebelen, etc. First used with Petri nets (or VAS) extensions, channel systems, counter machines, integral automata, etc.
- Still thriving today, with several new WSTS models (based on wqos on graphs, etc.), or applications (deciding data logics, modal logics, etc.) appearing every year
- Main question not answered during all these developments: what is the complexity of WSTS verification? Related question: what is the expressive power of these WSTS models?

Some Recent Developments (2008—)

Exact complexity determined for verification problems on Petri net extensions, lossy channel systems, timed-arc Petri nets, etc.

More generally, we have been developing a set of theoretical tools for the complexity analysis of algorithms that rely on WQO-theory:

– Length-function theorems to bound the length of bad sequences

– Robust encodings of Hardy computations in WSTS

– Ordinal-recursive complexity classes with catalog of complete problems

These tools borrow from proof theory, WQO and ordinals theory, combinatorics à la Ramsey, … but repackaging was required

Some Recent Developments (2008—)

Exact complexity determined for verification problems on Petri net extensions, lossy channel systems, timed-arc Petri nets, etc.

More generally, we have been developing a set of theoretical tools for the complexity analysis of algorithms that rely on WQO-theory:

– Length-function theorems to bound the length of bad sequences

– Robust encodings of Hardy computations in WSTS

– Ordinal-recursive complexity classes with catalog of complete problems

These tools borrow from proof theory, WQO and ordinals theory, combinatorics à la Ramsey, … but repackaging was required

Some Recent Developments (2008—)

Exact complexity determined for verification problems on Petri net extensions, lossy channel systems, timed-arc Petri nets, etc.

More generally, we have been developing a set of theoretical tools for the complexity analysis of algorithms that rely on WQO-theory:

– Length-function theorems to bound the length of bad sequences

– Robust encodings of Hardy computations in WSTS

– Ordinal-recursive complexity classes with catalog of complete problems

These tools borrow from proof theory, WQO and ordinals theory, combinatorics à la Ramsey, . . . but repackaging was required

Outline of the Talk

- Part 1: **Basics of WSTS.** Recalling the basic definition, with broadcast protocols as an example
- Part 2: **Verifying WSTS.** Two simple verification algorithms, deciding Termination and Coverability
- Part 3: **Bounding Running Time.** By bounding the length of controlled bad sequences
- Part 4: **Proving (Matching) Lower Bounds.** By weakly computing ordinal-recursive functions

Technical details mostly avoided, see CONCUR paper for more.
Also, see our lecture notes “Algorithmic Aspects of WQO Theory”.

Part 1

Basics of WSTS

What Are WSTS?

Def. A WSTS is an ordered TS $\mathcal{S} = (S, \to, \leqslant)$ that is monotonic and such that $(S, \leqslant)$ is a well-quasi-ordering (a wqo, more later).

Recall:

– transition system (TS): $\mathcal{S} = (S, \to)$ with steps e.g. "$s \to s'$"

– ordered TS: $\mathcal{S} = (S, \to, \leqslant)$ with smaller and larger states, e.g. $s \leqslant t$

– monotonic TS: ordered TS with
$\big(s_1 \to s_2 \text{ and } s_1 \leqslant t_1\big)$ implies $\exists t_2 \in S : \big(t_1 \to t_2 \text{ and } s_2 \leqslant t_2\big)$,
i.e., "larger states simulate smaller states".

Equivalently: $\leqslant$ is a wqo and a simulation.

NB. Starting from any $t_0 \geqslant s_0$, a run $s_0 \to s_1 \to \cdots \to s_n$ can be simulated "from above" with some $t_0 \to t_1 \to \cdots \to t_n$

What Are WSTS?

Def. A WSTS is an ordered TS $\mathcal{S} = (S, \to, \leqslant)$ that is monotonic and such that $(S, \leqslant)$ is a well-quasi-ordering (a wqo, more later).

Recall:

– transition system (TS): $\mathcal{S} = (S, \to)$ with steps e.g. "$s \to s'$"

– ordered TS: $\mathcal{S} = (S, \to, \leqslant)$ with smaller and larger states, e.g. $s \leqslant t$

– monotonic TS: ordered TS with

$\big(s_1 \to s_2$ and $s_1 \leqslant t_1\big)$ implies $\exists t_2 \in S : \big(t_1 \to t_2$ and $s_2 \leqslant t_2\big)$,

i.e., "larger states simulate smaller states".

Equivalently: $\leqslant$ is a wqo and a simulation.

NB. Starting from any $t_0 \geqslant s_0$, a run $s_0 \to s_1 \to \cdots \to s_n$ can be simulated "from above" with some $t_0 \to t_1 \to \cdots \to t_n$

WELL-QUASI-ORDERING (WQO)

Now what was meant by "$(S,\leqslant)$ is wqo"?

Def1. $(X,\leqslant)$ is a wqo $\overset{\text{def}}{\Leftrightarrow}$ any infinite sequence $x_0,x_1,x_2,\ldots$ contains an increasing pair: $x_i \leqslant x_j$ for some $i<j$.

Def2. $(X,\leqslant)$ is a wqo $\overset{\text{def}}{\Leftrightarrow}$ any infinite sequence $x_0,x_1,x_2,\ldots$ contains an infinite increasing subsequence: $x_{n_0} \leqslant x_{n_1} \leqslant x_{n_2} \leqslant \ldots$

NB. These definitions are equivalent (not trivially).

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

$$a=(a_1,\ldots,a_k)\leqslant_\times b=(b_1,\ldots,b_k) \overset{\text{def}}{\Leftrightarrow} a_1\leqslant b_1 \wedge\cdots\wedge a_k\leqslant b_k$$

Other important/useful wqos: words with the subword relation (Higman's Lemma), trees (also multisets) ordered by embedding (Kruskal's Theorem), and graphs with minors (Robertson & Seymour's Graph Minor Theorem).

WELL-QUASI-ORDERING (WQO)

Now what was meant by "$(S, \leqslant)$ is wqo"?

Def1. $(X, \leqslant)$ is a wqo $\overset{\text{def}}{\Leftrightarrow}$ any infinite sequence $x_0, x_1, x_2, \ldots$ contains an increasing pair: $x_i \leqslant x_j$ for some $i < j$.

Def2. $(X, \leqslant)$ is a wqo $\overset{\text{def}}{\Leftrightarrow}$ any infinite sequence $x_0, x_1, x_2, \ldots$ contains an infinite increasing subsequence: $x_{n_0} \leqslant x_{n_1} \leqslant x_{n_2} \leqslant \ldots$

NB. These definitions are equivalent (not trivially).

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

$$a = (a_1, \ldots, a_k) \leqslant_\times b = (b_1, \ldots, b_k) \overset{\text{def}}{\Leftrightarrow} a_1 \leqslant b_1 \wedge \cdots \wedge a_k \leqslant b_k$$

Other important/useful wqos: words with the subword relation (Higman's Lemma), trees (also multisets) ordered by embedding (Kruskal's Theorem), and graphs with minors (Robertson & Seymour's Graph Minor Theorem).

Well-Quasi-Ordering (WQO)

Now what was meant by "$(S, \leqslant)$ is wqo"?

Def1. $(X, \leqslant)$ is a wqo $\overset{\text{def}}{\Leftrightarrow}$ any infinite sequence $x_0, x_1, x_2, \ldots$ contains an increasing pair: $x_i \leqslant x_j$ for some $i < j$.

Def2. $(X, \leqslant)$ is a wqo $\overset{\text{def}}{\Leftrightarrow}$ any infinite sequence $x_0, x_1, x_2, \ldots$ contains an infinite increasing subsequence: $x_{n_0} \leqslant x_{n_1} \leqslant x_{n_2} \leqslant \ldots$

NB. These definitions are equivalent (not trivially).

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

$$\mathbf{a} = (a_1, \ldots, a_k) \leqslant_\times \mathbf{b} = (b_1, \ldots, b_k) \overset{\text{def}}{\Leftrightarrow} a_1 \leqslant b_1 \wedge \cdots \wedge a_k \leqslant b_k$$

Other important/useful wqos: words with the subword relation (Higman's Lemma), trees (also multisets) ordered by embedding (Kruskal's Theorem), and graphs with minors (Robertson & Seymour's Graph Minor Theorem).

Well-Quasi-Ordering (WQO)

Now what was meant by "$(S, \leqslant)$ is wqo"?

Def1. $(X, \leqslant)$ is a wqo $\stackrel{\text{def}}{\Leftrightarrow}$ any infinite sequence $x_0, x_1, x_2, \ldots$ contains an increasing pair: $x_i \leqslant x_j$ for some $i < j$.

Def2. $(X, \leqslant)$ is a wqo $\stackrel{\text{def}}{\Leftrightarrow}$ any infinite sequence $x_0, x_1, x_2, \ldots$ contains an infinite increasing subsequence: $x_{n_0} \leqslant x_{n_1} \leqslant x_{n_2} \leqslant \ldots$

NB. These definitions are equivalent (not trivially).

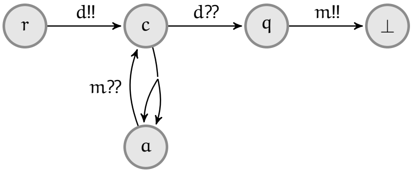
Example. (Dickson's Lemma) $(\mathbb{N}^k, \leqslant_\times)$ is a wqo, with

$$\mathbf{a} = (a_1, \ldots, a_k) \leqslant_\times \mathbf{b} = (b_1, \ldots, b_k) \stackrel{\text{def}}{\Leftrightarrow} a_1 \leqslant b_1 \wedge \cdots \wedge a_k \leqslant b_k$$

Other important/useful wqos: words with the subword relation (Higman's Lemma), trees (also multisets) ordered by embedding (Kruskal's Theorem), and graphs with minors (Robertson & Seymour's Graph Minor Theorem).

Example: Broadcast Protocols

Broadcast protocols (Esparza et al.'99) are dynamic & distributed collections of finite-state processes communicating via brodcasts and rendez-vous.



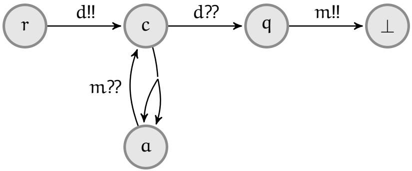
A configuration collects the local states of all processes. E.g., $s = \{c, r, c\}$, also denoted $\{c^2, r\}$.

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

We'll see later: The above protocol does not have infinite runs

EXAMPLE: BROADCAST PROTOCOLS

Broadcast protocols (Esparza et al.'99) are dynamic & distributed collections of finite-state processes communicating via brodcasts and rendez-vous.



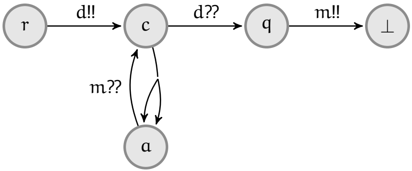
A configuration collects the local states of all processes. E.g., $s = \{c, r, c\}$, also denoted $\{c^2, r\}$.

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

We'll see later: The above protocol does not have infinite runs

EXAMPLE: BROADCAST PROTOCOLS

Broadcast protocols (Esparza et al.'99) are dynamic & distributed collections of finite-state processes communicating via brodcasts and rendez-vous.



A configuration collects the local states of all processes. E.g., $s = \{c, r, c\}$, also denoted $\{c^2, r\}$.

Steps: $\{c^2, q, r\} \xrightarrow{a} \{a^2, c, q, r\} \xrightarrow{a} \{a^4, q, r\} \xrightarrow{m} \{c^4, r, \perp\} \xrightarrow{d} \{c, q^4, \perp\}$

We'll see later: The above protocol does not have infinite runs

BRODCAST PROTOCOLS ARE WSTS

Ordering of configurations is multiset inclusion, e.g., $\{c, q\} \subseteq \{c^2, r, q\}$

Fact. Configurations $(\mathbb{N}^{\{r,c,a,q,\perp\}}, \subseteq)$ is a wqo.

Proof: this is exactly $(\mathbb{N}^5, \leqslant_\times)$

Fact. Brodcast protocols are monotonic TS

Proof Idea: assume $s_1 \subseteq t_1$ and consider all cases for a step $s_1 \to s_2$

Coro. Broadcast protocols are WSTS

Part 2

Verification of WSTS

Termination

Termination is the question, given a TS $\mathcal{S} = (S, \rightarrow, \ldots)$ and a state s_{init}, whether $\mathcal{S}$ has no infinite runs starting from s_{init}

Lem. [Finite Witnesses for Infinite Runs]
A WSTS $\mathcal{S}$ has an infinite run from s_{init} **iff** it has a finite run from s_{init} that is a good sequence.

Recall: $s_0, s_1, s_2, \ldots, s_n$ is good $\overset{\text{def}}{\Leftrightarrow}$ there exist $i < j$ s.t. $s_i \leqslant s_j$

$\Rightarrow$ one can decide Termination for a WSTS $\mathcal{S}$ by enumerating all finite runs from s_{init} until a good sequence is found.

NB: This requires some minimal effectiveness assumptions on the WSTS, e.g., that the ordering is decidable

Algorithm extends and allows deciding inevitability, finiteness, and regular simulation

Termination

Termination is the question, given a TS $\mathcal{S} = (S, \rightarrow, \ldots)$ and a state s_{init}, whether $\mathcal{S}$ has no infinite runs starting from s_{init}

Lem. [Finite Witnesses for Infinite Runs]
A WSTS $\mathcal{S}$ has an infinite run from s_{init} **iff** it has a finite run from s_{init} that is a good sequence.

Recall: $s_0, s_1, s_2, \ldots, s_n$ is good $\overset{\text{def}}{\Leftrightarrow}$ there exist $i < j$ s.t. $s_i \leqslant s_j$

$\Rightarrow$ one can decide Termination for a WSTS $\mathcal{S}$ by enumerating all finite runs from s_{init} until a good sequence is found.

NB: This requires some minimal effectiveness assumptions on the WSTS, e.g., that the ordering is decidable

Algorithm extends and allows deciding inevitability, finiteness, and regular simulation

Termination

Termination is the question, given a TS $\mathcal{S} = (S, \rightarrow, \ldots)$ and a state s_{init}, whether $\mathcal{S}$ has no infinite runs starting from s_{init}

Lem. [Finite Witnesses for Infinite Runs]
A WSTS $\mathcal{S}$ has an infinite run from s_{init} **iff** it has a finite run from s_{init} that is a good sequence.

Recall: $s_0, s_1, s_2, \ldots, s_n$ is good $\overset{\text{def}}{\Leftrightarrow}$ there exist $i < j$ s.t. $s_i \leqslant s_j$

$\Rightarrow$ one can decide Termination for a WSTS $\mathcal{S}$ by enumerating all finite runs from s_{init} until a good sequence is found.

NB: This requires some minimal effectiveness assumptions on the WSTS, e.g., that the ordering is decidable

Algorithm extends and allows deciding inevitability, finiteness, and regular simulation

Coverability

Coverability is the question, given $\mathcal{S} = (S, \to, \ldots)$, a state s_{init} and a target state t, whether $\mathcal{S}$ has a run $s_{init} \to s_1 \to s_2 \ldots \to s_n$ with $s_n \geqslant t$.

This is equivalent to having a pseudorun $s_{init}, s_1, \ldots, s_n$ with $s_n \geqslant t$, where a pseudorun is a sequence $s_0, s_1, \ldots$ such that for all $i > 0$, there is a step $s_{i-1} \to t_i$ with $t_i \geqslant s_i$.

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a pseudorun $s_{init}, \ldots, s_n$ covering t **iff** it has a minimal pseudorun from some $s_0 \leqslant s_{init}$ to t that is a bad sequence in reverse.
NB. a pseudorun $s_0, \ldots, s_n$ is minimal $\overset{\text{def}}{\Leftrightarrow}$ for all $0 \leqslant i < n$, s_i is a minimal (pseudo) predecessor of s_{i+1}.

$\Rightarrow$ one can decide Coverability by enumerating all pseudoruns ending in t (hence backward chaining) that are minimal and bad sequences in reverse.

Coverability

Coverability is the question, given $\mathcal{S} = (S, \to, \ldots)$, a state s_{init} and a target state t, whether $\mathcal{S}$ has a run $s_{init} \to s_1 \to s_2 \ldots \to s_n$ with $s_n \geqslant t$.

This is equivalent to having a pseudorun $s_{init}, s_1, \ldots, s_n$ with $s_n \geqslant t$, where a pseudorun is a sequence $s_0, s_1, \ldots$ such that for all $i > 0$, there is a step $s_{i-1} \to t_i$ with $t_i \geqslant s_i$.

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a pseudorun $s_{init}, \ldots, s_n$ covering t **iff** it has a minimal pseudorun from some $s_0 \leqslant s_{init}$ to t that is a bad sequence in reverse.

NB. a pseudorun $s_0, \ldots, s_n$ is minimal $\overset{\text{def}}{\Leftrightarrow}$ for all $0 \leqslant i < n$, s_i is a minimal (pseudo) predecessor of s_{i+1}.

$\Rightarrow$ one can decide Coverability by enumerating all pseudoruns ending in t (hence backward chaining) that are minimal and bad sequences in reverse.

COVERABILITY

Coverability is the question, given $\mathcal{S} = (S, \rightarrow, \ldots)$, a state s_{init} and a target state t, whether $\mathcal{S}$ has a run $s_{init} \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_n$ with $s_n \geqslant t$.

This is equivalent to having a pseudorun $s_{init}, s_1, \ldots, s_n$ with $s_n \geqslant t$, where a pseudorun is a sequence $s_0, s_1, \ldots$ such that for all $i > 0$, there is a step $s_{i-1} \rightarrow t_i$ with $t_i \geqslant s_i$.

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a pseudorun $s_{init}, \ldots, s_n$ covering t **iff** it has a minimal pseudorun from some $s_0 \leqslant s_{init}$ to t that is a bad sequence in reverse.

NB. a pseudorun $s_0, \ldots, s_n$ is minimal $\overset{\text{def}}{\Leftrightarrow}$ for all $0 \leqslant i < n$, s_i is a minimal (pseudo) predecessor of s_{i+1}.

$\Rightarrow$ one can decide Coverability by enumerating all pseudoruns ending in t (hence backward chaining) that are minimal and bad sequences in reverse.

Coverability

Coverability is the question, given $\mathcal{S} = (S, \rightarrow, \ldots)$, a state s_{init} and a target state t, whether $\mathcal{S}$ has a run $s_{\mathit{init}} \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_n$ with $s_n \geqslant t$.

This is equivalent to having a pseudorun $s_{\mathit{init}}, s_1, \ldots, s_n$ with $s_n \geqslant t$, where a pseudorun is a sequence $s_0, s_1, \ldots$ such that for all $i > 0$, there is a step $s_{i-1} \rightarrow t_i$ with $t_i \geqslant s_i$.

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a pseudorun $s_{\mathit{init}}, \ldots, s_n$ covering t **iff** it has a minimal pseudorun from some $s_0 \leqslant s_{\mathit{init}}$ to t that is a bad sequence in reverse.

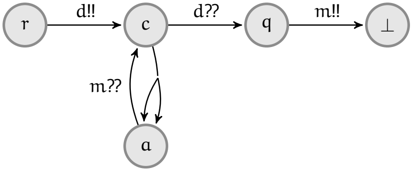
NB. a pseudorun $s_0, \ldots, s_n$ is minimal $\overset{\text{def}}{\Leftrightarrow}$ for all $0 \leqslant i < n$, s_i is a minimal (pseudo) predecessor of s_{i+1}.

$\Rightarrow$ one can decide Coverability by enumerating all pseudoruns ending in t (hence backward chaining) that are minimal and bad sequences in reverse.

Part 3

Bounding Running Time

Broadcast Protocols and Termination



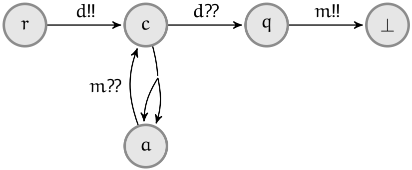
This broadcast protocol terminates: all its runs are bad sequences, hence are finite

Proof. Assume $s_0 \to s_1 \to \cdots \to s_n$ and pick two positions $i < j$.
Write $s_i = \{a^{n_1}, c^{n_2}, q^{n_3}, r^{n_4}, \bot^*\}$, and $s_j = \{a^{n'_1}, c^{n'_2}, q^{n'_3}, r^{n'_4}, \bot^*\}$.

– if $s_i \xrightarrow{+} s_j$ uses only spawn steps then $n'_2 < n_2$,
– if a m and no d have been broadcast, then $n'_3 < n_3$,
– if a d has been broadcast, and then $n'_4 < n_4$.

In all cases, $s_i \not\subseteq s_j$. QED

Broadcast Protocols and Termination



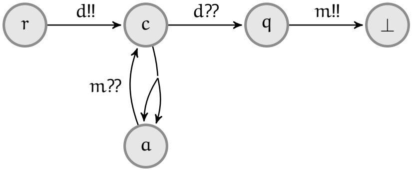
This broadcast protocol terminates: all its runs are bad sequences, hence are finite

Proof. Assume $s_0 \to s_1 \to \cdots \to s_n$ and pick two positions $i < j$.
Write $s_i = \{a^{n_1}, c^{n_2}, q^{n_3}, r^{n_4}, \bot^*\}$, and $s_j = \{a^{n'_1}, c^{n'_2}, q^{n'_3}, r^{n'_4}, \bot^*\}$.

– if $s_i \xrightarrow{+} s_j$ uses only spawn steps then $n'_2 < n_2$,
– if a m and no d have been broadcast, then $n'_3 < n_3$,
– if a d has been broadcast, and then $n'_4 < n_4$.

In all cases, $s_i \not\subseteq s_j$. QED

Broadcast Protocols and Termination



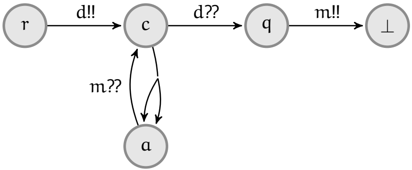
This broadcast protocol terminates: all its runs are bad sequences, hence are finite

Proof. Assume $s_0 \to s_1 \to \cdots \to s_n$ and pick two positions $i < j$. Write $s_i = \{a^{n_1}, c^{n_2}, q^{n_3}, r^{n_4}, \bot^*\}$, and $s_j = \{a^{n'_1}, c^{n'_2}, q^{n'_3}, r^{n'_4}, \bot^*\}$.

– if $s_i \xrightarrow{+} s_j$ uses only spawn steps then $n'_2 < n_2$,
– if a m and no d have been broadcast, then $n'_3 < n_3$,
– if a d has been broadcast, and then $n'_4 < n_4$.

In all cases, $s_i \not\subseteq s_j$. QED

Broadcast Protocols Take Their Time

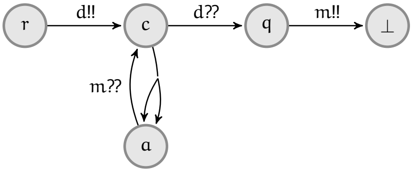


"Doubling" run: $\{c^{n}, q, (\bot^*)\} \xrightarrow{a^n} \{a^{2n}, q, (\bot^*)\} \xrightarrow{m} \{c^{2n}, (\bot^*)\}$

Building up: $\{c^{2^0}, q^{n}, r\} \xrightarrow{a^{2^0} m} \{c^{2^1}, q^{n-1}, r\} \xrightarrow{a^{2^1} m} \{c^{2^2}, q^{n-2}, r\} \to$
$\cdots \to \{c^{2^{n-1}}, q, r\} \xrightarrow{a^{2^{n-1}} m} \{c^{2^n}, r\} \xrightarrow{d} \{c^{2^0}, q^{2^n}\}$

Then: $\{c, q, r^{n}\} \xrightarrow{*} \{c, q^{2^n}, r^{n-1}\} \xrightarrow{*} \{c, q^{\text{tower}(n)}\}$

BROADCAST PROTOCOLS TAKE THEIR TIME

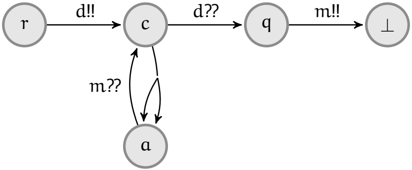


"Doubling" run: $\{c^{n}, q, (\bot^*)\} \xrightarrow{a^n} \{a^{2n}, q, (\bot^*)\} \xrightarrow{m} \{c^{2n}, (\bot^*)\}$

Building up: $\{c^{2^0}, q^{n}, r\} \xrightarrow{a^{2^0} m} \{c^{2^1}, q^{n-1}, r\} \xrightarrow{a^{2^1} m} \{c^{2^2}, q^{n-2}, r\} \rightarrow \cdots \rightarrow \{c^{2^{n-1}}, q, r\} \xrightarrow{a^{2^{n-1}} m} \{c^{2^n}, r\} \xrightarrow{d} \{c^{2^0}, q^{2^n}\}$

Then: $\{c, q, r^{n}\} \xrightarrow{*} \{c, q^{2^n}, r^{n-1}\} \xrightarrow{*} \{c, q^{\text{tower}(n)}\}$

Broadcast Protocols Take Their Time



"Doubling" run: $\{c^{n}, q, (\bot^*)\} \xrightarrow{a^n} \{a^{2n}, q, (\bot^*)\} \xrightarrow{m} \{c^{2n}, (\bot^*)\}$

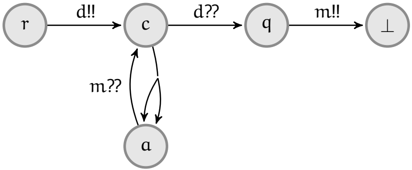
Building up: $\{c^{2^0}, q^{n}, r\} \xrightarrow{a^{2^0} m} \{c^{2^1}, q^{n-1}, r\} \xrightarrow{a^{2^1} m} \{c^{2^2}, q^{n-2}, r\} \to$

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

Then: $\{c, q, r^{n}\} \xrightarrow{*} \{c, q^{2^n}, r^{n-1}\} \xrightarrow{*} \{c, q^{\text{tower}(n)}\}$

where $\text{tower}(n) \stackrel{\text{def}}{=} \left.2^{2^{\cdot^{\cdot^{2}}}}\right\} n \text{ times}$

Broadcast Protocols Take Their Time



"Doubling" run: $\{c^n, q, (\bot^*)\} \xrightarrow{a^n} \{a^{2n}, q, (\bot^*)\} \xrightarrow{m} \{c^{2n}, (\bot^*)\}$

Building up: $\{c^{2^0}, q^n, r\} \xrightarrow{a^{2^0} m} \{c^{2^1}, q^{n-1}, r\} \xrightarrow{a^{2^1} m} \{c^{2^2}, q^{n-2}, r\} \to$
$\cdots \to \{c^{2^{n-1}}, q, r\} \xrightarrow{a^{2^{n-1}} m} \{c^{2^n}, r\} \xrightarrow{d} \{c^{2^0}, q^{2^n}\}$

Then: $\{c, q, r^n\} \xrightarrow{*} \{c, q^{2^n}, r^{n-1}\} \xrightarrow{*} \{c, q^{\text{tower}(n)}\}$

⇒ Runs of terminating systems may have nonelementary lengths

⇒ Running time of termination verification algorithm is not elementary (for broadcast protocols)

Complexity Analysis?

When analyzing the termination algorithm, the main question is "how long can a bad sequence be?"

WQO-theory only says that a bad sequence is finite

Over $(\mathbb{N}^k, \leqslant_\times)$, one can find arbitrarily long bad sequences:

— 999, 998, . . . , 1, 0

— $(2,2)$, $(2,1)$, $(2,0)$, $(1,999)$, . . . , $(1,0)$, $(0,999999999)$, . . .

Two tricks: unbounded start element, or unbounded increase in a step

The runs of a broadcast protocol don't play these tricks!

COMPLEXITY ANALYSIS?

When analyzing the termination algorithm, the main question is "how long can a bad sequence be?"

WQO-theory only says that a bad sequence is finite

Over $(\mathbb{N}^k, \leqslant_\times)$, one can find arbitrarily long bad sequences:

— 999, 998, . . . , 1, 0

— (2,2), (2,1), (2,0), (1,999), . . . , (1,0), (0,999999999), . . .

Two tricks: unbounded start element, or unbounded increase in a step

The runs of a broadcast protocol don't play these tricks!

COMPLEXITY ANALYSIS?

When analyzing the termination algorithm, the main question is "how long can a bad sequence be?"

WQO-theory only says that a bad sequence is finite

Over $(\mathbb{N}^k, \leqslant_\times)$, one can find arbitrarily long bad sequences:

— $999, 998, \ldots, 1, 0$

— $(2,2), (2,1), (2,0), (1,999), \ldots, (1,0), (0,999999999), \ldots$

Two tricks: unbounded start element, or unbounded increase in a step

The runs of a broadcast protocol don't play these tricks!

Complexity Analysis?

When analyzing the termination algorithm, the main question is "how long can a bad sequence be?"

WQO-theory only says that a bad sequence is finite

Over $(\mathbb{N}^k, \leqslant_\times)$, one can find arbitrarily long bad sequences:

— 999, 998, . . . , 1, 0

— $(2,2)$, $(2,1)$, $(2,0)$, $(1,999)$, . . . , $(1,0)$, $(0,999999999)$, . . .

Two tricks: unbounded start element, or unbounded increase in a step

The runs of a broadcast protocol don't play these tricks!

Controlled Bad Sequences

Def. A control is a pair of $n_0 \in \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$.

Def. A sequence $x_0, x_1, \ldots$ is controlled $\overset{\text{def}}{\Leftrightarrow} |x_i| \leqslant g^i(n_0)$ for all $i = 0, 1, \ldots$

Fact. For a fixed wqo $(A, \leqslant, |.|)$ and control (n_0, g), there is a bound on the length of controlled bad sequences.

Write $L_{g,A}(n_0)$ for this maximum length.

Length Function Theorem for $(\mathbb{N}^k, \leqslant_\times)$:

— $L_{g,\mathbb{N}^k}(n_0) \leqslant g^{\omega^k}(n_0)$

— $L_{g,\mathbb{N}^k}$ is in $\mathscr{F}_{k+m-1}$ for g in $\mathscr{F}_m$ [McAloon'84, Figueira^{2}SS'11]

(more later on Fast-Growing Hierarchy)

Controlled Bad Sequences

Def. A control is a pair of $n_0 \in \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$.

Def. A sequence $x_0, x_1, \ldots$ is controlled $\stackrel{\text{def}}{\Leftrightarrow} |x_i| \leqslant g^i(n_0)$ for all $i = 0, 1, \ldots$

Fact. For a fixed wqo $(A, \leqslant, |.|)$ and control (n_0, g), there is a bound on the length of controlled bad sequences.

Write $L_{g,A}(n_0)$ for this maximum length.

Length Function Theorem for $(\mathbb{N}^k, \leqslant_\times)$:

— $L_{g,\mathbb{N}^k}(n_0) \leqslant g^{\omega^k}(n_0)$

— $L_{g,\mathbb{N}^k}$ is in $\mathscr{F}_{k+m-1}$ for g in $\mathscr{F}_m$ [McAloon'84, Figueira[2]SS'11]

(more later on Fast-Growing Hierarchy)

CONTROLLED BAD SEQUENCES

Def. A control is a pair of $n_0 \in \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$.

Def. A sequence $x_0, x_1, \ldots$ is controlled $\overset{\text{def}}{\Leftrightarrow}$ $|x_i| \leqslant g^i(n_0)$ for all $i = 0, 1, \ldots$

Fact. For a fixed wqo $(A, \leqslant, |.|)$ and control (n_0, g), there is a bound on the length of controlled bad sequences.

Write $L_{g,A}(n_0)$ for this maximum length.

Length Function Theorem for $(\mathbb{N}^k, \leqslant_\times)$:

— $L_{g,\mathbb{N}^k}(n_0) \leqslant g^{\omega^k}(n_0)$

— $L_{g,\mathbb{N}^k}$ is in $\mathscr{F}_{k+m-1}$ for g in $\mathscr{F}_m$ [McAloon'84, Figueira[2]SS'11]

(more later on Fast-Growing Hierarchy)

APPLYING TO BROADCAST PROTOCOLS

Fact. The runs explored by the Termination algorithm are controlled with $|s_{init}|$ and $Succ : \mathbb{N} \to \mathbb{N}$.

$\Rightarrow$ Time/space bound in $\mathscr{F}_{k-1}$ for broadcast protocols with k states, and in $\mathscr{F}_{\omega}$ when k is not fixed.

Fact. The minimal pseudoruns explored by the backward-chaining Coverability algorithm are controlled by $|t|$ and *Succ*.

$\Rightarrow \cdots$ *same upper bounds* $\cdots$

This is a general situation:
— WSTS model (or WQO-based algorithm) provides g
— WQO-theory provides bounds for $L_{A,g}$
— Translates as complexity upper bounds for WQO-based algorithm

APPLYING TO BROADCAST PROTOCOLS

Fact. The runs explored by the Termination algorithm are controlled with $|s_{init}|$ and *Succ* : $\mathbb{N} \to \mathbb{N}$.

$\Rightarrow$ Time/space bound in $\mathscr{F}_{k-1}$ for broadcast protocols with k states, and in $\mathscr{F}_{\omega}$ when k is not fixed.

Fact. The minimal pseudoruns explored by the backward-chaining Coverability algorithm are controlled by $|t|$ and *Succ*.

$\Rightarrow \cdots$ *same upper bounds* $\cdots$

This is a general situation:
— WSTS model (or WQO-based algorithm) provides g
— WQO-theory provides bounds for $L_{A,g}$
— Translates as complexity upper bounds for WQO-based algorithm

Applying to Broadcast Protocols

Fact. The runs explored by the Termination algorithm are controlled with $|s_{init}|$ and $Succ : \mathbb{N} \to \mathbb{N}$.

$\Rightarrow$ Time/space bound in $\mathscr{F}_{k-1}$ for broadcast protocols with k states, and in $\mathscr{F}_{\omega}$ when k is not fixed.

Fact. The minimal pseudoruns explored by the backward-chaining Coverability algorithm are controlled by $|t|$ and *Succ*.

$\Rightarrow \cdots$ *same upper bounds* $\cdots$

This is a general situation:
— WSTS model (or WQO-based algorithm) provides g
— WQO-theory provides bounds for $L_{A,g}$
— Translates as complexity upper bounds for WQO-based algorithm

The Fast-Growing Hierarchy

An ordinal-indexed family $(F_\alpha)_{\alpha \in Ord}$ of functions $\mathbb{N} \to \mathbb{N}$

$$F_0(x) \stackrel{\text{def}}{=} x+1 \qquad F_{\alpha+1}(x) \stackrel{\text{def}}{=} \overbrace{F_\alpha(F_\alpha(\dots F_\alpha}^{x+1}(x)\dots))$$

$$F_\omega(x) \stackrel{\text{def}}{=} F_{x+1}(x)$$

gives $F_1(x) \sim 2x$, $F_2(x) \sim 2^x$, $F_3(x) \sim \text{tower}(x)$ and $F_\omega(x) \sim \text{Ackermann}(x)$, the first F_α that is not primitive recursive.

$F_\lambda(x) \stackrel{\text{def}}{=} F_{\lambda_x}(x)$ for λ a limit ordinal with a fundamental sequence $\lambda_0 < \lambda_1 < \lambda_2 < \cdots < \lambda$.

E.g. $F_{\omega^2}(x) = F_{\omega\cdot(x+1)}(x) = F_{\omega\cdot x+x+1}(x) = \overbrace{F_{\omega\cdot x+x}(F_{\omega\cdot x+x}(..F_{\omega\cdot x+x}}^{x+1}(x)..))$

$\mathscr{F}_\alpha \stackrel{\text{def}}{=}$ all functions computable in time $F_\alpha^{O(1)}$ (very robust).

The Fast-Growing Hierarchy

An ordinal-indexed family $(F_\alpha)_{\alpha\in Ord}$ of functions $\mathbb{N}\to\mathbb{N}$

$F_0(x) \stackrel{\text{def}}{=} x+1 \qquad F_{\alpha+1}(x) \stackrel{\text{def}}{=} \overbrace{F_\alpha(F_\alpha(\ldots F_\alpha}^{x+1}(x)\ldots))$

$F_\omega(x) \stackrel{\text{def}}{=} F_{x+1}(x)$

gives $F_1(x)\sim 2x$, $F_2(x)\sim 2^x$, $F_3(x)\sim \text{tower}(x)$ and $F_\omega(x)\sim \text{Ackermann}(x)$, the first F_α that is not primitive recursive.

$F_\lambda(x) \stackrel{\text{def}}{=} F_{\lambda_x}(x)$ for λ a limit ordinal with a fundamental sequence $\lambda_0<\lambda_1<\lambda_2<\cdots<\lambda$.

E.g. $F_{\omega^2}(x) = F_{\omega\cdot(x+1)}(x) = F_{\omega\cdot x+x+1}(x) = \overbrace{F_{\omega\cdot x+x}(F_{\omega\cdot x+x}(..F_{\omega\cdot x+x}}^{x+1}(x)..))$

$\mathscr{F}_\alpha \stackrel{\text{def}}{=}$ all functions computable in time $F_\alpha^{O(1)}$ (very robust).

The Fast-Growing Hierarchy

An ordinal-indexed family $(F_\alpha)_{\alpha \in Ord}$ of functions $\mathbb{N} \to \mathbb{N}$

$$F_0(x) \stackrel{\text{def}}{=} x+1 \qquad F_{\alpha+1}(x) \stackrel{\text{def}}{=} \overbrace{F_\alpha(F_\alpha(\dots F_\alpha}^{x+1}(x)\dots))$$
$$F_\omega(x) \stackrel{\text{def}}{=} F_{x+1}(x)$$

gives $F_1(x) \sim 2x$, $F_2(x) \sim 2^x$, $F_3(x) \sim \text{tower}(x)$ and $F_\omega(x) \sim \text{ACKERMANN}(x)$, the first F_α that is not primitive recursive.

$F_\lambda(x) \stackrel{\text{def}}{=} F_{\lambda_x}(x)$ for λ a limit ordinal with a fundamental sequence $\lambda_0 < \lambda_1 < \lambda_2 < \cdots < \lambda$.

E.g. $F_{\omega^2}(x) = F_{\omega\cdot(x+1)}(x) = F_{\omega\cdot x+x+1}(x) = \overbrace{F_{\omega\cdot x+x}(F_{\omega\cdot x+x}(..F_{\omega\cdot x+x}}^{x+1}(x)..))$

$\mathscr{F}_\alpha \stackrel{\text{def}}{=}$ all functions computable in time $F_\alpha^{O(1)}$ (very robust).

More Length Function Theorems

For finite words with embedding, L_{Σ^*} is in $\mathscr{F}_{\omega^{|\Sigma|-1}}$, and in $\mathscr{F}_{\omega^\omega}$ when alphabet is not fixed [Cichon'98, SS'11]. Applies e.g. to lossy channel systems.

For sequences over $\mathbb{N}^k$ with embedding, $L_{(\mathbb{N}^k)^*}$ is in $\mathscr{F}_{\omega^{\omega^k}}$, and in $\mathscr{F}_{\omega^{\omega^\omega}}$ when k is not fixed [SS'11]. Applies e.g. to timed-arc Petri nets.

For finite words with priority ordering, L_{Σ^*} is in $\mathscr{F}_{\varepsilon_0}$ [HaaseSS'13]. Applies e.g. to priority channel systems.

Bottom line: we can provide definite complexity upper bounds for WQO-based algorithms

Some research goals: more varied/complex wqos, less crude notion of controlled sequences, analysis of complex algorithms

More Length Function Theorems

For finite words with embedding, L_{Σ^*} is in $\mathscr{F}_{\omega^{|\Sigma|-1}}$, and in $\mathscr{F}_{\omega^\omega}$ when alphabet is not fixed [Cichon'98, SS'11]. Applies e.g. to lossy channel systems.

For sequences over $\mathbb{N}^k$ with embedding, $L_{(\mathbb{N}^k)^*}$ is in $\mathscr{F}_{\omega^{\omega^k}}$, and in $\mathscr{F}_{\omega^{\omega^\omega}}$ when k is not fixed [SS'11]. Applies e.g. to timed-arc Petri nets.

For finite words with priority ordering, L_{Σ^*} is in $\mathscr{F}_{\varepsilon_0}$ [HaaseSS'13]. Applies e.g. to priority channel systems.

Bottom line: we can provide definite complexity upper bounds for WQO-based algorithms

Some research goals: more varied/complex wqos, less crude notion of controlled sequences, analysis of complex algorithms

More Length Function Theorems

For finite words with embedding, L_{Σ^*} is in $\mathscr{F}_{\omega^{|\Sigma|-1}}$, and in $\mathscr{F}_{\omega^\omega}$ when alphabet is not fixed [Cichon'98, SS'11]. Applies e.g. to lossy channel systems.

For sequences over $\mathbb{N}^k$ with embedding, $L_{(\mathbb{N}^k)^*}$ is in $\mathscr{F}_{\omega^{\omega^k}}$, and in $\mathscr{F}_{\omega^{\omega^\omega}}$ when k is not fixed [SS'11]. Applies e.g. to timed-arc Petri nets.

For finite words with priority ordering, L_{Σ^*} is in $\mathscr{F}_{\varepsilon_0}$ [HaaseSS'13]. Applies e.g. to priority channel systems.

Bottom line: we can provide definite complexity upper bounds for WQO-based algorithms

Some research goals: more varied/complex wqos, less crude notion of controlled sequences, analysis of complex algorithms

More Length Function Theorems

For finite words with embedding, L_{Σ^*} is in $\mathscr{F}_{\omega^{|\Sigma|-1}}$, and in $\mathscr{F}_{\omega^\omega}$ when alphabet is not fixed [Cichon'98, SS'11]. Applies e.g. to lossy channel systems.

For sequences over $\mathbb{N}^k$ with embedding, $L_{(\mathbb{N}^k)^*}$ is in $\mathscr{F}_{\omega^{\omega^k}}$, and in $\mathscr{F}_{\omega^{\omega^\omega}}$ when k is not fixed [SS'11]. Applies e.g. to timed-arc Petri nets.

For finite words with priority ordering, L_{Σ^*} is in $\mathscr{F}_{\varepsilon_0}$ [HaaseSS'13]. Applies e.g. to priority channel systems.

Bottom line: we can provide definite complexity upper bounds for WQO-based algorithms

Some research goals: more varied/complex wqos, less crude notion of controlled sequences, analysis of complex algorithms

Part 4

Proving Lower Bounds

What About Lower Bounds?

Q. Are the upper bounds for Termination and Coverability optimal?

In the case of broadcast protocols:

The upper bound is tight for the algorithms we presented

But there may exist better algorithms (as with VASS, e.g.)

One can prove that the Termination and Coverability problems are F_ω-hard, hence F_ω-complete, for broadcast protocols [S'10]

and F_{ω^ω}-complete for lossy channel systems [ChambartS'08], $F_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12], F_{ϵ_0}-complete for priority channel systems [HaaseSS'13]

These results/characterizations have applications outside verification: WSTS models are often used for decidability (or hardness) of problems in logic.

What About Lower Bounds?

Q. Are the upper bounds for Termination and Coverability optimal?

In the case of broadcast protocols:

The upper bound is tight for the algorithms we presented

But there may exist better algorithms (as with VASS, e.g.)

One can prove that the Termination and Coverability problems are F_ω-hard, hence F_ω-complete, for broadcast protocols [S'10]

and F_{ω^ω}-complete for lossy channel systems [ChambartS'08], $F_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12], F_{ϵ_0}-complete for priority channel systems [HaaseSS'13]

These results/characterizations have applications outside verification: WSTS models are often used for decidability (or hardness) of problems in logic.

What About Lower Bounds?

Q. Are the upper bounds for Termination and Coverability optimal?

In the case of broadcast protocols:

The upper bound is tight for the algorithms we presented

But there may exist better algorithms (as with VASS, e.g.)

One can prove that the Termination and Coverability problems are F_ω-hard, hence F_ω-complete, for broadcast protocols [S'10]

and F_{ω^ω}-complete for lossy channel systems [ChambartS'08], $F_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12], F_{ϵ_0}-complete for priority channel systems [HaaseSS'13]

These results/characterizations have applications outside verification: WSTS models are often used for decidability (or hardness) of problems in logic.

WHAT ABOUT LOWER BOUNDS?

Q. Are the upper bounds for Termination and Coverability optimal?

In the case of broadcast protocols:

The upper bound is tight for the algorithms we presented

But there may exist better algorithms (as with VASS, e.g.)

One can prove that the Termination and Coverability problems are F_ω-hard, hence F_ω-complete, for broadcast protocols [S'10]

and F_{ω^ω}-complete for lossy channel systems [ChambartS'08], $F_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12], F_{ϵ_0}-complete for priority channel systems [HaaseSS'13]

These results/characterizations have applications outside verification: WSTS models are often used for decidability (or hardness) of problems in logic.

What About Lower Bounds?

Q. Are the upper bounds for Termination and Coverability optimal?

In the case of broadcast protocols:

The upper bound is tight for the algorithms we presented

But there may exist better algorithms (as with VASS, e.g.)

One can prove that the Termination and Coverability problems are F_ω-hard, hence F_ω-complete, for broadcast protocols [S'10]

and F_{ω^ω}-complete for lossy channel systems [ChambartS'08], $F_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12], F_{ϵ_0}-complete for priority channel systems [HaaseSS'13]

These results/characterizations have applications outside verification: WSTS models are often used for decidability (or hardness) of problems in logic.

Proving F_α-Hardness

The four hardness results we just mentioned have all been proved using the same techniques:

One shows how the WSTS model can weakly compute F_α and its inverse F_α^{-1}.

Encode initial ordinals in $(S, \leqslant)$ & implement Hardy computations in $\mathcal{S}$.

Hardy computations: $(\alpha + 1, x) \mapsto (\alpha, x + 1)$ and $(\lambda, x) \mapsto (\lambda_x, x)$.

Main technical issue: robustness

— One easily guarantee $s \leqslant t \Rightarrow \alpha(s) \leqslant \alpha(t)$ but this does not guarantee $F_{\alpha(s)}(x) \leqslant F_{\alpha(t)}(x)$ required for weak computation of F_α.

— We need $s \leqslant t \Rightarrow \alpha(s) \sqsubseteq \alpha(t)$, using an ad-hoc stronger relation.

Proving F_α-Hardness

The four hardness results we just mentioned have all been proved using the same techniques:

One shows how the WSTS model can weakly compute F_α and its inverse F_α^{-1}.

Encode initial ordinals in $(S, \leqslant)$ & implement Hardy computations in $\mathcal{S}$.
Hardy computations: $(\alpha + 1, x) \mapsto (\alpha, x + 1)$ and $(\lambda, x) \mapsto (\lambda_x, x)$.

Main technical issue: robustness

— One easily guarantee $s \leqslant t \Rightarrow \alpha(s) \leqslant \alpha(t)$ but this does not guarantee $F_{\alpha(s)}(x) \leqslant F_{\alpha(t)}(x)$ required for weak computation of F_α.

— We need $s \leqslant t \Rightarrow \alpha(s) \sqsubseteq \alpha(t)$, using an ad-hoc stronger relation.

Proving F_α-Hardness

The four hardness results we just mentioned have all been proved using the same techniques:

One shows how the WSTS model can weakly compute F_α and its inverse F_α^{-1}.

Encode initial ordinals in $(S, \leqslant)$ & implement Hardy computations in $\mathcal{S}$.

Hardy computations: $(\alpha + 1, x) \mapsto (\alpha, x + 1)$ and $(\lambda, x) \mapsto (\lambda_x, x)$.

Main technical issue: robustness

— One easily guarantee $s \leqslant t \Rightarrow \alpha(s) \leqslant \alpha(t)$ but this does not guarantee $F_{\alpha(s)}(x) \leqslant F_{\alpha(t)}(x)$ required for weak computation of F_α.

— We need $s \leqslant t \Rightarrow \alpha(s) \sqsubseteq \alpha(t)$, using an ad-hoc stronger relation.

Concluding remarks

Complexity analysis of WSTS models is possible

WSTS models are powerful, i.e., very expressive

WSTS have applications outside verification

Join the fun! Technical details are lighter than it seems, see our lecture notes “Algorithmic aspects of wqo theory”

CONCLUDING REMARKS

Complexity analysis of WSTS models is possible

WSTS models are powerful, i.e., very expressive

WSTS have applications outside verification

Join the fun! Technical details are lighter than it seems, see our lecture notes “Algorithmic aspects of wqo theory”

THANK YOU FOR YOUR INTEREST