Simple algorithms and fast-growing complexity for well-structured systems

Philippe Schnoebelen

LSV; CNRS & ENS Paris-Saclay

Journées Montoises 2018

Based on joint work with Sylvain Schmitz, Pierre Chambart, Prateek Karandikar, Simon Halfon, K. Narayan Kumar, Alain Finkel, ..

Background & motivations

- Well-Structured Systems (WSTS) are a family of infinite-state models where safety, inevitability, etc., properties are decidable
- There, decidability relies on the fact that states are well-quasi-ordered and uses generic algorithms.
- WSTS invented by Finkel (1987), developed and popularized by Abdulla & Jonsson, Finkel & Schnoebelen, etc. (1996–2005).
- First used with counters, queues, gap-order constraints, etc.
- The family encompasses many kinds of models: distributed systems, counter systems, out-of-order memory, communication protocols, automata for logic, . . . (still growing).
- In 2017, WSTS recognized as a fundamental contribution by the Computer-Aided Verification community.

Outline of the Talk

- Part 0: **Basics of WQOs.**
- Part 1: **Basics of WSTS.**
- Part 2: **Verifying WSTS.**
- Part 3: **Assessing Complexity.**

Part 0

Basics of WQOs

QUASI-ORDERINGS

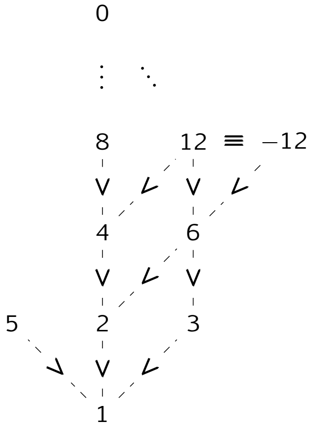
We consider orderings, like e.g.

$$(\mathbb{Z}, \leqslant) \qquad \begin{array}{c} \vdots \\ \vee \\ 2 \\ \vee \\ 1 \\ \vee \\ 0 \\ \vee \\ -1 \\ \vee \\ \vdots \end{array}$$

QUASI-ORDERINGS

We consider quasi-orderings, like e.g.

$(\mathbb{Z}, \leqslant_{\mathsf{div}})$

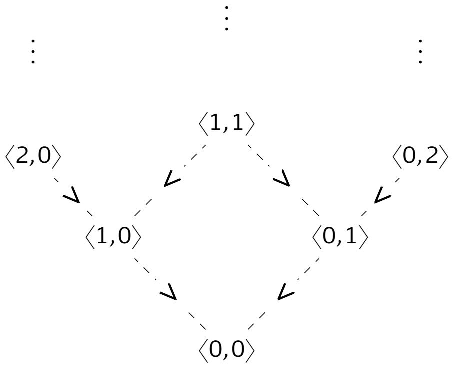


Quasi-orderings are more robust. If $(X, \leqslant)$ is an ordering, $(\mathcal{P}(X), \sqsubseteq)$ is in general not antisymmetric

QUASI-ORDERINGS

We consider quasi-orderings, like e.g.

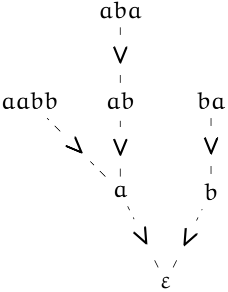
$$(\mathbb{N}^2, \leqslant_\times)$$



QUASI-ORDERINGS

We consider quasi-orderings, like e.g.

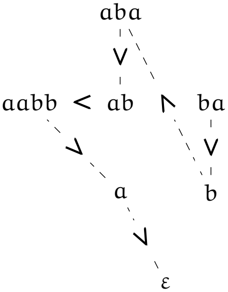
$$(A^*, \leqslant_{\text{pref}})$$



QUASI-ORDERINGS

We consider quasi-orderings, like e.g.

$(A^*, \leqslant_{\text{lex}})$

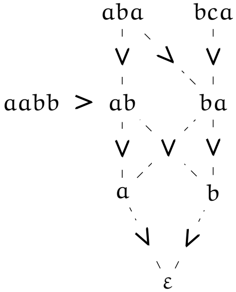


$(A^*, \leqslant_{\text{lex}})$ is a total/linear ordering that contains/extends $(A^*, \leqslant_{\text{pref}})$

QUASI-ORDERINGS

We consider quasi-orderings, like e.g.

$$(A^*, \leqslant_*)$$



$\leqslant_*$ is the subsequence/subword ordering. It extends $\leqslant_{\text{pref}}$

WELL-QUASI ORDERINGS (WQO)

A WQO is a quasi-ordering $(X, \leqslant)$

$(\mathbb{Z}, \leqslant)$

$(\mathbb{N}^2, \leqslant_\times)$

$(A^*, \leqslant_*)$

$(A^*, \leqslant_{\text{pref}})$

$(A^*, \leqslant_{\text{lex}})$

$(\mathbb{Z}, \leqslant_{\text{div}})$

WELL-QUASI ORDERINGS (WQO)

A WQO is a quasi-ordering $(X, \leqslant)$ that is well-founded

– WF: no infinite decreasing sequence $x_0 > x_1 > x_2 > \cdots$

$(\mathbb{Z}, \leqslant)$

$(\mathbb{N}^2, \leqslant_\times)$

$(A^*, \leqslant_*)$

$(A^*, \leqslant_{\text{pref}})$

$(A^*, \leqslant_{\text{lex}})$

$(\mathbb{Z}, \leqslant_{\text{div}})$

Well-quasi orderings (WQO)

A WQO is a quasi-ordering $(X,\leqslant)$ that is well-founded

– WF: no infinite decreasing sequence $x_0 > x_1 > x_2 > \cdots$

$(\mathbb{Z},\leqslant)$

$(\mathbb{N}^2,\leqslant_\times)$

$(A^*,\leqslant_*)$

$(A^*,\leqslant_{\mathsf{pref}})$

$(A^*,\leqslant_{\mathsf{lex}})$

$(\mathbb{Z},\leqslant_{\mathsf{div}})$

WELL-QUASI ORDERINGS (WQO)

A WQO is a quasi-ordering $(X, \leqslant)$ that is well-founded

– WF: no infinite decreasing sequence $x_0 > x_1 > x_2 > \cdots$

$(\mathbb{Z}, \leqslant)$ ✗

$(\mathbb{N}^2, \leqslant_\times)$

$(A^*, \leqslant_*)$

$(A^*, \leqslant_{\text{pref}})$

$(A^*, \leqslant_{\text{lex}})$ ✗

$(\mathbb{Z}, \leqslant_{\text{div}})$

WELL-QUASI ORDERINGS (WQO)

A WQO is a quasi-ordering $(X, \leqslant)$ that is well-founded and has the finite antichain property

– WF: no infinite decreasing sequence $x_0 > x_1 > x_2 > \cdots$

– FAC: no infinite set $\{x_0, x_1, x_2, \ldots\}$ of pairwise incomparable elements

$(\mathbb{Z}, \leqslant)$ ✗ $\quad (\mathbb{N}^2, \leqslant_\times)$

$(A^*, \leqslant_*)$

$(A^*, \leqslant_{\text{pref}})$ $\quad (A^*, \leqslant_{\text{lex}})$ ✗

$(\mathbb{Z}, \leqslant_{\text{div}})$

WELL-QUASI ORDERINGS (WQO)

A WQO is a quasi-ordering $(X, \leqslant)$ that is well-founded and has the finite antichain property

– WF: no infinite decreasing sequence $x_0 > x_1 > x_2 > \cdots$

– FAC: no infinite set $\{x_0, x_1, x_2, \ldots\}$ of pairwise incomparable elements

~~$(\mathbb{Z}, \leqslant)$~~

$(\mathbb{N}^2, \leqslant_\times)$

$(A^*, \leqslant_*)$

$(A^*, \leqslant_{\text{pref}})$

~~$(A^*, \leqslant_{\text{lex}})$~~

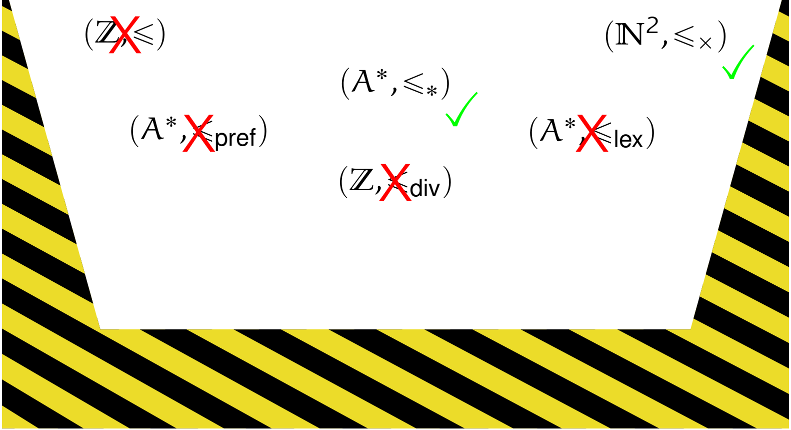
$(\mathbb{Z}, \leqslant_{\text{div}})$

WELL-QUASI ORDERINGS (WQO)

A WQO is a quasi-ordering $(X, \leqslant)$ that is well-founded and has the finite antichain property

– WF: no infinite decreasing sequence $x_0 > x_1 > x_2 > \cdots$

– FAC: no infinite set $\{x_0, x_1, x_2, \ldots\}$ of pairwise incomparable elements



Other WQO facts

Many characterizations: $(X, \leqslant)$ is wqo iff

- it is WF and FAC;
- every infinite sequence $x_0, x_1, x_2, \ldots$ is good, i.e. contains an increasing pair $x_i \leqslant x_j$ (for some $i < j$);
- every infinite sequence $x_0, x_1, x_2, \ldots$ is perfect, i.e. contains an infinite increasing subsequence $x_{i_0} \leqslant x_{i_1} \leqslant x_{i_2} \leqslant \cdots$ (with $i_0 < i_1 < i_2 < \cdots$);
- every linearization $(X, \leqslant')$ of $\leqslant$ is a wellorder.

Many ways to construct WQOs:

- Cartesian products, powersets, ..
- sequences, trees, graphs.

Part 1

Basics of WSTS

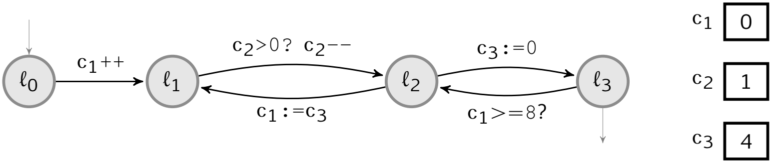
Well-structured systems

In program verification, wqo's appear prominently in well-structured systems (WSTS).

Def. A WSTS is a system $(S, \rightarrow, \leqslant)$ where

1. $(S, \rightarrow)$ with $\rightarrow \subseteq S \times S$ is a transition system
2. the set of states $(S, \leqslant)$ is wqo, and
3. the transition relation is compatible with the ordering (also called "monotonic"): $s \rightarrow t$ and $s \leqslant s'$ imply $s' \rightarrow t'$ for some $t' \geqslant t$

SOME WSTS'S: MONOTONIC COUNTER MACHINES



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

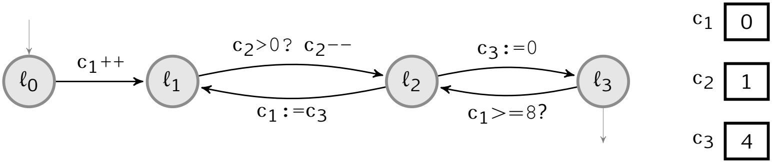
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \not\leqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3,\leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

Related models: Petri nets; vector additions systems; broadcast protocols; etc.

NB. Minsky (counter) machines have zero-tests, hence don't enjoy monotonicity

SOME WSTS'S: MONOTONIC COUNTER MACHINES



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

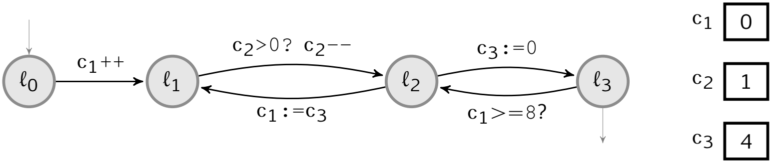
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \not\leqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3, \leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

Related models: Petri nets; vector additions systems; broadcast protocols; etc.

NB. Minsky (counter) machines have zero-tests, hence don't enjoy monotonicity

Some WSTS's: Monotonic Counter Machines



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

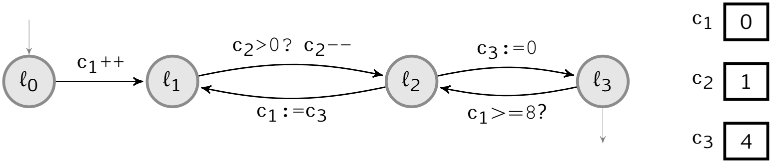
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \not\leqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3,\leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

Related models: Petri nets; vector additions systems; broadcast protocols; etc.

NB. Minsky (counter) machines have zero-tests, hence don't enjoy monotonicity

Some WSTS's: Monotonic Counter Machines



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

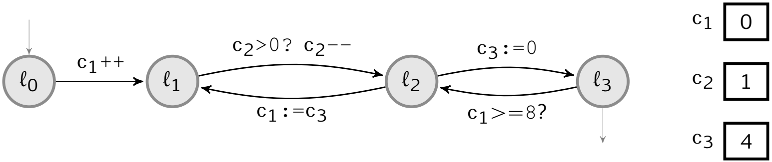
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \nleqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3,\leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

Related models: Petri nets; vector additions systems; broadcast protocols; etc.

NB. Minsky (counter) machines have zero-tests, hence don't enjoy monotonicity

Some WSTS's: Monotonic Counter Machines



A run of M: $(\ell_0,0,1,4) \to (\ell_1,1,1,4) \to (\ell_2,1,0,4) \to (\ell_3,1,0,0)$

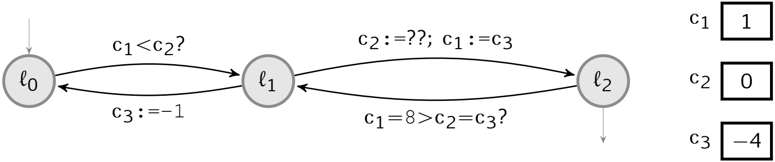
Ordering states: $(\ell_1,0,0,0) \leqslant (\ell_1,0,1,2)$ but $(\ell_1,0,0,0) \not\leqslant (\ell_2,0,1,2)$.
This is wqo as a product of wqo's: $(Loc,=) \times (\mathbb{N}^3,\leqslant_\times)$

Compatibility: easily checked when guards are upward-closed and assignments are monotonic functions of the variables.

Related models: Petri nets; vector additions systems; broadcast protocols; etc.

NB. Minsky (counter) machines have zero-tests, hence don't enjoy monotonicity

Some WSTS's: Integral Relational automata



Guards: comparisons between counters and constants

Updates: assignments with counter values, constants, & "??"

One does not use $\leqslant_\times$ to compare states!! Rather

$$(a_1,\ldots,a_k)\leqslant_{\text{sparse}}(b_1,\ldots,b_k)$$

$$\overset{\text{def}}{\Leftrightarrow} \forall i,j=1,\ldots,k: \big(a_i \leqslant a_j \text{ iff } b_i \leqslant b_j\big) \wedge \big(|a_i-a_j| \leqslant |b_i-b_j|\big)\,.$$

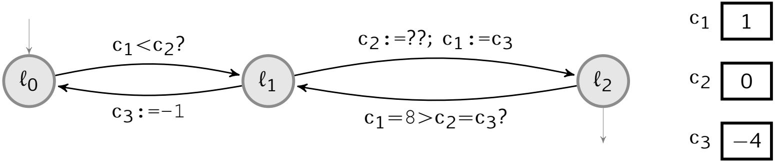
Fact. $(\mathbb{Z}^k,\leqslant_{\text{sparse}})$ is wqo

Compatibility: We use

$$(\ell,a_1,\ldots,a_k)\leqslant(\ell',b_1,\ldots,b_k) \overset{\text{def}}{\Leftrightarrow} \tag{1}$$

$$\ell=\ell' \wedge (a_1,\ldots,a_k,-1,8) \leqslant_{\text{sparse}} (b_1,\ldots,b_k,-1,8)\,.$$

Some WSTS's: Integral Relational automata



Guards: comparisons between counters and constants

Updates: assignments with counter values, constants, & "??"

One does not use $\leqslant_\times$ to compare states!! Rather

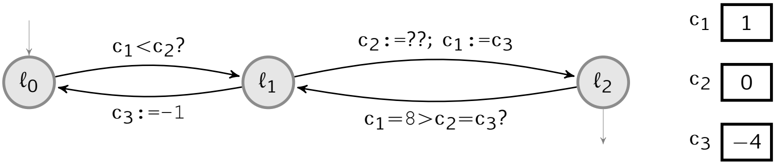
$$(a_1,\ldots,a_k) \leqslant_{\text{sparse}} (b_1,\ldots,b_k)$$
$$\overset{\text{def}}{\Leftrightarrow} \forall i,j = 1,\ldots,k : \left(a_i \leqslant a_j \text{ iff } b_i \leqslant b_j\right) \wedge \left(|a_i - a_j| \leqslant |b_i - b_j|\right).$$

Fact. $(\mathbb{Z}^k, \leqslant_{\text{sparse}})$ is wqo

Compatibility: We use

$$(\ell, a_1,\ldots,a_k) \leqslant (\ell', b_1,\ldots,b_k) \overset{\text{def}}{\Leftrightarrow} \quad (1)$$
$$\ell = \ell' \wedge (a_1,\ldots,a_k,-1,8) \leqslant_{\text{sparse}} (b_1,\ldots,b_k,-1,8).$$

Some WSTS's: Integral Relational automata



Guards: comparisons between counters and constants
Updates: assignments with counter values, constants, & "??"

One does not use $\leqslant_\times$ to compare states!! Rather

$$(a_1,\ldots,a_k) \leqslant_{\text{sparse}} (b_1,\ldots,b_k)$$

$$\overset{\text{def}}{\Leftrightarrow} \forall i,j = 1,\ldots,k : \big(a_i \leqslant a_j \text{ iff } b_i \leqslant b_j\big) \wedge \big(|a_i - a_j| \leqslant |b_i - b_j|\big)\,.$$

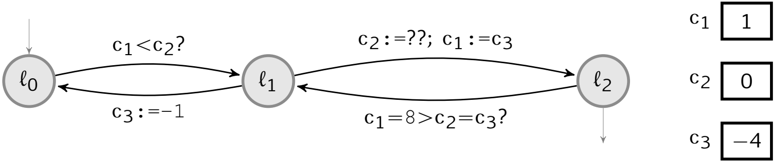
Fact. $(\mathbb{Z}^k, \leqslant_{\text{sparse}})$ is wqo

Compatibility: We use

$$(\ell, a_1,\ldots,a_k) \leqslant (\ell', b_1,\ldots,b_k) \overset{\text{def}}{\Leftrightarrow} \qquad (1)$$

$$\ell = \ell' \wedge (a_1,\ldots,a_k,-1,8) \leqslant_{\text{sparse}} (b_1,\ldots,b_k,-1,8)\,.$$

SOME WSTS'S: INTEGRAL RELATIONAL AUTOMATA



Guards: comparisons between counters and constants

Updates: assignments with counter values, constants, & "??"

One does not use $\leqslant_\times$ to compare states!! Rather

$$(a_1,\ldots,a_k)\leqslant_{\text{sparse}}(b_1,\ldots,b_k)$$

$$\overset{\text{def}}{\Leftrightarrow}\forall i,j=1,\ldots,k:\Big(a_i\leqslant a_j \text{ iff } b_i\leqslant b_j\Big)\wedge\Big(|a_i-a_j|\leqslant|b_i-b_j|\Big).$$

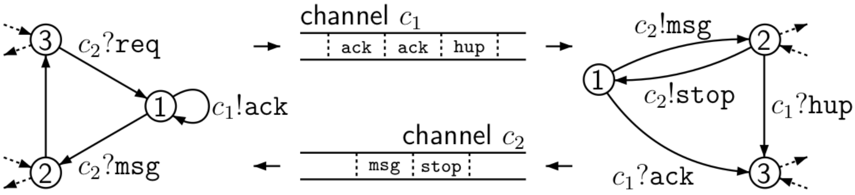
Fact. $(\mathbb{Z}^k,\leqslant_{\text{sparse}})$ is wqo

Compatibility: We use

$$(\ell,a_1,\ldots,a_k)\leqslant(\ell',b_1,\ldots,b_k)\overset{\text{def}}{\Leftrightarrow} \tag{1}$$

$$\ell=\ell'\wedge(a_1,\ldots,a_k,-1,8)\leqslant_{\text{sparse}}(b_1,\ldots,b_k,-1,8).$$

Some WSTS's: Lossy Channel Systems (LCS)



A state $s = (\ell_1, \ell_2, w_1, w_2)$ with $w_i \in A^*$.
E.g., $w_1 = \texttt{hup.ack.ack}$.

Reliable steps: $s \to_{\text{rel}} s'$ read in front of channels, write at end (FIFO)

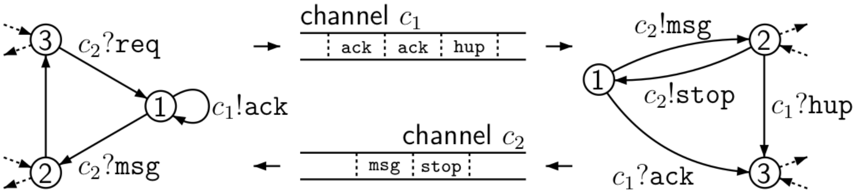
Lossy steps: messages may be lost nondeterministically

$$s \to s' \overset{\text{def}}{\Leftrightarrow} s \geqslant_* t \to_{\text{rel}} t' \geqslant_* s' \text{ for some } t, t' \in S$$

where $(S, \sqsubseteq)$ is the wqo $(Loc_1, =) \times (Loc_2, =) \times (A^*_{c_1}, \leqslant_*) \times (A^*_{c_2}, \leqslant_*)$

A model useful for concurrent protocols but also timed automata, metric temporal logic, products of modal logics, ...

SOME WSTS'S: LOSSY CHANNEL SYSTEMS (LCS)



A state $s = (\ell_1, \ell_2, w_1, w_2)$ with $w_i \in A^*$.
E.g., $w_1 = \texttt{hup.ack.ack}$.

Reliable steps: $s \rightarrow_{\text{rel}} s'$ read in front of channels, write at end (FIFO)

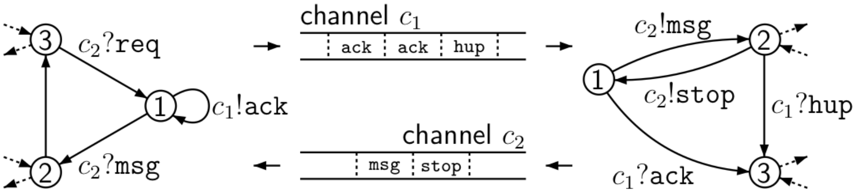
Lossy steps: messages may be lost nondeterministically

$$s \rightarrow s' \overset{\text{def}}{\Leftrightarrow} s \geqslant_* t \rightarrow_{\text{rel}} t' \geqslant_* s' \text{ for some } t, t' \in S$$

where $(S, \sqsubseteq)$ is the wqo $(Loc_1, =) \times (Loc_2, =) \times (A^*_{c_1}, \leqslant_*) \times (A^*_{c_2}, \leqslant_*)$

A model useful for concurrent protocols but also timed automata, metric temporal logic, products of modal logics, ...

Some WSTS's: Lossy Channel Systems (LCS)



A state $s = (\ell_1, \ell_2, w_1, w_2)$ with $w_i \in A^*$.
E.g., $w_1 = \texttt{hup.ack.ack}$.

Reliable steps: $s \to_{\text{rel}} s'$ read in front of channels, write at end (FIFO)

Lossy steps: messages may be lost nondeterministically

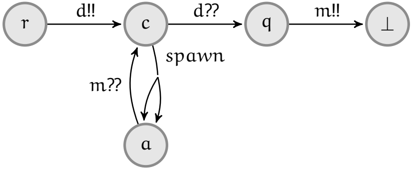
$$s \to s' \stackrel{\text{def}}{\Leftrightarrow} s \geqslant_* t \to_{\text{rel}} t' \geqslant_* s' \text{ for some } t, t' \in S$$

where $(S, \sqsubseteq)$ is the wqo $(Loc_1, =) \times (Loc_2, =) \times (A^*_{c_1}, \leqslant_*) \times (A^*_{c_2}, \leqslant_*)$

A model useful for concurrent protocols but also timed automata, metric temporal logic, products of modal logics, ...

SOME WSTS'S: BROADCAST PROTOCOLS

Broadcast protocols (Esparza, Finkel, Mayr 1999), aka population protocols, are dynamic & distributed collections of finite-state processes communicating via brodcasts (and rendez-vous, not shown here).



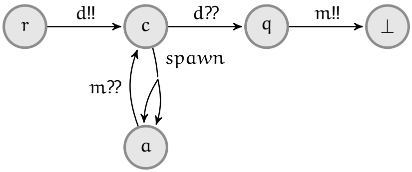
A configuration collects the local states of all processes. E.g., $s = \{c, r, c\}$, also denoted $\{c^2, r\}$.

Steps:

$\{c^2, q, r\} \xrightarrow{s(pawn)} \{a^2, c, q, r\} \xrightarrow{s} \{a^4, q, r\} \xrightarrow{m} \{c^4, r, \perp\} \xrightarrow{d} \{c, q^4, \perp\}$

SOME WSTS'S: BROADCAST PROTOCOLS

Broadcast protocols (Esparza, Finkel, Mayr 1999), aka population protocols, are dynamic & distributed collections of finite-state processes communicating via brodcasts (and rendez-vous, not shown here).

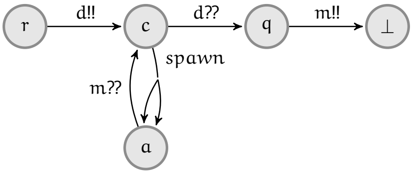


A configuration collects the local states of all processes. E.g., $s = \{c, r, c\}$, also denoted $\{c^2, r\}$.

Steps:

$$\{c^2, q, r\} \xrightarrow{s(pawn)} \{a^2, c, q, r\} \xrightarrow{s} \{a^4, q, r\} \xrightarrow{m} \{c^4, r, \perp\} \xrightarrow{d} \{c, q^4, \perp\}$$

PROVING TERMINATION

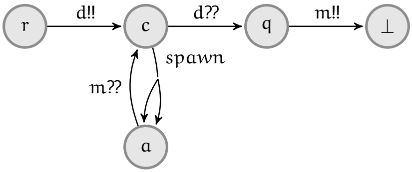


- This protocol has no infinite runs

Proof. Write $s = \{r^{n_1}, q^{n_2}, c^{n_3}, a^*, \perp^*\}$.
In any step $s \to s'$ the triple $\langle n_1, n_2, n_3 \rangle$ decreases in the lexicographic ordering

This is the pattern for proofs of termination: one invents a well-founded measure that decreases with every step

Proving Termination

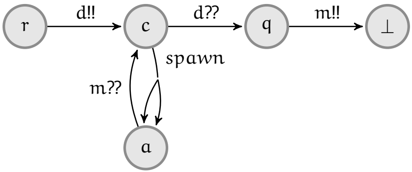


- This protocol has no infinite runs

Proof. Write $s = \{r^{n_1}, q^{n_2}, c^{n_3}, a^*, \bot^*\}$.
In any step $s \to s'$ the triple $\langle n_1, n_2, n_3 \rangle$ decreases in the lexicographic ordering

This is the pattern for proofs of termination: one invents a well-founded measure that decreases with every step

PROVING TERMINATION



- This protocol has no infinite runs

Proof. Write $s = \{r^{n_1}, q^{n_2}, c^{n_3}, a^*, \bot^*\}$.
In any step $s \to s'$ the triple $\langle n_1, n_2, n_3 \rangle$ decreases in the lexicographic ordering

This is the pattern for proofs of termination: one invents a well-founded measure that decreases with every step

BRODCAST PROTOCOLS ARE WSTS

1. Order the configurations by multiset inclusion, e.g.,
$\{c,q\} \subseteq \{c^2,r,q\}$

2. Observe that steps are monotonic:

$$s \to t \wedge s \subseteq s' \implies \exists t' : s' \to t' \wedge t \subseteq t'$$

3. Further observe that $(S,\subseteq)$ is wqo : it is isomoprhic to $\mathbb{N}^{|Loc|}$

BRODCAST PROTOCOLS ARE WSTS

1. Order the configurations by multiset inclusion, e.g., $\{c,q\} \subseteq \{c^2,r,q\}$

2. Observe that steps are monotonic:

$$s \to t \wedge s \subseteq s' \implies \exists t' : s' \to t' \wedge t \subseteq t'$$

Proof. Case analysis: is $s \to t$ an internal move? or a spawning step? or a broadcasting? or a rendez-vous?

3. Further observe that $(S, \subseteq)$ is wqo : it is isomoprhic to $\mathbb{N}^{|Loc|}$

BRODCAST PROTOCOLS ARE WSTS

1. Order the configurations by multiset inclusion, e.g.,
$\{c, q\} \subseteq \{c^2, r, q\}$

2. Observe that steps are monotonic:

$$s \to t \land s \subseteq s' \implies \exists t' : s' \to t' \land t \subseteq t'$$

3. Further observe that $(S, \subseteq)$ is wqo : it is isomoprhic to $\mathbb{N}^{|Loc|}$

Part 2

Verification of WSTS

Deciding Termination for WSTS

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. $s_i \leqslant s_j$

Corollary. One may decide Termination by enumerating all finite runs from s_{init} until a good sequence is encountered.
If all runs are bad, the enumeration will eventually exhaust them

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
A similar algorithm allows deciding safety properties

Deciding Termination for WSTS

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. $s_i \leqslant s_j$

Proof. $\Rightarrow$: by definition since $\leqslant$ is wqo
$\Leftarrow$: good finite run $s_0 \xrightarrow{*} s_i \xrightarrow{+} s_j$ can be extended by simulating $s_i \xrightarrow{+} s_j$ from above: $s_j \xrightarrow{+} s_{2j-i}$, then $s_{2j-i} \xrightarrow{+} s_{3j-2i}$, etc.

Corollary. One may decide Termination by enumerating all finite runs from s_{init} until a good sequence is encountered.
If all runs are bad, the enumeration will eventually exhaust them

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
A similar algorithm allows deciding safety properties

DECIDING TERMINATION FOR WSTS

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. $s_i \leqslant s_j$

Corollary. One may decide Termination by enumerating all finite runs from s_{init} until a good sequence is encountered.
If all runs are bad, the enumeration will eventually exhaust them

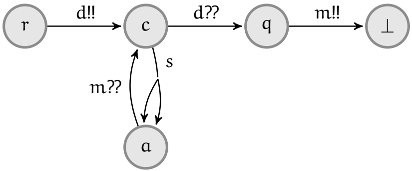
NB: This requires some minimal effectiveness assumptions on the WSTS, e.g., that the ordering is decidable

Algorithm extends and allows deciding inevitability, finiteness, and regular simulation
A similar algorithm allows deciding safety properties

Part 3a

Complexity: Upper Bounds

Broadcast Protocols Take Their Time

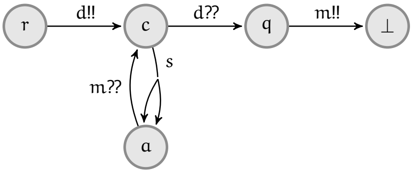


"Doubling" run: $\{c^{n}, q, (\bot^*)\} \xrightarrow{s^n} \{a^{2n}, q, (\bot^*)\} \xrightarrow{m} \{c^{2n}, (\bot^*)\}$

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

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

Broadcast Protocols Take Their Time

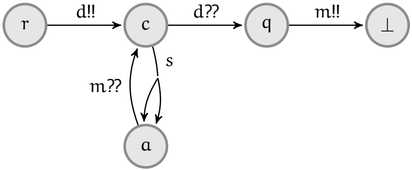


"Doubling" run: $\{c^{n}, q, (\bot*)\} \xrightarrow{s^n} \{a^{2n}, q, (\bot*)\} \xrightarrow{m} \{c^{2n}, (\bot*)\}$

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

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

Broadcast Protocols Take Their Time



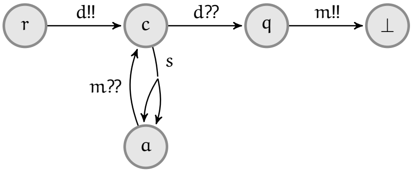
"Doubling" run: $\{c^{n}, q, (\bot^*)\} \xrightarrow{s^n} \{a^{2n}, q, (\bot^*)\} \xrightarrow{m} \{c^{2n}, (\bot^*)\}$

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

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

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

Broadcast Protocols Take Their Time



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

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

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 generic algorithm verifying termination is not elementary for broadcast protocols

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 $\mathbb{N} \to \mathbb{N}$

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

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

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 $\lambda_0 < \lambda_1 < \lambda_2 < \dots < \lambda$.

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).

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

One can exhibit arbitrarily long bad sequences. E.g. over $(\mathbb{N}^k, \leqslant_\times)$:

— 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 broadcast protocols don't have unbounded increases, and the starting configuration is the input of our problem

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

One can exhibit arbitrarily long bad sequences. E.g. over $(\mathbb{N}^k, \leqslant_\times)$:

— 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 broadcast protocols don't have unbounded increases, and the starting configuration is the input of our problem

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

One can exhibit arbitrarily long bad sequences. E.g. over $(\mathbb{N}^k, \leqslant_\times)$:

— 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 broadcast protocols don't have unbounded increases, and the starting configuration is the input of our problem

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

One can exhibit arbitrarily long bad sequences. E.g. over $(\mathbb{N}^k, \leqslant_\times)$:

— $999, 998, \ldots, 1, 0$

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

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

The runs of broadcast protocols don't have unbounded increases, and the starting configuration is the input of our problem

CONTROLLED BAD SEQUENCES

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

Here the control is the pair (n_0, g) of $n_0 \in \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$.

Fact. For a fixed wqo $(A, \leqslant, |.|)$ and control (n_0, g), there is max length on controlled bad sequences (Kőnig's Lemma again)

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

Length Function Theorem for $(\mathbb{N}^k, \leqslant_\times)$ [McAloon,Schmitz & S.,...]

— $L_{g,\mathbb{N}^k}(n_0) \leqslant g'_k(n_0)$ with g' polynomial in g

— g'_k and $L_{g,\mathbb{N}^k}$ are in $\mathscr{F}_{k+m-1}$ for g in $\mathscr{F}_m$

Controlled Bad Sequences

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

Here the control is the pair (n_0, g) of $n_0 \in \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$.

Fact. For a fixed wqo $(A, \leqslant, |.|)$ and control (n_0, g), there is max length on controlled bad sequences (Kőnig's Lemma again)

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

Length Function Theorem for $(\mathbb{N}^k, \leqslant_\times)$ [McAloon,Schmitz & S.,...]

— $L_{g,\mathbb{N}^k}(n_0) \leqslant g'_k(n_0)$ with g' polynomial in g

— g'_k and $L_{g,\mathbb{N}^k}$ are in $\mathscr{F}_{k+m-1}$ for g in $\mathscr{F}_m$

Controlled Bad Sequences

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

Here the control is the pair (n_0, g) of $n_0 \in \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$.

Fact. For a fixed wqo $(A, \leqslant, |.|)$ and control (n_0, g), there is max length on controlled bad sequences (Kőnig's Lemma again)

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

Length Function Theorem for $(\mathbb{N}^k, \leqslant_\times)$ [McAloon,Schmitz & S.,...]

— $L_{g,\mathbb{N}^k}(n_0) \leqslant g'_k(n_0)$ with g' polynomial in g

— g'_k and $L_{g,\mathbb{N}^k}$ are in $\mathscr{F}_{k+m-1}$ for g in $\mathscr{F}_m$

APPLYING TO BROADCAST PROTOCOLS

The runs explored by the Termination algorithm are controlled with $|s_{\text{init}}|$ 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.

NB. Similar controls for the backward-chaining Coverability algorithm: $|s_{target}|$ and $Succ$.

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

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

Applying to Broadcast Protocols

The runs explored by the Termination algorithm are controlled with $|s_{\text{init}}|$ 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.

NB. Similar controls for the backward-chaining Coverability algorithm: $|s_{target}|$ and $Succ$.

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

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

Applying to Broadcast Protocols

The runs explored by the Termination algorithm are controlled with $|s_{\text{init}}|$ 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.

NB. Similar controls for the backward-chaining Coverability algorithm: $|s_{target}|$ and $Succ$.

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

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

More Length Function Theorems

For finite words with embedding, $L_{\Sigma*}$ is in $\mathscr{F}_{\omega^{|\Sigma|-1}}$, and in $\mathscr{F}_{\omega^\omega}$ when alphabet is not fixed [Cichon,Schmitz& S.]. 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 [S.S.]. Applies e.g. to timed-arc Petri nets.

For finite words with priority ordering, $L_{\Sigma*}$ is in $\mathscr{F}_{\varepsilon_0}$. Applies e.g. to priority channel systems and higher-order LCS.

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

Some research goals: more varied/complex wqos (powerset, restricted families of graphs, ..) & analysis of complex algorithms

More Length Function Theorems

For finite words with embedding, $L_{\Sigma*}$ is in $\mathscr{F}_{\omega^{|\Sigma|-1}}$, and in $\mathscr{F}_{\omega^\omega}$ when alphabet is not fixed [Cichon,Schmitz& S.]. 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 [S.S.]. Applies e.g. to timed-arc Petri nets.

For finite words with priority ordering, $L_{\Sigma*}$ is in $\mathscr{F}_{\varepsilon_0}$. Applies e.g. to priority channel systems and higher-order LCS.

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

Some research goals: more varied/complex wqos (powerset, restricted families of graphs, ..) & analysis of complex algorithms

More Length Function Theorems

For finite words with embedding, $L_{\Sigma*}$ is in $\mathscr{F}_{\omega^{|\Sigma|-1}}$, and in $\mathscr{F}_{\omega^\omega}$ when alphabet is not fixed [Cichon,Schmitz& S.]. 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 [S.S.]. Applies e.g. to timed-arc Petri nets.

For finite words with priority ordering, $L_{\Sigma*}$ is in $\mathscr{F}_{\varepsilon_0}$. Applies e.g. to priority channel systems and higher-order LCS.

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

Some research goals: more varied/complex wqos (powerset, restricted families of graphs, ..) & analysis of complex algorithms

More Length Function Theorems

For finite words with embedding, $L_{\Sigma*}$ is in $\mathscr{F}_{\omega^{|\Sigma|-1}}$, and in $\mathscr{F}_{\omega^\omega}$ when alphabet is not fixed [Cichon,Schmitz& S.]. 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 [S.S.]. Applies e.g. to timed-arc Petri nets.

For finite words with priority ordering, $L_{\Sigma*}$ is in $\mathscr{F}_{\varepsilon_0}$. Applies e.g. to priority channel systems and higher-order LCS.

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

Some research goals: more varied/complex wqos (powerset, restricted families of graphs, ..) & analysis of complex algorithms

Part 3b

Complexity: 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 $\mathscr{F}_\omega$-hard, hence $\mathscr{F}_\omega$-complete, for broadcast protocols [Urquhart,..]

and $\mathscr{F}_{\omega^\omega}$-complete for lossy channel systems [ChambartS'08],
$\mathscr{F}_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12],
$\mathscr{F}_{\varepsilon_0}$-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 $\mathscr{F}_\omega$-hard, hence $\mathscr{F}_\omega$-complete, for broadcast protocols [Urquhart,..]

and $\mathscr{F}_{\omega^\omega}$-complete for lossy channel systems [ChambartS'08], $\mathscr{F}_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12], $\mathscr{F}_{\varepsilon_0}$-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 $\mathscr{F}_\omega$-hard, hence $\mathscr{F}_\omega$-complete, for broadcast protocols [Urquhart,..]

and $\mathscr{F}_{\omega^\omega}$-complete for lossy channel systems [ChambartS'08], $\mathscr{F}_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12], $\mathscr{F}_{\varepsilon_0}$-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 $\mathscr{F}_\omega$-hard, hence $\mathscr{F}_\omega$-complete, for broadcast protocols [Urquhart,..]

and $\mathscr{F}_{\omega^\omega}$-complete for lossy channel systems [ChambartS'08],
$\mathscr{F}_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12],
$\mathscr{F}_{\varepsilon_0}$-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 $\mathscr{F}_\omega$-hard, hence $\mathscr{F}_\omega$-complete, for broadcast protocols [Urquhart,..]

and $\mathscr{F}_{\omega^\omega}$-complete for lossy channel systems [ChambartS'08], $\mathscr{F}_{\omega^{\omega^\omega}}$-complete for timed-arc Petri nets [HaddadSS'12], $\mathscr{F}_{\varepsilon_0}$-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}. (Recall: broadcast protocol computing tower function)

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 $\alpha \sqsubseteq \beta$ that entails $F_\alpha(x) \leqslant F_\beta(x)$.

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}. (Recall: broadcast protocol computing tower function)

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 $\alpha \sqsubseteq \beta$ that entails $F_\alpha(x) \leqslant F_\beta(x)$.

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}. (Recall: broadcast protocol computing tower function)

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 $\alpha \sqsubseteq \beta$ that entails $F_\alpha(x) \leqslant F_\beta(x)$.

Concluding remarks

- Executive Summary

Complexity analysis of WSTS models is possible

We have complexity classes, generic techniques for upper bounds, catalog of $\mathscr{F}_\alpha$-complete problems, see S. Schmitz. Complexity hierarchies beyond Elementary. ACM Trans. Computation Theory, 8(1), 2016.

Many applications in verification and logic

- Perspectives

Need more length function theorems

There are many models for which complexity has not been narrowed

Would love to have alternative to Hardy computations …