Well-Structured

Parameterized Networks of Systems

Philippe Schnoebelen

LSV, CNRS & ENS Cachan

1st Workshop on Parameterized Verification, Roma, Sep. 6th, 2014

WSTS FOR PV?

- *Well-structured 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/VASS extensions, channel systems, counter machines, integral automata, etc.
- Used in software verification, communication protocols, . . . In particular, for distributed algorithms, WSTS have been used for verification of parameterized networks. Useful for proving safety/for finding minimal unsafe start configurations.
- WSTS still thriving today, with several new models (based on wqos on graphs, etc.), or applications (deciding data logics, modal logics, etc.) proposed every year.
- Meanwhile, the generic WSTS theory saw recent new developments: (1) techniques for wqo-based complexity;

WSTS FOR PV?

- *Well-structured 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/VASS extensions, channel systems, counter machines, integral automata, etc.

- Used in software verification, communication protocols, . . . In particular, for distributed algorithms, WSTS have been used for verification of parameterized networks. Useful for proving safety/for finding minimal unsafe start configurations.

- WSTS still thriving today, with several new models (based on wqos on graphs, etc.), or applications (deciding data logics, modal logics, etc.) proposed every year.

- Meanwhile, the generic WSTS theory saw recent new developments: (1) techniques for wqo-based complexity;

WSTS FOR PV?

- *Well-structured 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/VASS extensions, channel systems, counter machines, integral automata, etc.
- Used in software verification, communication protocols, ... In particular, for distributed algorithms, WSTS have been used for verification of parameterized networks. Useful for proving safety/for finding minimal unsafe start configurations.
- WSTS still thriving today, with several new models (based on wqos on graphs, etc.), or applications (deciding data logics, modal logics, etc.) proposed every year.
- Meanwhile, the generic WSTS theory saw recent new developments: (1) techniques for wqo-based complexity;

WSTS FOR PV?

- WSTS invented in 1987, developed and popularized in 1996–2005 by Abdulla & Jonsson, Finkel & Schnoebelen, etc. First used with Petri nets/VASS extensions, channel systems, counter machines, integral automata, etc.

- Used in software verification, communication protocols, ... In particular, for distributed algorithms, WSTS have been used for verification of parameterized networks. Useful for proving safety/for finding minimal unsafe start configurations.

- WSTS still thriving today, with several new models (based on wqos on graphs, etc.), or applications (deciding data logics, modal logics, etc.) proposed every year.

- Meanwhile, the generic WSTS theory saw recent new developments: (1) techniques for wqo-based complexity; (2) completion theory for forward acceleration; ...

Outline of the Talk

- Part 1: **Basics of WSTS.**

  Recalling the basic definition, with Broadcast protocols and Timed-arc nets as examples

- Part 2: **Verifying WSTS.**

  Two simple verification algorithms, deciding Termination and Coverability

- Part 3: **A few words on complexity.**

  Looking at controlled bad sequences and bounding their length

Part 1 What are 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 $\mathcal{S} = (S, \rightarrow, \leqslant)$ that is monotonic and such that $(S, \leqslant)$ is a well-quasi-ordering (a wqo, more later).

Recall:

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

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

– monotonic TS: ordered TS with
$\big(s_1 \rightarrow s_2$ and $s_1 \leqslant t_1\big)$ implies $\exists t_2 \in S : \big(t_1 \rightarrow t_2$ 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 $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n$ can be simulated "from above" with some $t_0 \rightarrow t_1 \rightarrow \cdots \rightarrow t_n$

Well-Quasi-Ordering (WQO)

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

Def. $(X, \leqslant)$ 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 $i < j$.

$\Leftrightarrow$ "*every infinite sequence is a good sequence*"

$\Leftrightarrow$ "*every bad sequence is finite*"

Well-Quasi-Ordering (WQO)

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

Def. $(X, \leqslant)$ 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 $i < j$.

$\Leftrightarrow$ "*every infinite sequence is a good sequence*"

$\Leftrightarrow$ "*every bad sequence is finite*"

Alternatively: $(X, \leqslant)$ 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. Equivalence of these two definitions is not trivial

Well-Quasi-Ordering (WQO)

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

Def. $(X,\leqslant)$ 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 $i<j$.

$\Leftrightarrow$ "*every infinite sequence is a good sequence*"

$\Leftrightarrow$ "*every bad sequence is finite*"

Alternatively: $(X,\leqslant)$ 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. Equivalence of these two definitions is not trivial

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

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

Well-Quasi-Ordering (WQO)

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

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

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

Example. (Cartesian product) $(X_1 \times \cdots \times X_k, \leqslant_\times)$ is a wqo when $(X_1, \leqslant_1), \ldots, (X_k, \leqslant_k)$ are wqos, with

$$\mathbf{x} = (x_1, \ldots, x_k) \leqslant_\times \mathbf{y} = (y_1, \ldots, y_k) \overset{\text{def}}{\Leftrightarrow} x_1 \leqslant_1 y_1 \wedge \cdots \wedge x_k \leqslant_k y_k$$

Well-Quasi-Ordering (WQO)

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

Example. (Cartesian product) $(X_1 \times \cdots \times X_k, \leqslant_\times)$ is a wqo when $(X_1, \leqslant_1), \ldots, (X_k, \leqslant_k)$ are wqos, with

$$\boldsymbol{x} = (x_1, \ldots, x_k) \leqslant_\times \boldsymbol{y} = (y_1, \ldots, y_k) \overset{\text{def}}{\Leftrightarrow} x_1 \leqslant_1 y_1 \wedge \cdots \wedge x_k \leqslant_k y_k$$

Example. (Kleene star) $(X^*, \leqslant_*)$ is a wqo when $(X, \leqslant)$ is a wqo, with

$$\boldsymbol{x} = (x_1 \cdots x_k) \leqslant_* \boldsymbol{y} = (y_1 \cdots y_\ell)$$

$\overset{\text{def}}{\Leftrightarrow} x_1 \leqslant y_{i_1} \wedge \cdots \wedge x_k \leqslant y_{i_k}$ for some $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant \ell$

$\overset{\text{def}}{\Leftrightarrow} \boldsymbol{x} \leqslant_\times \boldsymbol{y}'$ for some subsequence $\boldsymbol{y}'$ of $\boldsymbol{y}$

WELL-QUASI-ORDERING (WQO)

Example. (Cartesian product) $(X_1 \times \cdots \times X_k, \leqslant_\times)$ is a wqo when $(X_1, \leqslant_1), \ldots, (X_k, \leqslant_k)$ are wqos, with

$$\boldsymbol{x} = (x_1, \ldots, x_k) \leqslant_\times \boldsymbol{y} = (y_1, \ldots, y_k) \overset{\text{def}}{\Leftrightarrow} x_1 \leqslant_1 y_1 \wedge \cdots \wedge x_k \leqslant_k y_k$$

Example. (Kleene star) $(X^*, \leqslant_*)$ is a wqo when $(X, \leqslant)$ is a wqo, with

$$\boldsymbol{x} = (x_1 \cdots x_k) \leqslant_* \boldsymbol{y} = (y_1 \cdots y_\ell)$$

$\overset{\text{def}}{\Leftrightarrow} x_1 \leqslant y_{i_1} \wedge \cdots \wedge x_k \leqslant y_{i_k}$ for some $1 \leqslant i_1 < i_2 < \cdots < i_k \leqslant \ell$

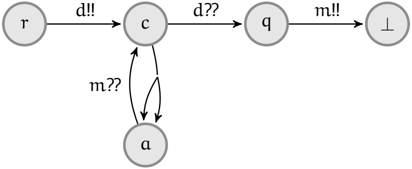
$\overset{\text{def}}{\Leftrightarrow} \boldsymbol{x} \leqslant_\times \boldsymbol{y}'$ for some subsequence $\boldsymbol{y}'$ of $\boldsymbol{y}$

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

Two examples of WSTS

EXAMPLE 1: BROADCAST PROTOCOLS

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



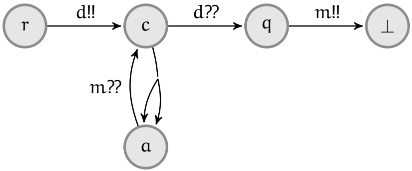
A configuration collects the local states of all processes. E.g., $s = \{c, r, c\}$, also denoted $\{c^2, r\}$.

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

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

EXAMPLE 1: BROADCAST PROTOCOLS

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



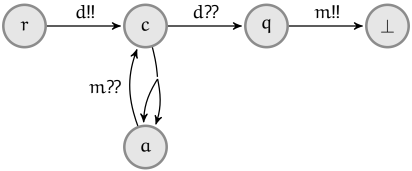
A configuration collects the local states of all processes. E.g., $s = \{c, r, c\}$, also denoted $\{c^2, r\}$.

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

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

Example 1: 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: $\{c^2, q, r\} \to \{a^2, c, q, r\} \to \{a^4, q, r\} \xrightarrow{m} \{c^4, r, \perp\} \xrightarrow{d} \{c, q^4, \perp\}$

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. *Conf* $= \mathcal{M}_f(\{r, c, a, q, \perp\})$ equipped with $\subseteq$ 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

Example 2: Timed-arc Nets

Timed-arc Nets (Abdulla & Nylén 2001), aka TPN, are dynamic & distributed collections of finite-state processes, each carrying a real-valued clock.

EXAMPLE 2: TIMED-ARC NETS

Timed-arc Nets (Abdulla & Nylén 2001), aka TPN, are dynamic & distributed collections of finite-state processes, each carrying a real-valued clock.

Control states of individual processes taken from some finite $Q = \{r, c, a, q, ..\}$ (same as Broadcast protocols)

A configuration collects the local states of all processes, e.g., $s = \{c : 1.4, r : 3.0, q : 2.5\}$, this time with clock values.

I.e. *Conf* $\stackrel{\text{def}}{=} \mathcal{M}_f(Q \times \mathbb{R}_{\geqslant 0})$

Example 2: Timed-arc Nets

Control states of individual processes taken from some finite $Q = \{r, c, a, q, ..\}$ (same as Broadcast protocols)

A configuration collects the local states of all processes, e.g., $s = \{c : 1.4, r : 3.0, q : 2.5\}$, this time with clock values.

I.e. *Conf* $\stackrel{\text{def}}{=} \mathcal{M}_f(Q \times \mathbb{R}_{\geqslant 0})$

TPNs have rules like e.g. $\delta = \left\{ \begin{array}{c} c \in [1;2) \\ q \in [2;\infty) \end{array} \mapsto \begin{array}{c} r \in [0;2] \\ q \in [1;1] \\ a \in (0;4) \end{array} \right\}$

Example 2: Timed-arc Nets

A configuration collects the local states of all processes, e.g., $s = \{c : 1.4, r : 3.0, q : 2.5\}$, this time with clock values.
I.e. *Conf* $\stackrel{\text{def}}{=} \mathcal{M}_f(Q \times \mathbb{R}_{\geqslant 0})$

TPNs have rules like e.g. $\delta = \left\{ \begin{array}{c} c \in [1;2) \\ q \in [2;\infty) \end{array} \mapsto \begin{array}{c} r \in [0;2] \\ q \in [1;1] \\ a \in (0;4) \end{array} \right\}$

Yielding steps like
$s = \{c : 1.4, r : 3.0, q : 2.5\} \xrightarrow{\delta} \{r : 3.0, r : 0.73, q : 1.0, a : 2.1\} = s'$

EXAMPLE 2: TIMED-ARC NETS

A configuration collects the local states of all processes, e.g., $s = \{c : 1.4, r : 3.0, q : 2.5\}$, this time with clock values.
I.e. *Conf* $\stackrel{\text{def}}{=} \mathcal{M}_f(Q \times \mathbb{R}_{\geqslant 0})$

TPNs have rules like e.g. $\delta = \left\{ \begin{array}{c} c \in [1;2) \\ q \in [2;\infty) \end{array} \mapsto \begin{array}{c} r \in [0;2] \\ q \in [1;1] \\ a \in (0;4) \end{array} \right\}$

Yielding steps like
$s = \{c : 1.4, r : 3.0, q : 2.5\} \xrightarrow{\delta} \{r : 3.0, r : 0.73, q : 1.0, a : 2.1\} = s'$

also time-elapse steps like
$s' = \{r : 3.0, r : 0.73, q : 1.0, s : 2.1\} \xrightarrow{+0.8} \{r : 3.8, r : 1.53, q : 1.8, a : 2.9\}$

Example 2: Timed-arc Nets

TPNs have rules like e.g. $\delta = \left\{ \begin{array}{c} c \in [1;2) \\ q \in [2;\infty) \end{array} \mapsto \begin{array}{c} r \in [0;2] \\ q \in [1;1] \\ a \in (0;4) \end{array} \right\}$

Yielding steps like

$s = \{c:1.4, r:3.0, q:2.5\} \xrightarrow{\delta} \{r:3.0, r:0.73, q:1.0, a:2.1\} = s'$

also time-elapse steps like

$s' = \{r:3.0, r:0.73, q:1.0, s:2.1\} \xrightarrow{+0.8} \{r:3.8, r:1.53, q:1.8, a:2.9\}$

Fact. Steps are monotonic for multiset inclusion

But $(\mathcal{M}_f(Q \times \mathbb{R}_{\geqslant 0}), \subseteq)$ is not wqo —since already $(\mathbb{R}_{\geqslant 0}, =)$ is not

Timed-arc Nets are WSTS

$$s = \{r : 3.0, r : 0.73, q : 1.0, a : 2.1\} \quad \approx \quad \widetilde{s} = \{r : 3, q : 1\} \bullet \{a : 2\} \bullet \{r : 0\}$$

Timed-arc Nets are WSTS

$$s = \{r:3.0, r:0.73, q:1.0, a:2.1\} \quad \approx \quad \widetilde{s} = \{r:3, q:1\} \bullet \{a:2\} \bullet \{r:0\}$$

$$\begin{array}{ccccccc} [0 & < & x_1 & < & x_2 & < & 1) \\ | & & | & & | & & \\ r:3 & & a:2(+x_1) & & r:0(+x_2) & & \\ q:1 & & & & & & \end{array}$$

In general $\widetilde{s}$ is a sequence over $\mathcal{M}_f(Q \times \{0,1,2,3,4,5+\})$

TIMED-ARC NETS ARE WSTS

In general $\widetilde{s}$ is a sequence over $\mathcal{M}_f(Q \times \{0,1,2,3,4,5+\})$

Fact. The abstracted system is bisimilar with the original one (NB: durations of time-elapse steps are not preserved).

$$\{r:3, q:1\} \bullet \{a:2\} \bullet \{r:0\} \xrightarrow{+?} \{\} \bullet \{r:3, q:1\} \bullet \{a:2\} \bullet \{r:0\}$$

$$\xrightarrow{+?} \{r:1\} \bullet \{r:3, q:1\} \bullet \{a:2\} \to \cdots$$

TIMED-ARC NETS ARE WSTS

In general $\widetilde{s}$ is a sequence over $\mathcal{M}_f(Q \times \{0, 1, 2, 3, 4, 5+\})$

Fact. The abstracted system is bisimilar with the original one (NB: durations of time-elapse steps are not preserved).

$$\begin{aligned} \{r:3, q:1\} \bullet \{a:2\} \bullet \{r:0\} &\xrightarrow{+?} \{\} \bullet \{r:3, q:1\} \bullet \{a:2\} \bullet \{r:0\} \\ &\xrightarrow{+?} \{r:1\} \bullet \{r:3, q:1\} \bullet \{a:2\} \to \cdots \end{aligned}$$

Fact. This new semantics is monotonic wrt pointed sequence embedding $\leqslant_*$ over $\big(\mathcal{M}_f(Q \times \{0, \ldots, 4, 5+\})\big)^+$, a wqo.
Hence TPN 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 $\stackrel{\text{def}}{\Leftrightarrow}$ there exist $i < j$ s.t. $s_i \leqslant s_j$

Proof: "$\Rightarrow$" by def of wqo. "$\Leftarrow$" by simulating $s_i \xrightarrow{+} s_j$ from 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 $\mathcal{S} = (S, \to, \ldots)$ 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. $s_i \leqslant s_j$

Proof: "$\Rightarrow$" by def of wqo. "$\Leftarrow$" by simulating $s_i \xrightarrow{+} s_j$ from 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 $\mathcal{S} = (S, \to, \ldots)$ 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 $\stackrel{\text{def}}{\Leftrightarrow}$ there exist $i < j$ s.t. $s_i \leqslant s_j$

Proof: "$\Rightarrow$" by def of wqo. "$\Leftarrow$" by simulating $s_i \xrightarrow{+} s_j$ from 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 asks, given $\mathcal{S} = (S, \rightarrow, \ldots)$, a state s_{init} and a target state t, whether $\mathcal{S}$ has a covering run $s_{init} \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_n$ with $s_n \geqslant t$.

This is equivalent to having a covering pseudorun of the form

$$s_{init} = s_0 \geqslant t_0 \rightarrow s_1 \geqslant t_1 \rightarrow s_2 \geqslant \cdots t_{n-1} \rightarrow s_n \geqslant t_n = t$$

Fact. In a covering pseudorun, we can assume that each t_i is a minimal (pseudo) predecessor of t_{i+1}

Fact. In a shortest covering pseudorun, the (reversed) sequence $t_n, \ldots, t_1, t_0$ is bad

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a covering pseudorun from s_{init} to t **iff** it has one that is minimal and reverse-bad

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

Coverability

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

This is equivalent to having a covering pseudorun of the form

$$s_{init} = s_0 \geqslant t_0 \to s_1 \geqslant t_1 \to s_2 \geqslant \cdots t_{n-1} \to s_n \geqslant t_n = t$$

Fact. In a covering pseudorun, we can assume that each t_i is a minimal (pseudo) predecessor of t_{i+1}

Fact. In a shortest covering pseudorun, the (reversed) sequence $t_n, \ldots, t_1, t_0$ is bad

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a covering pseudorun from s_{init} to t **iff** it has one that is minimal and reverse-bad

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

Coverability

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

This is equivalent to having a covering pseudorun of the form

$$s_{init} = s_0 \geqslant t_0 \to s_1 \geqslant t_1 \to s_2 \geqslant \cdots t_{n-1} \to s_n \geqslant t_n = t$$

Fact. In a covering pseudorun, we can assume that each t_i is a minimal (pseudo) predecessor of t_{i+1}

Fact. In a shortest covering pseudorun, the (reversed) sequence $t_n, \ldots, t_1, t_0$ is bad

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a covering pseudorun from s_{init} to t **iff** it has one that is minimal and reverse-bad

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

COVERABILITY

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

This is equivalent to having a covering pseudorun of the form

$$s_{init} = s_0 \geqslant t_0 \to s_1 \geqslant t_1 \to s_2 \geqslant \cdots t_{n-1} \to s_n \geqslant t_n = t$$

Fact. In a covering pseudorun, we can assume that each t_i is a minimal (pseudo) predecessor of t_{i+1}

Fact. In a shortest covering pseudorun, the (reversed) sequence $t_n, \ldots, t_1, t_0$ is bad

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a covering pseudorun from s_{init} to t **iff** it has one that is minimal and reverse-bad

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

COVERABILITY

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

This is equivalent to having a covering pseudorun of the form

$$s_{init} = s_0 \geqslant t_0 \to s_1 \geqslant t_1 \to s_2 \geqslant \cdots t_{n-1} \to s_n \geqslant t_n = t$$

Fact. In a covering pseudorun, we can assume that each t_i is a minimal (pseudo) predecessor of t_{i+1}

Fact. In a shortest covering pseudorun, the (reversed) sequence $t_n, \ldots, t_1, t_0$ is bad

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a covering pseudorun from s_{init} to t **iff** it has one that is minimal and reverse-bad

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

COVERABILITY

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

This is equivalent to having a covering pseudorun of the form

$$s_{init} = s_0 \geqslant t_0 \to s_1 \geqslant t_1 \to s_2 \geqslant \cdots t_{n-1} \to s_n \geqslant t_n = t$$

Fact. In a covering pseudorun, we can assume that each t_i is a minimal (pseudo) predecessor of t_{i+1}

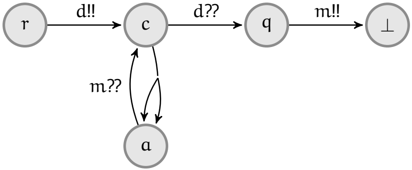
Fact. In a shortest covering pseudorun, the (reversed) sequence $t_n, \ldots, t_1, t_0$ is bad

Lem. [Finite Witnesses for Covering]
A WSTS $\mathcal{S}$ has a covering pseudorun from s_{init} to t **iff** it has one that is minimal and reverse-bad

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

Part 3 Bounding complexity

Broadcast Protocols and Termination



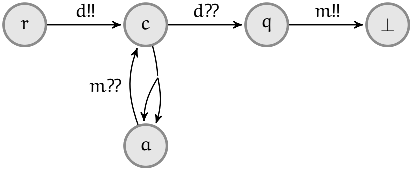
This broadcast protocol terminates: all its runs are bad sequences, hence are finite

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

– 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

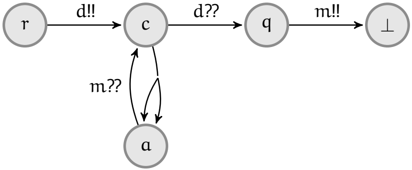


This broadcast protocol terminates: all its runs are bad sequences, hence are finite

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



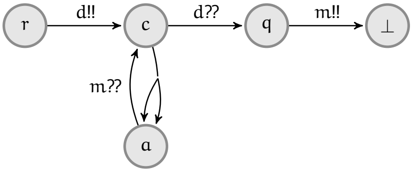
This broadcast protocol terminates: all its runs are bad sequences, hence are finite

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

– 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

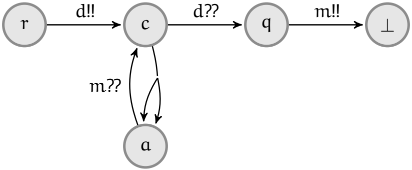


"Doubling" run: $\{c^{n}, q, (\perp^*)\} \xrightarrow{a^n} \{a^{2n}, q, (\perp^*)\} \xrightarrow{m} \{c^{2n}, (\perp^*)\}$

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

Broadcast Protocols Take Their Time

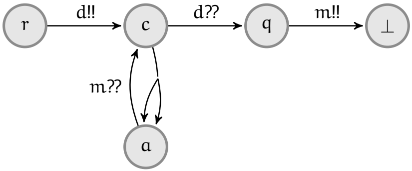


"Doubling" run: $\{c^{n}, q, (\perp^*)\} \xrightarrow{a^n} \{a^{2n}, q, (\perp^*)\} \xrightarrow{m} \{c^{2n}, (\perp^*)\}$

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

Broadcast Protocols Take Their Time



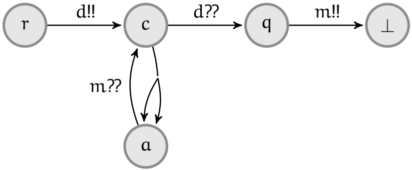
"Doubling" run: $\{c^{n}, q, (\perp^*)\} \xrightarrow{a^n} \{a^{2n}, q, (\perp^*)\} \xrightarrow{m} \{c^{2n}, (\perp^*)\}$

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

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

$\Rightarrow$ Runs of terminating systems may have nonelementary lengths

$\Rightarrow$ Running time of termination verification algorithm is not elementary (for broadcast protocols)

Complexity Analysis?

Key point: 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?

Key point: 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?

Key point: 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?

Key point: 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!

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 $\mathscr{F}_m$ [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 $\mathscr{F}_m$ [Figueira²SS'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 $\mathscr{F}_m$ [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 $n_0 = |s_{init}|$ and $g = \mathit{Succ} : \mathbb{N} \to \mathbb{N}$.

$\Rightarrow$ 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 $n_0 = |s_{init}|$ and $g = \mathit{Succ} : \mathbb{N} \to \mathbb{N}$.

$\Rightarrow$ 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 $n_0 = |s_{init}|$ and $g = \mathit{Succ}: \mathbb{N} \to \mathbb{N}$.

$\Rightarrow$ 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

NOW APPLYING TO TIMED-ARC NETS

Fact. The runs of a Timed-arc net N are controlled with $n_0 = |s_{init}|$ and $g : x \mapsto x + |N|$,
or with $n_0 = |s_{init}| + |N|$ and $g = \mathit{Double} : x \mapsto 2x$ if we want fixed g.

For *Conf* $= \mathcal{M}_f(Q \times \{0, 1, \ldots, M+\})^*$ ordered with pointed sequence embedding, the Length Function theorem [SS '11] gives

$$L_{g,\mathit{Conf}} \text{ in } \mathscr{F}_{\omega^{\omega^k}} \text{ where } k = |Q| \times M$$

$\Rightarrow$ Time/space bound in $\mathscr{F}_{\omega^{\omega^\omega}}$ for Timed-arc Nets verification

These bounds are optimal!
— Verification of Timed-arc nets is $\mathscr{F}_{\omega^{\omega^\omega}}$-complete [HSS '12]
— Verification of Broadcast protocols is $\mathscr{F}_{\omega}$-complete, or "Ackermann-complete" [S '10]

Bottom line: we can provide definite complexity for many WSTS models

Now applying to Timed-arc nets

Fact. The runs of a Timed-arc net N are controlled with $n_0 = |s_{init}|$ and $g : x \mapsto x + |N|$,
or with $n_0 = |s_{init}| + |N|$ and $g = \mathit{Double} : x \mapsto 2x$ if we want fixed g.

For $\mathit{Conf} = \mathcal{M}_f(Q \times \{0, 1, \ldots, M+\})^*$ ordered with pointed sequence embedding, the Length Function theorem [SS '11] gives

$$L_{g,\mathit{Conf}} \text{ in } \mathscr{F}_{\omega^{\omega^k}} \text{ where } k = |Q| \times M$$

$\Rightarrow$ Time/space bound in $\mathscr{F}_{\omega^{\omega^\omega}}$ for Timed-arc Nets verification

These bounds are optimal!

— Verification of Timed-arc nets is $\mathscr{F}_{\omega^{\omega^\omega}}$-complete [HSS '12]

— Verification of Broadcast protocols is $\mathscr{F}_{\omega}$-complete, or "Ackermann-complete" [S '10]

Bottom line: we can provide definite complexity for many WSTS models

Now applying to Timed-arc nets

Fact. The runs of a Timed-arc net N are controlled with $n_0 = |s_{\mathit{init}}|$ and $g : x \mapsto x + |N|$, or with $n_0 = |s_{\mathit{init}}| + |N|$ and $g = \mathit{Double} : x \mapsto 2x$ if we want fixed g.

For $\mathit{Conf} = \mathcal{M}_f(Q \times \{0,1,\ldots,M+\})^*$ ordered with pointed sequence embedding, the Length Function theorem [SS '11] gives

$$L_{g,\mathit{Conf}} \text{ in } \mathscr{F}_{\omega^{\omega^k}} \text{ where } k = |Q| \times M$$

$\Rightarrow$ Time/space bound in $\mathscr{F}_{\omega^{\omega^\omega}}$ for Timed-arc Nets verification

These bounds are optimal!

— Verification of Timed-arc nets is $\mathscr{F}_{\omega^{\omega^\omega}}$-complete [HSS '12]

— Verification of Broadcast protocols is $\mathscr{F}_\omega$-complete, or "Ackermann-complete" [S '10]

Bottom line: we can provide definite complexity for many WSTS models

THE FAST-GROWING HIERARCHY

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

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

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

gives $F_1(x) \sim 2x$, $F_2(x) \sim 2^x$, $F_3(x) \sim \text{tower}(x)$ and $F_\omega(x) \sim \textsc{Ackermann}(x)$, 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).

CONCLUDING REMARKS

- WSTS are a powerful tool for the verification of parameterized networks
- WSTS allow complexity analysis

Join the fun!

Technical details are lighter than it seems.
See [Sch '10] [HSS '12] [HSS '13]
and tutorial notes
– "Algorithmic Aspects of WQO Theory" (with S. Schmitz)
– "Complexity Hierarchies Beyond Elementary" (by S. Schmitz)

CONCLUDING REMARKS

- WSTS are a powerful tool for the verification of parameterized networks
- WSTS allow complexity analysis

Join the fun!

Technical details are lighter than it seems.
See [Sch '10] [HSS '12] [HSS '13]
and tutorial notes
– "Algorithmic Aspects of WQO Theory" (with S. Schmitz)
– "Complexity Hierarchies Beyond Elementary" (by S. Schmitz)

THANK YOU FOR YOUR INTEREST

REFERENCES: CLICK TO DOWNLOAD

[AN01] P. A. Abdulla and A. Nylén. Timed Petri nets and BQOs. ICATPN 2001.
[A+00] P. A. Abdulla, B. Jonsson *et al.* Algorithmic analysis of programs with well quasi-ordered domains. *Inf. & Comp.* 2000.
[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. Proc. LICS '99.
[F87] A. Finkel. A generalization of the procedure of Karp and Miller to well structured transition systems. Proc. ICALP '87
[FS01] A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comp. Sci.* 2001.
[F+11] D. Figueira et al. Ackermannian and primitive-recursive bounds with Dickson's lemma. LICS 2011.
[HSS12] S. Haddad, S. Schmitz, and Ph. Schnoebelen. The ordinal-recursive complexity of timed-arc Petri nets, data nets, and other enriched nets. LICS 2012
[S13] S. Schmitz. Complexity hierarchies beyond elementary. ArXiv Report cs.CC/1312.5686, 2013.
[SS11] S. Schmitz and Ph. Schnoebelen. Multiply-recursive upper bounds with Higman's lemma. ICALP 2011.
[SS12] S. Schmitz and Ph. Schnoebelen. Algorithmic aspects of WQO theory. ESSLLI lecture notes, 2012.
[S10] Ph. Schnoebelen. Revisiting Ackermann-hardness for lossy counter machines and reset Petri nets. MFCS 2010.