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 $S = (S, \rightarrow, \leq)$ that is monotonic and such that $(S, \leq)$ is a well-quasi-ordering (a wqo, more later).

Recall:
- transition system (TS): $S = (S, \rightarrow)$ with steps e.g. “$s \rightarrow s’$”
- ordered TS: $S = (S, \rightarrow, \leq)$ with smaller and larger states, e.g. $s \leq t$
- monotonic TS: ordered TS with $(s_1 \rightarrow s_2$ and $s_1 \leq t_1$) implies $\exists t_2 \in S : (t_1 \rightarrow t_2$ and $s_2 \leq t_2$), i.e., “larger states simulate smaller states”.

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

NB. Starting from any $t_0 \geq s_0$, a run $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n$ can be simulated “from above” with some $t_0 \rightarrow t_1 \rightarrow \cdots \rightarrow t_n$.
What Are WSTS?

**Def.** A WSTS is an ordered TS \( S = (S, \rightarrow, \leq) \) that is monotonic and such that \((S, \leq)\) is a well-quasi-ordering (a wqo, more later).

**Recall:**
- **transition system (TS):** \( S = (S, \rightarrow) \) with steps e.g. “\( s \rightarrow s' \)”
- **ordered TS:** \( S = (S, \rightarrow, \leq) \) with smaller and larger states, e.g. \( s \leq t \)
- **monotonic TS:** ordered TS with
  \[ (s_1 \rightarrow s_2 \text{ and } s_1 \leq t_1) \text{ implies } \exists t_2 \in S : (t_1 \rightarrow t_2 \text{ and } s_2 \leq t_2), \]
  i.e., “larger states simulate smaller states”.

**Equivalently:** \( \leq \) is a wqo and a simulation.

**NB.** Starting from any \( t_0 \geq s_0 \), 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 “\((S, \leq)\) is wqo”? 

**Def.** \((X, \leq)\) is a wqo if \(\iff\) any infinite sequence \(x_0, x_1, x_2, \ldots\) contains an increasing pair: \(x_i \leq x_j\) for some \(i < j\). 

\[\iff\text{“every infinite sequence is a good sequence”}\]

\[\iff\text{“every bad sequence is finite”}\]
**Well-Quasi-Ordering (WQO)**

Now what was meant by “\((S, \leq) \text{ is wqo} \)”?

**Def.** \((X, \leq) \text{ is a wqo } \iff \text{ any infinite sequence } x_0, x_1, x_2, \ldots \text{ contains an increasing pair: } x_i \leq x_j \text{ for some } i < j.\)**

\[\iff \text{ “every infinite sequence is a good sequence”}\]

\[\iff \text{ “every bad sequence is finite”}\]

**Alternatively:** \((X, \leq) \text{ is a wqo } \iff \text{ any infinite sequence } x_0, x_1, x_2, \ldots \text{ contains an infinite increasing subsequence: } x_{n_0} \leq x_{n_1} \leq x_{n_2} \leq \ldots\)

**NB.** Equivalence of these two definitions is **not trivial**
Well-Quasi-Ordering (WQO)

Now what was meant by “\((S, \leq) \text{ is wqo}\)?

**Def.** \((X, \leq) \text{ is a wqo } \iff \) any infinite sequence \(x_0, x_1, x_2, \ldots\) contains an increasing pair: \(x_i \leq x_j\) for some \(i < j\).

\[\iff \text{“every infinite sequence is a good sequence”}\]

\[\iff \text{“every bad sequence is finite”}\]

**Alternatively:** \((X, \leq) \text{ is a wqo } \iff \) any infinite sequence \(x_0, x_1, x_2, \ldots\) contains an infinite increasing subsequence: \(x_{n_0} \leq x_{n_1} \leq x_{n_2} \leq \ldots\)

**NB.** Equivalence of these two definitions is not trivial

**Example.** (Dickson’s Lemma) \(\mathbb{N}^k, \leq_{\times}\) is a wqo, with

\[a = (a_1, \ldots, a_k) \leq_{\times} b = (b_1, \ldots, b_k) \iff a_1 \leq b_1 \land \cdots \land a_k \leq b_k\]
**Well-Quasi-Ordering (WQO)**

**Def.** $(X, \leq)$ is a wqo $\iff$ any infinite sequence $x_0, x_1, x_2, \ldots$ contains an infinite increasing subsequence: $x_{n_0} \leq x_{n_1} \leq x_{n_2} \leq \ldots$

**Example.** (Dickson’s Lemma) $(\mathbb{N}^k, \leq_x)$ is a wqo, with
\[
\alpha = (a_1, \ldots, a_k) \leq_x \beta = (b_1, \ldots, b_k) \iff a_1 \leq b_1 \land \cdots \land a_k \leq b_k
\]

**Example.** (Cartesian product) $(X_1 \times \cdots \times X_k, \leq_x)$ is a wqo when $(X_1, \leq_1), \ldots, (X_k, \leq_k)$ are wqos, with
\[
x = (x_1, \ldots, x_k) \leq_x y = (y_1, \ldots, y_k) \iff x_1 \leq_1 y_1 \land \cdots \land x_k \leq_k y_k
\]
**WELL-QUASI-ORDERING (WQO)**

**Def.** \((X, \leq)\) is a wqo \(\iff\) any infinite sequence \(x_0, x_1, x_2, \ldots\) contains an infinite increasing subsequence: \(x_{n_0} \leq x_{n_1} \leq x_{n_2} \leq \ldots\)

**Example.** (Cartesian product) \((X_1 \times \cdots \times X_k, \leq)\) is a wqo when \((X_1, \leq_1), \ldots, (X_k, \leq_k)\) are wqos, with

\[
x = (x_1, \ldots, x_k) \leq_{\times} y = (y_1, \ldots, y_k) \iff x_1 \leq_1 y_1 \land \cdots \land x_k \leq_k y_k
\]

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

\[
x = (x_1 \cdots x_k) \leq^* y = (y_1 \cdots y_\ell)
\]

\(\iff\) \(x_1 \leq y_{i_1} \land \cdots \land x_k \leq y_{i_k}\) for some \(1 \leq i_1 < i_2 < \cdots < i_k \leq \ell\)

\(\iff\) \(x \leq_{\times} y'\) for some subsequence \(y'\) of \(y\)
**Well-Quasi-Ordering (WQO)**

**Example.** (Cartesian product) \((X_1 \times \cdots \times X_k, \leq_x)\) is a wqo when \((X_1, \leq_1), \ldots, (X_k, \leq_k)\) are wqos, with

\[
\mathbf{x} = (x_1, \ldots, x_k) \leq_x \mathbf{y} = (y_1, \ldots, y_k) \iff x_1 \leq_1 y_1 \land \cdots \land x_k \leq_k y_k
\]

**Example.** (Kleene star) \((X^*, \leq_*)\) is a wqo when \((X, \leq)\) is a wqo, with

\[
\mathbf{x} = (x_1 \cdots x_k) \leq_* \mathbf{y} = (y_1 \cdots y_\ell) \defeq x_1 \leq y_{i_1} \land \cdots \land x_k \leq y_{i_k} \text{ for some } 1 \leq i_1 < i_2 < \cdots < i_k \leq \ell
\]

\[
\iff \mathbf{x} \leq_x \mathbf{y}' \text{ for some subsequence } \mathbf{y}' \text{ of } \mathbf{y}
\]

**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: \( \{c^2, q, r\} \rightarrow \{a^2, c, q, r\} \rightarrow \{a^4, q, r\} \xrightarrow{m} \{c^4, r, \perp\} \xrightarrow{d} \{c, q^4, \perp\} \)

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 broadcasts and rendez-vous.

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

**Steps:**

\[
\begin{align*}
\{c^2, q, r\} &\rightarrow \{a^2, c, q, r\} \\
&\rightarrow \{a^4, q, r\} \\
&\rightarrow \{c^4, r, \perp\} \\
&\rightarrow \{c, q^4, \perp\}
\end{align*}
\]

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:**

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

**We’ll see later:** The above protocol does not have infinite runs.
**Broadcast Protocols Are WSTS**

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

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

**Proof:** this is exactly \((\mathbb{N}^5, \leq_x)\)

**Fact.** Broadcast protocols are monotonic TS

**Proof Idea:** assume \( s_1 \subseteq t_1 \) and consider all cases for a step \( s_1 \rightarrow s_2 \)

**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. $Conf \overset{\text{def}}{=} \mathcal{M}_f(Q \times \mathbb{R}_{\geq 0})$
**Example 2: Timed-arc Nets**

Control states of individual processes taken from some finite set $Q = \{r, c, a, q, \ldots\}$ (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. $Conf \overset{\text{def}}{=} \mathcal{M}_f(Q \times \mathbb{R}_{\geq 0})$

TPNs have rules like e.g. $\delta = \left\{ \begin{array}{l} c \in [1;2) \quad r \in [0;2] \\ q \in [2;\infty) \quad q \in [1;1] \\ a \in (0;4) \end{array} \right\}$
**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. \[ \text{Conf} \overset{\text{def}}{=} \mathcal{M}_f (Q \times \mathbb{R}_{\geq 0}) \]

TPNs have rules like e.g.
\[
\delta = \begin{cases} 
  c \in [1;2) & \Rightarrow \ r \in [0;2] \\
  q \in [2;\infty) & \Rightarrow \ q \in [1;1] \\
  a \in (0;4) & 
\end{cases}
\]

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. \( Conf \overset{\text{def}}{=} \mathcal{M}_f(Q \times \mathbb{R}_{\geq 0}) \)

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

Yielding steps like
\[ s = \{c : 1.4, r : 3.0, q : 2.5\} \implies \{r : 3.0, r : 0.73, q : 1.0, a : 2.1\} = s' \]

also time-elapse steps like
\[ s' = \{r : 3.0, r : 0.73, q : 1.0, s : 2.1\} \overset{+0.8}{\implies} \{r : 3.8, r : 1.53, q : 1.8, a : 2.9\} \]
Example 2: Timed-Arc Nets

TPNs have rules like e.g.

\[ \delta = \left\{ \begin{array}{c}
  c \in [1; 2) \\
  q \in [2; \infty) \\
  a \in (0; 4)
\end{array} \rightarrow \begin{array}{c}
  r \in [0; 2] \\
  q \in [1; 1] \\
  a \in (0; 4)
\end{array} \right\} \]

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 \((\mathcal{M}_f(Q \times \mathbb{R}_{\geq 0}), \subseteq)\) is not wqo —since already \((\mathbb{R}_{\geq 0}, =)\) is not
TIMED-ARC NETS ARE WSTS

\[ s = \{ r : 3.0, r : 0.73, q : 1.0, a : 2.1 \} \quad \approx \quad \tilde{s} = \{ r : 3, q : 1 \} \cdot \{ a : 2 \} \cdot \{ r : 0 \} \]
**TIMED-ARC NETS ARE WSTS**

\[
s = \{r : 3.0, r : 0.73, q : 1.0, a : 2.1\} \approx \tilde{s} = \{r : 3, q : 1\} \bullet \{a : 2\} \bullet \{r : 0\}
\]

\[
\begin{bmatrix}
0 < x_1 < x_2 < 1 \\
\mid \quad \mid \quad \mid \\
r : 3 \quad a : 2(+x_1) \quad r : 0(+x_2) \\
q : 1
\end{bmatrix}
\]

In general \(\tilde{s}\) is a sequence over \(\mathcal{M}_f(Q \times \{0, 1, 2, 3, 4, 5+\})\).
Timed-arc Nets are WSTS

In general, $\tilde{s}$ 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).

\[
\{r : 3, q : 1\} \bullet \{a : 2\} \bullet \{r : 0\} \xrightarrow{+?} \{\} \bullet \{r : 3, q : 1\} \bullet \{a : 2\} \bullet \{r : 0\} \\
\xrightarrow{+?} \{r : 1\} \bullet \{r : 3, q : 1\} \bullet \{a : 2\} \rightarrow \cdots
\]
Timed-arc Nets are WSTS

In general $\tilde{s}$ 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).

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

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

**Fact.** This new semantics is monotonic wrt pointed sequence embedding $\leq^*$ over $(\mathcal{M}_f(Q \times \{0, \ldots, 4, 5+\}))^+$, a wqo. Hence TPN are WSTS!!!
Part 2 Verification of WSTS
**TERMINATION**

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

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

**Recall:** $s_0, s_1, s_2, \ldots, s_n$ is good $\Leftrightarrow$ there exist $i < j$ s.t. $s_i \leq s_j$

**Proof:** “$\Rightarrow$” by def of wqo. “$\Leftarrow$” by simulating $s_i \vdash s_j$ from $s_j$

⇒ one can decide Termination for a WSTS $S$ by enumerating all finite runs from $s_{\text{init}}$ 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 $S = (S, \rightarrow, \ldots)$ and a state $s_{\text{init}}$, whether $S$ has no infinite runs starting from $s_{\text{init}}$

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

**Recall:** $s_0, s_1, s_2, \ldots, s_n$ is good $\defeq$ there exist $i < j$ s.t. $s_i \leq s_j$

**Proof:** “$\Rightarrow$” by def of wqo. “$\Leftarrow$” by simulating $s_i \trans{s_{\text{init}}} s_j$ from $s_j$

⇒ one can decide Termination for a WSTS $S$ by enumerating all finite runs from $s_{\text{init}}$ 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 $S = (S, \rightarrow, \ldots)$ and a state $s_{init}$, whether $S$ has no infinite runs starting from $s_{init}$.

**Lem. [Finite Witnesses for Infinite Runs]**
A WSTS $S$ 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 $\iff$ there exist $i < j$ s.t. $s_i \leq s_j$

**Proof:** “$\Rightarrow$” by def of wqo. “$\Leftarrow$” by simulating $s_i \rightarrow s_j$ from $s_j$

$\Rightarrow$ one can decide Termination for a WSTS $S$ 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 $S = (S, \rightarrow, \ldots)$, a state $s_{\text{init}}$ and a target state $t$, whether $S$ has a **covering run** $s_{\text{init}} \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_n$ with $s_n \geq t$.

This is equivalent to having a **covering pseudorun** of the form

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

**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 $S$ has a covering pseudorun from $s_{\text{init}}$ 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 asks, given \( S = (S, \rightarrow, \ldots) \), a state \( s_{\text{init}} \) and a target state \( t \), whether \( S \) has a covering run \( s_{\text{init}} \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_n \) with \( s_n \geq t \).

This is equivalent to having a covering pseudorun of the form

\[
s_{\text{init}} = s_0 \geq t_0 \rightarrow s_1 \geq t_1 \rightarrow s_2 \geq \cdots \rightarrow t_{n-1} \rightarrow s_n \geq t_n = t
\]

**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 \( S \) has a covering pseudorun from \( s_{\text{init}} \) 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 asks, given $S = (S, \rightarrow, \ldots)$, a state $s_{\text{init}}$ and a target state $t$, whether $S$ has a covering run $s_{\text{init}} \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_n$ with $s_n \geq t$.

This is equivalent to having a covering pseudorun of the form

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

**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 $S$ has a covering pseudorun from $s_{\text{init}}$ 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 $S = (S, \rightarrow, \ldots)$, a state $s_{init}$ and a target state $t$, whether $S$ has a covering run $s_{init} \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_n$ with $s_n \geq t$.

This is equivalent to having a covering pseudorun of the form

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

**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 $S$ 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 asks, given $S = (S, \rightarrow, \ldots)$, a state $s_{\text{init}}$ and a target state $t$, whether $S$ has a covering run $s_{\text{init}} \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_n$ with $s_n \geq t$.

This is equivalent to having a covering pseudorun of the form

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

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 $S$ has a covering pseudorun from $s_{\text{init}}$ to $t$ iff it has one that is minimal and reverse-bad

⇒ one can decide Coverability by enumerating all pseudoruns ending in $t$ (hence backward chaining) that are minimal and reverse-bad.
Coverability asks, given $S = (S, \rightarrow, \ldots)$, a state $s_{\text{init}}$ and a target state $t$, whether $S$ has a covering run $s_{\text{init}} \rightarrow s_1 \rightarrow s_2 \ldots \rightarrow s_n$ with $s_n \geq t$.

This is equivalent to having a covering pseudorun of the form

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

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 $S$ has a covering pseudorun from $s_{\text{init}}$ 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
This broadcast protocol terminates: all its runs are bad sequences, hence are finite

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

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

In all cases, \( s_i \not\subset s_j \). QED
This broadcast protocol terminates: all its runs are bad sequences, hence are finite

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

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

In all cases, $s_i \not\subseteq s_j$. QED
This broadcast protocol terminates: all its runs are bad sequences, hence are finite

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

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

In all cases, \( s_i \not\in s_j \). QED
Broadcast Protocols Take Their Time

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

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

Then: $\{c, q, r^n\} \rightarrow^* \{c, q^{2^n}, r^{n-1}\} \rightarrow^* \{c, q^{\text{tower}(n)}\}$
“Doubling” run: \( \{c^n, q, (\bot^*)\} \xrightarrow{a^n} \{a^{2n}, q, (\bot^*)\} \xrightarrow{m} \{c^{2n}, (\bot^*)\} \)

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

Then: \( \{c, q, r^n\} \rightarrow \{c, q^{2^n}, r^{n-1}\} \rightarrow \{c, q^{\text{tower}(n)}\} \)
**Broadcast Protocols Take Their Time**

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

**Building up:**
\[
\begin{align*}
\{c^{20}, q^n, r\} & \xrightarrow{a^{20}m} \{c^{21}, q^{n-1}, r\} \\
& \xrightarrow{a^{21}m} \{c^{22}, q^{n-2}, r\} \\
& \cdots \\
& \xrightarrow{a^{2n-1}m} \{c^{2n}, r\} \\
& \xrightarrow{d} \{c^{20}, q^{2n}\}
\end{align*}
\]

**Then:**
\[
\begin{align*}
\{c, q, r^n\} & \xrightarrow{*} \{c, q^{2n}, r^{n-1}\} \\
& \xrightarrow{*} \{c, q^\text{tower}(n)\}
\end{align*}
\]

where \(\text{tower}(n) \overset{\text{def}}{=} 2^{2^\cdots^2}\) \(n\) times
"Doubling" run: \( \{ c^n, q, (\bot*) \} \xrightarrow{a^n} \{ a^{2n}, q, (\bot*) \} \xrightarrow{m} \{ c^{2n}, (\bot*) \} \)

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

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

\[ \Rightarrow \text{Runs of terminating systems may have nonelementary lengths} \]
\[ \Rightarrow \text{Running time of termination verification algorithm is not elementary (for broadcast protocols)} \]
**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 \((\mathbb{N}^k, \leq_x)\), 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 \((\mathbb{N}^k, \leq_X)\), 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 \((\mathbb{N}^k, \leq \times)\), one can find arbitrarily long bad sequences:

- 999, 998, …, 1, 0
- \((2,2), (2,1), (2,0), (1,999), \ldots, (1,0), (0,999999999), \ldots\)

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

The runs of a broadcast protocol don’t play these tricks!
**Complexity Analysis?**

**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 \((\mathbb{N}^k, \leq_x)\), one can find arbitrarily long bad sequences:

— 999, 998, ..., 1, 0
— (2,2), (2,1), (2,0), (1,999), ..., (1,0), (0,999999999), ...

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

The runs of a broadcast protocol don’t play these tricks!
CONTROLLED BAD SEQUENCES

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

Def. A sequence $x_0, x_1, \ldots$ is controlled $\iff |x_i| \leq g^i(n_0)$ for all $i = 0, 1, \ldots$

Fact. For a fixed wqo $(\mathcal{A}, \leq, |.|)$ and control $(n_0, g)$, there is a bound on the length of controlled bad sequences. Write $L_{g,\mathcal{A}}(n_0)$ for this maximum length.

Length Function Theorem for $(\mathbb{N}^k, \leq_X)$:
— $L_{g,\mathbb{N}^k}(n_0) \leq g^{\omega^k}(n_0)$
— $L_{g,\mathbb{N}^k}$ is in $\mathcal{F}_{k+m-1}$ for $g$ in $\mathcal{F}_m$ [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 $g : \mathbb{N} \rightarrow \mathbb{N}$.

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

**Fact.** For a fixed wqo $(\mathcal{A}, \leq, |.|)$ and control $(n_0, g)$, there is a bound on the length of controlled bad sequences. Write $L_{g, \mathcal{A}}(n_0)$ for this maximum length.

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

- $L_{g, \mathbb{N}^k}(n_0) \leq g^{\omega^k}(n_0)$
- $L_{g, \mathbb{N}^k}$ is in $\mathcal{F}_{k+m-1}$ for $g$ in $\mathcal{F}_m$ [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 $\overset{\text{def}}{\iff} |x_i| \leq g^i(n_0)$ for all $i = 0, 1, \ldots$

**Fact.** For a fixed wqo $(\mathcal{A}, \leq, |.|)$ and control $(n_0, g)$, there is a bound on the length of controlled bad sequences. Write $L_{g, \mathcal{A}}(n_0)$ for this maximum length.

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

- $L_{g, \mathbb{N}^k}(n_0) \leq g^{\omega^k}(n_0)$
- $L_{g, \mathbb{N}^k}$ is in $\mathcal{F}_{k+m-1}$ for $g$ in $\mathcal{F}_m$ [Figueira'2SS’11]

(more later on Fast-Growing Hierarchy)
**Applying to Broadcast Protocols**

**Fact.** The runs explored by the Termination algorithm are controlled with \( n_0 = |s_{\text{init}}| \) and \( g = \text{Succ} : \mathbb{N} \to \mathbb{N} \).

\[ \Rightarrow \] Time/space bound in \( \mathcal{F}_{k-1} \) for broadcast protocols with \( k \) states, and in \( \mathcal{F}_\omega \) when \( k \) is not fixed.

**Fact.** The minimal pseudoruns explored by the backward-chaining Coverability algorithm are controlled by \( |t| \) and \( \text{Succ} \).

\[ \Rightarrow \cdots \text{ same upper bounds } \cdots \]

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 \( n_0 = |s_{\text{init}}| \) and \( g = \text{Succ} : \mathbb{N} \rightarrow \mathbb{N} \).

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

**Fact.** The minimal pseudoruns explored by the backward-chaining Coverability algorithm are controlled by \(|t|\) and \( \text{Succ} \).

\[ \Rightarrow \text{same upper bounds} \]

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 \( n_0 = |s_{\text{init}}| \) and \( g = \text{Succ} : \mathbb{N} \rightarrow \mathbb{N} \).

\[ \Rightarrow \] Time/space bound in \( F_{k-1} \) for broadcast protocols with \( k \) states, and in \( F_\omega \) when \( k \) is not fixed.

Fact. The minimal pseudoruns explored by the backward-chaining Coverability algorithm are controlled by \( |t| \) and \( \text{Succ} \).

\[ \Rightarrow \cdots \text{same upper bounds} \cdots \]

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 $g = \text{Double} : x \mapsto 2x$ 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

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

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

These bounds are optimal!

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

— Verification of Broadcast protocols is $\mathcal{F}_\omega$-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 $n_0 = |s_{\text{init}}|$ and $g : x \mapsto x + |N|$, or with $n_0 = |s_{\text{init}}| + |N|$ and $g = \text{Double} : x \mapsto 2x$ if we want fixed $g$.

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

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

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

These bounds are optimal!

— Verification of Timed-arc nets is $\mathcal{F}_{\omega \omega \omega}$-complete [HSS ’12]
— Verification of Broadcast protocols is $\mathcal{F}_{\omega}$-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 $n_0 = |s_{\text{init}}|$ and $g : x \mapsto x + |N|$, or with $n_0 = |s_{\text{init}}| + |N|$ and $g = \text{Double} : x \mapsto 2x$ if we want fixed $g$.

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

$$L_{g,Conf} \in F_{\omega \omega k} \text{ where } k = |Q| \times M$$

$\Rightarrow$ Time/space bound in $F_{\omega \omega \omega}$ for Timed-arc Nets verification

These bounds are optimal!

— Verification of Timed-arc nets is $F_{\omega \omega \omega}$-complete [HSS ’12]
— Verification of Broadcast protocols is $F_{\omega}$-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 \((F_\alpha)_{\alpha \in \text{Ord}}\) of functions \(\mathbb{N} \rightarrow \mathbb{N}\)

- \(F_0(x) \overset{\text{def}}{=} x + 1\)
- \(F_{\alpha+1}(x) \overset{\text{def}}{=} F_\alpha(F_\alpha(\ldots F_\alpha(x) \ldots))\)
- \(F_\omega(x) \overset{\text{def}}{=} F_{x+1}(x)\)

This 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 \(F_\alpha\) that is not primitive recursive.

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

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

\(\mathcal{F}_\alpha \overset{\text{def}}{=} \) all functions computable in time \(F^{O(1)}_\alpha\) (very robust).
The Fast-Growing Hierarchy

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

\[
F_0(x) \overset{\text{def}}{=} x + 1 \\
F_{\alpha + 1}(x) \overset{\text{def}}{=} F_\alpha(F_\alpha(\ldots F_\alpha(x) \ldots)) \\
F_\omega(x) \overset{\text{def}}{=} F_{x+1}(x)
\]

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 \(F_\alpha\) that is not primitive recursive.

\[
F_\lambda(x) \overset{\text{def}}{=} F_{\lambda_\chi}(x)\quad \text{for } \lambda \text{ a limit ordinal with a fundamental sequence } \\
\lambda_0 < \lambda_1 < \lambda_2 < \ldots < \lambda.
\]

\[
\text{E.g. } F_{\omega^2}(x) = F_{\omega \cdot (x+1)}(x) = F_{\omega \cdot x + x + 1}(x) = F_{\omega \cdot x + x + 1}(F_{\omega \cdot x + x}(F_{\omega \cdot x + x}(\ldots F_{\omega \cdot x + x}(x)\ldots)))
\]

\(\mathcal{F}_\alpha \overset{\text{def}}{=} \text{all functions computable in time } F_\alpha^{O(1)} \text{ (very robust)}.\)
An ordinal-indexed family \((F_\alpha)_{\alpha \in \text{Ord}}\) of functions \(\mathbb{N} \rightarrow \mathbb{N}\)

- \(F_0(x) \overset{\text{def}}{=} x + 1\)
- \(F_{\alpha + 1}(x) \overset{\text{def}}{=} F_\alpha(F_\alpha(...F_\alpha(x)...))\)
- \(F_\omega(x) \overset{\text{def}}{=} F_{x+1}(x)\)

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 \(F_\alpha\) that is not primitive recursive.

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

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

- \(O(1)\) (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