Branching in Well-Structured Transition Systems

Sylvain Schmitz

CSL 2021, January 25, 2021
well-structured transition systems (WSTS)
  ▶ algorithmic framework for decidability

complexity analysis for WSTS
  ▶ controlled bad sequences
  ▶ subrecursive complexity

this talk
  ▶ branching systems
  ▶ examples:
    ▶ forward XPath [Jurdziński & Lazić ’11],
    ▶ relevance logics [Kripke ’59; Urquhart ’99]
  ▶ lifting branching systems to linear ones
  ▶ leaf sub-coverability
well-structured transition systems (WSTS)
  ▶ algorithmic framework for decidability

complexity analysis for WSTS
  ▶ controlled bad sequences
  ▶ subrecursive complexity

this talk
  ▶ branching systems
  ▶ examples:
    ▶ forward XPath [Jurdziński & Lazić ’11],
    ▶ relevance logics [Kripke ’59; Urquhart ’99]
  ▶ lifting branching systems to linear ones
  ▶ leaf sub-coverability
Outline

well-structured transition systems (WSTS)
  ▶ algorithmic framework for decidability

complexity analysis for WSTS
  ▶ controlled bad sequences
  ▶ subrecursive complexity

this talk
  ▶ branching systems
  ▶ examples:
    ▶ forward XPath [Jurdziński & Lazić ’11],
    ▶ relevance logics [Kripke ’59; Urquhart ’99]
  ▶ lifting branching systems to linear ones
  ▶ leaf sub-coverability
WELL STRUCTURED TRANSITION SYSTEMS

[Abdulla, Čerãns, Jonsson & Tsay ‘00; Finkel & Schnoebelen ‘01]

- general algorithmic framework
- algorithms for several verification problems
- exploit an underlying well-quasi-order (wqo) for termination
Well Structured Transition Systems

[Abdulla, Čerāns, Jonsson & Tsay ’00; Finkel & Schnoebelen ’01]

- general algorithmic framework
- algorithms for several verification problems
- exploit an underlying well-quasi-order (wqo) for termination

Abstract

Well-structured transition systems (WSTS) are a general class of infinite-state systems for which decidability results rely on the existence of a well-quasi-ordering between states that is compatible with the transitions. In this article, we provide an extensive treatment of the WSTS, which consist of a finite control part operating on an infinite data domain.

Keywords: Infinite systems; Verification; Well-structured ordering

1. Introduction

Well-structured transition systems everywhere!
Well Structured Transition Systems

[Abdulla, Čerāns, Jonsson & Tsay ’00; Finkel & Schnoebelen ’01]

- general algorithmic framework
- algorithms for several verification problems
- exploit an underlying well-quasi-order (wqo) for termination
**Example: Lossy Counter Machines**

Example

![Diagram of Lossy Counter Machines]

Lossy Semantics

$q_1(0,2) \xrightarrow{c_1++} q_2(1,1) \xrightarrow{c_2 \neq 0} q_1(0,0)$
**Example: Lossy Counter Machines**

**Example**

Lossy Semantics

\[ q_1(0, 2) \xrightarrow{c_1++} q_2(1, 1) \xrightarrow{c_2 = 0} q_1(0, 0) \]
**Example: Lossy Counter Machines**

**Example**

![Diagram of a state transition system](image)

**Lossy Semantics**

\[
q_1(0, 2) \xrightarrow{c_1++} q_2(1, 1) \xrightarrow{c_2 \equiv 0} q_1(0, 0)
\]

\[
\lor
q_1(0, 2) \xrightarrow{c_1++} q_2(1, 2)
\]

\[
\land
\]
**Example: Lossy Counter Machines**

Example

Lossy Semantics

\[ q_1(0,2) \xrightarrow{c_1++} q_2(1,1) \xrightarrow{c_2 \not= 0} q_1(0,0) \]

\[ q_2(1,0) \xrightarrow{c_2 \not= 0} q_1(1,0) \]
**COVERABILITY**

Coverability Problem for LCM

**input** an LCM $M$, initial configuration $q_0(v_0)$, target configuration $q_f(v_f)$

**question** $\exists v \geq v_f . q_0(v_0) \rightarrow^*_l q_f(v)$?
COVERABILITY

Coverability Problem for LCM

input an LCM $M$, initial configuration $q_0(v_0)$, target configuration $q_f(v_f)$

question $\exists v \geq v_f . q_0(v_0) \xrightarrow{\ell}^* q_f(v)$?

Remark
equivalent to reachability:

question $q_0(v_0) \xrightarrow{\ell}^* q_f(v_f)$?
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

Example: coverability of $q_2(1,1)$ in

\[
\begin{align*}
q_1 & \leftarrow c_1- \\
q_2 & \rightarrow c_1+ \\
q_3 & \rightarrow c_2+ \\
? & = c_2 = 0
\end{align*}
\]
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

**Example:** coverability of $q_2(1, 1)$ in

$$U_k \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1, 1). q(v) \xrightarrow{\leq_k} q_2(v') \}$$
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

**Example:** coverability of $q_2(1,1)$ in

$$U_0 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \rightarrow^\ell_0 q_2(v') \}$$
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

**Example:** coverability of $q_2(1,1)$ in

$$U_0 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \to_{\ell}^0 q_2(v') \}$$
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

**Example:** coverability of $q_2(1,1)$ in

$$U_1 \overset{\text{def}}{=} \{ q(\mathbf{v}) \mid \exists \mathbf{v}' \geq (1,1). q(\mathbf{v}) \rightarrow^1 q_2(\mathbf{v}') \}$$
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

**Example:** coverability of $q_2(1,1)$ in

$$U_1 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1) \cdot q(v) \rightarrow_\ell^1 q_2(v') \}$$

Diagram showing the states $q_1$, $q_2$, $q_3$ and their transitions.
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

Example: coverability of $q_2(1,1)$ in

$$U_2 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \rightarrow^{\leq 2} q_2(v') \}$$
**Backward Coverability Algorithm**

[Arnold & Latteux'78]

**Example:** coverability of $q_2(1,1)$ in

$$U_3 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \rightarrow_\ell^3 q_2(v') \}$$
Backward Coverability Algorithm

[Arnold & Latteux'78]

Example: coverability of $q_2(1,1)$ in

$$U_4 \overset{\text{def}}{=} \{ q(\mathbf{v}) \mid \exists \mathbf{v}' \geq (1,1). q(\mathbf{v}) \rightarrow_\ell^4 q_2(\mathbf{v}') \}$$
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

**Example:** coverability of $q_2(1,1)$ in

$$U_4 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \rightarrow_{\ell}^{\leq 4} q_2(v') \}$$
**Backward Coverability Algorithm**

Example: coverability of $q_2(1,1)$ in

$$U_5 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \rightarrow_{\ell}^{\leq 5} q_2(v') \}$$
Backward Coverability Algorithm

Example: coverability of $q_2(1,1)$ in

$$U_6 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \rightarrow^\ell \leq 6 q_2(v') \}$$
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

**Example:** coverability of $q_2(1,1)$ in

$$U_6 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \xrightarrow{\ell}^* q_2(v') \}$$
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

Example: coverability of $q_2(1,1)$ in

$$U_6 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \rightarrow^* q_2(v') \}$$

The sequence $U_0 \subsetneq U_1 \subsetneq U_3 \subsetneq U_4 \subsetneq U_5 \subsetneq U_6$ is defined by

$$U_0 \overset{\text{def}}{=} \uparrow q_2(1,1) \text{ and } U_{k+1} \overset{\text{def}}{=} U_k \cup \text{Pre}_\exists(U_k).$$
Backward Coverability Algorithm

[Arnold & Latteux’78]

Example: coverability of $q_2(1,1)$ in

$$U_6 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \rightarrow^* q_2(v') \}$$

This sequence of upwards-closed sets $U_0 \subset U_1 \subset U_3 \subset U_4 \subset U_5 \subset U_6$ is ascending and thus finite.
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

**Example:** coverability of \( q_2(1,1) \) in

\[
U_6 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \xrightarrow{\ell}^* q_2(v') \}
\]

The sequence \( q_2(1,1), q_1(0,1), q_3(0,0), q_1(1,0), q_2(1,0), q_1(0,0), q_2(0,0) \) is defined by picking \( x_0 \overset{\text{def}}{=} q_2(1,1) \) and \( x_{k+1} \in \min(\text{Pre}_\exists(\uparrow x_k) \setminus U_k) \).
**Backward Coverability Algorithm**

[Arnold & Latteux’78]

**Example:** coverability of $q_2(1,1)$ in

$$U_6 \overset{\text{def}}{=} \{ q(v) \mid \exists v' \geq (1,1). q(v) \rightarrow^\ast q_2(v') \}$$

This sequence $q_2(1,1), q_1(0,1), q_3(0,0), q_1(1,0), q_2(1,0), q_1(0,0), q_2(0,0)$ is a **bad** sequence and thus finite.
**Well-quasi-orders**

- \((X, \leq)\) is a **quasi-order** (qo) if \(\leq\) is transitive and reflexive.

- **Upwards-closure** of \(S \subseteq X\):
  \[
  \uparrow S = \{y \mid \exists x \in S. x \leq y\}
  \]
  (notation: \(\uparrow x = \uparrow \{x\}\) for \(x \in X\)).

- \(U \subseteq X\) is upwards-closed if \(\uparrow U = U\).

- A sequence \(x_0, x_1, \ldots\) over \(X\) is bad if \(\forall i < j. x_i \not\leq x_j\).
**WELL-QUASI-ORDERS**

- $(X, \leq)$ is a quasi-order (qo) if $\leq$ is transitive and reflexive

- **upwards-closure** of $S \subseteq X$: $\uparrow S \overset{\text{def}}{=} \{ y \mid \exists x \in S. x \leq y \}$
  (notation: $\uparrow x \overset{\text{def}}{=} \uparrow \{x\}$ for $x \in X$)

- $U \subseteq X$ is upwards-closed if $\uparrow U = U$

- A sequence $x_0, x_1, \ldots$ over $X$ is bad if $\forall i < j. x_i \not\leq x_j$
Well-quasi-orders

- $(X, \leq)$ is a quasi-order (qo) if $\leq$ is transitive and reflexive

- upwards-closure of $S \subseteq X$: $\uparrow S \overset{\text{def}}{=} \{ y \mid \exists x \in S. x \leq y \}$
  (notation: $\uparrow x \overset{\text{def}}{=} \uparrow\{ x \}$ for $x \in X$)

- $U \subseteq X$ is upwards-closed if $\uparrow U = U$

- a sequence $x_0, x_1, \ldots$ over $X$ is bad if $\forall i < j. x_i \not\leq x_j$
Well-quasi-orders

- $(X, \leq)$ is a quasi-order (qo) if $\leq$ is transitive and reflexive.
- Upwards-closure of $S \subseteq X$: $\uparrow S \overset{\text{def}}{=} \{y | \exists x \in S. x \leq y\}$
  (notation: $\uparrow x \overset{\text{def}}{=} \uparrow \{x\}$ for $x \in X$)
- $U \subseteq X$ is upwards-closed if $\uparrow U = U$
- A sequence $x_0, x_1, \ldots$ over $X$ is bad if $\forall i < j. x_i \not\leq x_j$
Well-quasi-orders

- \((X, \leq)\) is a quasi-order (qo) if \(\leq\) is transitive and reflexive

- upwards-closure of \(S \subseteq X\): \(\uparrow S \overset{\text{def}}{=} \{y \mid \exists x \in S. x \leq y\}\)
  (notation: \(\uparrow x \overset{\text{def}}{=} \uparrow \{x\}\) for \(x \in X\))

- \(U \subseteq X\) is upwards-closed if \(\uparrow U = U\)

- a sequence \(x_0, x_1, \ldots\) over \(X\) is bad if \(\forall i < j. x_i \not\preceq x_j\)
**Well-quasi-orders**

- $(X, \leq)$ is a **well-quasi-order (wqo)** if bad sequences are finite.

- Equivalently, ascending chain condition: ascending chains $U_0 \subset U_1 \subset \cdots$ of upwards-closed subsets are finite.

- Equivalently, finite basis property: non-empty subsets $S \subseteq X$ have at least one, and up to equivalence, finitely many minimal elements.

- Equivalently, $\leq$ is well-founded and has no infinite antichain.

- Etc.: very robust notion [Kruskal ’72]
**WELL-QUASI-ORDERS**

- $(X, \subseteq)$ is a well-quasi-order (wqo) if bad sequences are finite

- equivalently, **ascending chain condition**: ascending chains $U_0 \subsetneq U_1 \subsetneq \cdots$ of upwards-closed subsets are finite

- equivalently, finite basis property: non-empty subsets $S \subseteq X$ have at least one, and up to equivalence, finitely many minimal elements

- equivalently, $\subseteq$ is well-founded and has no infinite antichain

- etc.: very robust notion [Kruskal ’72]
**Well-quasi-orders**

- $(X, \subseteq)$ is a well-quasi-order (wqo) if bad sequences are finite.
- Equivalently, ascending chain condition: ascending chains $U_0 \subsetneq U_1 \subsetneq \cdots$ of upwards-closed subsets are finite.
- Equivalently, **finite basis property**: non-empty subsets $S \subseteq X$ have at least one, and up to equivalence, finitely many minimal elements.
- Equivalently, $\subseteq$ is well-founded and has no infinite antichain.
- Etc.: very robust notion [Kruskal ’72]
**Well-quasi-orders**

- $(X, \leq)$ is a well-quasi-order (wqo) if bad sequences are finite

- equivalently, ascending chain condition: ascending chains $U_0 \subsetneq U_1 \subsetneq \cdots$ of upwards-closed subsets are finite

- equivalently, finite basis property: non-empty subsets $S \subseteq X$ have at least one, and up to equivalence, finitely many minimal elements

- equivalently, $\leq$ is well-founded and has no infinite antichain

- etc.: very robust notion [Kruskal ’72]
**Well-quasi-orders**

- $(X, \leq)$ is a well-quasi-order (wqo) if bad sequences are finite.

- Equivalently, ascending chain condition: ascending chains $U_0 \subsetneq U_1 \subsetneq \cdots$ of upwards-closed subsets are finite.

- Equivalently, finite basis property: non-empty subsets $S \subseteq X$ have at least one, and up to equivalence, finitely many minimal elements.

- Equivalently, $\leq$ is well-founded and has no infinite antichain.

- Etc.: very robust notion [Kruskal '72]
Examples of WQOs

Ordinals

ordinal: well-founded linear order

bad sequences are descending sequences:

\[ \alpha \not\leq \beta \text{ iff } \alpha > \beta \]
Examples of WQOs

**Lemma** (Finite products, Dickson 1913)

If $(X, \leq_X)$ and $(Y, \leq_Y)$ are two wqos, then $(X \times Y, \leq)$ is a wqo for the product ordering:

$$\langle x, y \rangle \leq \langle x', y' \rangle \overset{\text{def}}{\iff} x \leq_X x' \land y \leq_Y y'.$$

**Example**

LCM configurations in $(Q \times \mathbb{N}^d, \leq)$ for a finite set $(Q, =)$
ordered transition system \((X, \leq, \rightarrow)\) with \(\rightarrow \subseteq X \times X\)

upward (reflexive) compatibility

\[\forall x, x', y
\]

\[\begin{align*}
x' \\
\forall / \\
x \quad \rightarrow \quad y
\end{align*}\]

\((X, \leq)\) is a wqo
ordered transition system \((X, \leq, \rightarrow)\) with \(\rightarrow \subseteq X \times X\)

upward (reflexive) compatibility

\[
\forall x, x', y \quad x' \\
\forall / \\
x \quad \rightarrow \quad y
\]

\((X, \leq)\) is a wqo
ordered transition system \((X, \leq, \rightarrow)\) with \(\rightarrow \subseteq X \times X\)

upward (reflexive) compatibility

\[
\forall x, x', y \quad \exists y' \\
\quad x' \quad \rightarrow \quad y' \\
\forall/ \quad \forall/ \\
\quad x \quad \rightarrow \quad y
\]

\((X, \leq)\) is a wqo
ordered transition system \((X, \leq, \rightarrow)\) with \(\rightarrow \subseteq X \times X\)

upward (reflexive) compatibility

\[
\forall x, x', y \\
\exists y' \\
\begin{array}{c}
x' \\
\rightarrow \\
\forall/ \\
x \\
\rightarrow \\
y
\end{array} = \\
\begin{array}{c}
x' \\
\rightarrow \\
\forall/ \\
y' \\
\rightarrow \\
y
\end{array}
\]

\((X, \leq)\) is a wqo
ordered transition system \((X, \leq, \rightarrow)\) with \(\rightarrow \subseteq X \times X\)

upward (reflexive) compatibility:

\[
\forall x, x', y \quad \exists y' \\
\begin{array}{c}
\begin{array}{c}
x' \\
\forall
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\rightarrow
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
y'
\end{array}
\end{array}
\]

\((X, \leq)\) is a \textit{wqo}\)
ordered transition system \((X, \leq, \to)\) with \(\to \subseteq X \times X\)

upward (reflexive) compatibility

\[
\forall x, x', y \quad = \quad \exists y' \\
\quad x' \quad \to \quad y' \\
\quad \forall / \quad \forall / \\
\quad x \quad \to \quad y
\]

Example: lossy counter machines

\[
q(v) \to^\ell q'(v') \quad \text{if} \\
q(v) \geq q(u) \to q'(u') \geq q'(v')
\]

\((X, \leq)\) is a wqo
WSTS

[Abdulla, Čerăns, Jonsson & Tsay ’00; Finkel & Schnoebelen ’01]

Coverability Problem

input ordered transition system \((X, \leq, \rightarrow)\), initial configuration \(s \in X\), target configuration \(t \in X\)

question \(\exists t' \geq t \cdot s \rightarrow^* t'\)?

**Theorem**

Coverability is decidable for WSTS, using the backward coverability algorithm.
**DOWNWARD WSTS**

[Finkel & Schnoebelen ‘01]

- **downward (reflexive) compatibility**

\[
\forall x, x', y \\
\quad x \quad \rightarrow \\ \\
\quad \forall/ \\ \\
\quad x'
\]

- **Subcoverability Problem**

Input: ordered transition system \((X, \leq, \rightarrow)\), initial configuration \(s \in X\), target configuration \(t \in X\)

Question: \(\exists t' \leq t \cdot s \rightarrow^* t'\)?

- Subcoverability is decidable for downward WSTS, using a forward subcoverability algorithm.
Downward WSTS

[Finkel & Schnoebelen ‘01]

- **Downward (reflexive) compatibility**
  \[
  \forall x, x', y \quad \exists y' \\
  x \quad \xrightarrow{\text{l}} \quad y \\
  \forall / \quad \forall / \\
  x' \quad = \quad y'
  \]

- **Subcoverability Problem**

  input ordered transition system \((X, \leq, \rightarrow)\), initial configuration \(s \in X\), target configuration \(t \in X\)

  question \(\exists t' \leq t \cdot s \rightarrow^* t'\)

- Subcoverability is decidable for downward WSTS, using a forward subcoverability algorithm.
DOWNWARD WSTS

[Finkel & Schnoebelen ‘01]

- **downward (reflexive) compatibility**

\[ \forall x, x', y \quad \exists y' \]

\[ x \rightarrow y \]

\[ = \]

\[ x' \rightarrow y' \]

- **Subcoverability Problem**

input ordered transition system \((X, \leq, \rightarrow)\), initial configuration \(s \in X\), target configuration \(t \in X\)

question \(\exists t' \leq t \cdot s \rightarrow^* t'\)?
DOWNWARD WSTS

[Finkel & Schnoebelen ’01]

- **downward (reflexive) compatibility**

  \[ \forall x, x', y \quad \exists y' \]

  \[ x \rightarrow y \]

- **Subcoverability Problem**

  input ordered transition system \((X, \leq, \rightarrow)\), initial configuration \(s \in X\), target configuration \(t \in X\)

  question \( \exists t' \leq t \cdot s \rightarrow^* t' \) ?

- **Subcoverability is decidable for downward WSTS, using a forward subcoverability algorithm.**
**DOWNWARD WSTS**

[Finkel & Schnoebelen ’01]

- **downward (reflexive) compatibility**
  \[
  \forall x, x', y \quad \exists y' \\
  x \rightarrow y \\
  \forall / \quad \forall / \\
  x' \rightarrow y'
  \]

  Example: incrementing counter machines
  \[
  q(v) \rightarrow_i q'(v') \text{ if} \\
  q(v) \leq q(u) \rightarrow q'(u') \leq q'(v')
  \]

- **Subcoverability Problem**

  input ordered transition system \((X, \leq, \rightarrow)\), initial configuration \(s \in X\), target configuration \(t \in X\)

  question
  \[
  \exists t' \leq t \cdot s \rightarrow^* t'?
  \]

- **Subcoverability** is decidable for downward WSTS, using a forward subcoverability algorithm.
Bad Sequences

Over a qo $(\mathcal{X}, \leq)$

- $x_0, x_1, \ldots$ is **bad** if $\forall i < j. x_i \not\leq x_j$

- $(\mathcal{X}, \leq)$ wqo if all bad sequences are **finite**
**Bad Sequences**

Over a qo \((X, \leq)\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j . x_i \not\leq x_j\)

- \((X, \leq)\) wqo if all bad sequences are finite...

... but can be of **arbitrary** length

**Example (in \(\mathbb{N}^2\))**
Bad Sequences

Over a qo \((X, \leq)\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j. x_i \not\leq x_j\)

- \((X, \leq)\) wqo if all bad sequences are finite...

... but can be of arbitrary length

Example (in \(\mathbb{N}^2\))

\(0, 2\)
**Bad Sequences**

Over a wqo \((X, \leq)\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j. x_i \not\leq x_j\)
- \((X, \leq)\) wqo if all bad sequences are finite...

... but can be of arbitrary length

Example (in \(\mathbb{N}^2\))

\((0,2), (2,1)\)
Bad Sequences

Over a qo $(X, \leq)$

▶ $x_0, x_1, \ldots$ is bad if $\forall i < j . x_i \nleq x_j$

▶ $(X, \leq)$ wqo if all bad sequences are finite...

... but can be of arbitrary length

Example (in $\mathbb{N}^2$)

$(0, 2), (2, 1), (0, 1)$
Bad Sequences

Over a qo \((X, \leq)\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j. x_i \not\leq x_j\)

- \((X, \leq)\) wqo if all bad sequences are finite...

... but can be of arbitrary length

Example (in \(\mathbb{N}^2\))

\((0, 2), (2, 1), (0, 1), (6, 0)\)
Bad Sequences

Over a qo $(X, \leq)$

- $x_0, x_1, \ldots$ is bad if $\forall i < j. x_i \not\leq x_j$

- $(X, \leq)$ wqo if all bad sequences are finite...

... but can be of arbitrary length

Example (in $\mathbb{N}^2$)

$(0, 2), (2, 1), (0, 1), (6, 0), (5, 0)$
Bad Sequences

Over a qo $(X, \leq)$

- $x_0, x_1, \ldots$ is bad if $\forall i < j . x_i \not\leq x_j$

- $(X, \leq)$ wqo if all bad sequences are finite...

... but can be of arbitrary length

Example (in $\mathbb{N}^2$)

$(0, 2), (2, 1), (0, 1), (6, 0), (5, 0), (4, 0)$
Bad Sequences

Over a qos $(X, \leq)$

- $x_0, x_1, \ldots$ is bad if $\forall i < j . x_i \not\leq x_j$
- $(X, \leq)$ wqo if all bad sequences are finite…

… but can be of arbitrary length

Example (in $\mathbb{N}^2$)

$(0,2), (2,1), (0,1), (6,0), (5,0), (4,0), (3,0)$
**Bad Sequences**

Over a qo \((X, \leq)\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j. x_i \not\leq x_j\)

- \((X, \leq)\) wqo if all bad sequences are finite...

... but can be of arbitrary length

**Example (in \(\mathbb{N}^2\))**

\[(0, 2), (2, 1), (0, 1), (6, 0), (5, 0), (4, 0), (3, 0), (2, 0)\]
**Bad Sequences**

Over a qo \((\mathcal{X}, \sqsubseteq)\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j . x_i \not\sqsubseteq x_j\)

- \((\mathcal{X}, \sqsubseteq)\) wqo if all bad sequences are finite...

... but can be of *arbitrary* length

Example (in \(\mathbb{N}^2\))

\[(0, 2), (2, 1), (0, 1), (6, 0), (5, 0), (4, 0), (3, 0), (2, 0), (1, 0)\]
Bad Sequences

Over a qo $(X, \leq)$

- $x_0, x_1, \ldots$ is bad if $\forall i < j . x_i \not\leq x_j$
- $(X, \leq)$ wqo if all bad sequences are finite...

... but can be of arbitrary length

Example (in $\mathbb{N}^2$)

$(0, 2), (2, 1), (0, 1), (6, 0), (5, 0), (4, 0), (3, 0), (2, 0), (1, 0), (0, 0)$
**CONTROLLED BAD SEQUENCES**

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j . x_i \not\leq x_j\)

- \(x_0, x_1, \ldots\) is **controlled** by \(g : \mathbb{N} \to \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if

  \[
  \forall i . \|x_i\| \leq g^i(n_0)
  \]

[Cichoń & Tahhan Bittar’98]
**Controlled Bad Sequences**

Over a qo $(X, \leq)$ with norm $\| \cdot \|$:

- $x_0, x_1, \ldots$ is bad if $\forall i < j. x_i \nleq x_j$

- $x_0, x_1, \ldots$ is controlled by $g: \mathbb{N} \rightarrow \mathbb{N}$ and $n_0 \in \mathbb{N}$ if $\forall i. \| x_i \| \leq g^i(n_0)$

[Cichoń & Tahhan Bittar’98]

Example (in $\mathbb{N}^2$ with $n_0 = 2$ and $g(n) = n + 1$)

$(0, 2)$
**CONTROLLED BAD SEQUENCES**

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j . x_i \not\preceq x_j\)
- \(x_0, x_1, \ldots\) is controlled by \(g: \mathbb{N} \rightarrow \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if
  \[
  \forall i . \| x_i \| \leq g^i(n_0)
  \]

[Cichoń & Tahhan Bittar’98]

Example (in \(\mathbb{N}^2\) with \(n_0 = 2\) and \(g(n) = n + 1\))

\((0,2), (2,1)\)
**CONTROLLED BAD SEQUENCES**

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j \cdot x_i \not\leq x_j\)

- \(x_0, x_1, \ldots\) is controlled by \(g: \mathbb{N} \rightarrow \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if
  \[
  \forall i \cdot \|x_i\| \leq g^i(n_0)
  \]

[Cichoń & Tahhan Bittar’98]

Example (in \(\mathbb{N}^2\) with \(n_0 = 2\) and \(g(n) = n + 1\))

\((0,2), (2,1), (0,1)\)
**controlled bad sequences**

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j . x_i \not\leq x_j\)

- \(x_0, x_1, \ldots\) is controlled by \(g: \mathbb{N} \to \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if
  \[\forall i . \|x_i\| \leq g^i(n_0)\]

[Cichoń & Tahhan Bittar’98]

Example (in \(\mathbb{N}^2\) with \(n_0 = 2\) and \(g(n) = n + 1\))

\((0,2), (2,1), (0,1), (5,0)\)
**Controlled Bad Sequences**

Over a qo \((X, \leq)\) with norm \(||\cdot||\):

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j. x_i \not\leq x_j\)
- \(x_0, x_1, \ldots\) is controlled by \(g: \mathbb{N} \rightarrow \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if

  \[\forall i. ||x_i|| \leq g^i(n_0)\]

[Cichoń & Tahhan Bittar'98]

Example (in \(\mathbb{N}^2\) with \(n_0 = 2\) and \(g(n) = n + 1\))

\((0,2), (2,1), (0,1), (5,0), (4,0)\)
**Controlled Bad Sequences**

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\):

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j. x_i \not\leq x_j\)

- \(x_0, x_1, \ldots\) is **controlled** by \(g: \mathbb{N} \to \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if
  \[
  \forall i. \|x_i\| \leq g^i(n_0)
  \]

[Cichoń & Tahhan Bittar’98]

Example (in \(\mathbb{N}^2\) with \(n_0 = 2\) and \(g(n) = n + 1\))

\((0, 2), (2, 1), (0, 1), (5, 0), (4, 0), (3, 0)\)
**Controlled Bad Sequences**

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j . x_i \not\leq x_j\)

- \(x_0, x_1, \ldots\) is **controlled** by \(g: \mathbb{N} \to \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if
  \[
  \forall i . \| x_i \| \leq g^i(n_0)
  \]

[Cichoń & Tahhan Bittar’98]

Example (in \(\mathbb{N}^2\) with \(n_0 = 2\) and \(g(n) = n + 1\))

- \((0, 2), (2, 1), (0, 1), (5, 0), (4, 0), (3, 0), (2, 0)\)
**CONTROLLED BAD SEQUENCES**

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j \cdot x_i \not\approx x_j\)

- \(x_0, x_1, \ldots\) is **controlled** by \(g : \mathbb{N} \rightarrow \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if
  \[
  \forall i, \|x_i\| \leq g^i(n_0)
  \]

[Cichoń & Tahhan Bittar’98]

Example (in \(\mathbb{N}^2\) with \(n_0 = 2\) and \(g(n) = n + 1\))

\((0,2), (2,1), (0,1), (5,0), (4,0), (3,0), (2,0), (1,0)\)
**CONTROLLED BAD SEQUENCES**

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j. \ x_i \not\leq x_j\)

- \(x_0, x_1, \ldots\) is **controlled** by \(g: \mathbb{N} \rightarrow \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if
  \[
  \forall i. \ |x_i| \leq g^i(n_0)
  \]

[Cichoń & Tahhan Bittar’98]

Example (in \(\mathbb{N}^2\) with \(n_0 = 2\) and \(g(n) = n + 1\))

\((0, 2), (2, 1), (0, 1), (5, 0), (4, 0), (3, 0), (2, 0), (1, 0), (0, 0)\)
CONTROLLED BAD SEQUENCES

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\):

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j . x_i \not\leq x_j\)

- \(x_0, x_1, \ldots\) is controlled by \(g : \mathbb{N} \to \mathbb{N}\) and \(n_0 \in \mathbb{N}\) if
  \[\forall i . \| x_i \| \leq g^i(n_0)\]

[Cichoń & Tahhan Bittar’98]

PROPOSITION

In a wqo \((X, \leq)\), if \(\forall n \{ x \in X \mid \| x \| \leq n \}\) is finite, then \((g, n_0)\)-controlled bad sequences have a maximal length, denoted \(L_{g,X}(n_0)\).
**Controlled Bad Sequences**

Over a qo \((X, \leq)\) with norm \(\| \cdot \|\)

- \(x_0, x_1, \ldots\) is bad if \(\forall i < j. x_i \not\leq x_j\)

- \(x_0, x_1, \ldots\) is controlled by \(g: \mathbb{N} \to \mathbb{N}\)

and \(n_0 \in \mathbb{N}\) if

\[\|x_0\| \leq n_0\] and \(\forall i. \|x_{i+1}\| \leq g(\|x_i\|)\)

**Corollary**

In a wqo \((X, \leq)\), if \(\forall n \{x \in X \mid \|x\| \leq n\}\) is finite, then

\((g, n_0)\)-controlled bad sequences have a maximal length,

denoted \(L_{g,X}(n_0)\).

Length function theorems give explicit upper bounds on

\(L_{g,X}(n_0)\) for various \(X\) in terms of fast-growing functions \(F_\alpha\).
Fast-growing Complexity

[Löb & Wainer ’70]

\[ F_0(x) = x + 1 \]
\[ F_1(x) = \underbrace{F_0 \circ \cdots \circ F_0}_x(x) = 2x + 1 \]
\[ F_2(x) = \underbrace{F_1 \circ \cdots \circ F_1}_x(x) \approx 2^x \]
\[ F_3(x) = \underbrace{F_2 \circ \cdots \circ F_2}_x(x) \approx \text{tower}(x) \]
\[ \vdots \]
\[ F_\omega(x) = \underbrace{F_{x+1}}_{x+1 \text{ times}}(x) \approx \text{ackermann}(x) \]
\[ F_{\omega+1}(x) = \underbrace{F_\omega \circ \cdots \circ F_\omega}_x(x) \]
\[ \vdots \]
\[ F_{\omega \cdot 2}(x) = F_{\omega + x + 1}(x) \]
\[ \vdots \]
\[ F_{\omega \cdot 3}(x) = F_{\omega \cdot 2 + x + 1}(x) \]
\[ \vdots \]
\[ F_{\omega 2}(x) = F_{\omega \cdot (x+1)}(x) \]
\[ \vdots \]
\[ F_{\omega \omega}(x) = F_{\omega x + 1}(x) \]

Yields corresponding complexity classes [S. ’16]
Fast-growing Complexity

[Löb & Wainer '70]

\[ F_0(x) = x + 1 \]
\[ F_1(x) = \underbrace{F_0 \circ \cdots \circ F_0}_x(x) = 2x + 1 \]
\[ F_2(x) = \underbrace{F_1 \circ \cdots \circ F_1}_x(x) \approx 2^x \]
\[ F_3(x) = \underbrace{F_2 \circ \cdots \circ F_2}_x(x) \approx \text{tower}(x) \]
...  
\[ F_\omega(x) = F_{x+1}(x) \approx \text{ackermann}(x) \]
\[ F_{\omega+1}(x) = \underbrace{F_\omega \circ \cdots \circ F_\omega}_x(x) \]
...  
\[ F_{\omega\cdot 2}(x) = F_{\omega+x+1}(x) \]
...  
\[ F_{\omega\cdot 3}(x) = F_{\omega\cdot 2+x+1}(x) \]
...  
\[ F_{\omega^2}(x) = F_{\omega}(x+1)(x) \]
...  
\[ F_{\omega\omega}(x) = F_{\omega x+1}(x) \]

Yields corresponding complexity classes [S. '16]

Examples: Coverability is

- \( F_\omega \)-complete for lossy counter systems
- \( F_{\omega\cdot 2} \)-complete for \( \nu \)-Petri nets
- \( F_{\omega\omega} \)-complete for lossy channel systems
Fast-growing Complexity

[Löb & Wainer ’70]

\[
F_0(x) = x + 1
\]

\[
F_1(x) = \underbrace{F_0 \circ \cdots \circ F_0}_x(x) = 2x + 1
\]

\[
F_2(x) = \underbrace{F_1 \circ \cdots \circ F_1}_x(x) \approx 2^x
\]

\[
F_3(x) = \underbrace{F_2 \circ \cdots \circ F_2}_x(x) \approx \text{tower}(x)
\]

\[
F_\omega(x) = F_{x+1}(x) \approx \text{ackermann}(x)
\]

\[
F_{\omega+1}(x) = \underbrace{F_\omega \circ \cdots \circ F_\omega}_x(x)
\]

\[
F_{\omega \cdot 2}(x) = F_{\omega+x+1}(x)
\]

\[
F_{\omega \cdot 3}(x) = F_{\omega \cdot 2+x+1}(x)
\]

\[
F_{\omega 2}(x) = F_{\omega \cdot (x+1)}(x)
\]

\[
F_{\omega \omega}(x) = F_{\omega x+1}(x)
\]

Yields corresponding complexity classes [S.’16]

Examples Coverability is

- \(F_\omega\)-complete for lossy counter systems
- \(F_{\omega \cdot 2}\)-complete for \(\nu\)-Petri nets
- \(F_{\omega \omega}\)-complete for lossy channel systems
**Ordered Branching Systems**

\((X, \leq, \rightarrow)\) where \(\rightarrow \subseteq X \times \mathcal{P}(X)\)

(we often assume \(\rightarrow \subseteq X \times \mathcal{P}_{\text{fin}}(X)\))

Tree executions

where \(x \rightarrow \{y_1, y_2, \ldots, y_n, \ldots\}\)
**Ordered Branching Systems**

\[(X, \leq, \rightarrow)\text{ where } \rightarrow \subseteq X \times \mathcal{P}(X)\]

(we often assume \(\rightarrow \subseteq X \times \mathcal{P}_{\text{fin}}(X)\))

**Coverability Problems**

**input** an ordered branching system \((X, \leq, \rightarrow)\), a root configuration \(r \in X\) and a leaf configuration \(\ell \in X\)

**question** existence of a tree execution:

- leaf coverability

  \[r \rightsquigarrow \ell \ldots \ell\]

  (top-down coverability)

- root coverability

  \[r \\land \ell \ldots \ell\]

  (bottom-up coverability)

- leaf subcoverability

  \[r \land \ell \ldots \ell\]

  (top-down subcoverability)

- root subcoverability

  \[r \lor \ell \ldots \ell\]

  (bottom-up subcoverability)
**Ordered Branching Systems**

\((X, \leq, \rightarrow)\) where \(\rightarrow \subseteq X \times \mathcal{P}(X)\)

(we often assume \(\rightarrow \subseteq X \times \mathcal{P}_{\text{fin}}(X)\))

**Coverability Problems**

input an ordered branching system \((X, \leq, \rightarrow)\), a root configuration \(r \in X\) and a leaf configuration \(\ell \in X\)

question existence of a tree execution:

![Diagram showing different types of coverability and subcoverability](image.png)
**Ordered Branching Systems**

\[(X, \leq, \rightarrow)\ where \ \rightarrow \subseteq X \times \mathcal{P}(X)\]

(we often assume \(\rightarrow \subseteq X \times \mathcal{P}_{\text{fin}}(X)\))

**Coverability Problems**

*Input:* an ordered branching system \((X, \leq, \rightarrow)\), a root configuration \(r \in X\) and a leaf configuration \(\ell \in X\)

*Question:* existence of a tree execution:

- **Leaf Coverability:** \(r \leq_{\text{top-down}} \ell\)
- **Root Coverability:** \(r \geq_{\text{bottom-up}} \ell\)
- **Leaf Subcoverability:** \(r \geq_{\text{top-down}} \ell\)
- **Root Subcoverability:** \(r \leq_{\text{bottom-up}} \ell\)
**Ordered Branching Systems**

\[(X, \leq, \rightarrow) \text{ where } \rightarrow \subseteq X \times \mathcal{P}(X)\]

(we often assume \(\rightarrow \subseteq X \times \mathcal{P}_{\text{fin}}(X)\))

**Coverability Problems**

- **Input**: an ordered branching system \((X, \leq, \rightarrow)\), a root configuration \(r \in X\) and a leaf configuration \(\ell \in X\)
- **Question**: existence of a tree execution:

  - **Leaf Coverability**: \(r \leq \ell \ldots \leq \ell\) (top-down coverability)
  - **Root Coverability**: \(r \geq \ell \ldots \geq \ell\) (bottom-up coverability)
  - **Leaf Subcoverability**: \(r \geq \ell \ldots \geq \ell\) (top-down subcoverability)
  - **Root Subcoverability**: \(r \leq \ell \ldots \leq \ell\) (bottom-up subcoverability)
**Ordered Branching Systems**

\[(X, \leq, \rightarrow)\] where \(\rightarrow \subseteq X \times \mathbb{P}(X)\)

(we often assume \(\rightarrow \subseteq X \times \mathbb{P}_{\text{fin}}(X)\))

**Coverability Problems**

*input* an ordered branching system \((X, \leq, \rightarrow)\), a root configuration \(r \in X\) and a leaf configuration \(\ell \in X\)

*question* existence of a tree execution:

- **Leaf coverability**
- **Root coverability**
- **Leaf subcoverability**
- **Root subcoverability**

- (top-down coverability)
- (bottom-up coverability)
- (top-down subcoverability)
- (bottom-up subcoverability)
Example: Forward XPath $\varepsilon$

[Jurdziński & Lazíc ’11]
Process XML files

EXAMPLE: FORWARD XPATH

Example: Forward XPath

\[ \varepsilon \]

[Jurdziński & Lazic '11]

XSL
XQuery
Process XML files

<xml:template name="qanda:section:level">
  <xml:variable name="section:level">
    <xml:call-template name="qanda:section:level"/>
  </xml:variable>
  <xml:variable name="anc:divs">
    <xml:call-template name="ancestor::qanda:divs">
      <xml:value-of select="count($anc:divs) + number($section:level)"/>
    </xml:call-template>
  </xml:variable>
</xml:template>

<xml:template name="question:answer:label">
  <xml:variable name="deflabel">
    <xml:choose>
      <xml:when test="$deflabel = 'qanda'">
        <xml:call-template name="gentext">
          <xml:with-param name="key">
            <!-- Insert key parameters here -->
          </xml:with-param>
        </xml:call-template>
      </xml:when>
      <!-- Other cases... -->
    </xml:choose>
  </xml:variable>
</xml:template>
EXAMPLE: FORWARD XPath $\varepsilon$

[Jurdziński & Lazić ’11]

XSL

Process XML files

XQuery

Alternating Automata on Data Trees and XPath Satisfiability

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

1. INTRODUCTION

Current methods for reasoning about data words and tree data are unsatisfactory for XML and the need for formal verification and synthesis of finite-state systems, trees that have infinite structure.

Initial progress made on reasoning about data words and data trees is summarized in the survey by Jurdziński [2006]. A data word is a word over $\Sigma = \{\varepsilon\}$, where $\varepsilon$ is a finite alphabet, and $D$ is an infinite set (domain) whose elements (data) can only be compared every node is labeled by a pair in $\Sigma \times D$.

First-order logic for data words was considered by Steenikz and R. [2006], and related automata were studied further by Riek/K and Schewe [2007]. The logic for each value that range over word positions (x, y, 1, 2, N), unary predicates for every letter from the finite alphabet, and binary predicates $x = y$ that denotes equality.

This article is a revised and extended version of Jurdziński and Lazić [2007].

A full version of the article is available at [URL].
Example: Forward XPath $\varepsilon$

[Jurdziński & Lazić '11]

XML document

```xml
<e>
  <g/>
  <f a="foo">
    <g/>
    <e>
      <h/>
      <f b="foo"></f>
    </e>
  </f>
</e>
```

XPath expression

```
//[a = ./*[following-sibling::f/@b]]
```
Example: Forward XPath $\varepsilon$

[Jurdziński & Lazić ’11]

A data tree is an intended ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("data") from an infinite set, where the latter can only be compared for equality. The register for storing data. The main results are that emptiness over finite data trees is decidable, but not the presence of a witness automaton with finitely many elements. Allowing complex queries, such as shortest path queries, is obtained for two data-encoding fragments.

A data tree is an intended ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("data") from an infinite set, where the latter can only be compared for equality. The register for storing data. The main results are that emptiness over finite data trees is decidable, but not the presence of a witness automaton with finitely many elements. Allowing complex queries, such as shortest path queries, is obtained for two data-encoding fragments.

1. INTRODUCTION

Context: Logic and automata for words and trees over finite alphabets are relatively well understood. Motivated partly by the search for automated reasoning techniques for XML and the need to formalize and verify the correctness of infinite-state systems, there is an active and broad research program on the theory of automata and logic for words and trees that have richer structure.

Initial progress made on reasoning about data words and data trees is summarized in the survey by Kaplan (2006). A data word is a word over $\Sigma = \{\},$ where $\Sigma$ is a finite alphabet, and $\Delta$ is an infinite set (domain), whose elements are data; can only be compared for equality. Similarly, a data tree is a tree (countable, unranked, and ordered) whose every node is labeled by a letter from $\Sigma = \{\},$ first-order logic for data words was considered by Saks, et al. (2006), and related automata were studied further by Raskinen and Schreiber (2007). The logic has variables that range over word positions, $\{0, 1, \ldots \},$ one unary predicate for each letter from the finite alphabet, and a binary predicate $=$ that denotes equality.

XPath expression

```
//f[@a = ./*h/following-sibling::f/@b]
```
**Example: Forward XPath**

[Jurdziński & Lazic ’11]

A data tree is an intended ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("label") from an infinite set, where the latter can only be prepared for specific data. The main results are that computations over finite data trees are decidable but not elementary. The results require each class undecidability for certain, decidability is obtained for two data-accessible fragments.

---

**XPath expression**

\[//f[@a = ../h/following-sibling::f/@b]\]
Example: Forward XPath $\epsilon$

[Jurdziński & Lazić ’11]

```xml
data tree

@b

f

h

@a

foo

//f[@a = ./*/h/following-sibling::f/@b]
```
**Example: Forward XPath**

[@a foo]  
[@b foo]  
\[\text{XPath expression} \]

\[
//f[@a = ./*/h/following-sibling::f/@b]
\]

---

**Data tree**

- e
- g
- f
- h

---

**Example: Forward XPath**

[@a foo]  
[@b foo]  
\[\text{XPath expression} \]

\[
//f[@a = ./*/h/following-sibling::f/@b]
\]
Example: Forward XPath ε

[Jurdziński & Lazić ’11]

reductions

satisfiability for Forward XPath ε

emptiness of alternating 1-register tree automata

counter abstraction

subcoverability of incrementing tree counter machines
Example: Forward XPath $\varepsilon$

[Jurdziński & Lazić ’11]

reductions

satisfiability for Forward XPath $\varepsilon$

emptiness of alternating 1-register tree automata

counter abstraction

subcoverability of incrementing tree counter machines
**INCREMENTSING TREE COUNTER MACHINES**

[Jurdziński & Lazić ’11]

**Example**

![Graph of incrementing tree counter machines](image)

**Incrementing Semantics**

\[
\begin{align*}
q_1(1,2) & \leq q_1(2,4) & c_1-- & q_3(1,4) & \leq q_3(3,4) \\
q_1(0,2) & \leq q_1(0,2) & c_1++ \\
q_2(1,2) & \leq q_2(5,2) & c_2++ & q_2 ? = 0 & c_2++
\end{align*}
\]

**Theorem** (Jurdziński & Lazić ’11)

Subcoverability in incrementing tree counter machines is decidable. Thus satisfiability of Forward XPath^ε is also decidable.
INCREMENTING TREE COUNTER MACHINES

[Jurdziński & Lazić ’11]

Example

Incrementing Semantics

\[
q_1(0,2) \leq q_1(0,2) \\
q_1(1,2) \leq q_1(2,4) \xrightarrow{c_1\rightarrow} q_3(1,4) \leq q_3(3,4) \\
q_2(1,2) \leq q_2(5,2)
\]

Theorem (Jurdziński & Lazić ’11)

Subcoverability in incrementing tree counter machines is decidable. Thus satisfiability of Forward XPath\(\varepsilon\) is also decidable.
INCREMENTING TREE COUNTER MACHINES

[Jurdziński & Lazić ’11]

Example

Incrementing Semantics

\[ q_1(0,2) \leq q_1(0,2) \]
\[ q_1(1,2) \leq q_1(2,4) \xrightarrow{c_1-} q_3(1,4) \leq q_3(3,4) \]
\[ q_2(1,2) \leq q_2(5,2) \]

**Theorem** (Jurdziński & Lazić ’11)

Subcoverability in incrementing tree counter machines is decidable. Thus satisfiability of Forward XPath$^\varepsilon$ is also decidable.
Example: Relevance Logic \( R \rightarrow \)

[Kripke ’59; Riche & Meyer ’98; Urquhart ’99]

Example: \( A \rightarrow (B \rightarrow A) \)

“if it’s raining (A), then if your favorite colour is green (B) then it’s raining (A)”

A theorem in classical logic, not in relevance logic.

Gentzen-style sequent calculus

A, B, C formulæ; \( \Gamma, \Delta \) finite multisets of formulæ; no weakening

\[
\frac{A \vdash A}{\Gamma, A \vdash A} \quad \text{(Id)}
\]

\[
\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B} \quad \text{(C)}
\]

\[
\frac{\Gamma \vdash A}{\Gamma, A \vdash B} \quad \text{(→L)}
\]

\[
\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \quad \text{(→R)}
\]

Theorem (Kripke ’59)

Provability in \( R \rightarrow \) is decidable.
**Example: Relevance Logic**

\[ \rightarrow \]

【Kripke ’59; Riche & Meyer ’98; Urquhart ’99】

**Example:** \( A \rightarrow (B \rightarrow A) \)

“if it’s raining (A), then if your favorite colour is green (B) then it’s raining (A)”

A theorem in classical logic, not in relevance logic.

**Gentzen-style sequent calculus**

- \( A, B, C \) formulae; \( \Gamma, \Delta \) finite multisets of formulæ;
- no weakening

\[
\Gamma \vdash A \quad \text{(Id)}
\]

\[
\Gamma, A, A \vdash B \quad \text{(C)}
\]

\[
\Gamma \vdash A \quad \Delta, B \vdash C \quad \text{\( \rightarrow \)L}
\]

\[
\Gamma, \Delta, A \rightarrow B \vdash C \quad \text{\( \rightarrow \)R}
\]

Theorem (Kripke ’59)

Provability in \( R \rightarrow \) is decidable.
Example: Relevance Logic $\textbf{R}\rightarrow$

Example: $A \rightarrow (B \rightarrow A)$

"if it’s raining (A), then if your favorite colour is green (B) then it’s raining (A)"

A theorem in classical logic, not in relevance logic.

Gentzen-style sequent calculus

A, B, C formulae; $\Gamma, \Delta$ finite multisets of formulæ; no weakening

- $A \vdash A$ (Id)
- $\Gamma, A, A \vdash B$ (C)
- $\Gamma \vdash A, \Delta, B \vdash C$ ($\rightarrow L$)
- $\Gamma, \Delta, A \rightarrow B \vdash C$
- $\Gamma \vdash A \rightarrow B$ ($\rightarrow R$)

Theorem (Kripke ’59)

Provability in $\textbf{R}\rightarrow$ is decidable.
Example: **Relevance Logic LR⁺**

[Kripke ‘59; Riche & Meyer ’98; Urquhart ’99]

**Gentzen-style sequent calculus**

A, B, C formulæ; Γ, Δ finite multisets of formulæ; no weakening

\[
\frac{A \vdash A}{\Gamma \vdash A} \quad (\text{Id})
\]

\[
\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \rightarrow B \vdash C} \quad (\rightarrow_L)
\]

\[
\frac{\Gamma, A \vdash B}{\Gamma, A \vdash A \rightarrow B} \quad (\rightarrow_R)
\]

\[
\frac{\Gamma, A \vdash A, B}{\Gamma, A \vdash B} \quad (C)
\]

Thus \(\rightarrow\) behaves like intuitionistic linear implication \(\rightarrow\) with added contraction. LR⁺ adds the symbols and rules for \(\otimes\), \(\oplus\), &, and \(1\).

**Theorem** (Kripke ‘59; Anderson & Belnap ’75; Meyer ’66)

*Provability in LR⁺ is decidable.*
Example: **Relevance Logic LR+**

[Kripke ’59; Riche & Meyer ’98; Urquhart ’99]

**Gentzen-style sequent calculus**

$A, B, C$ formulae; $\Gamma, \Delta$ finite multisets of formulae; no weakening

\[
\begin{align*}
A \vdash A & \quad (\text{Id}) \\
\Gamma, A, A \vdash B & \quad (C)
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash A & \quad (\rightarrow_L) \\
\Gamma, \Delta, A \rightarrow B \vdash C & \quad (\rightarrow_L) \\
\Gamma, A, A \vdash B & \quad (C)
\end{align*}
\]

\[
\begin{align*}
\Gamma, A \rightarrow B & \quad (\rightarrow_R) \\
\Gamma \vdash A \rightarrow B & \quad (\rightarrow_R)
\end{align*}
\]

Thus $\rightarrow$ behaves like intuitionistic linear implication $\rightarrow$ with added contraction. $\text{LR}+$ adds the symbols and rules for $\otimes$, $\oplus$, $\&$, and $\mathbf{1}$.

**Theorem** (Kripke ’59; Anderson & Belnap ’75; Meyer ’66)

*Provability in LR+ is decidable.*
**Counter Abstraction**

[Urquhart ‘90]

- **Subformula property**: in every rule, the formulæ in the premisses are subformulæ of those in the conclusion.

- In a proof search for a sequent \( \Gamma \vdash A \)

\[
\begin{array}{ccc}
B \vdash B & \cdots & C \vdash C \\
\end{array}
\]

we only need to work with formulæ from the finite set

\[ \Sigma \overset{\text{def}}{=} \text{Sub}(\Gamma, A) \]

- For a sequent \( \Delta \vdash B \) built over \( \Sigma \), associate an element 

\[ \gamma \Delta \vdash B \overset{\text{def}}{=} (\text{dom}(\Delta), v_\Delta, B) \]

of

\[ X \overset{\text{def}}{=} \mathbb{P}(\Sigma) \times \mathbb{N}^{\left|\Sigma\right|} \times \Sigma \]

where \( v_\Delta(C) \) is the number of occurrences of \( C \) in the multiset \( \Delta \)

- \( (X, \leq) \) as the product of \( (\mathbb{P}(\Sigma), =), \)

\( (\mathbb{N}^{\left|\Sigma\right|}, \leq) \), and \( (\Sigma, =) \) is a wqo by Dickson’s Lemma

- \( \gamma \Delta \vdash B \leq \gamma \Delta' \vdash B' \) iff \( \Delta \vdash B \) can be obtained by contracting \( \Delta' \vdash B' \) a number of times

- Proof search for a sequent \( \Gamma \vdash A \) is an instance of leaf subcoverability with root \( \gamma \Gamma \vdash A \) and leaves in

\[ \{ \gamma B \vdash B \mid B \in \Sigma \} \]
**Counter Abstraction**

[Urquhart '90]

- Subformula property: in every rule, the formulæ in the premisses are subformulæ of those in the conclusion.

- In a proof search for a sequent $\Gamma \vdash A$

  \[
  \begin{array}{c}
  B \vdash B \\
  \vdash \\
  C \vdash C \\
  \vdash \\
  \end{array}
  \]

  we only need to work with formulæ from the finite set

  \[
  \Sigma \overset{\text{def}}{=} \text{Sub}(\Gamma, A)
  \]

  For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\Gamma \Delta \vdash B \overset{\text{def}}{=} (\text{dom}(\Delta), v_\Delta, B)$ of

  \[
  \chi \overset{\text{def}}{=} \mathcal{P}(\Sigma) \times \mathbb{N}^{\left|\Sigma\right|} \times \Sigma
  \]

  where $v_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$.

  $(\chi, \leq)$ as the product of $(\mathcal{P}(\Sigma), =)$, $(\mathbb{N}^{\left|\Sigma\right|}, \leq)$, and $(\Sigma, =)$ is a wqo by Dickson’s Lemma.

  $\Gamma \Delta \vdash B \leq \Gamma \Delta' \vdash B'$ iff $\Delta \vdash B$ can be obtained by contracting $\Delta' \vdash B'$ a number of times.

  Proof search for a sequent $\Gamma \vdash A$ is an instance of leaf subcoverability with root $\Gamma \vdash A \vdash$ and leaves in $\{\Gamma B \vdash B \vdash | B \in \Sigma\}$.
COUNTER ABSTRACTION

[Urquhart ’90]

▶ Subformula property: in every rule, the formulæ in the premisses are subformulæ of those in the conclusion.

▶ In a proof search for a sequent $\Gamma \vdash A$

$B \vdash B \quad \cdots \quad C \vdash C$

$\Gamma \vdash A$

we only need to work with formulæ from the finite set

$\Sigma \overset{\text{def}}{=} \text{Sub}(\Gamma, A)$

▶ For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\Gamma \Delta \vdash B \overset{\text{def}}{=} (\text{dom}(\Delta), v_\Delta, B)$ of

$X \overset{\text{def}}{=} \mathcal{P}(\Sigma) \times \mathbb{N}^{|\Sigma|} \times \Sigma$

where $v_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

▶ $(X, \preceq)$ as the product of $(\mathcal{P}(\Sigma), =)$, $(\mathbb{N}^{|\Sigma|}, \preceq)$, and $(\Sigma, =)$ is a wqo by Dickson’s Lemma

▶ $\Gamma \Delta \vdash B \preceq \Gamma \Delta' \vdash B'$ iff $\Delta \vdash B$ can be obtained by contracting $\Delta' \vdash B'$ a number of times

▶ Proof search for a sequent $\Gamma \vdash A$ is an instance of leaf subcoverability with root $\Gamma \vdash A \downarrow$ and leaves in $
\{\Gamma B \vdash B \downarrow \mid B \in \Sigma\}$
Counter Abstraction

[Urquhart ’90]

- **Subformula property:** in every rule, the formulæ in the premisses are subformulæ of those in the conclusion.

- In a proof search for a sequent $\Gamma \vdash A$

  \[
  \begin{array}{c}
  B \vdash B \\
  \ldots \\
  C \vdash C \\
  \end{array}
  \]

  we only need to work with formulæ from the finite set

  \[
  \Sigma \overset{\text{def}}{=} \text{Sub}(\Gamma, A)
  \]

- For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\Gamma \Delta \vdash B \overset{\text{def}}{=} (\text{dom}(\Delta), v_\Delta, B)$ of

  \[
  X \overset{\text{def}}{=} P(\Sigma) \times \mathbb{N}^{\mid \Sigma \mid} \times \Sigma
  \]

  where $v_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

- $(X, \leq)$ as the product of $(P(\Sigma), =)$, $(\mathbb{N}^{\mid \Sigma \mid}, \leq)$, and $(\Sigma, =)$ is a wqo by Dickson’s Lemma

- $\Gamma \Delta \vdash B \leq \Gamma \Delta' \vdash B'$ iff $\Delta \vdash B$ can be obtained by contracting $\Delta' \vdash B'$ a number of times

- Proof search for a sequent $\Gamma \vdash A$ is an instance of leaf subcoverability with root $\Gamma \vdash A$ and leaves in

  \[
  \{ \Gamma B \vdash B \mid B \in \Sigma \} \]
Counter Abstraction

[Urquhart ’90]

- Subformula property: in every rule, the formulæ in the premisses are subformulæ of those in the conclusion.

- In a proof search for a sequent $\Gamma \vdash A$

\[
\begin{array}{c}
B \vdash B \quad \cdots \quad C \vdash C
\end{array}
\]

we only need to work with formulæ from the finite set

\[
\Sigma \overset{\text{def}}{=} \text{Sub}(\Gamma, A)
\]

- For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\Gamma \Delta \vdash B \overset{\text{def}}{=} (\text{dom}(\Delta), v_\Delta, B)$ of

\[
X \overset{\text{def}}{=} \mathbb{P}(\Sigma) \times \mathbb{N}^{\mid \Sigma \mid} \times \Sigma
\]

where $v_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

- $(X, \leq)$ as the product of $(\mathbb{P}(\Sigma), =)$, $(\mathbb{N}^{\mid \Sigma \mid}, \leq)$, and $(\Sigma, =)$ is a wqo by Dickson’s Lemma

- $\Gamma \Delta \vdash B \overset{\leq}{\leq} \Gamma \Delta' \vdash B'$ iff $\Delta \vdash B$ can be obtained by contracting $\Delta' \vdash B'$ a number of times

- Proof search for a sequent $\Gamma \vdash A$ is an instance of leaf subcoverability with root $\Gamma \vdash A \overset{\leq}{\leq}$ and leaves in

\[
\{\Gamma B \vdash B \mid B \in \Sigma\}\]
### COUNTER ABSTRACTION

[Urquhart ‘90]

- **Subformula property:** in every rule, the formulæ in the premisses are subformulæ of those in the conclusion.

- In a proof search for a sequent $\Gamma \vdash A$

\[
\begin{array}{c}
B \vdash B \\
\vdots \\
C \vdash C
\end{array}
\]

$\Gamma \vdash A$

we only need to work with formulæ from the finite set

\[
\Sigma \defeq \text{Sub}(\Gamma, A)
\]

- For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\Gamma \vdash B \vdash (\text{dom}(\Delta), v_\Delta, B)$ of

\[
X \defeq \mathcal{P}(\Sigma) \times \mathbb{N}^{||\Sigma||} \times \Sigma
\]

where $v_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

- $(X, \leqslant)$ as the product of $(\mathcal{P}(\Sigma), =)$, $(\mathbb{N}^{||\Sigma||}, \leqslant)$, and $(\Sigma, =)$ is a wqo by Dickson’s Lemma

- $\Gamma \vdash B \vdash \Gamma \vdash B'$ iff $\Delta \vdash B$ can be obtained by contracting $\Delta' \vdash B'$ a number of times

- Proof search for a sequent $\Gamma \vdash A$ is an instance of leaf subcoverability with root $\Gamma \vdash A$ and leaves in $\{\Gamma B \vdash B| B \in \Sigma\}$
**Complexity Lower Bounds**

**Theorem** (Demri & Lazic ‘09, Figueira & Segoufin, ’09)

Satisfiability of Forward XPath$^\varepsilon$ expressions is $F_\omega$-hard.

**Theorem** (Urquhart ’99)

Provability in $LR^+$ is $F_\omega$-hard.
LIFTING WQOS

► If \((X, \leq)\) is a wqo, then
\((\mathcal{P}_{\text{fin}}(X), \subseteq_H)\) is a wqo
where \(\subseteq_H\) is the **Hoare quasi-ordering** (or majoring quasi-ordering)

\[
S \subseteq_H S' \iff \forall x \in S. \exists x' \in S'. x \leq x'
\]

► If \((X, \leq)\) is an \(\omega^2\)-wqo, then
\((\mathcal{P}_{\text{fin}}(X), \subseteq_S)\) is a wqo
where \(\subseteq_S\) is the **Smyth quasi-ordering** (or minoring quasi-ordering)

\[
S \subseteq_S S' \iff \forall x' \in S'. \exists x \in S. x \leq x'
\]
LIFTING WQOS

▶ If $(X, \leq)$ is a wqo, then $(\mathbb{P}_{\text{fin}}(X), \subseteq_H)$ is a wqo where $\subseteq_H$ is the Hoare quasi-ordering (or majoring quasi-ordering)

\[ S \subseteq_H S' \iff \forall x \in S. \exists x' \in S'. x \leq x' \]

▶ If $(X, \leq)$ is an $\omega^2$-wqo, then $(\mathbb{P}_{\text{fin}}(X), \subseteq_S)$ is a wqo where $\subseteq_S$ is the Smyth quasi-ordering (or minoring quasi-ordering)

\[ S \subseteq_S S' \iff \forall x' \in S'. \exists x \in S. x \leq x' \]
LIFTING WQOS

If \((X, \leq)\) is a wqo, then 
\((\mathcal{P}_{\text{fin}}(X), \subseteq_{\mathcal{H}})\) is a wqo
where \(\subseteq_{\mathcal{H}}\) is the Hoare quasi-ordering (or majoring quasi-ordering)

\[ S \subseteq_{\mathcal{H}} S' \iff \forall x \in S. \exists x' \in S'. x \leq x' \]

If \((X, \leq)\) is an \(\omega^2\)-wqo, then 
\((\mathcal{P}_{\text{fin}}(X), \subseteq_{\mathcal{S}})\) is a wqo
where \(\subseteq_{\mathcal{S}}\) is the Smyth quasi-ordering (or minorizing quasi-ordering)

\[ S \subseteq_{\mathcal{S}} S' \iff \forall x' \in S'. \exists x \in S. x \leq x' \]
Lifting wqos

- If $(X, \leq)$ is a wqo, then $(P_{\text{fin}}(X), \subseteq_{H})$ is a wqo where $\subseteq_{H}$ is the Hoare quasi-ordering (or majoring quasi-ordering)

$$S \subseteq_{H} S' \iff \forall x \in S . \exists x' \in S'. x \leq x'$$

- If $(X, \leq)$ is an $\omega^2$-wqo, then $(P_{\text{fin}}(X), \subseteq_{S})$ is a wqo where $\subseteq_{S}$ is the Smyth quasi-ordering (or minoring quasi-ordering)

$$S \subseteq_{S} S' \iff \forall x' \in S'. \exists x \in S. x \leq x'$$
Lifting wqos

- If \((X, \leq)\) is a wqo, then 
  \((P_{\text{fin}}(X), \subseteq_{\text{H}})\) is a wqo
  where \(\subseteq_{\text{H}}\) is the Hoare quasi-ordering (or majoring quasi-ordering)

  \[ S \subseteq_{\text{H}} S' \iff \forall x \in S. \exists x' \in S'. x \leq x' \]

- If \((X, \leq)\) is an \(\omega^2\)-wqo, then 
  \((P_{\text{fin}}(X), \subseteq_{\text{S}})\) is a wqo
  where \(\subseteq_{\text{S}}\) is the Smyth quasi-ordering (or minoring quasi-ordering)

  \[ S \subseteq_{\text{S}} S' \iff \forall x' \in S'. \exists x \in S. x \leq x' \]
Compatibility for Branching Systems

Leaf Coverability

Root Coverability

Leaf Subcoverability

Root Subcoverability

∀x, x′, S

∀x, x′, S

∀x, x′, S

∀x, x′, S

x′

x′

x′

x

x

x

x

S

S

S

S

∀x, x′, S

∀x, x′, S

∀x, x′, S

∀x, x′, S

\lor

\lor

\lor

\lor

\lor

\lor

\lor
Compatibility for Branching Systems

Leaf coverability

Root coverability

Leaf subcoverability

Root subcoverability

t-d. compatibility

b-up compatibility

t-d. downward compatibility

b-up downward compatibility
Compatibility for Branching Systems

- Leaf coverability
  \[ r \leq \ell \]
- Root coverability
  \[ r \geq \ell \]
- Leaf subcoverability
  \[ r \geq \ell \]
- Root subcoverability
  \[ r \leq \ell \]

- t.-d. compatibility
  \[ \forall x, x', S \quad x' \quad \Rightarrow \quad S' \]
  \[ \forall x, x', S \quad x' \quad \iff \quad S' \]
  \[ \forall x, x', S \quad x' \quad \Rightarrow \quad S \]
- b.-up compatibility
  \[ \forall x, x', S \quad x' \quad \Rightarrow \quad S \]
  \[ \forall x, x', S \quad x' \quad \iff \quad S \]
  \[ \forall x, x', S \quad x' \quad \Rightarrow \quad S \]

- t.-d. downward compatibility
  \[ \forall x, x', S \quad x' \quad \Rightarrow \quad S \]
  \[ \forall x, x', S \quad x' \quad \iff \quad S \]
  \[ \forall x, x', S \quad x' \quad \Rightarrow \quad S \]
- b.-up downward compatibility
  \[ \forall x, x', S \quad x' \quad \Rightarrow \quad S \]
  \[ \forall x, x', S \quad x' \quad \iff \quad S \]
  \[ \forall x, x', S \quad x' \quad \Rightarrow \quad S \]
Compatibility for Branching Systems

- **Leaf coverability**: \( r \geq \ell \)
- **Root coverability**: \( r \leq \ell \)
- **Leaf subcoverability**: \( r \geq \ell \)
- **Root subcoverability**: \( r \leq \ell \)

- **t.-d. compatibility**: \( \forall x, x', S, x' \rightarrow S' \)
- **b.-up compatibility**: \( \forall x, x', S, x' \rightarrow S' \)
- **t.-d. downward compatibility**: \( \forall x, x', S, x_S \rightarrow S' \)
- **b.-up downward compatibility**: \( \forall x, x', S, x_S \rightarrow S' \)
Compatibility for Branching Systems

- Leaf coverability
- Root coverability
- Leaf subcoverability
- Root subcoverability

T.-d. compatibility

∀x, x', S  =  ∃S'  
\[ x' \rightarrow S' \]
\[ \lor \rightarrow S \]

B.-up compatibility

∀x, x', S  =  ∃S'  
\[ x' \rightarrow S' \]
\[ \lor \rightarrow S \]

T.-d. downward compatibility

∀x, x', S  =  ∃S'  
\[ x \rightarrow S \]
\[ \lor \rightarrow S \]

B.-up downward compatibility

∀x, x', S  =  ∃S'  
\[ x \rightarrow S \]
\[ \lor \rightarrow S \]
\[ x' \rightarrow S \]
Compatibility for Branching Systems

- **Leaf Coverability**
  - \( \forall x, x', S \quad x' \rightarrow S' \)
  -\( x \rightarrow S \)

- **Root Coverability**
  - \( \forall x, x', S \quad x' \rightarrow S' \)
  -\( x \rightarrow S \)

- **Leaf Subcoverability**
  - \( \forall x, x', S \quad x' \rightarrow S' \)
  -\( x \rightarrow S \)

- **Root Subcoverability**
  - \( \forall x, x', S \quad x' \rightarrow S' \)
  -\( x \rightarrow S \)

- **t.-d. Compatibility**
  - \( \forall x, x', S \quad x' \rightarrow S' \)
  -\( x \rightarrow S \)

- **b.-up Compatibility**
  - \( \forall x, x', S \quad x' \rightarrow S' \)
  -\( x \rightarrow S \)

- **t.-d. Downward Compatibility**
  - \( \forall x, x', S \quad x' \rightarrow S' \)
  -\( x \rightarrow S \)

- **b.-up Downward Compatibility**
  - \( \forall x, x', S \quad x' \rightarrow S' \)
  -\( x \rightarrow S \)
Compatibility for Branching Systems

Leaf coverability

Root coverability

Leaf subcoverability

Root subcoverability

T.-d. compatibility

B.-up compatibility

T.-d. downward compatibility

B.-up downward compatibility
Compatibility for Branching Systems

Leaf coverability

\[
\begin{align*}
\forall x, x', S : x' \quad \Rightarrow \quad S' \\
\forall x, x', S : x \quad \Rightarrow \quad S
\end{align*}
\]

Root coverability

\[
\begin{align*}
\forall x, x', S : x' \quad \Rightarrow \quad S' \\
\forall x, x', S : x \quad \Rightarrow \quad S
\end{align*}
\]

Leaf subcoverability

\[
\begin{align*}
\forall x, x', S : x' \quad \Rightarrow \quad S' \\
\forall x, x', S : x \quad \Rightarrow \quad S
\end{align*}
\]

Root subcoverability

\[
\begin{align*}
\forall x, x', S : x' \quad \Rightarrow \quad S' \\
\forall x, x', S : x \quad \Rightarrow \quad S
\end{align*}
\]

t.-d. compatibility

\[
\begin{align*}
\forall x, x', S : x' \quad \Rightarrow \quad S' \\
\forall x, x', S : x \quad \Rightarrow \quad S
\end{align*}
\]

b.-up compatibility

\[
\begin{align*}
\forall x, x', S : x' \quad \Rightarrow \quad S' \\
\forall x, x', S : x \quad \Rightarrow \quad S
\end{align*}
\]

t.-d. downward compatibility

\[
\begin{align*}
\exists S' : \quad \exists S' \\
\forall x, x', S : x' \quad \Rightarrow \quad S' \\
\forall x, x', S : x \quad \Rightarrow \quad S
\end{align*}
\]

b.-up downward compatibility

\[
\begin{align*}
\exists S' : \quad \exists S' \\
\forall x, x', S : x' \quad \Rightarrow \quad S' \\
\forall x, x', S : x \quad \Rightarrow \quad S
\end{align*}
\]
Compatibility for Branching Systems

- **Leaf Coverability**
  - t.-d. compatibility:
    \[ \forall x, x', S \implies x' \rightarrow S' \]
  - B.-up compatibility:
    \[ \forall x, x', S \implies x' \rightarrow S' \]

- **Root Coverability**
  - t.-d. compatibility:
    \[ \forall x, x', S \implies x \rightarrow S \]
  - B.-up compatibility:
    \[ \forall x, x', S \implies x \rightarrow S \]

- **Leaf Subcoverability**
  - t.-d. downward compatibility:
    \[ \forall x, x', S \implies x' \rightarrow S' \]
  - B.-up downward compatibility:
    \[ \forall x, x', S \implies x' \rightarrow S' \]

- **Root Subcoverability**
  - t.-d. compatibility:
    \[ \forall x, x', S \implies x \rightarrow S \]
  - B.-up compatibility:
    \[ \forall x, x', S \implies x \rightarrow S \]
LIFTING WSTS

[Lazić & S. ’21]

**FOR LEAF (SUB)COVERABILITY**

\[ S \Rightarrow T \overset{\text{def}}{\iff} T = \bigcup_{x \in S} T_x \text{ where } \forall x \in S . x \rightarrow T_x \]

- \( \{r\} \Rightarrow^* S \) whenever \( S \) is a cut of an execution tree
- Leaf coverability of \( \ell \) from \( r \) \( \iff \) \( \{r\} \Rightarrow^* S \supseteq_S \{\ell\} \)
- Leaf subcoverability of \( \ell \) from \( r \) \( \iff \) \( \{r\} \Rightarrow^* S \subseteq_{IH} \{\ell\} \)
Lifting WSTS

[Lazić & S. ’21]

For leaf (sub)coverability

\[ S \models T \overset{\text{def}}{\iff} T = \bigcup_{x \in S} T_x \text{ where } \forall x \in S. x \to T_x \]

- \{r\} \models^* S \text{ whenever } S \text{ is a cut of an execution tree}

- leaf coverability of \( \ell \) from \( r \) \( \iff \) \{r\} \models^* S \supseteq \{\ell\}

- leaf subcoverability of \( \ell \) from \( r \) \( \iff \) \{r\} \models^* S \subseteq \text{IH} \{\ell\}
Lifting WSTS

[Lazić & S. ’21]

For leaf (sub)coverability

\[ S \to T \iff T = \bigcup_{x \in S} T_x \text{ where } \forall x \in S. x \xrightarrow{\sigma} T_x \]

- \{r\} \Rightarrow^* S \text{ whenever } S \text{ is a cut of an execution tree}

- leaf coverability of \( \ell \) from \( r \) \( \iff \) \{r\} \Rightarrow^* S \supseteq S \{\ell\}

- leaf subcoverability of \( \ell \) from \( r \) \( \iff \) \{r\} \Rightarrow^* S \subseteq H \{\ell\}
**Lifting WSTS**

[Lazić & S. ’21]

**For leaf (sub)coverability**

\[
S \Rightarrow T \overset{\text{def}}{\iff} T = \bigcup_{x \in S} T_x \text{ where } \forall x \in S. x \rightarrow T_x
\]

- \( \{r\} \Rightarrow^* S \) whenever \( S \) is a cut of an execution tree

- Leaf coverability of \( \ell \) from \( r \) \( \iff \) \( \{r\} \Rightarrow^* S \supseteq S \{\ell\} \)

- Leaf subcoverability of \( \ell \) from \( r \) \( \iff \) \( \{r\} \Rightarrow^* S \subseteq_{IH} \{\ell\} \)
**Top-Down Downward Compatibility**

**Lemma**

An ordered branching system \((X, \leq, \to)\) has top-down downward compatibility if and only if \((\mathcal{P}(X), \subseteq_H, \supseteq)\) has downward compatibility.

\[
\forall x, x', S \\
x \quad \quad \quad S \\
\forall \quad \quad \quad \subseteq_H \\
x' \quad \quad \quad S' \\
\exists S' \\
\]

\[
\forall S, S', T \\
S \quad \quad \quad T \\
\subseteq_H \\
S' \quad \quad \quad T' \\
\exists T' \\
\]

\[
\text{In } (X, \leq, \to): \\
\text{In } (\mathcal{P}_{\text{fin}}(X), \subseteq_H, \supseteq): \\
\]
**Top-Down Downward Compatibility**

**Lemma**

An ordered branching system \((X, \leq, \rightarrow)\) has top-down downward compatibility if and only if \((\mathcal{P}(X), \subseteq_H, \supseteq)\) has downward compatibility.

\[
\begin{align*}
\ln (X, \leq, \rightarrow): & \quad \forall x, x', S \quad \exists S' \\
& \quad x \rightarrow S \\
& \quad \forall \exists S' \\
& \quad x' \quad = \quad S' \\
\ln (\mathcal{P}_{\text{fin}}(X), \subseteq_H, \supseteq): & \quad \forall S, S', T \\
& \quad S \quad \rightarrow T \\
& \quad \exists T' \\
& \quad \bigcup S' \quad = \quad \bigcup S' \\
& \quad \bigcup S' \quad \rightarrow \quad T'
\end{align*}
\]
Top-Down Downward Compatibility

The sequent calculus for $R \to$ does not satisfy t.-d. downward compatibility

\[
\frac{D \vdash A \quad D, B \vdash C}{D, D, A \to B \vdash C} \quad (\to_L)
\]

Incrementing tree counter machines satisfy t.-d. downward compatibility

$\Gamma, D, A \to B \vdash C \preceq \Gamma, D, D, A \to B \vdash C$

Solution: replace $(\to_L)$ with a rule with built-in contraction

\[
\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{[\Gamma, \Delta, A \to B] \vdash C} \quad ([\to_L])
\]

where $[\Gamma, \Delta, A \to B]$ is any multiset of formulae embeddable into $\Gamma, \Delta, A \to B$ and with the same domain
**Top-Down Downward Compatibility**

The sequent calculus for $\mathbf{R} \rightarrow$ does not satisfy t.-d. downward compatibility

$$
\frac{D \vdash A \quad D, B \vdash C}{D, D, A \rightarrow B \vdash C} \quad (\rightarrow_L)
$$

$\Gamma, D, A \rightarrow B \vdash C^\triangleright \leq \Gamma, D, D, A \rightarrow B \vdash C^\triangleright$

Solution: replace $(\rightarrow_L)$ with a rule with built-in contraction

$$
\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{[\Gamma, \Delta, A \rightarrow B] \vdash C} \quad ([\rightarrow_L])
$$

where $[\Gamma, \Delta, A \rightarrow B]$ is any multiset of formulae embeddable into $\Gamma, \Delta, A \rightarrow B$ and with the same domain.
**TOP-DOWN DOWNWARD COMPATIBILITY**

The sequent calculus for $\mathcal{R} \rightarrow$ does not satisfy t.-d. downward compatibility

\[
\frac{D \vdash A \quad D, B \vdash C}{D, D, A \rightarrow B \vdash C} \quad (\rightarrow_L)
\]

Incrementing tree counter machines satisfy t.-d. downward compatibility

\[\Gamma, A \rightarrow B \vdash C \preceq \Gamma, D, A \rightarrow B \vdash C\]

Solution: replace $(\rightarrow_L)$ with a rule with built-in contraction

\[
\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{[\Gamma, \Delta, A \rightarrow B] \vdash C} \quad ([\rightarrow_L])
\]

where $[\Gamma, \Delta, A \rightarrow B]$ is any multiset of formulae embeddable into $\Gamma, \Delta, A \rightarrow B$ and with the same domain
Top-Down Downward Compatibility

The sequent calculus for $R \rightarrow$ does not satisfy t.-d. downward compatibility

$$
\frac{D \vdash A \quad D, B \vdash C}{D, D, A \rightarrow B \vdash C} \quad (\rightarrow_L)
$$

Incrementing tree counter machines satisfy t.-d. downward compatibility

$$
D, A \rightarrow B \vdash C \subseteq D, D, A \rightarrow B \vdash C
$$

Solution: replace $(\rightarrow_L)$ with a rule with built-in contraction

$$
\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{[\Gamma, \Delta, A \rightarrow B] \vdash C} \quad ([\rightarrow_L])
$$

where $[\Gamma, \Delta, A \rightarrow B]$ is any multiset of formulae embeddable into $\Gamma, \Delta, A \rightarrow B$ and with the same domain
Theorem (Balasubramanian ’20)
Let $g$ be primitive recursive and $X = \mathbb{P}_{\text{fin}}(\mathbb{N}^d)$. For sufficiently large $n$,

\[
\begin{align*}
L_g, X(n) &\leq F_{\omega_1} (n) \quad \text{for the Hoare quasi-ordering } \subseteq_H, \\
L_g, X(n) &\leq F_{\omega_{d+1}} (n) \quad \text{for the Smyth quasi-ordering } \subseteq_S.
\end{align*}
\]

Yields $F_{\omega} \omega$ upper bounds for

\[\begin{align*}
\text{satisfiability of Forward XPath}^\epsilon \quad \text{and} \\
\text{provability in } R_{\to, \wedge}.
\end{align*}\]
**Theorem** (Balasubramanian ’20)

Let $g$ be primitive recursive and $X = \mathbb{P}_{\text{fin}}(\mathbb{N}^d)$. For sufficiently large $n$,

- $L_{g,X}(n) \leq F_{\omega \cdot d - 1 + 1}(n)$ for the Hoare quasi-ordering $\subseteq_H$,

- $L_{g,X}(n) \leq F_{\omega \cdot d - 1.2d + 1}(n)$ for the Smyth quasi-ordering $\subseteq_S$.

Yields $F_{\omega \omega}$ upper bounds for

- satisfiability of Forward XPath $\varepsilon$ and

- provability in $\mathcal{R}_{\to, \wedge}$.
Theorem (Balasubramanian ’20)

Let $g$ be primitive recursive and $X = \mathbb{P}_{\text{fin}}(\mathbb{N}^d)$. For sufficiently large $n$,

\[ L_g, X(n) \leq F_{\omega}^{d-1,1}(n) \text{ for the Hoare quasi-ordering } \subseteq_H, \]
\[ L_g, X(n) \leq F_{\omega}^{d-1,2d+1}(n) \text{ for the Smyth quasi-ordering } \subseteq_S. \]

Yields $F_{\omega}^\omega$ upper bounds for

- satisfiability of Forward XPath $\epsilon$ and
- provability in $\mathbb{R}^\rightarrow, \wedge$. 

**Abstract**

We prove upper and lower bounds for the length of controlled bed sequences over the Hoare and the Smyth quasi-orderings on finite sets of $\mathbb{N}^d$. These results are obtained by bounding the length of such sequence by functions from the complexity of branching systems. This allows us to translate these results to bounds over the fast-growing hierarchy classes. The obtained bounds are proven to be tight for the major ordering, which solves a problem left open by Abriola, Figueira & Senno (Theor. Comp. Sci., Vol. 435). Finally, we see the results on controlled bed sequences in prove upper bounds for the emptiness problem of some classes of automata.

**CoC Concepts:** Theory of computation $\rightarrow$ Complexity classes, Program verification

Keywords: well-quasi orders, controlled bed sequences, reasoning and inductive inference

**ACM Reference Format:**


**1 Introduction**

A well-quasi-order (wqo) over a set $A$ is a reflexive and transitive relation $\preceq$ such that every infinite sequence $a_0, a_1, a_2, \ldots$ in $A$ has a subsequence $a_{i_0}, a_{i_1}, a_{i_2}, \ldots$ that is $\preceq$-finite for every $n$. A sequence over a wqo is called a bed sequence.

A bed sequence over a wqo is called a bed sequence over a wqo. We define a bed sequence over a wqo as a sequence that contains infinitely many $\preceq$-finite subsequences.

Yield $F_{\omega}^\omega$ upper bounds for

- satisfiability of Forward XPath $\epsilon$ and
- provability in $\mathbb{R}^\rightarrow, \wedge$. 

**Complexity of controlled bed sequences over finite sets of $\mathbb{N}^d$**

A. R. Balasubramanian
Technical University of Munich, Germany
bala@in.tum.de

**Abstract**

We prove upper and lower bounds for the length of controlled bed sequences over the Hoare and the Smyth quasi-orderings on finite sets of $\mathbb{N}^d$. These results are obtained by bounding the length of such sequence by functions from the complexity of branching systems. This allows us to translate these results to bounds over the fast-growing hierarchy classes. The obtained bounds are proven to be tight for the major ordering, which solves a problem left open by Abriola, Figueira & Senno (Theor. Comp. Sci., Vol. 435). Finally, we see the results on controlled bed sequences in prove upper bounds for the emptiness problem of some classes of automata.

**CoC Concepts:** Theory of computation $\rightarrow$ Complexity classes, Program verification

Keywords: well-quasi orders, controlled bed sequences, reasoning and inductive inference

**ACM Reference Format:**


**1 Introduction**

A well-quasi-order (wqo) over a set $A$ is a reflexive and transitive relation $\preceq$ such that every infinite sequence $a_0, a_1, a_2, \ldots$ in $A$ has a subsequence $a_{i_0}, a_{i_1}, a_{i_2}, \ldots$ that is $\preceq$-finite for every $n$. A sequence over a wqo is called a bed sequence over a wqo. We define a bed sequence over a wqo as a sequence that contains infinitely many $\preceq$-finite subsequences.

Yield $F_{\omega}^\omega$ upper bounds for

- satisfiability of Forward XPath $\epsilon$ and
- provability in $\mathbb{R}^\rightarrow, \wedge$. 

**Complexity of controlled bed sequences over finite sets of $\mathbb{N}^d$**

A. R. Balasubramanian
Technical University of Munich, Germany
bala@in.tum.de

**Abstract**

We prove upper and lower bounds for the length of controlled bed sequences over the Hoare and the Smyth quasi-orderings on finite sets of $\mathbb{N}^d$. These results are obtained by bounding the length of such sequence by functions from the complexity of branching systems. This allows us to translate these results to bounds over the fast-growing hierarchy classes. The obtained bounds are proven to be tight for the major ordering, which solves a problem left open by Abriola, Figueira & Senno (Theor. Comp. Sci., Vol. 435). Finally, we see the results on controlled bed sequences in prove upper bounds for the emptiness problem of some classes of automata.

**CoC Concepts:** Theory of computation $\rightarrow$ Complexity classes, Program verification

Keywords: well-quasi orders, controlled bed sequences, reasoning and inductive inference

**ACM Reference Format:**


**1 Introduction**

A well-quasi-order (wqo) over a set $A$ is a reflexive and transitive relation $\preceq$ such that every infinite sequence $a_0, a_1, a_2, \ldots$ in $A$ has a subsequence $a_{i_0}, a_{i_1}, a_{i_2}, \ldots$ that is $\preceq$-finite for every $n$. A sequence over a wqo is called a bed sequence over a wqo. We define a bed sequence over a wqo as a sequence that contains infinitely many $\preceq$-finite subsequences.

Yield $F_{\omega}^\omega$ upper bounds for

- satisfiability of Forward XPath $\epsilon$ and
- provability in $\mathbb{R}^\rightarrow, \wedge$. 

**Complexity of controlled bed sequences over finite sets of $\mathbb{N}^d$**

A. R. Balasubramanian
Technical University of Munich, Germany
bala@in.tum.de

**Abstract**

We prove upper and lower bounds for the length of controlled bed sequences over the Hoare and the Smyth quasi-orderings on finite sets of $\mathbb{N}^d$. These results are obtained by bounding the length of such sequence by functions from the complexity of branching systems. This allows us to translate these results to bounds over the fast-growing hierarchy classes. The obtained bounds are proven to be tight for the major ordering, which solves a problem left open by Abriola, Figueira & Senno (Theor. Comp. Sci., Vol. 435). Finally, we see the results on controlled bed sequences in prove upper bounds for the emptiness problem of some classes of automata.

**CoC Concepts:** Theory of computation $\rightarrow$ Complexity classes, Program verification

Keywords: well-quasi orders, controlled bed sequences, reasoning and inductive inference

**ACM Reference Format:**


**1 Introduction**

A well-quasi-order (wqo) over a set $A$ is a reflexive and transitive relation $\preceq$ such that every infinite sequence $a_0, a_1, a_2, \ldots$ in $A$ has a subsequence $a_{i_0}, a_{i_1}, a_{i_2}, \ldots$ that is $\preceq$-finite for every $n$. A sequence over a wqo is called a bed sequence over a wqo. We define a bed sequence over a wqo as a sequence that contains infinitely many $\preceq$-finite subsequences.

Yield $F_{\omega}^\omega$ upper bounds for

- satisfiability of Forward XPath $\epsilon$ and
- provability in $\mathbb{R}^\rightarrow, \wedge$. 
Complexity Upper Bounds

[Urquhart ’90, ’99]

Theorem (Urquhart ’90)

Provability in $\mathbf{LR}^+$ is in $\mathbf{F}_\omega$.

based on [Kripke ’59] and a length function theorem by [McAloon ’84].
WITNESSES
WITNESSES

\[ x \rightarrow \{ y_1, \ldots, y_n \} \]
Pseudo Witnesses

\[ r \wedge \ldots \wedge \ell \]

\[ x \rightarrow \{ y'_1, \ldots, y'_n \} \]
**Pseudo Witnesses**

\[
\begin{align*}
r & \quad \mathcal{X} \\
\downarrow & \quad \downarrow \\
y'_1 \cdots y'_n & \quad \{y'_1, \ldots, y'_n\} \in \text{Post}_\exists(\uparrow\mathcal{X})
\end{align*}
\]
IRREDUNDANT PSEUDO WITNESSES

\[ x \preceq x' \]
Irredundant Pseudo Witnesses

otherwise if $x \leq x'$

\[ r \]
**Irredundant Pseudo Witnesses**

\[ \{y'_1, \ldots, y'_n\} \in \min_{\subseteq H} \text{Post}_\exists (\uparrow x) \]
IRREDUNDANT PSEUDO WITNESSES

- there exists an irredundant pseudo witness iff subcoverability holds

- the sequences $r = x_0, x_1, \ldots, x_m \leq \ell$ along the branches are bad

- because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq H} \text{Post}_{\exists}(\uparrow x_i)$, we may be able to control those sequences

- also need to control the “and-branching degree”, e.g. if $x \rightarrow S$ then $|S| \leq f(||x||)$ for some reasonable $f$

- length function theorems then allow to bound the height and degree of irredundant pseudo witnesses, thus their size
IRREDUNDANT PSEUDO WITNESSES

- there exists an irredundant pseudo witness iff subcoverability holds

- the sequences $r = x_0, x_1, \ldots, x_m \leq \ell$ along the branches are **bad**

- because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq H} \text{Post}_{\exists}(\uparrow x_i)$, we may be able to control those sequences

- also need to control the “and-branching degree”, e.g. if $x \rightarrow S$ then $|S| \leq f(||x||)$ for some reasonable $f$

- length function theorems then allow to bound the height and degree of irredundant pseudo witnesses, thus their size
IRREDUNDANT PSEUDO WITNESSES

- there exists an irredundant pseudo witness iff subcoverability holds

- the sequences $r = x_0, x_1, \ldots, x_m \leq \ell$ along the branches are bad

- because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq H} \text{Post}_\exists(\uparrow x_i)$, we may be able to control those sequences

- also need to control the "and-branching degree", e.g. if $x \rightarrow S$ then $|S| \leq f(\|x\|)$ for some reasonable $f$

- length function theorems then allow to bound the height and degree of irredundant pseudo witnesses, thus their size
IRREDUNDANT PSEUDO WITNESSES

- there exists an irredundant pseudo witness iff subcoverability holds

- the sequences $r = x_0, x_1, \ldots, x_m \leq \ell$ along the branches are bad

- because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq H} \text{Post}_\exists(\uparrow x_i)$, we may be able to control those sequences

- also need to control the “and-branching degree”, e.g. if $x \rightarrow S$ then $|S| \leq f(\|x\|)$ for some reasonable $f$

- length function theorems then allow to bound the height and degree of irredundant pseudo witnesses, thus their size
IRREDUNDANT PSEUDO WITNESSES

- There exists an irredundant pseudo witness iff subcoverability holds.

- The sequences $r = x_0, x_1, \ldots, x_m \leq \ell$ along the branches are bad.

- Because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq H} \text{Post}_\exists(\uparrow x_i)$, we may be able to control those sequences.

- Also need to control the “and-branching degree”, e.g. if $x \rightarrow S$ then $|S| \leq f(||x||)$ for some reasonable $f$.

- Length function theorems then allow to bound the height and degree of irredundant pseudo witnesses, thus their size.
CONCLUDING REMARKS

- lack of a systematic investigation of branching WSTS
- complexity analysis through lifting might not be optimal
- in this talk, straightforward generalisation of techniques originally developed by [Kripke ’59] to leaf subcoverability
- in [Lazić & S. ’21], algorithms for root and leaf coverability
More examples of WQOs

**Examples of WQOs**

**Lemma** (Finite sequences, Higman 1952)

If $(X, \leq_X)$ is a wqo, then $(X^{<\omega}, \sqsubseteq)$ is a wqo for the subword embedding:

$$a_1 \cdots a_m \sqsubseteq b_1 \cdots b_n \overset{\text{def}}{\iff} \begin{cases} \exists 1 \leq i_1 < \cdots < i_m \leq n, \\ \land_{j=1}^m a_j \leq_X b_{i_j}. \end{cases}$$

**Example**

$$aba \sqsubseteq baaacabbbab$$
More examples of WQOs

**Examples of WQOs**

**Lemma** (Bounded tree-depth graphs, Ding 1992)

> For all \(k\), \((\text{Graphs} \uparrow P_k, \subseteq)\) and \((\text{Graphs} \downarrow P_k, \subseteq_i)\) are wqo.

**Non-Examples**

```
\[ \begin{array}{ccccccc}

\text{H} & \text{H} & \text{H} & \text{H} & \text{...} \\
\text{H} & \text{H} & \text{H} & \text{H} & \text{...} \\
\end{array} \]
```

```
\[ \begin{array}{ccccccc}

\text{...} & \text{...} & \text{...} & \text{...} & \text{...} \\
\end{array} \]
```