# Branching in Well-Structured Transition Systems

Sylvain Schmitz

CSL 2021, January 25, 2021

# 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

# 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

# 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, Čerāns, 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


Theoretical Computer Science
www.elsevier.com/locatetcs
ELSEVIER
Theoretical Computer Science 256 (2001) 63–92

Well-structured transition systems everywhere!

A. Finkel, Ph. Schnoebelen*
Lab. Spec cation and Veri cation, ENS de Cachan & CNRS UMR 8643, 61 av. Pdt Wilson, 94235 Cachan Cedex, France

Abstract
Well-structured transition systems (WSTSs) are a general class of in- nite-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 idea and show several new results. Our improved de- nitions allow many examples of classical systems to be seen as instances of WSTSs. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: In- nite systems; Veri- cation; Well-quasi-ordering

1. Introduction

1.1. Veri cation of in nite-state systems

Formal veri- cation of programs and systems is a very active - eld for both theoretical research and practical developments, especially since impressive advances in formal veri- cation technology proved feasible in several realistic applications from the industrial world. The highly successful model-checking approach for - nite systems [16] suggested that a working veri- cation technology could well be developed for systems


Information and Computation 160, 109–127 (2000)
doi:10.1006/inco.1999.2843, available online at http://www.idealibrary.com on IDEAL®

Algorithmic Analysis of Programs with Well Quasi-ordered Domains[1]

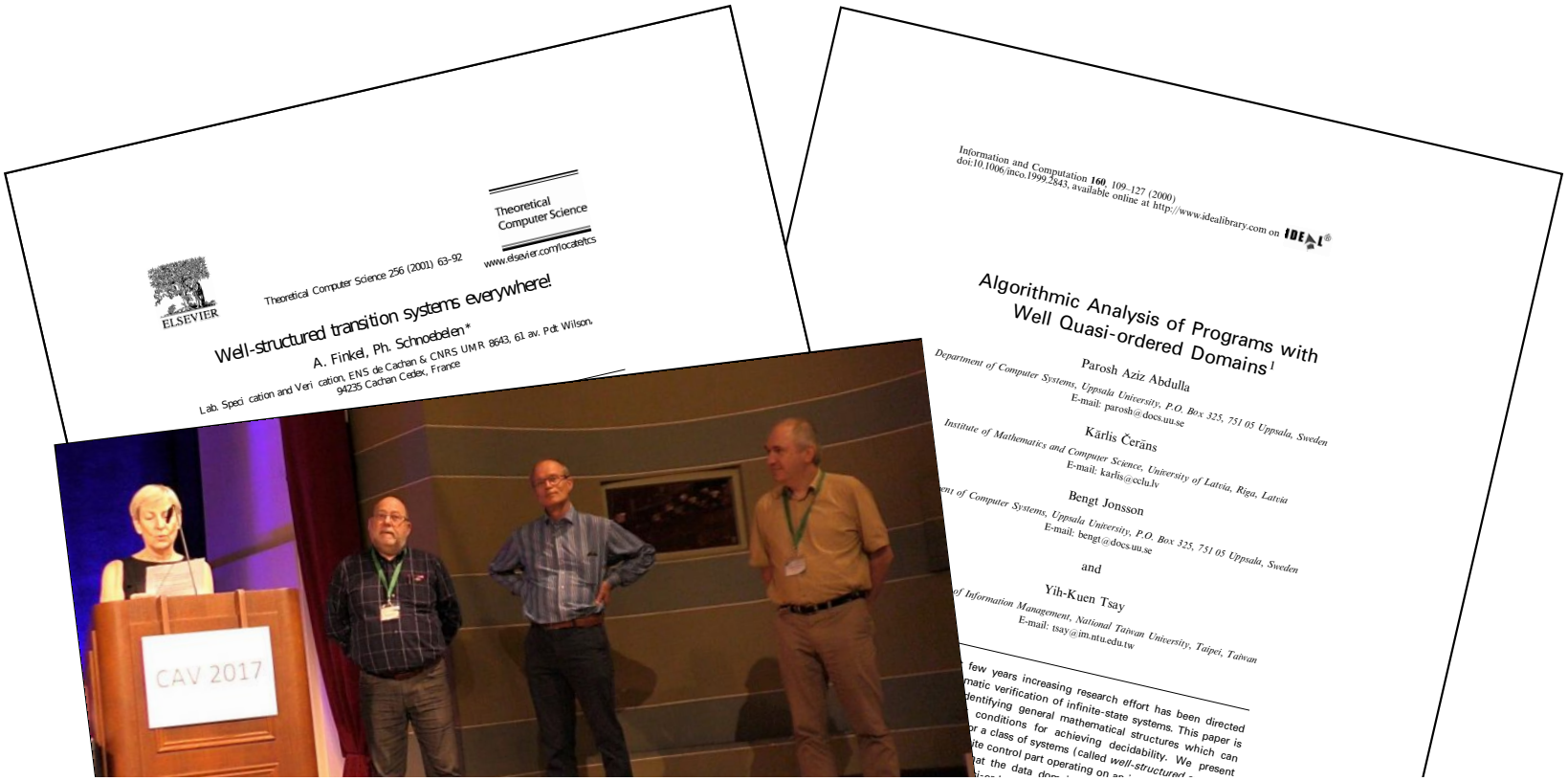
Parosh Aziz Abdulla
Department of Computer Systems, Uppsala University, P.O. Box 325, 751 05 Uppsala, Sweden
E-mail: parosh@docs.uu.se
Kārlis Čerāns
Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia
E-mail: karlis@cclu.lv
Bengt Jonsson
Department of Computer Systems, Uppsala University, P.O. Box 325, 751 05 Uppsala, Sweden
E-mail: bengt@docs.uu.se
and
Yih-Kuen Tsay
Department of Information Management, National Taiwan University, Taipei, Taiwan
E-mail: tsay@im.ntu.edu.tw

Over the past few years increasing research effort has been directed towards the automatic verification of infinite-state systems. This paper is concerned with identifying general mathematical structures which can be as sufficient conditions for achieving decidability. We present decidability results for a class of systems (called *well-structured*
consist of a finite control part operating on an
ults assume that the data do
a well quasi-

# Well Structured Transition Systems

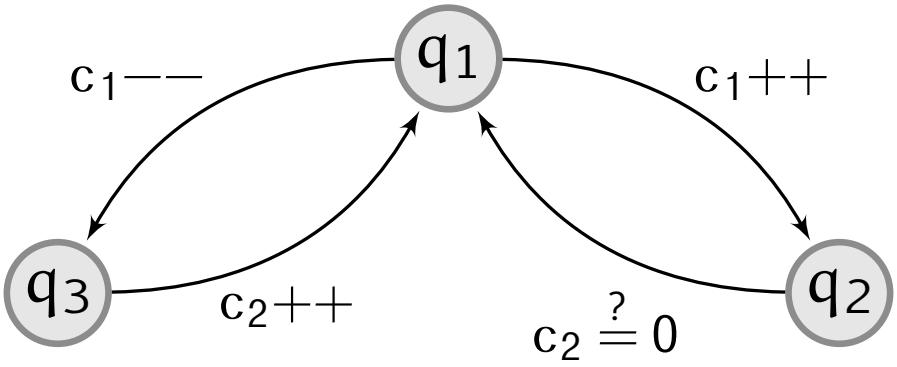
[Abdulla, Čerāns, Jonsson & Tsay '00; Finkel & Schnoebelen '01]

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



# Example: Lossy Counter Machines

## Example

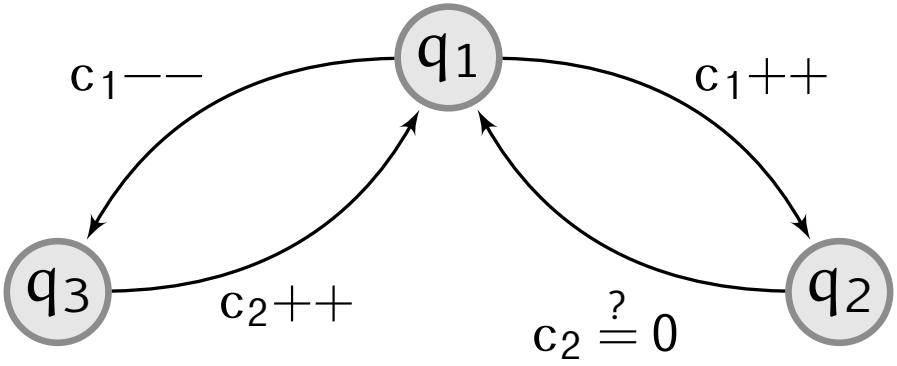



## Lossy Semantics

$$q_1(0,2) \xrightarrow{c_1{+}{+}}_{\ell} q_2(1,1) \xrightarrow{c_2 \stackrel{?}{=} 0}_{\ell} q_1(0,0)$$

# Example: Lossy Counter Machines

## Example

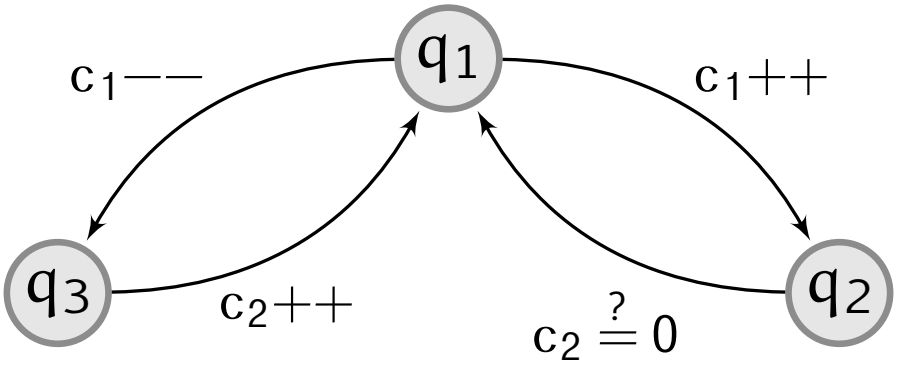



## Lossy Semantics

$$q_1(0,2) \xrightarrow{c_1{+}{+}}_{\ell} q_2(1,1) \xrightarrow{c_2 \stackrel{?}{=} 0}_{\ell} q_1(0,0)$$

# Example: Lossy Counter Machines

## Example




## Lossy Semantics

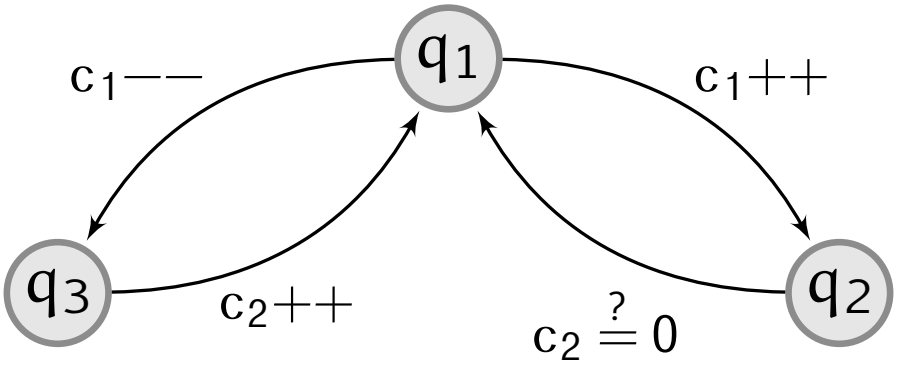
$$q_1(0,2) \xrightarrow{c_1{+}{+}}_{\ell} q_2(1,1) \xrightarrow{c_2 \stackrel{?}{=} 0}_{\ell} q_1(0,0)$$

$$\rotatebox{-90}{$\leq$} \qquad\qquad \rotatebox{-90}{$\geq$}$$

$$q_1(0,2) \xrightarrow{c_1{+}{+}} q_2(1,2)$$

# Example: Lossy Counter Machines

## Example




## Lossy Semantics

$$q_1(0,2) \xrightarrow{c_1{+}{+}}_{\ell} q_2(1,1) \xrightarrow{c_2 \stackrel{?}{=} 0}_{\ell} q_1(0,0)$$

$$\rotatebox{90}{$\leq$} \qquad\qquad \rotatebox{90}{$\leq$}$$

$$q_2(1,0) \xrightarrow{c_2 \stackrel{?}{=} 0} q_1(1,0)$$

# Coverability

**Coverability Problem for LCM**

input: an LCM $M$, initial configuration $q_0(\mathbf{v}_0)$, target configuration $q_f(\mathbf{v}_f)$

question: $\exists \mathbf{v} \geqslant \mathbf{v}_f \,.\, q_0(\mathbf{v}_0) \to_\ell^* q_f(\mathbf{v})$?

# Coverability

Coverability Problem for LCM

input an LCM $M$, initial configuration $q_0(\mathbf{v}_0)$, target configuration $q_f(\mathbf{v}_f)$

question $\exists \mathbf{v} \geqslant \mathbf{v}_f \,.\, q_0(\mathbf{v}_0) \to_\ell^* q_f(\mathbf{v})$?

Remark

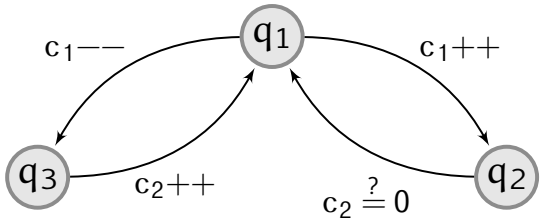
equivalent to reachability:

question $q_0(\mathbf{v}_0) \to_\ell^* q_f(\mathbf{v}_f)$?

# Backward Coverability Algorithm

[Arnold & Latteux'78]

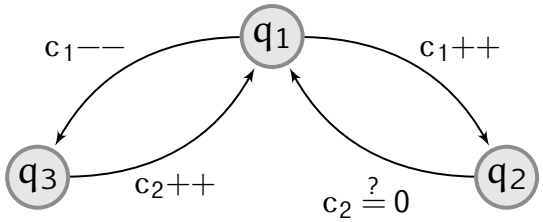
Example: coverability of $q_2(1,1)$ in



# Backward Coverability Algorithm

[Arnold & Latteux'78]

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

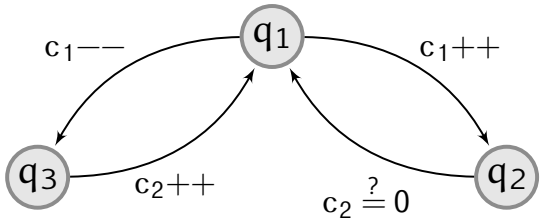



$$U_k \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant k} q_2(\mathbf{v}')\}$$

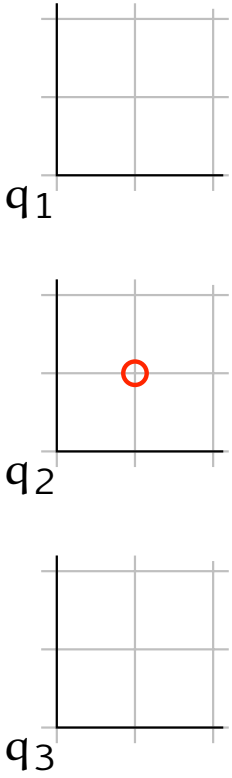
# Backward Coverability Algorithm

[Arnold & Latteux'78]

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




$$U_0 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 0} q_2(\mathbf{v}')\}$$

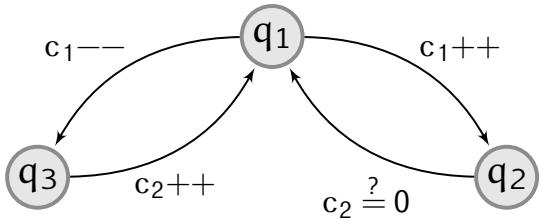



$q_2(1,1)$

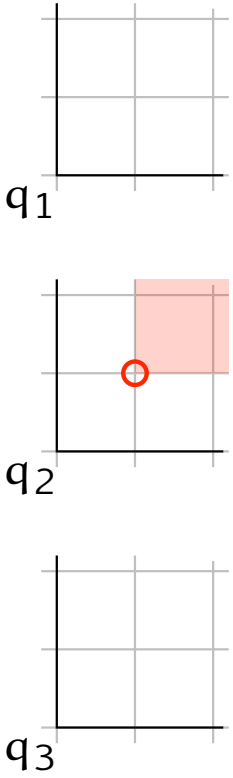
# Backward Coverability Algorithm

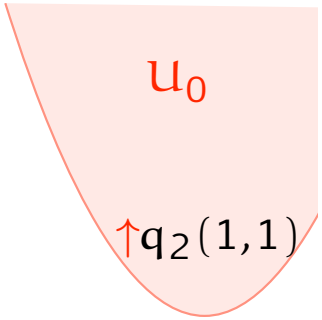
[Arnold & Latteux'78]

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




$$U_0 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 0} q_2(\mathbf{v}')\}$$

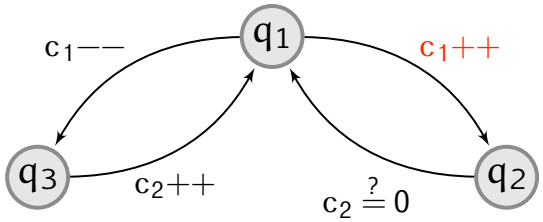


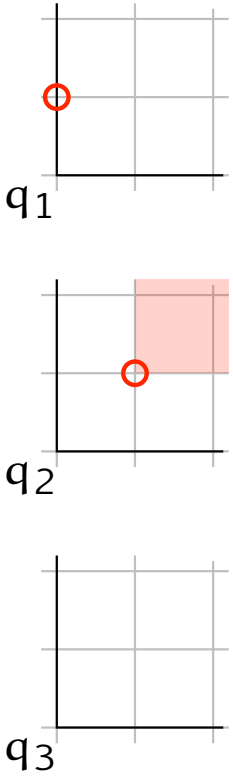
# Backward Coverability Algorithm

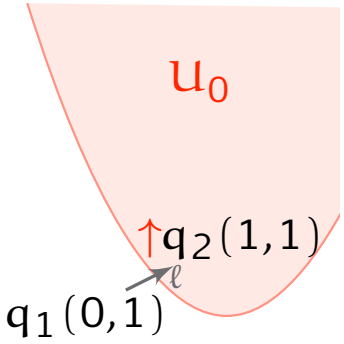
[Arnold & Latteux'78]

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




$$U_1 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 1} q_2(\mathbf{v}')\}$$

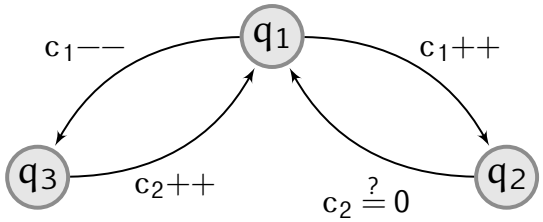


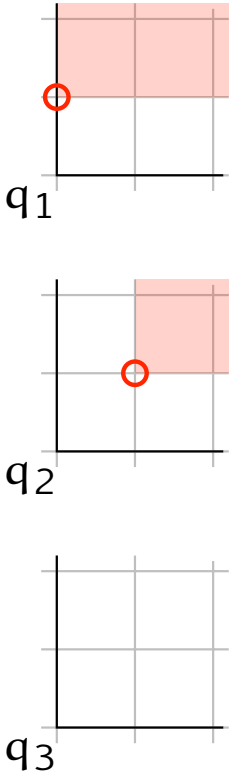
# Backward Coverability Algorithm

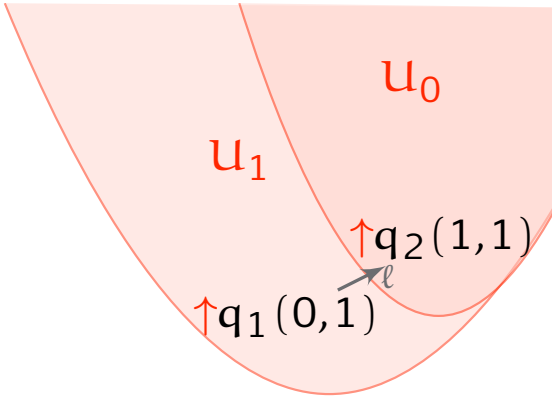
[Arnold & Latteux'78]

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




$$U_1 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 1} q_2(\mathbf{v}')\}$$

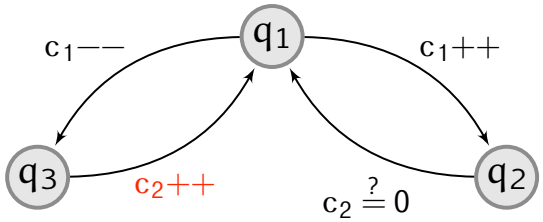


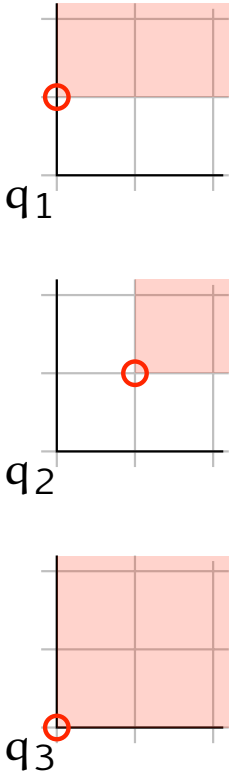
# Backward Coverability Algorithm

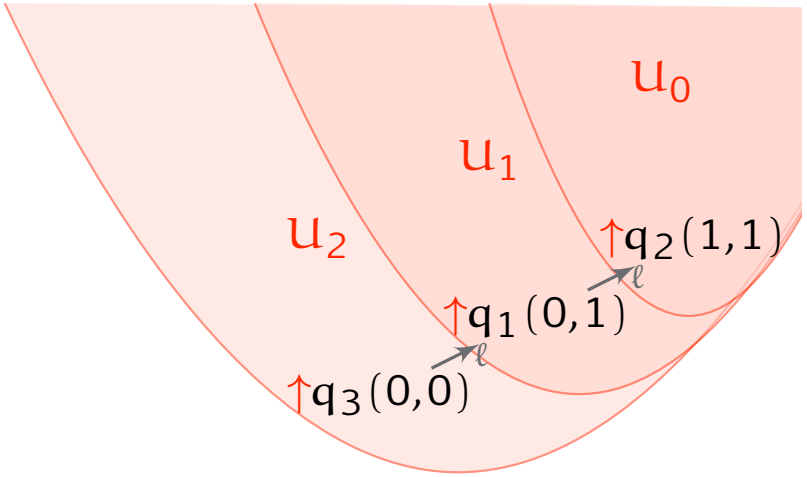
[Arnold & Latteux'78]

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




$$U_2 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 2} q_2(\mathbf{v}')\}$$

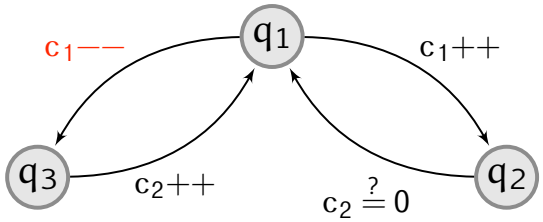


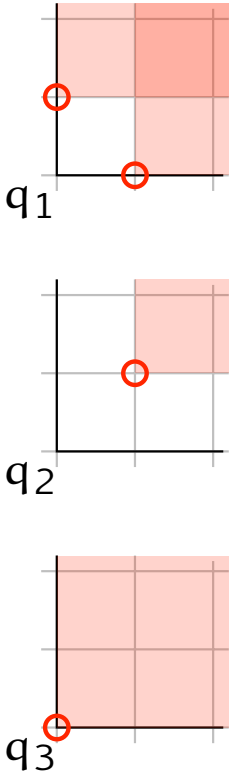
# Backward Coverability Algorithm

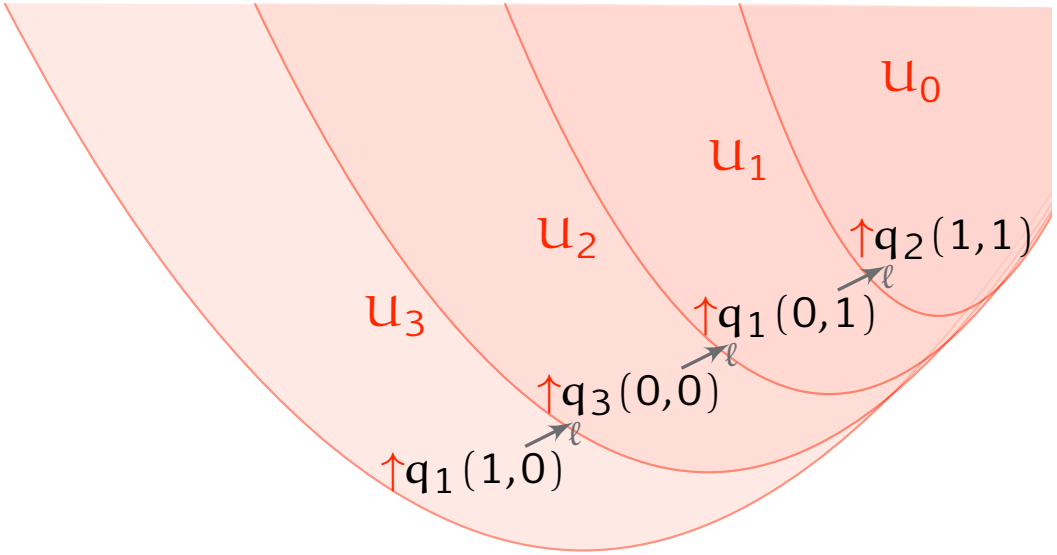
[Arnold & Latteux'78]

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




$$U_3 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 3} q_2(\mathbf{v}')\}$$

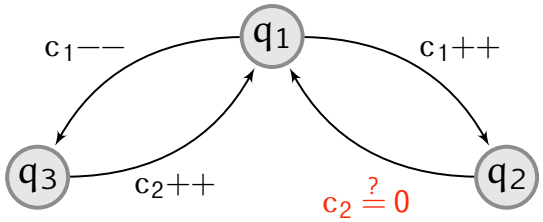


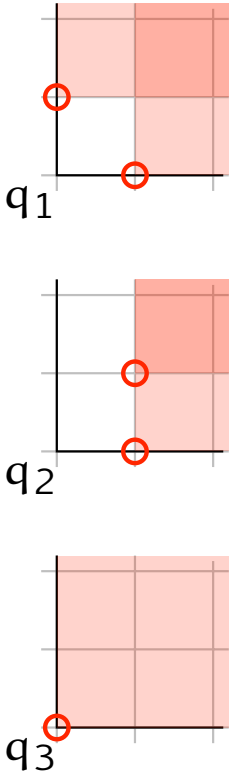
# Backward Coverability Algorithm

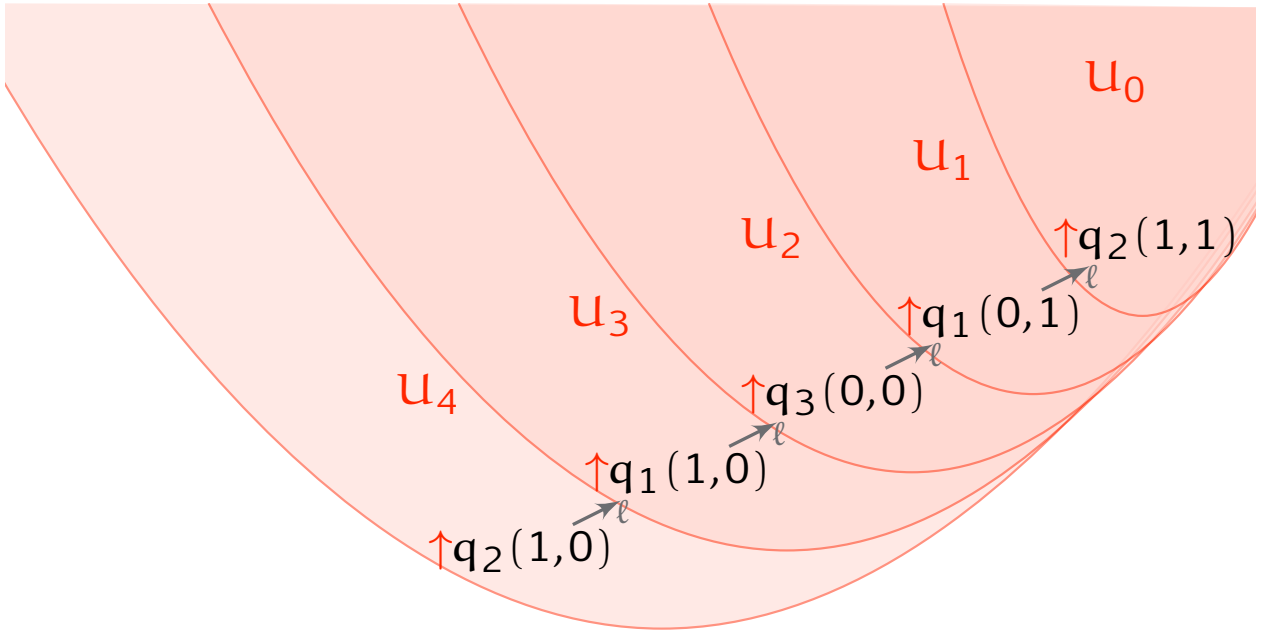
[Arnold & Latteux'78]

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




$$U_4 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 4} q_2(\mathbf{v}')\}$$

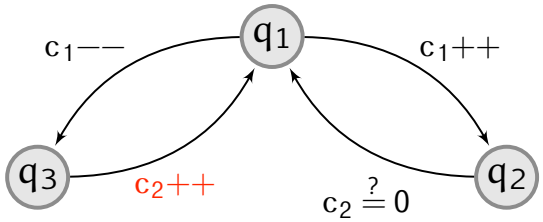


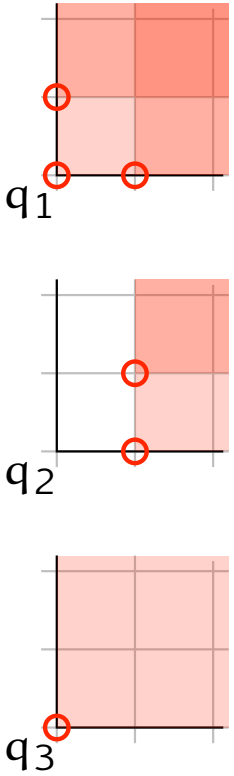
# Backward Coverability Algorithm

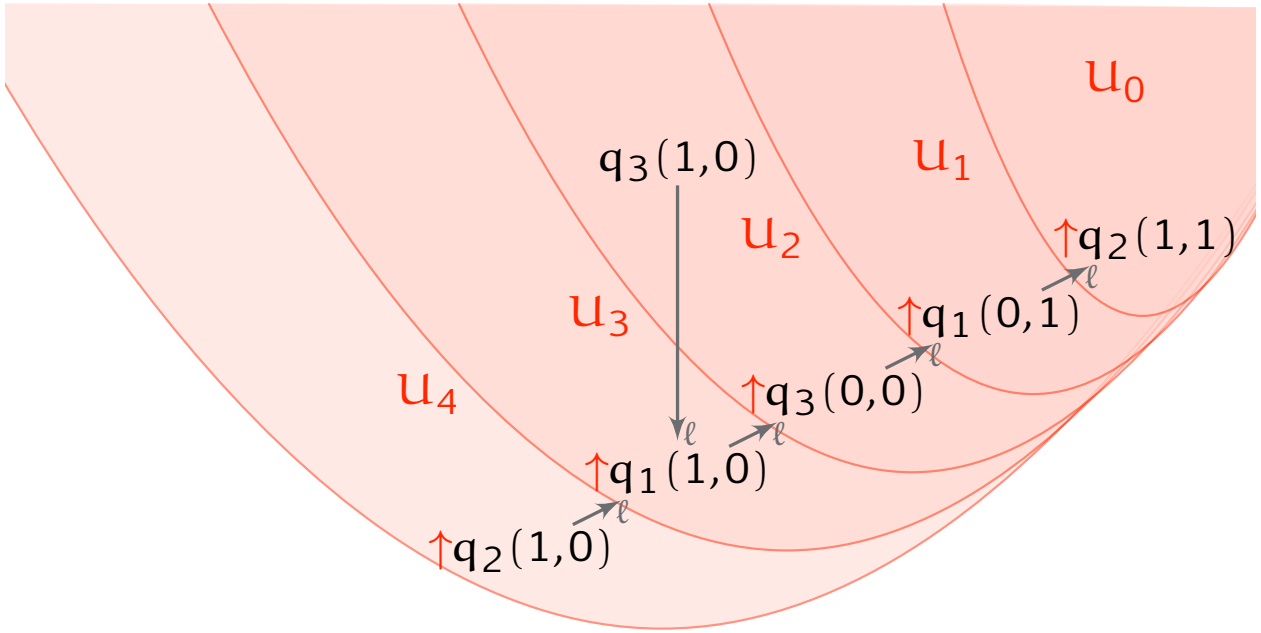
[Arnold & Latteux'78]

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




$$U_4 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 4} q_2(\mathbf{v}')\}$$

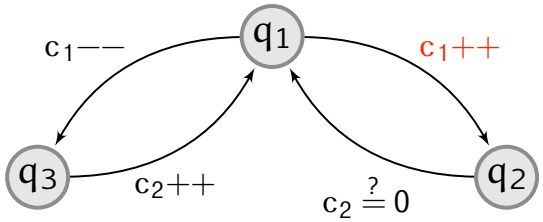


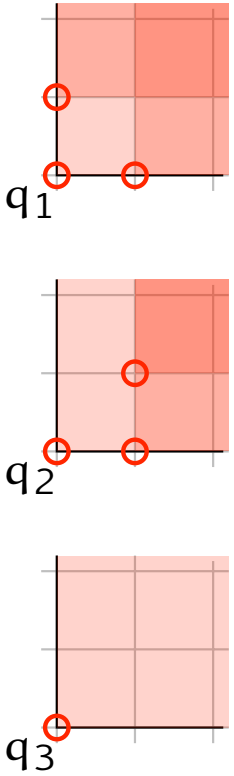
# Backward Coverability Algorithm

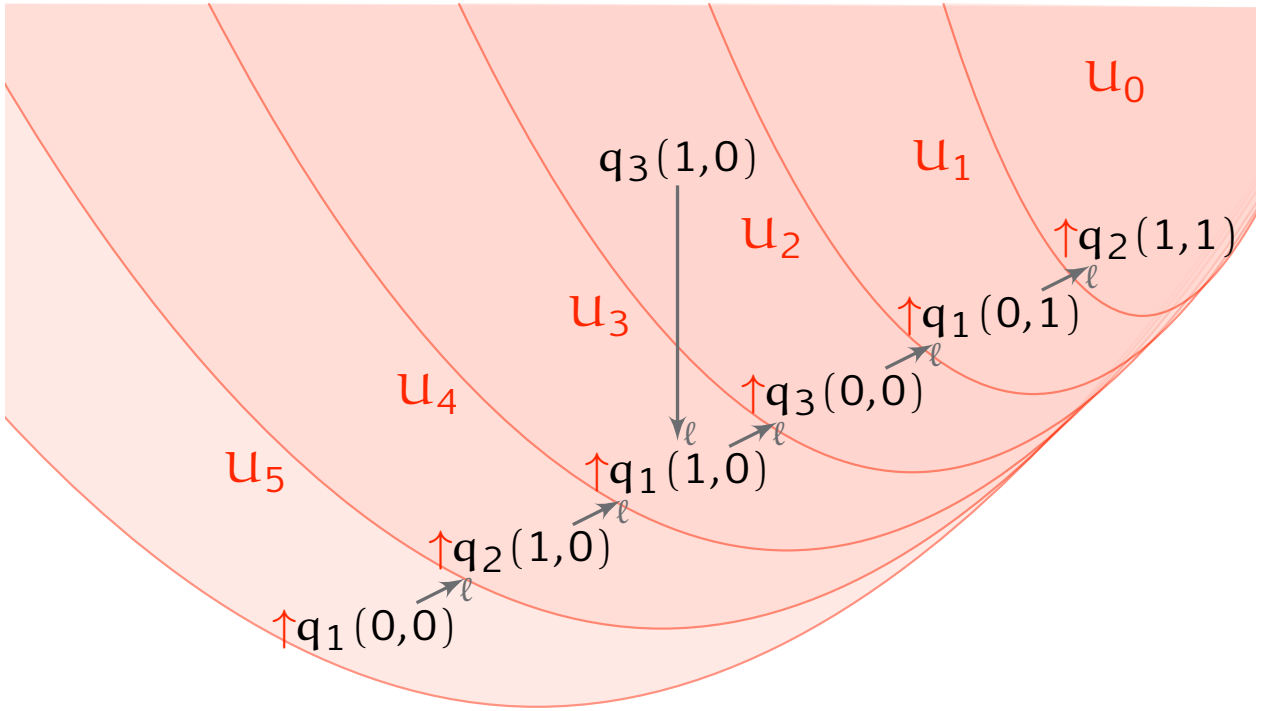
[Arnold & Latteux'78]

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




$$U_5 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 5} q_2(\mathbf{v}')\}$$

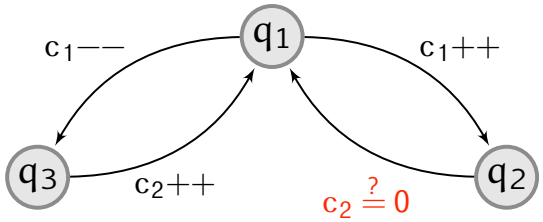


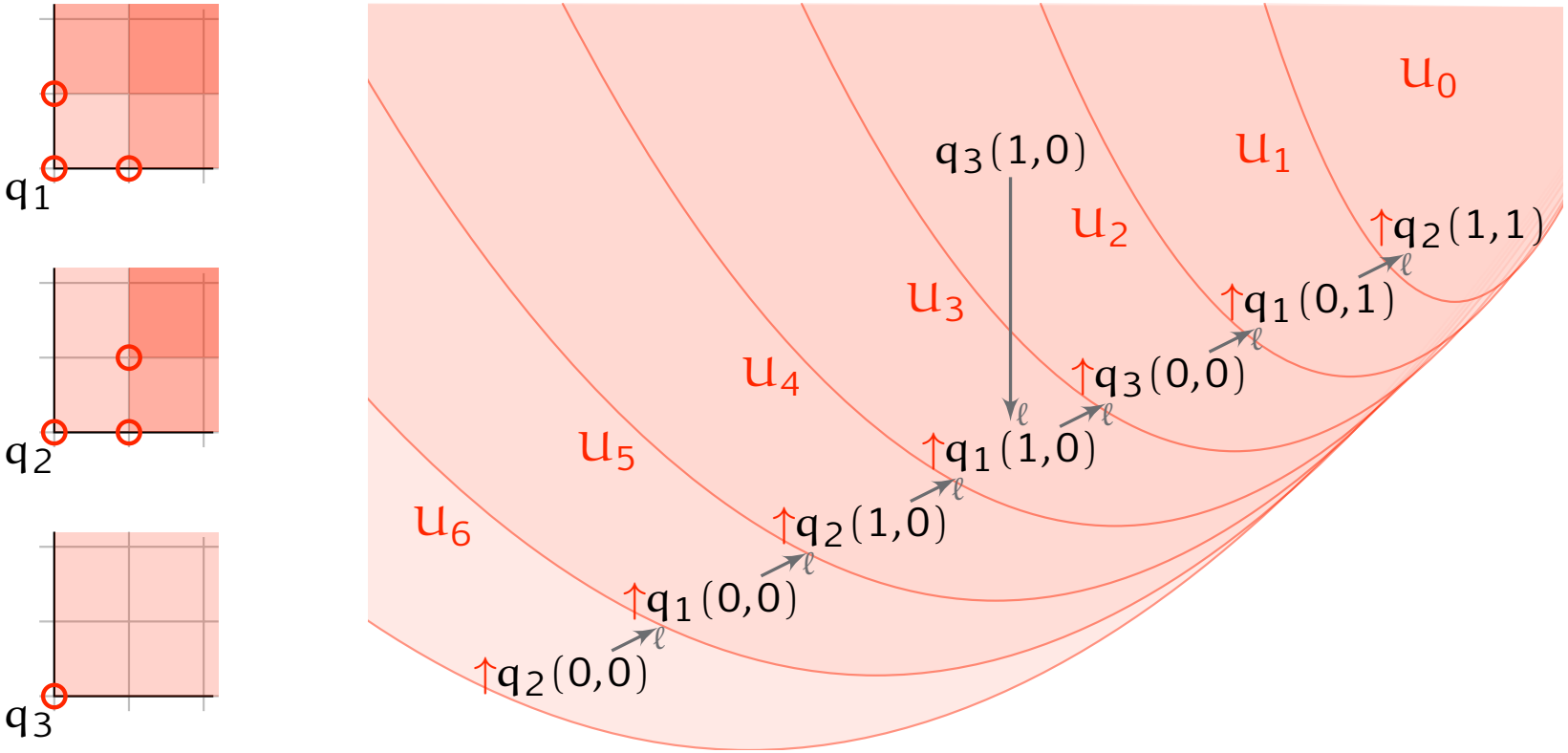
# Backward Coverability Algorithm

[Arnold & Latteux'78]

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



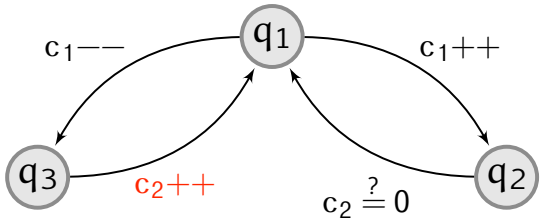

$$U_6 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^{\leqslant 6} q_2(\mathbf{v}')\}$$



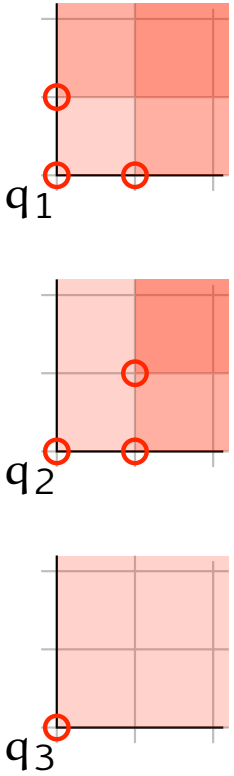
# Backward Coverability Algorithm

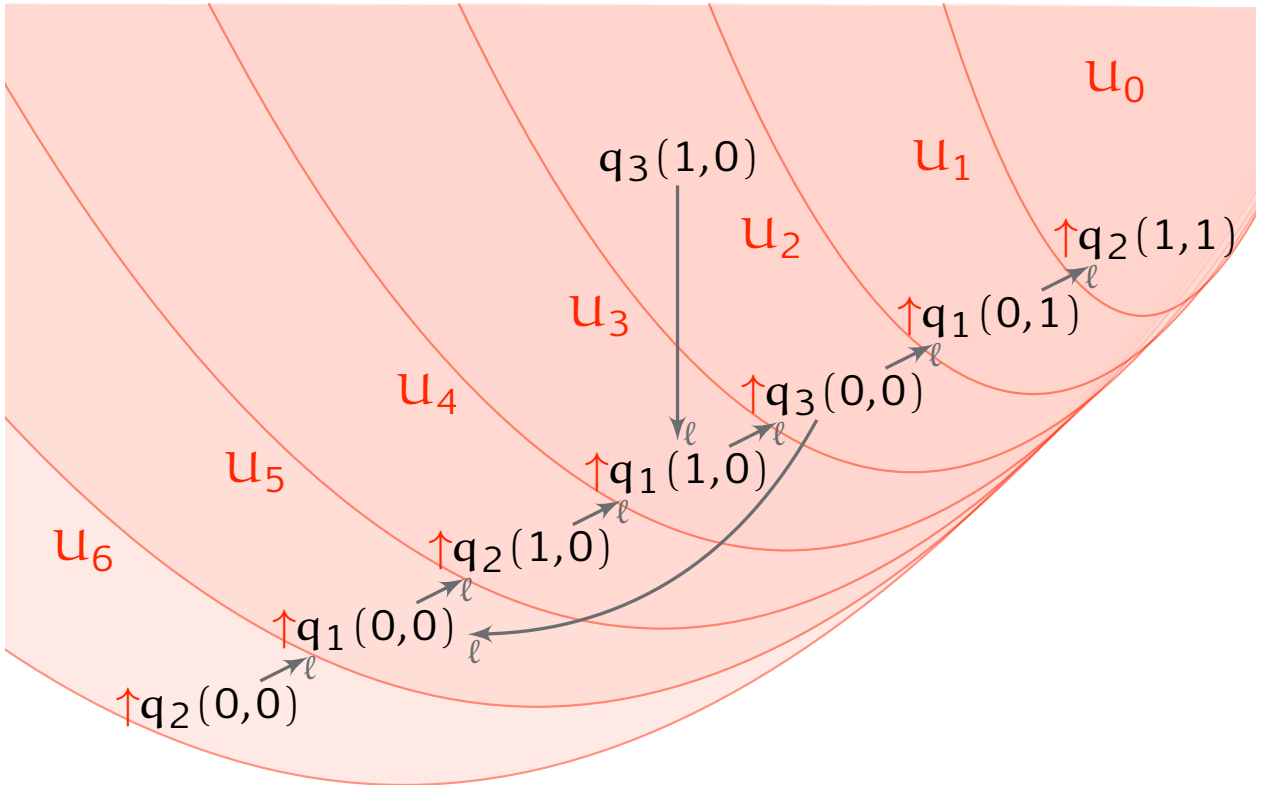
[Arnold & Latteux'78]

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



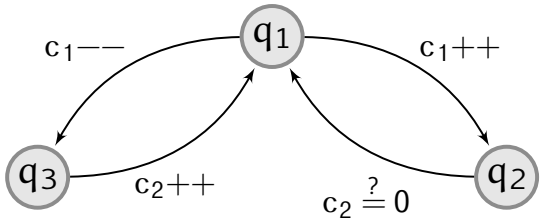

$$U_6 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^* q_2(\mathbf{v}')\}$$



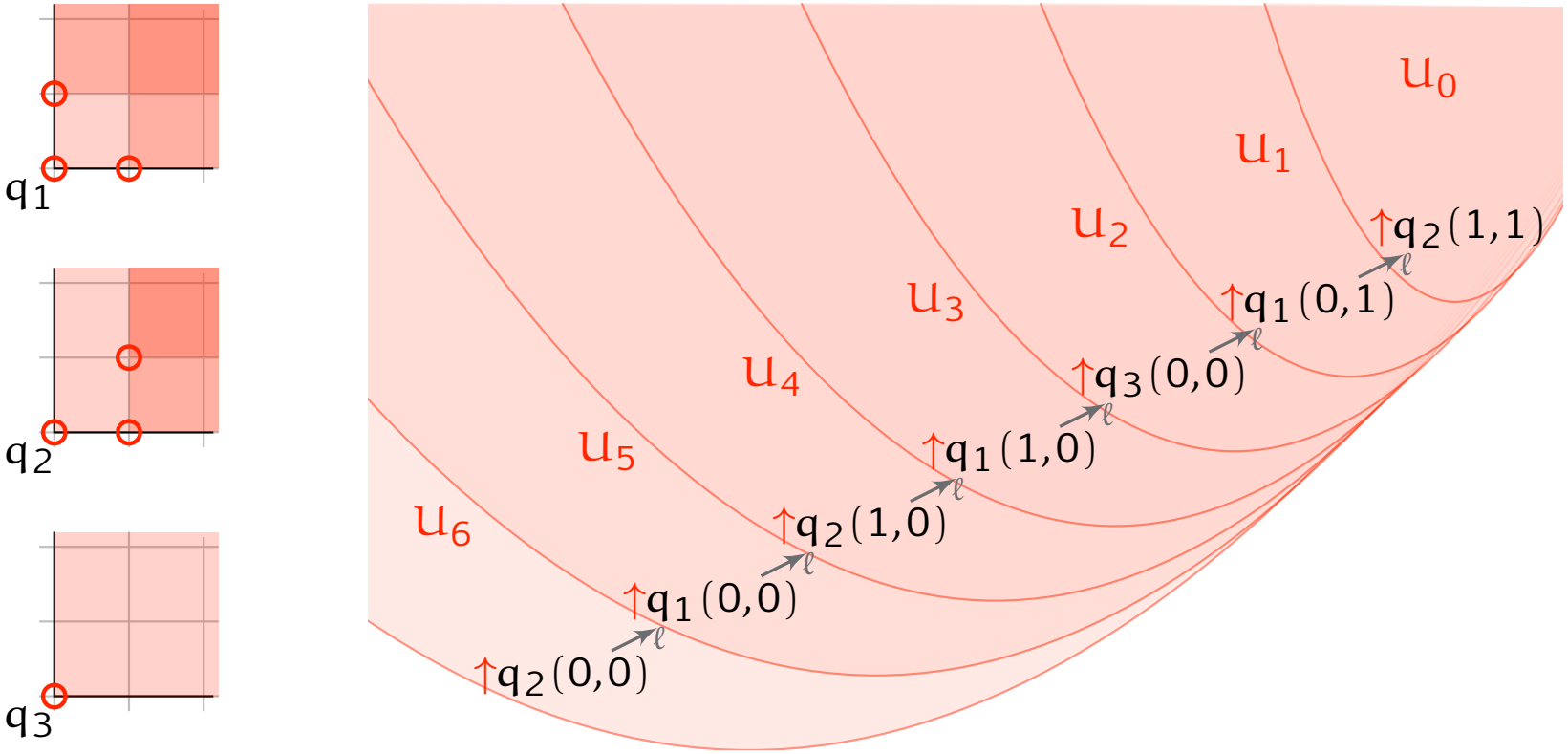
# Backward Coverability Algorithm

[Arnold & Latteux'78]




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

$$U_6 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^* q_2(\mathbf{v}')\}$$



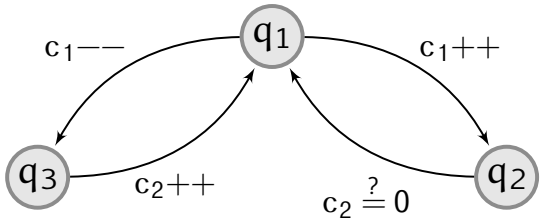

The sequence $U_0 \subsetneq U_1 \subsetneq U_3 \subsetneq U_4 \subsetneq U_5 \subsetneq U_6$ is defined by

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

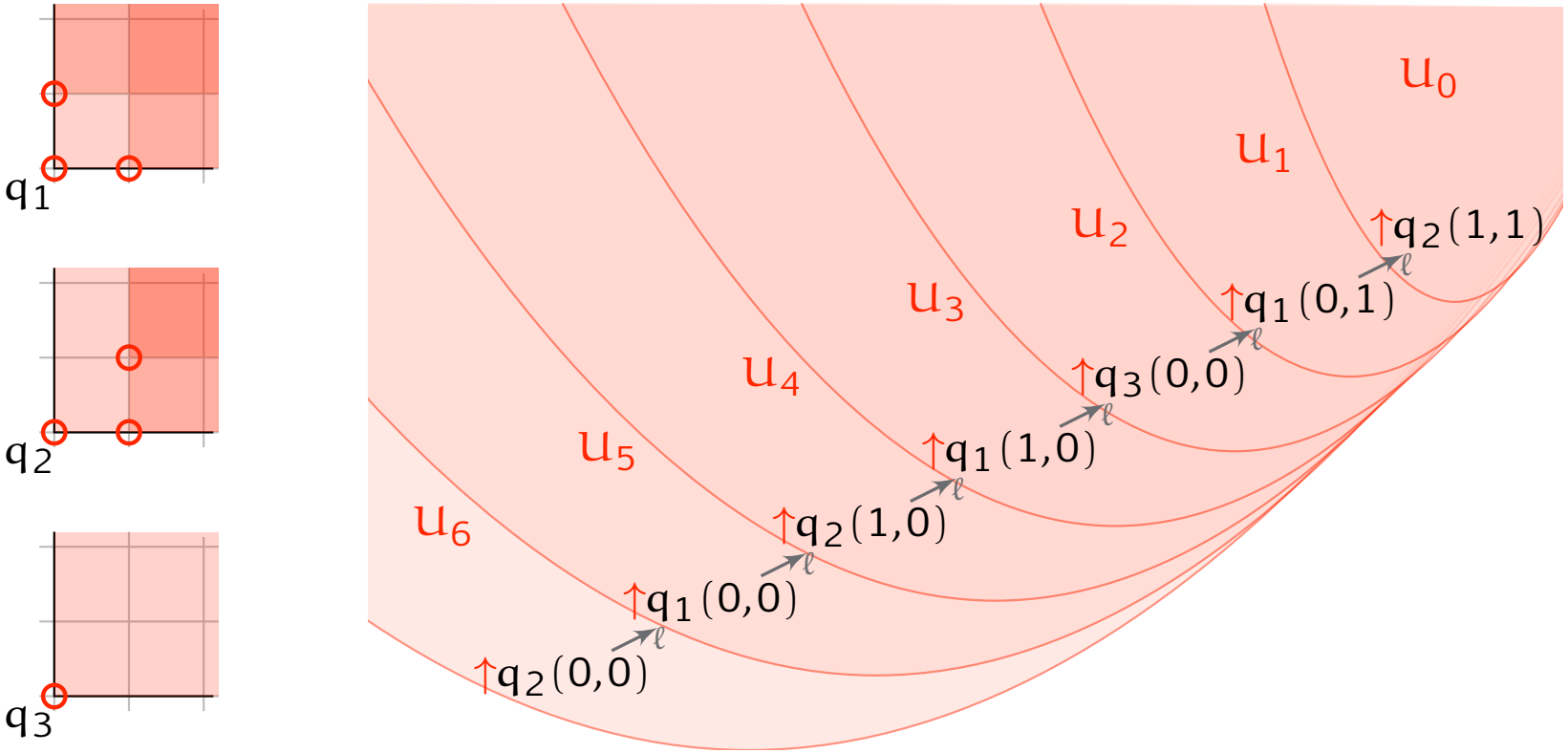
# Backward Coverability Algorithm

[Arnold & Latteux'78]

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



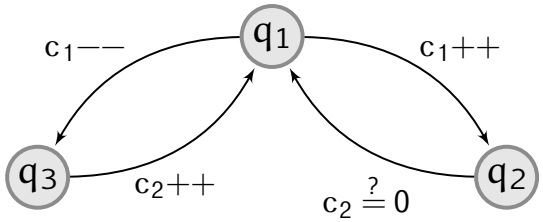

$$U_6 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^* q_2(\mathbf{v}')\}$$




This sequence of upwards-closed sets $U_0 \subsetneq U_1 \subsetneq U_3 \subsetneq U_4 \subsetneq U_5 \subsetneq U_6$ is ascending and thus finite.

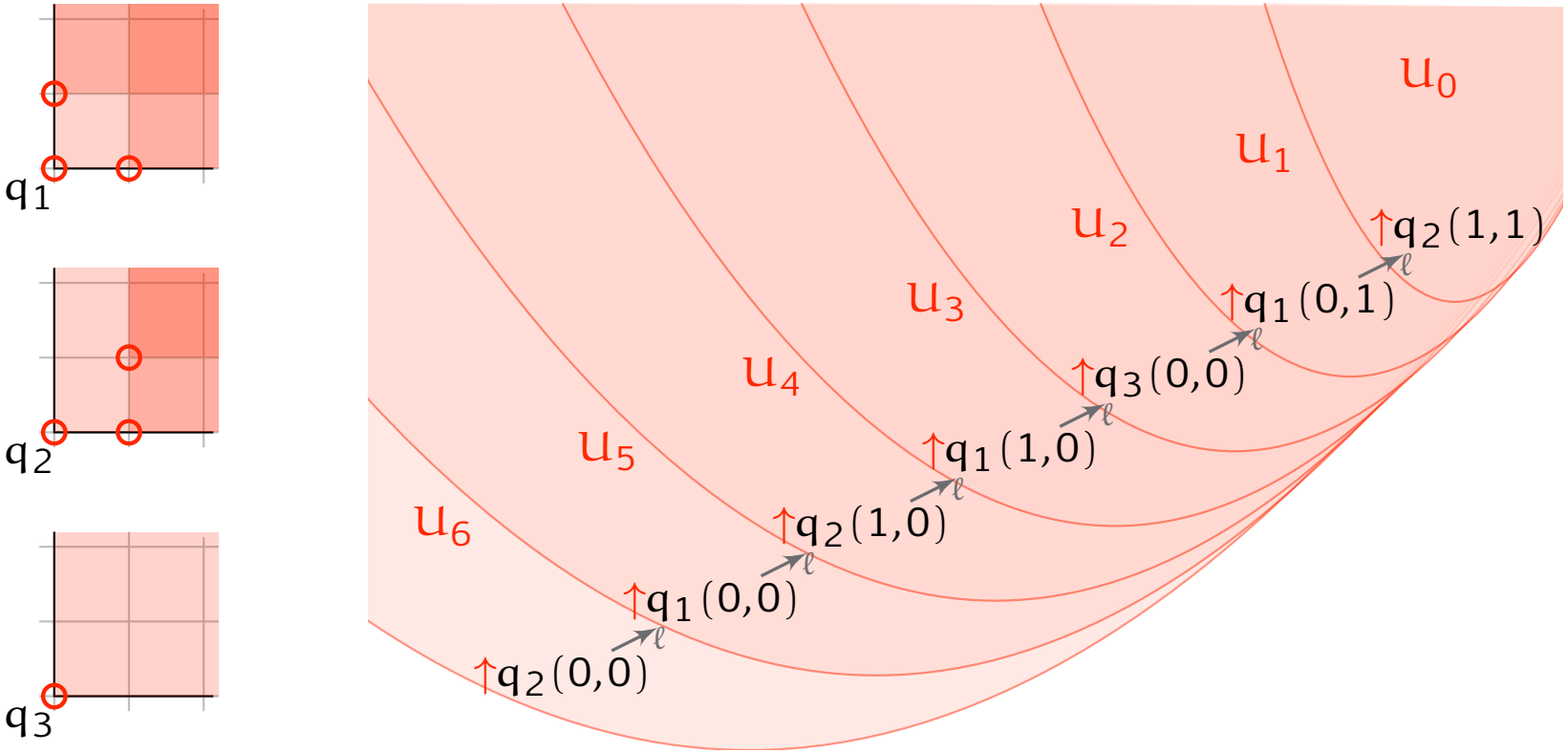
# Backward Coverability Algorithm

[Arnold & Latteux'78]




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

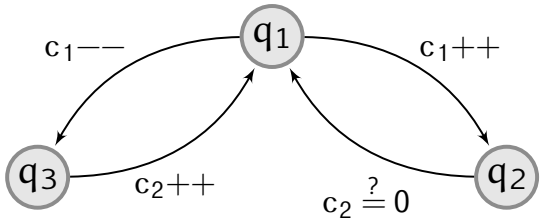
$$U_6 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^* q_2(\mathbf{v}')\}$$




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 $x_0 \stackrel{\text{def}}{=} q_2(1,1)$ and $x_{k+1} \in \min(\mathrm{Pre}_\exists(\uparrow x_k) \setminus U_k)$.

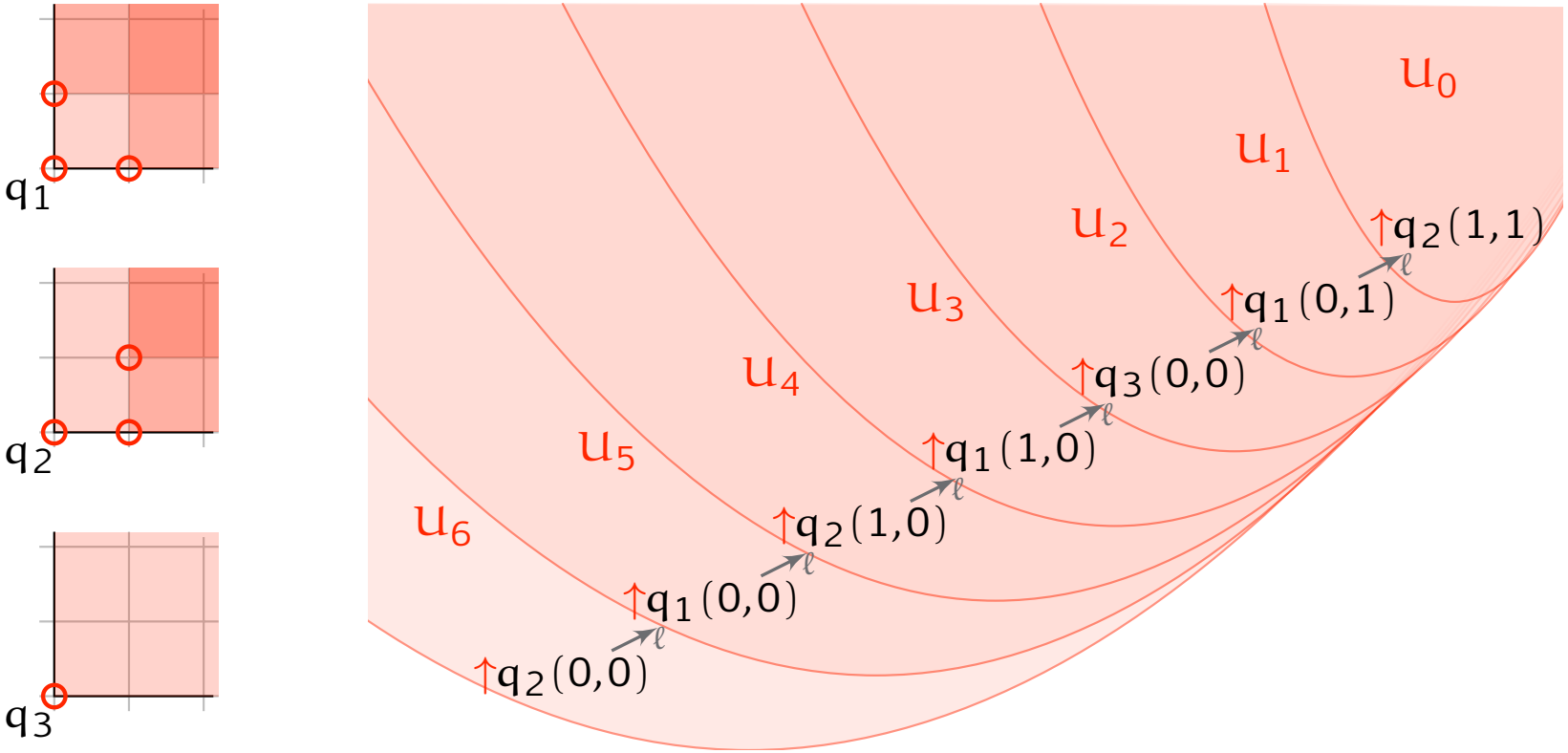
# Backward Coverability Algorithm

[Arnold & Latteux'78]




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

$$U_6 \stackrel{\text{def}}{=} \{q(\mathbf{v}) \mid \exists \mathbf{v}' \geqslant (1,1) \,.\, q(\mathbf{v}) \to_\ell^* q_2(\mathbf{v}')\}$$




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

- $(X, \leqslant)$ is a quasi-order (qo) if $\leqslant$ is transitive and reflexive
- upwards-closure of $S \subseteq X$: $\uparrow S \stackrel{\text{def}}{=} \{y \mid \exists x \in S . x \leqslant y\}$
  (notation: $\uparrow x \stackrel{\text{def}}{=} \uparrow\{x\}$ 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 $\forall i < j . x_i \not\leqslant x_j$

# Well-quasi-orders

- $(X, \leqslant)$ is a quasi-order (qo) if $\leqslant$ is transitive and reflexive
- upwards-closure of $S \subseteq X$: $\uparrow S \stackrel{\text{def}}{=} \{y \mid \exists x \in S . x \leqslant y\}$
  (notation: $\uparrow x \stackrel{\text{def}}{=} \uparrow\{x\}$ 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 $\forall i < j . x_i \not\leqslant x_j$

# Well-quasi-orders

- $(X, \leqslant)$ is a quasi-order (qo) if $\leqslant$ is transitive and reflexive
- upwards-closure of $S \subseteq X$: $\uparrow S \stackrel{\text{def}}{=} \{y \mid \exists x \in S . x \leqslant y\}$
  (notation: $\uparrow x \stackrel{\text{def}}{=} \uparrow\{x\}$ 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 $\forall i < j . x_i \not\leqslant x_j$

# Well-quasi-orders

- $(X, \leqslant)$ is a quasi-order (qo) if $\leqslant$ is transitive and reflexive
- upwards-closure of $S \subseteq X$: $\uparrow S \stackrel{\text{def}}{=} \{y \mid \exists x \in S . x \leqslant y\}$
  (notation: $\uparrow x \stackrel{\text{def}}{=} \uparrow\{x\}$ 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 $\forall i < j . x_i \not\leqslant x_j$

# Well-quasi-orders

- $(X, \leqslant)$ is a quasi-order (qo) if $\leqslant$ is transitive and reflexive
- upwards-closure of $S \subseteq X$: $\uparrow S \stackrel{\text{def}}{=} \{y \mid \exists x \in S . x \leqslant y\}$
  (notation: $\uparrow x \stackrel{\text{def}}{=} \uparrow\{x\}$ 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 $\forall i < j . x_i \not\leqslant x_j$

# Well-quasi-orders

- $(X, \leqslant)$ 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, $\leqslant$ is well-founded and has no infinite antichain
- etc.: very robust notion [Kruskal '72]

# Well-quasi-orders

- $(X, \leqslant)$ 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, $\leqslant$ is well-founded and has no infinite antichain
- etc.: very robust notion [Kruskal '72]

# Well-quasi-orders

- $(X, \leqslant)$ 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, $\leqslant$ is well-founded and has no infinite antichain
- etc.: very robust notion [Kruskal '72]

# Well-quasi-orders

- $(X, \leqslant)$ 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, $\leqslant$ is well-founded and has no infinite antichain
- etc.: very robust notion [Kruskal '72]

# Well-quasi-orders

- $(X, \leqslant)$ 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, $\leqslant$ is well-founded and has no infinite antichain
- etc.: very robust notion [Kruskal '72]

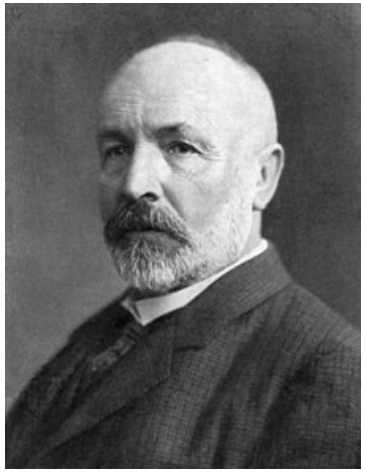
# Examples of WQOs

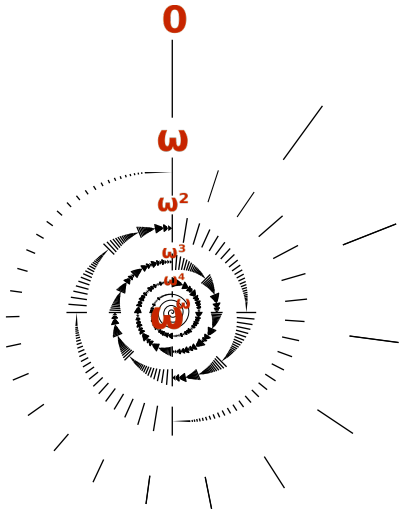
## Ordinals

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* $(X, \leqslant_X)$ *and* $(Y, \leqslant_Y)$ *are two wqos,*
*then* $(X \times Y, \leqslant)$ *is a wqo for the product ordering:*

$$\langle x, y \rangle \leqslant \langle x', y' \rangle \stackrel{\text{def}}{\Leftrightarrow} x \leqslant_X x' \wedge y \leqslant_Y y' .$$

**Example**

LCM configurations in $(Q \times \mathbb{N}^d, \leqslant)$ for a finite set $(Q, =)$

# WSTS

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

- ordered transition system $(X, \leqslant, \rightarrow)$ with $\rightarrow \subseteq X \times X$
- upward (reflexive) compatibility

$$\forall x, x', y$$

$$\begin{array}{ccc} x' & & \\ \rotatebox{90}{$\leqslant$} & & \\ x & \longrightarrow & y \end{array}$$

- $(X, \leqslant)$ is a wqo

# WSTS

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

- ordered transition system $(X, \leqslant, \rightarrow)$ with $\rightarrow \subseteq X \times X$
- upward (reflexive) compatibility

$$\forall x, x', y \quad \begin{array}{ccc} x' & & \\ \rotatebox{90}{$\leqslant$} & & \\ x & \longrightarrow & y \end{array}$$

- $(X, \leqslant)$ is a wqo

# WSTS

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

- ordered transition system $(X, \leqslant, \rightarrow)$ with $\rightarrow \subseteq X \times X$
- upward (reflexive) compatibility

$$\begin{tikzcd}
\forall x, x', y & & \exists y' \\
& x' \arrow[r, dotted] \arrow[d, phantom, "\leqslant"] & y' \arrow[d, phantom, "\leqslant"] \\
& x \arrow[r] & y
\end{tikzcd}$$

- $(X, \leqslant)$ is a wqo

# WSTS

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

- ordered transition system $(X, \leqslant, \rightarrow)$ with $\rightarrow \subseteq X \times X$
- upward (reflexive) compatibility

$$\forall x, x', y \qquad \exists y'$$

$$\begin{tikzcd} x' \arrow[r, dotted, "="] \arrow[d, phantom, "\leqslant"] & y' \arrow[d, phantom, "\leqslant"] \\ x \arrow[r] & y \end{tikzcd}$$

- $(X, \leqslant)$ is a wqo

# WSTS

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

- ordered transition system $(X, \leqslant, \rightarrow)$ with $\rightarrow \subseteq X \times X$
- upward (reflexive) compatibility

$$\forall x, x', y \qquad \exists y'$$

$$\begin{tikzcd}
x' \arrow[r, dotted, "="] \arrow[d, phantom, "\leqslant"] & y' \arrow[d, phantom, "\leqslant"] \\
x \arrow[r] & y
\end{tikzcd}$$

- $(X, \leqslant)$ is a wqo

# WSTS

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

- ordered transition system $(X, \leqslant, \rightarrow)$ with $\rightarrow \subseteq X \times X$
- upward (reflexive) compatibility

$$\begin{tikzcd}
\forall x, x', y & & \exists y' \\
x' \arrow[r, dotted, "=" ] & y' \\
\rotatebox{90}{$\leqslant$} & \rotatebox{90}{$\leqslant$} \\
x \arrow[r] & y
\end{tikzcd}$$

Example: lossy counter machines

$q(\mathbf{v}) \rightarrow_{\ell} q'(\mathbf{v}')$ if

$q(\mathbf{v}) \geqslant q(\mathbf{u}) \rightarrow q'(\mathbf{u}') \geqslant q'(\mathbf{v}')$

- $(X, \leqslant)$ is a wqo

# WSTS

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

## Coverability Problem

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

**question** $\exists t' \geqslant t \,.\, s \rightarrow^* t'$?

**Theorem**

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

# Downward WSTS

[Finkel & Schnoebelen '01]

- downward (reflexive) compatibility

$$\forall x, x', y \quad \begin{array}{ccc} x & \longrightarrow & y \\ \leqslant\!\!/ & & \\ x' & & \end{array}$$

- Subcoverability Problem

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

  question $\exists t' \leqslant t \,.\, s \rightarrow^* t'$?

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

# Downward WSTS

[Finkel & Schnoebelen '01]

- downward (reflexive) compatibility

$$\begin{tikzcd}
\forall x, x', y & & \exists y' \\
x \arrow[r] \arrow[d, phantom, "\leqslant"] & y \arrow[d, phantom, "\leqslant"] \\
x' \arrow[r, dotted, "="] & y'
\end{tikzcd}$$

- Subcoverability Problem

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

  question: $\exists t' \leqslant t \,.\, s \rightarrow^* t'$?

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

# Downward WSTS

[Finkel & Schnoebelen '01]

- downward (reflexive) compatibility

$$\begin{array}{lcc} \forall x, x', y & & \exists y' \\ & x \longrightarrow & y \\ & \rotatebox{90}{$\leqslant$} & \rotatebox{90}{$\leqslant$} \\ & x' \overset{=}{\dashrightarrow} & y' \end{array}$$

- Subcoverability Problem

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

  **question** $\exists t' \leqslant t \,.\, s \rightarrow^* t'$?

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

# Downward WSTS

[Finkel & Schnoebelen '01]

- downward (reflexive) compatibility

$$\begin{tikzcd} \forall x, x', y & & \exists y' \\ x \arrow[r] \arrow[d, phantom, "\geqslant"] & y \arrow[d, phantom, "\geqslant"] \\ x' \arrow[r, dotted, "="] & y' \end{tikzcd}$$

- Subcoverability Problem

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

  **question** $\exists t' \leqslant t \,.\, s \rightarrow^* t'$?

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

# Downward WSTS

[Finkel & Schnoebelen '01]

- downward (reflexive) compatibility

$$\begin{array}{lcc} \forall x, x', y & & \exists y' \\ x & \longrightarrow & y \\ \rotatebox{90}{$\leqslant$} & & \rotatebox{90}{$\leqslant$} \\ x' & \overset{=}{\dashrightarrow} & y' \end{array}$$

Example: incrementing counter machines

$q(\mathbf{v}) \to_i q'(\mathbf{v}')$ if

$q(\mathbf{v}) \leqslant q(\mathbf{u}) \to q'(\mathbf{u}') \leqslant q'(\mathbf{v}')$

- Subcoverability Problem

  input: ordered transition system $(X, \leqslant, \to)$, initial configuration $s \in X$, target configuration $t \in X$

  question: $\exists t' \leqslant t \,.\, s \to^* t'$?

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

# Bad Sequences

Over a qo $(X, \leqslant)$

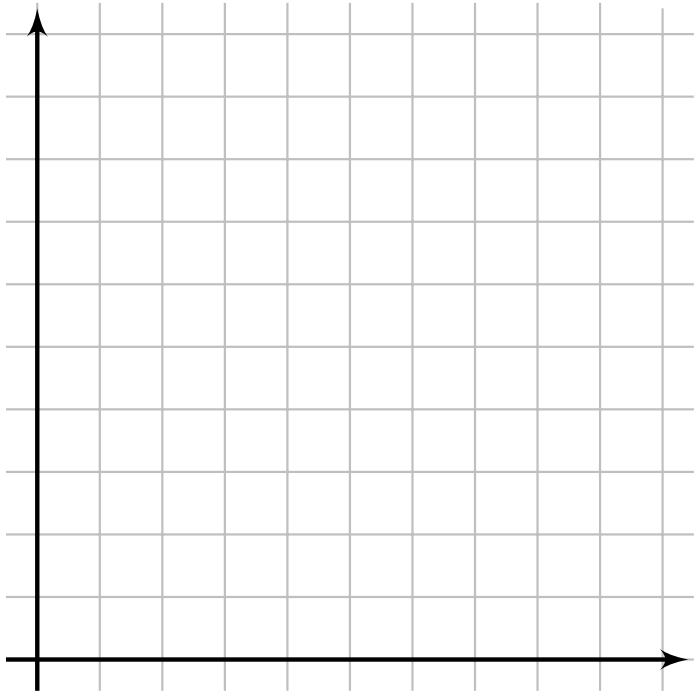
- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leq x_j$
- $(X, \leqslant)$ wqo if all bad sequences are finite

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



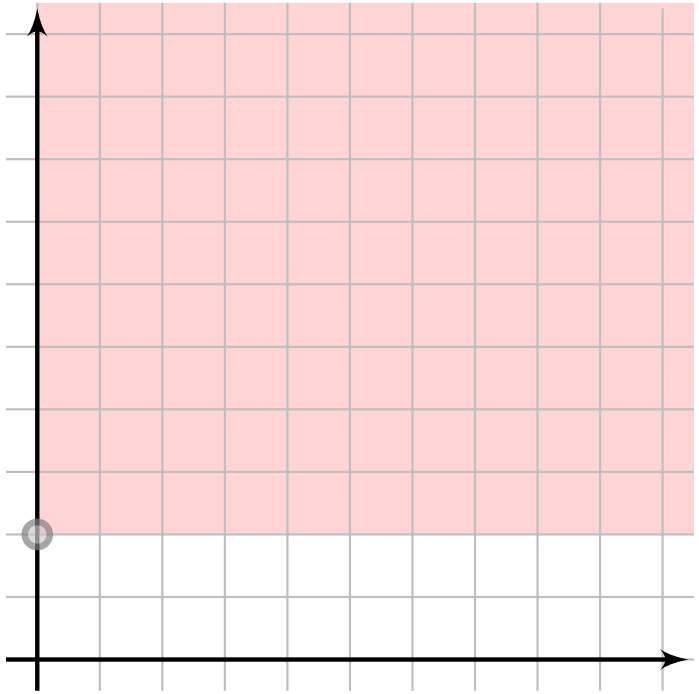
Example (in $\mathbb{N}^2$)

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



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

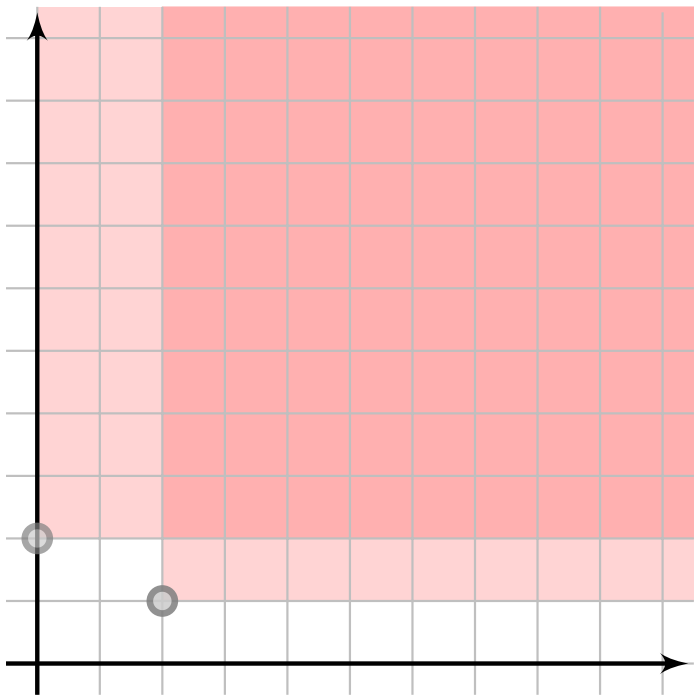
$(0, 2)$

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



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

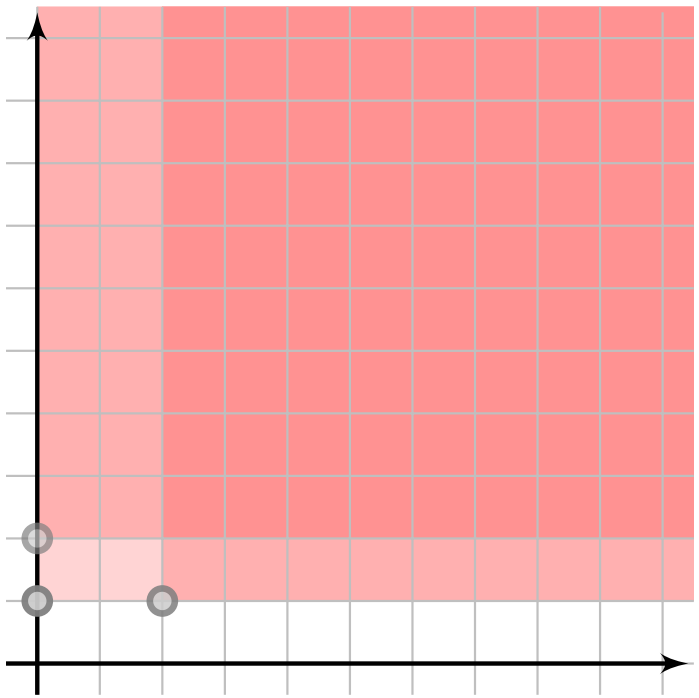
$(0,2), (2,1)$

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



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

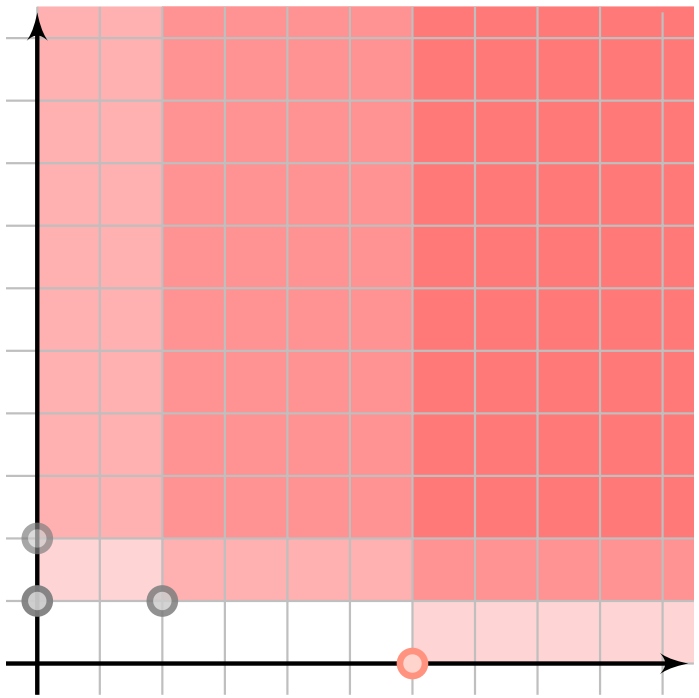
$(0,2), (2,1), (0,1)$

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



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

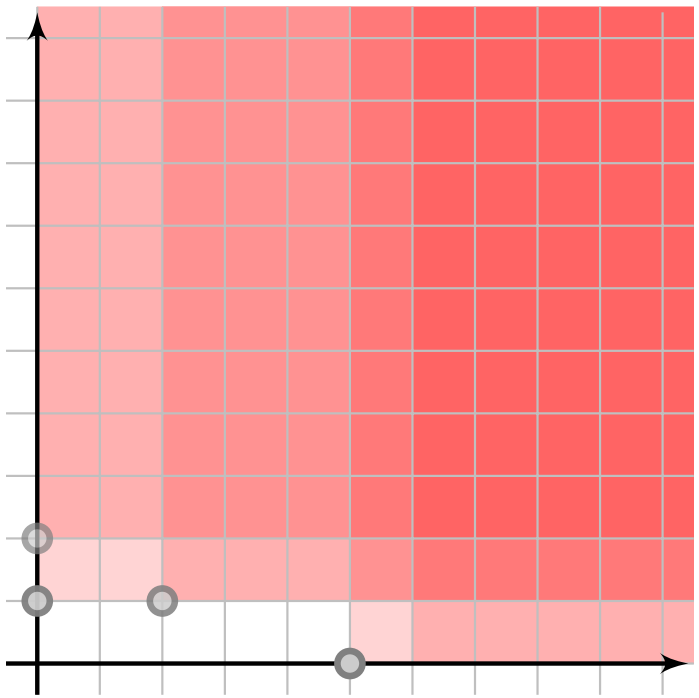
$(0,2), (2,1), (0,1), (6,0)$

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



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

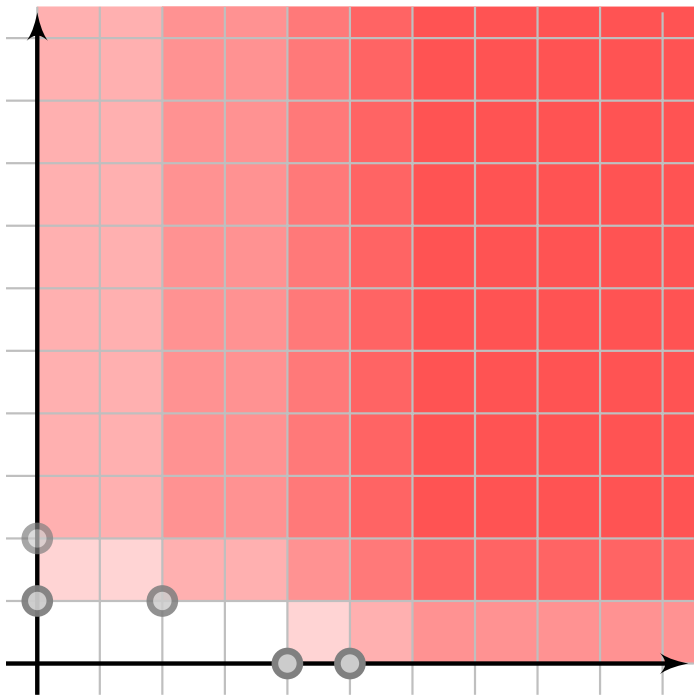
$(0,2), (2,1), (0,1), (6,0), (5,0)$

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



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

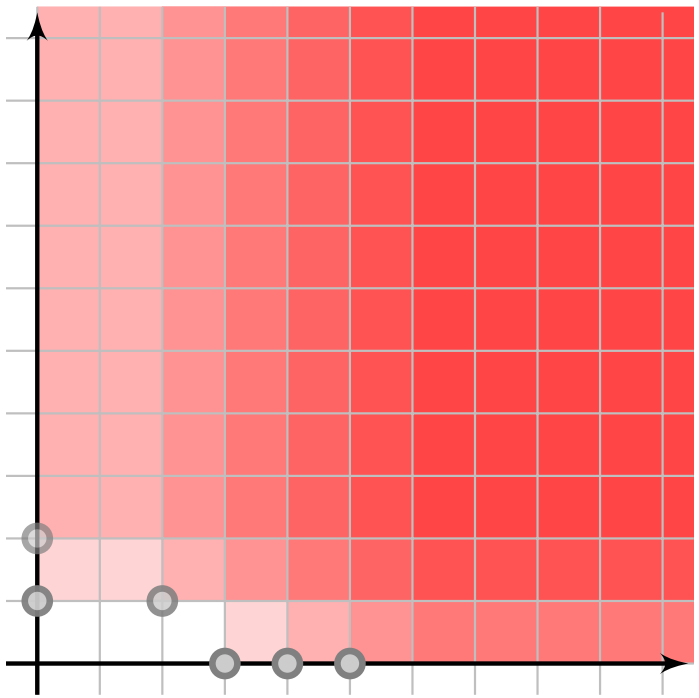
$(0,2), (2,1), (0,1), (6,0), (5,0), (4,0)$

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



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

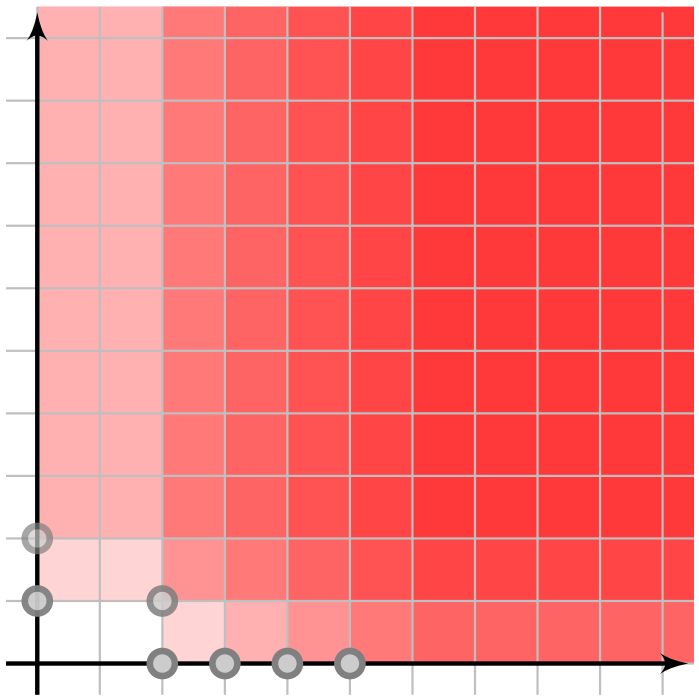
$(0,2), (2,1), (0,1), (6,0), (5,0), (4,0), (3,0)$

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



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

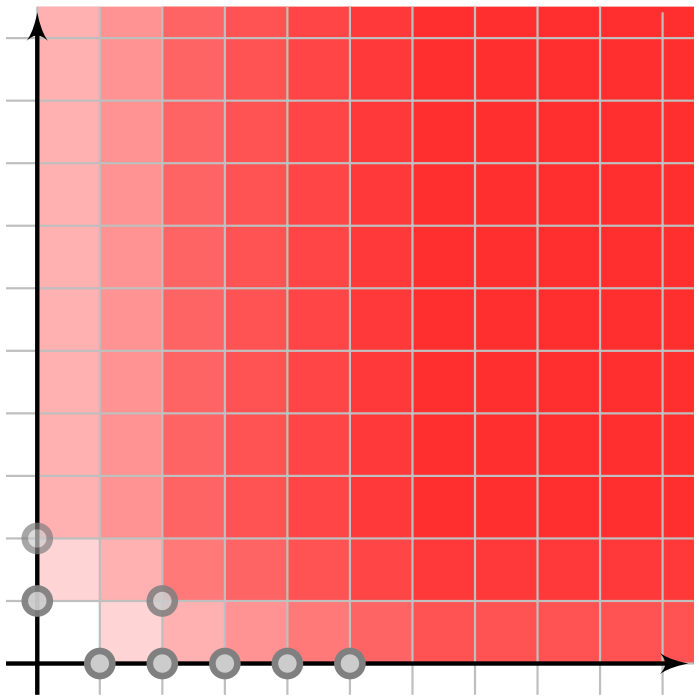
$(0,2), (2,1), (0,1), (6,0), (5,0), (4,0), (3,0), (2,0)$

# Bad Sequences

Over a qo $(X, \leqslant)$

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

... but can be of arbitrary length



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

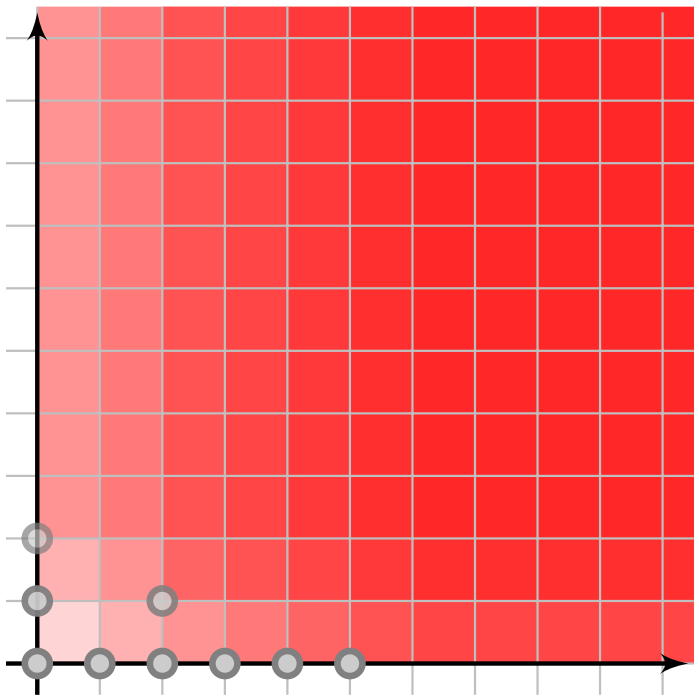
$(0,2), (2,1), (0,1), (6,0), (5,0), (4,0), (3,0), (2,0), (1,0)$

# Bad Sequences

Over a qo $(X, \leqslant)$

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leq x_j$
- $(X, \leqslant)$ 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 $(X, \leqslant)$ 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 $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if
$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]

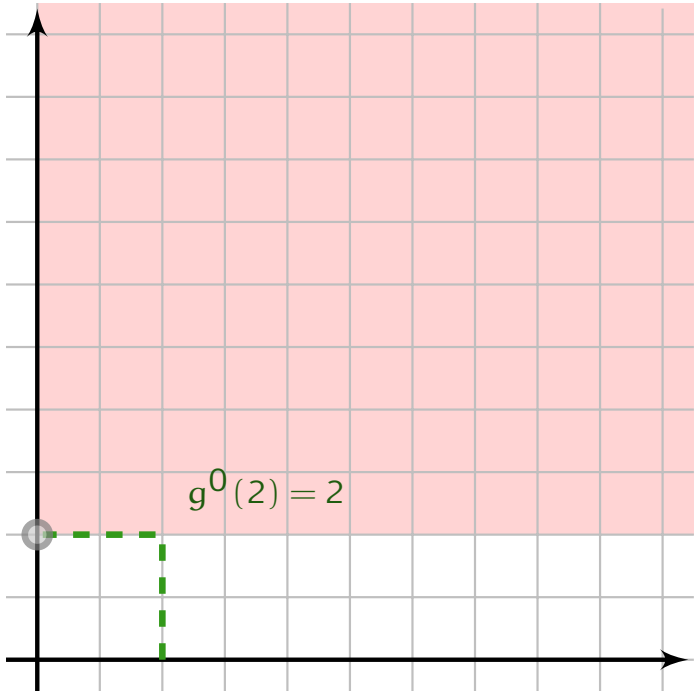
# Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $x_0, x_1, \ldots$ is controlled by $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if

$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]




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

$$(0, 2)$$

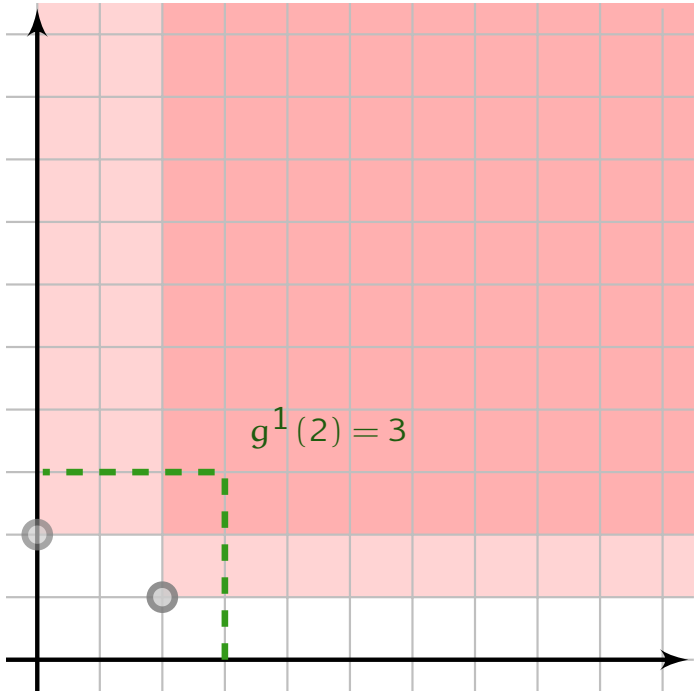
# Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $x_0, x_1, \ldots$ is controlled by $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if

$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]




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

$$(0,2), (2,1)$$

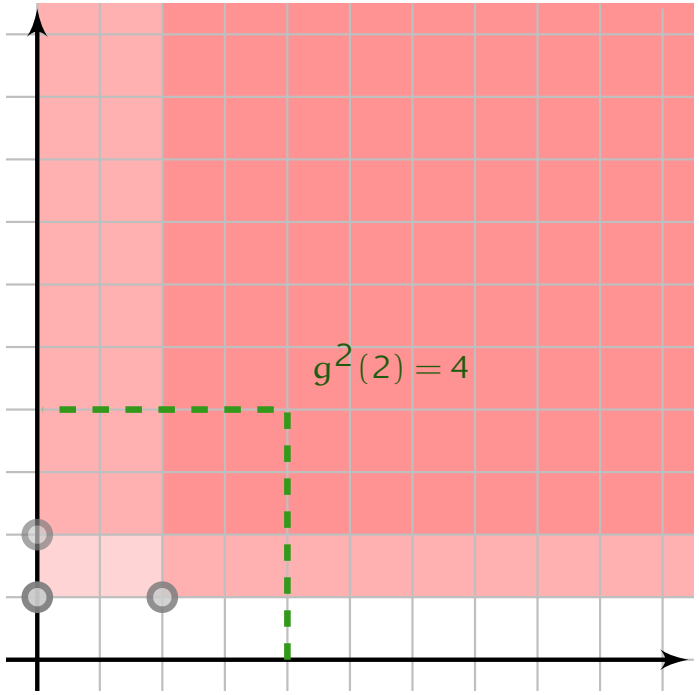
# Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $x_0, x_1, \ldots$ is controlled by $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if

$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]




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

$$(0,2), (2,1), (0,1)$$

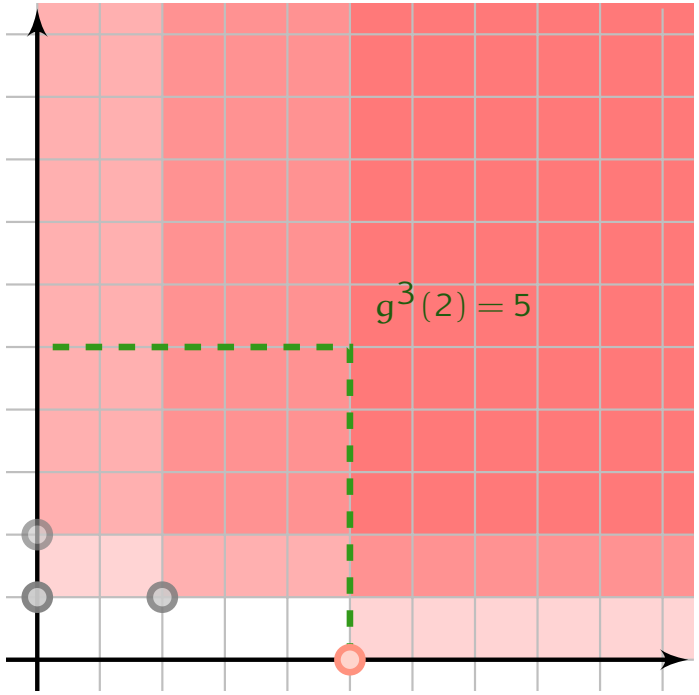
# Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $x_0, x_1, \ldots$ is controlled by $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if

$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]




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

$$(0,2), (2,1), (0,1), (5,0)$$

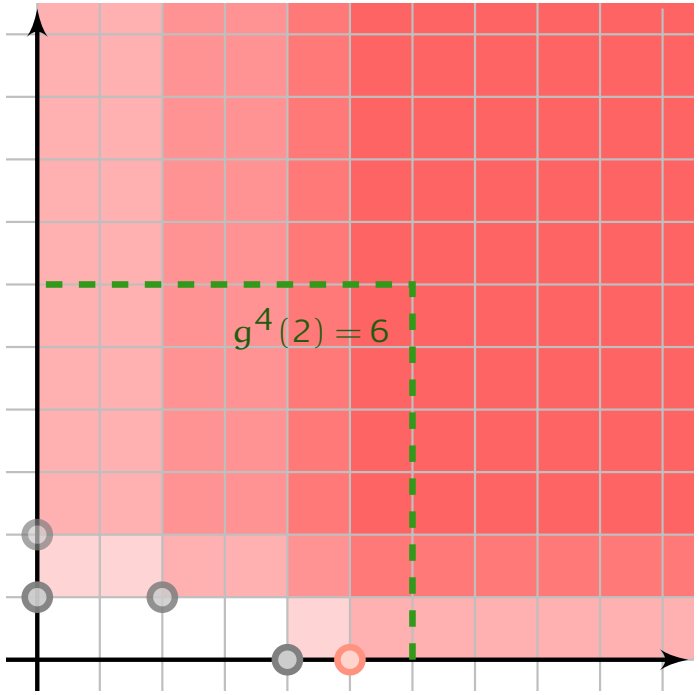
# Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $x_0, x_1, \ldots$ is controlled by $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if

$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]




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

$$(0,2), (2,1), (0,1), (5,0), (4,0)$$

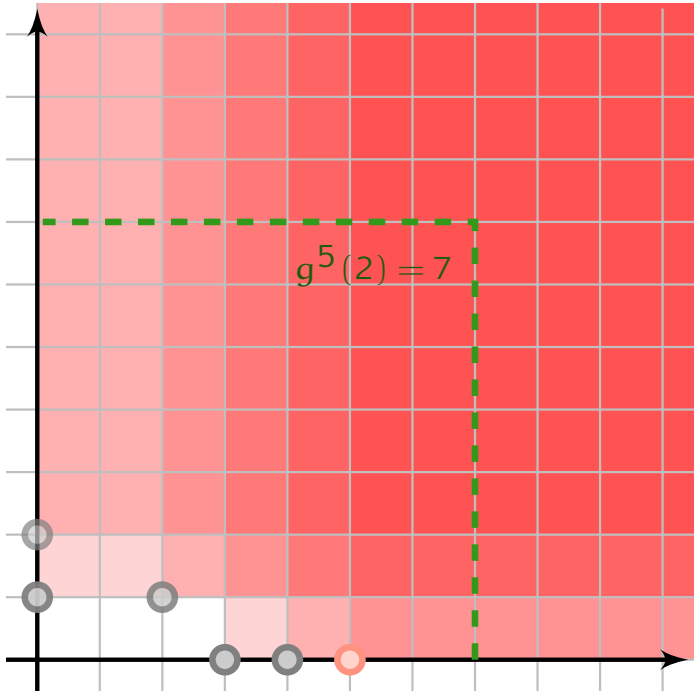
# Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $x_0, x_1, \ldots$ is controlled by $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if

$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]




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

$$(0,2), (2,1), (0,1), (5,0), (4,0), (3,0)$$

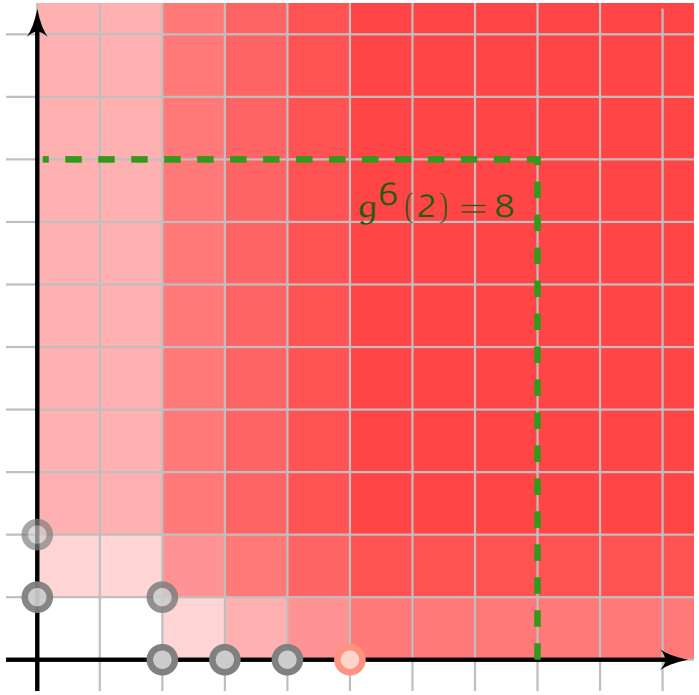
# Controlled Bad Sequences

Over a qo $(X, \leqslant)$ 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 $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if

$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]




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

$$(0,2), (2,1), (0,1), (5,0), (4,0), (3,0), (2,0)$$

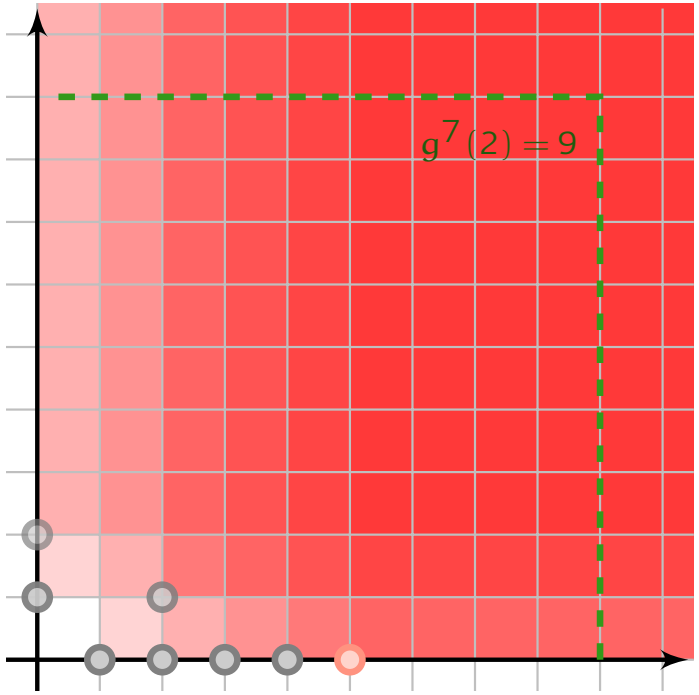
# Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $x_0, x_1, \ldots$ is controlled by $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if

$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]




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

$$(0,2), (2,1), (0,1), (5,0), (4,0), (3,0), (2,0), (1,0)$$

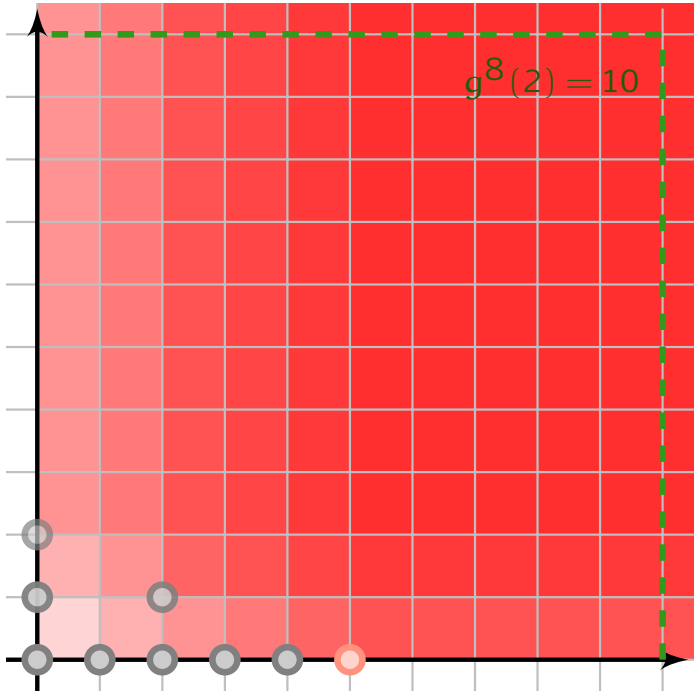
# Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $x_0, x_1, \ldots$ is controlled by $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if

$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[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 $(X, \leqslant)$ 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
$$\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$$

[Cichoń & Tahhan Bittar'98]

**Proposition**

*In a wqo $(X, \leqslant)$, if $\forall n \{x \in X \mid \|x\| \leqslant n\}$ 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 $(X, \leqslant)$ with norm $\|\cdot\|$

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $x_0, x_1, \ldots$ is controlled by $g\colon \mathbb{N} \to \mathbb{N}$ and $n_0 \in \mathbb{N}$ if $\|x_0\| \leqslant n_0$ and $\forall i. \|x_{i+1}\| \leqslant g(\|x_i\|)$

**Corollary**

*In a wqo $(X, \leqslant)$, if $\forall n \{x \in X \mid \|x\| \leqslant n\}$ 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]

$$
\begin{aligned}
F_0(x) &= x+1 \\
F_1(x) &= \overbrace{F_0 \circ \cdots \circ F_0}^{x+1 \text{ times}}(x) && = 2x+1 \\
F_2(x) &= \overbrace{F_1 \circ \cdots \circ F_1}^{x+1 \text{ times}}(x) && \approx 2^x \\
F_3(x) &= \overbrace{F_2 \circ \cdots \circ F_2}^{x+1 \text{ times}}(x) && \approx \mathrm{tower}(x) \\
&\vdots \\
F_\omega(x) &= F_{x+1}(x) && \approx \mathrm{ackermann}(x) \\
F_{\omega+1}(x) &= \overbrace{F_\omega \circ \cdots \circ F_\omega}^{x+1 \text{ times}}(x) \\
&\vdots \\
F_{\omega\cdot 2}(x) &= F_{\omega+x+1}(x) \\
&\vdots \\
F_{\omega\cdot 3}(x) &= F_{\omega\cdot 2+x+1}(x) \\
&\vdots \\
F_{\omega^2}(x) &= F_{\omega\cdot(x+1)}(x) \\
&\vdots \\
F_{\omega^\omega}(x) &= F_{\omega^{x+1}}(x)
\end{aligned}
$$

Yields corresponding complexity classes [S. '16]




Examples Coverability is

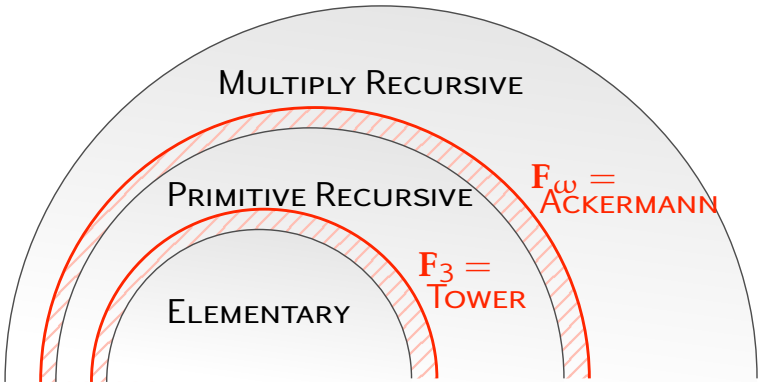
- $F_\omega$-complete for lossy counter systems
- $F_{\omega\cdot 2}$-complete for ν-Petri nets
- $F_{\omega^\omega}$-complete for lossy channel systems

# Fast-growing Complexity

[Löb & Wainer '70]

$$
\begin{aligned}
F_0(x) &= x+1 \\
F_1(x) &= \overbrace{F_0 \circ \cdots \circ F_0}^{x+1 \text{ times}}(x) && = 2x+1 \\
F_2(x) &= \overbrace{F_1 \circ \cdots \circ F_1}^{x+1 \text{ times}}(x) && \approx 2^x \\
F_3(x) &= \overbrace{F_2 \circ \cdots \circ F_2}^{x+1 \text{ times}}(x) && \approx \mathrm{tower}(x) \\
&\vdots \\
F_\omega(x) &= F_{x+1}(x) && \approx \mathrm{ackermann}(x) \\
F_{\omega+1}(x) &= \overbrace{F_\omega \circ \cdots \circ F_\omega}^{x+1 \text{ times}}(x) \\
&\vdots \\
F_{\omega\cdot 2}(x) &= F_{\omega+x+1}(x) \\
&\vdots \\
F_{\omega\cdot 3}(x) &= F_{\omega\cdot 2+x+1}(x) \\
&\vdots \\
F_{\omega^2}(x) &= F_{\omega\cdot(x+1)}(x) \\
&\vdots \\
F_{\omega^\omega}(x) &= F_{\omega^{x+1}}(x)
\end{aligned}
$$

Yields corresponding complexity classes [S. '16]




Examples Coverability is

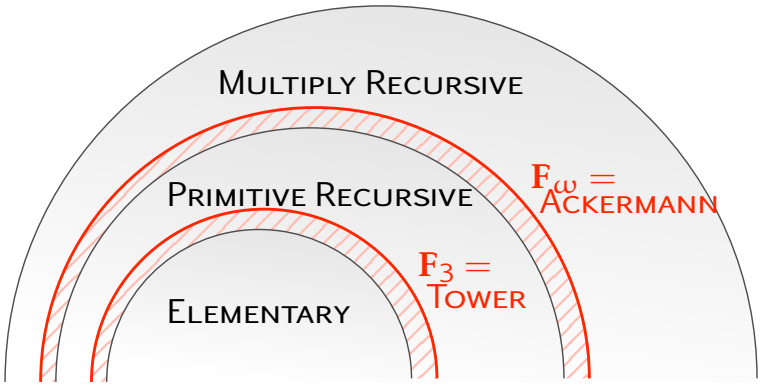
- $F_\omega$-complete for lossy counter systems
- $F_{\omega\cdot 2}$-complete for $\nu$-Petri nets
- $F_{\omega^\omega}$-complete for lossy channel systems

# Fast-growing Complexity

[Löb & Wainer '70]

$$
\begin{aligned}
F_0(x) &= x+1 \\
F_1(x) &= \overbrace{F_0 \circ \cdots \circ F_0}^{x+1 \text{ times}}(x) && = 2x+1 \\
F_2(x) &= \overbrace{F_1 \circ \cdots \circ F_1}^{x+1 \text{ times}}(x) && \approx 2^x \\
F_3(x) &= \overbrace{F_2 \circ \cdots \circ F_2}^{x+1 \text{ times}}(x) && \approx \mathrm{tower}(x) \\
&\vdots \\
F_\omega(x) &= F_{x+1}(x) && \approx \mathrm{ackermann}(x) \\
F_{\omega+1}(x) &= \overbrace{F_\omega \circ \cdots \circ F_\omega}^{x+1 \text{ times}}(x) \\
&\vdots \\
F_{\omega\cdot 2}(x) &= F_{\omega+x+1}(x) \\
&\vdots \\
F_{\omega\cdot 3}(x) &= F_{\omega\cdot 2+x+1}(x) \\
&\vdots \\
F_{\omega^2}(x) &= F_{\omega\cdot(x+1)}(x) \\
&\vdots \\
F_{\omega^\omega}(x) &= F_{\omega^{x+1}}(x)
\end{aligned}
$$

Yields corresponding complexity classes [S. '16]




Examples Coverability is

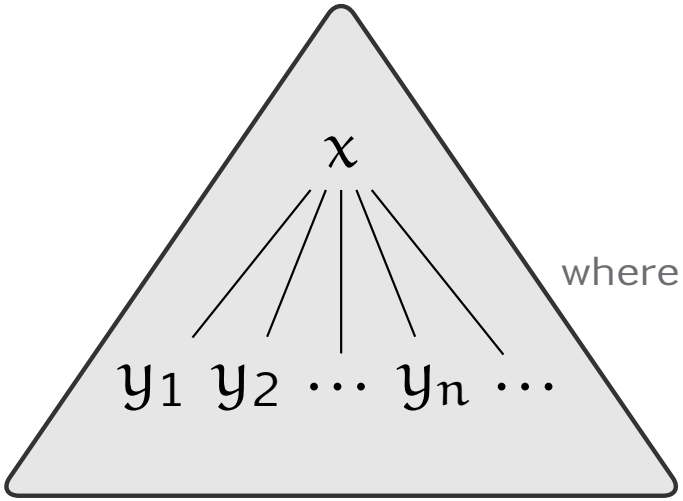
- $\mathbf{F}_\omega$-complete for lossy counter systems
- $\mathbf{F}_{\omega\cdot 2}$-complete for $\nu$-Petri nets
- $\mathbf{F}_{\omega^\omega}$-complete for lossy channel systems

# Ordered Branching Systems

$(X, \leqslant, \rightarrow)$ where $\rightarrow \subseteq X \times \mathbb{P}(X)$

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

Tree executions




where $x \rightarrow \{y_1, y_2, \ldots, y_n, \ldots\}$

# Ordered Branching Systems

$(X, \leqslant, \rightarrow)$ 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 $(X, \leqslant, \rightarrow)$, a root configuration $r \in X$ and a leaf configuration $\ell \in X$

question existence of a tree execution:

leaf coverability




(top-down coverability)

root coverability




(bottom-up coverability)

leaf subcoverability




(top-down subcoverability)

root subcoverability




(bottom-up subcoverability)

# Ordered Branching Systems

$(X, \leqslant, \rightarrow)$ where $\rightarrow \subseteq X \times \mathbb{P}(X)$

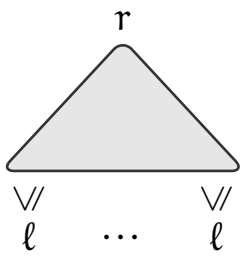
(we often assume $\rightarrow \subseteq X \times \mathbb{P}_{\text{fin}}(X)$)

## Coverability Problems

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

question existence of a tree execution:

leaf coverability




(top-down coverability)

root coverability




(bottom-up coverability)

leaf subcoverability




(top-down subcoverability)

root subcoverability




(bottom-up subcoverability)

# Ordered Branching Systems

$(X, \leqslant, \rightarrow)$ where $\rightarrow \subseteq X \times \mathbb{P}(X)$

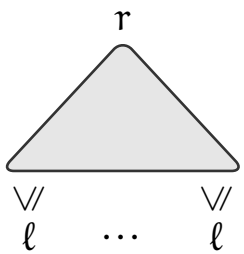
(we often assume $\rightarrow \subseteq X \times \mathbb{P}_{\text{fin}}(X)$)

## Coverability Problems

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

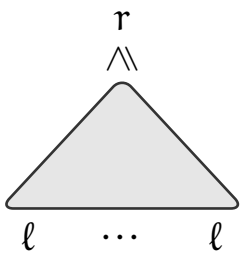
question existence of a tree execution:

leaf coverability




(top-down coverability)

root coverability




(bottom-up coverability)

leaf subcoverability




(top-down subcoverability)

root subcoverability




(bottom-up subcoverability)

# Ordered Branching Systems

$(X, \leqslant, \rightarrow)$ where $\rightarrow \subseteq X \times \mathbb{P}(X)$

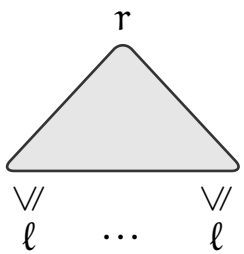
(we often assume $\rightarrow \subseteq X \times \mathbb{P}_{\text{fin}}(X)$)

## Coverability Problems

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

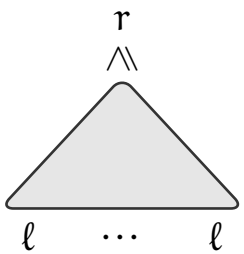
question existence of a tree execution:

leaf coverability



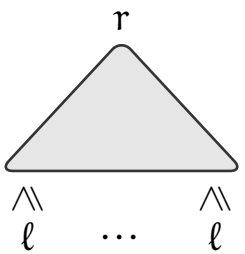

(top-down coverability)

root coverability




(bottom-up coverability)

leaf subcoverability




(top-down subcoverability)

root subcoverability




(bottom-up subcoverability)

# Ordered Branching Systems

$(X, \leqslant, \rightarrow)$ where $\rightarrow \subseteq X \times \mathbb{P}(X)$

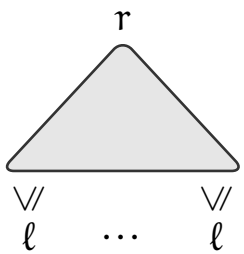
(we often assume $\rightarrow \subseteq X \times \mathbb{P}_{\text{fin}}(X)$)

## Coverability Problems

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

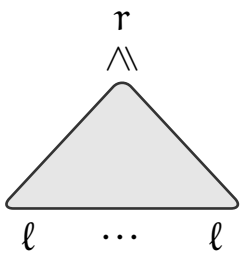
question existence of a tree execution:

leaf coverability



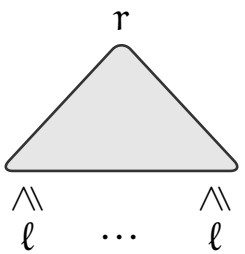

(top-down coverability)

root coverability



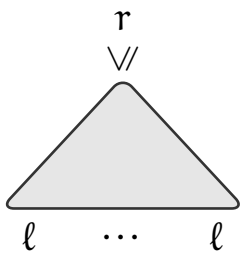

(bottom-up coverability)

leaf subcoverability




(top-down subcoverability)

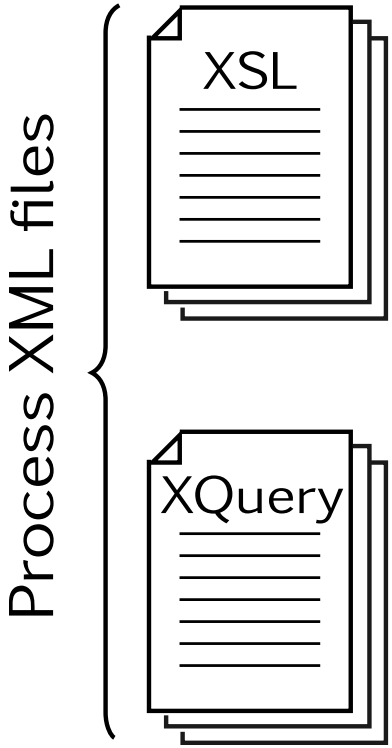
root subcoverability

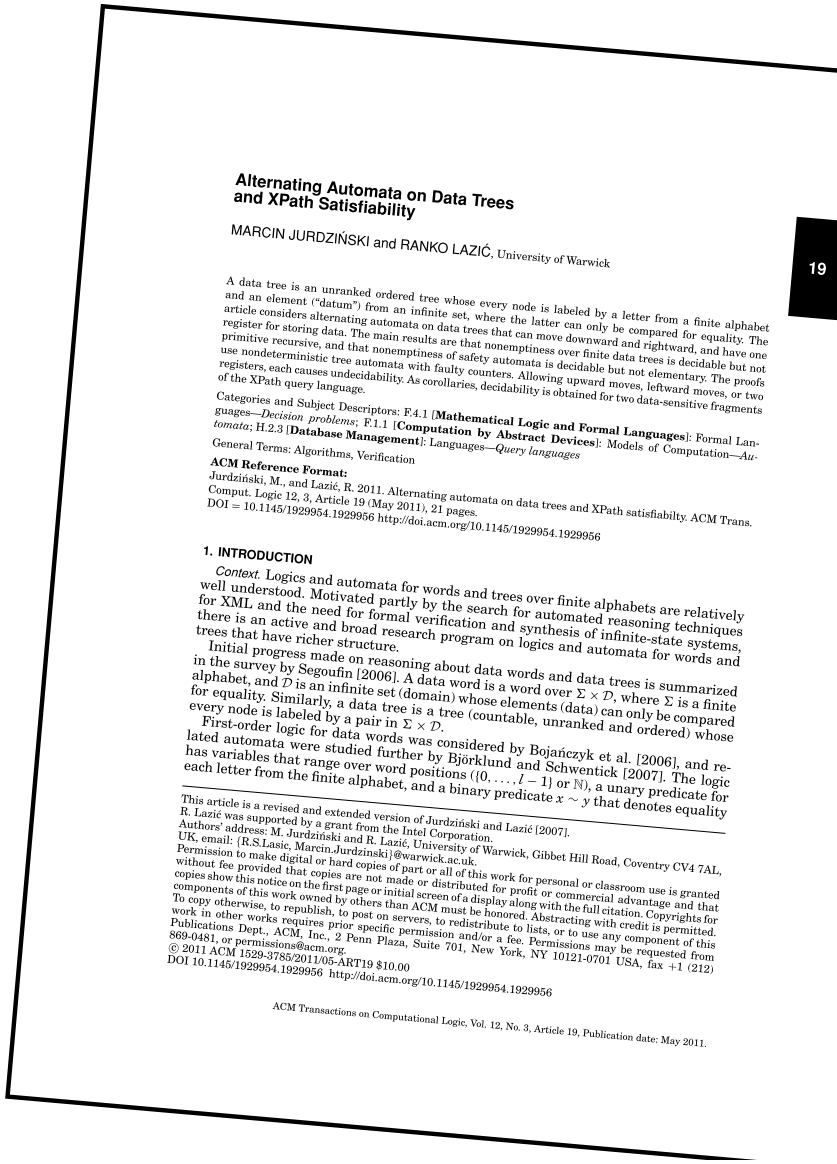



(bottom-up subcoverability)

# Example: Forward XPath$^{\varepsilon}$

[Jurdziński & Lazić ′11]



**Alternating Automata on Data Trees and XPath Satisfiability**

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

19

A data tree is an unranked ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("datum") from an infinite set, where the latter can only be compared for equality. The article considers alternating automata on data trees that can move downward and rightward, and have one register for storing data. The main results are that nonemptiness over finite data trees is decidable but not primitive recursive, and that nonemptiness of safety automata is decidable but not elementary. The proofs use nondeterministic tree automata with faulty counters. Allowing upward moves, leftward moves, or two registers, each causes undecidability. As corollaries, decidability is obtained for two data-sensitive fragments of the XPath query language.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Verification

**ACM Reference Format:**
Jurdziński, M., and Lazić, R. 2011. Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Logic 12, 3, Article 19 (May 2011), 21 pages.
DOI = 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

**1. INTRODUCTION**

*Context.* Logics 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 for formal verification and synthesis of infinite-state systems, there is an active and broad research program on logics and automata 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 Segoufin [2006]. A data word is a word over $\Sigma \times \mathcal{D}$, where $\Sigma$ is a finite alphabet, and $\mathcal{D}$ is an infinite set (domain) whose elements (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 pair in $\Sigma \times \mathcal{D}$.

First-order logic for data words was considered by Bojańczyk et al. [2006], and related automata were studied further by Björklund and Schwentick [2007]. The logic has variables that range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), a unary predicate for each letter from the finite alphabet, and a binary predicate $x \sim y$ that denotes equality

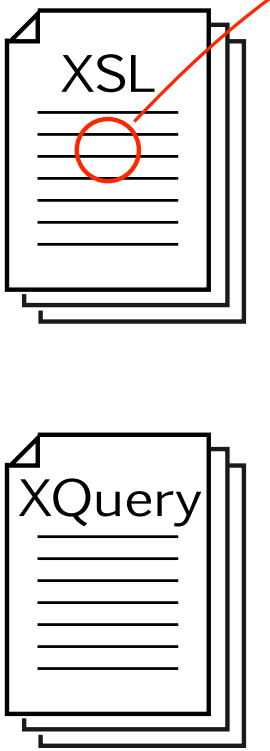
This article is a revised and extended version of Jurdziński and Lazić [2007].
R. Lazić was supported by a grant from the Intel Corporation.
Authors' address: M. Jurdziński and R. Lazić, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK, email: {R.S.Lasic, Marcin.Jurdzinski}@warwick.ac.uk.

DOI 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

# Example: Forward XPath$^{\varepsilon}$

[Jurdziński & Lazić ′11]

Process XML files

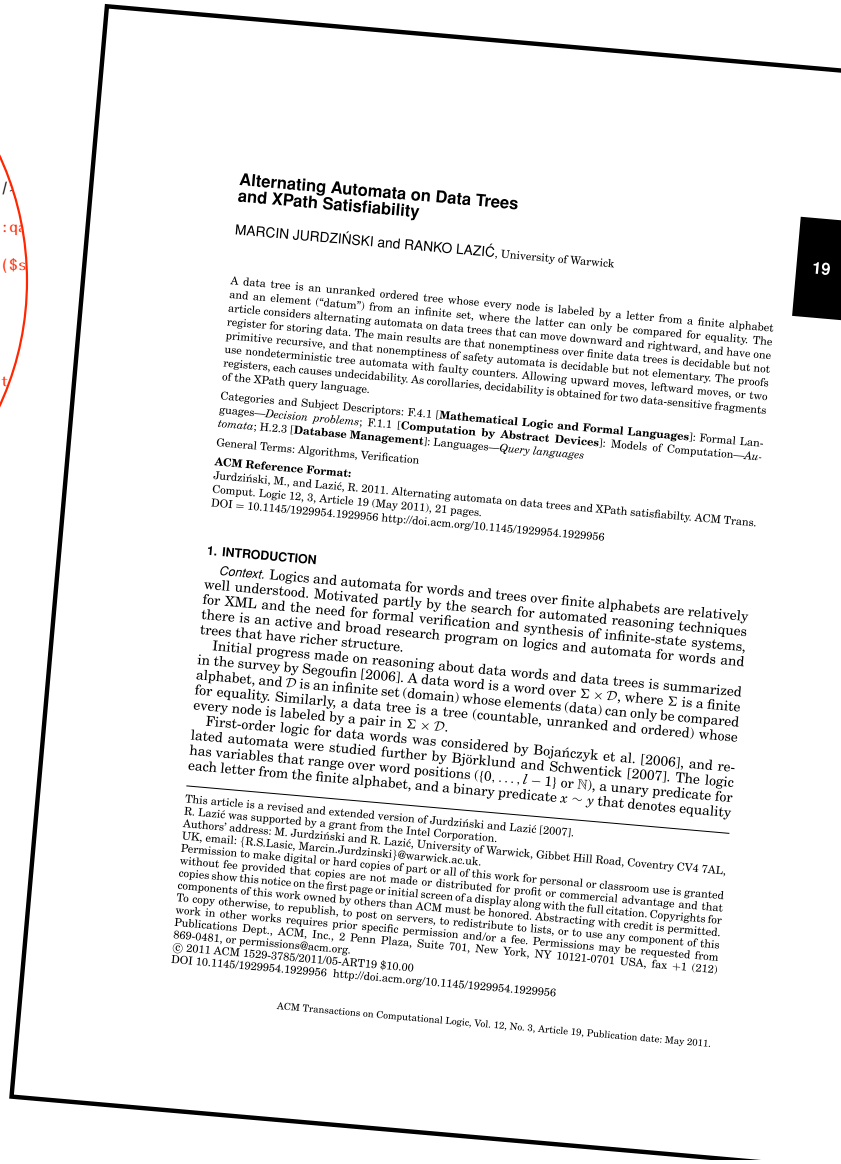



```
      </xsl:call-template>
      </xsl:variable>
      <xsl:value-of select="$slevel +
    </xsl:when>
    <xsl:otherwise>1</xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template name="qandadiv.section.level">
  <xsl:variable name="section.level">
    <xsl:call-template name="qanda.section.level"/
  </xsl:variable>
  <xsl:variable name="anc.divs" select="ancestor::q

  <xsl:value-of select="count($anc.divs) + number($s
</xsl:template>

<xsl:template name="question.answer.label">
  <xsl:variable name="deflabel">
    <xsl:choose>
      <xsl:when test="ancestor-or-self::*[@default
        <xsl:value-of select="(ancestor-or-self::
                              /@defaultlabel"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="$qanda.default
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:variable name="label" se
```



**Alternating Automata on Data Trees and XPath Satisfiability**

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

19

A data tree is an unranked ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("datum") from an infinite set, where the latter can only be compared for equality. The article considers alternating automata on data trees that can move downward and rightward, and have one register for storing data. The main results are that nonemptiness over finite data trees is decidable but not primitive recursive, and that nonemptiness of safety automata is decidable but not elementary. The proofs use nondeterministic tree automata with faulty counters. Allowing upward moves, leftward moves, or two registers, each causes undecidability. As corollaries, decidability is obtained for two data-sensitive fragments of the XPath query language.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Verification

**ACM Reference Format:**
Jurdziński, M., and Lazić, R. 2011. Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Logic 12, 3, Article 19 (May 2011), 21 pages.
DOI = 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

**1. INTRODUCTION**

*Context.* Logics 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 for formal verification and synthesis of infinite-state systems, there is an active and broad research program on logics and automata 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 Segoufin [2006]. A data word is a word over $\Sigma \times \mathcal{D}$, where $\Sigma$ is a finite alphabet, and $\mathcal{D}$ is an infinite set (domain) whose elements (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 pair in $\Sigma \times \mathcal{D}$.

First-order logic for data words was considered by Bojańczyk et al. [2006], and related automata were studied further by Björklund and Schwentick [2007]. The logic has variables that range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), a unary predicate for each letter from the finite alphabet, and a binary predicate $x \sim y$ that denotes equality

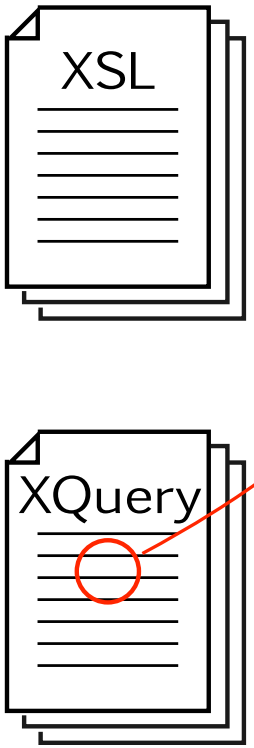
This article is a revised and extended version of Jurdziński and Lazić [2007].
R. Lazić was supported by a grant from the Intel Corporation.
Authors' address: M. Jurdziński and R. Lazić, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK, email: {R.S.Lasic, Marcin.Jurdzinski}@warwick.ac.uk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1529-3785/2011/05-ART19 $10.00
DOI 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

ACM Transactions on Computational Logic, Vol. 12, No. 3, Article 19, Publication date: May 2011.

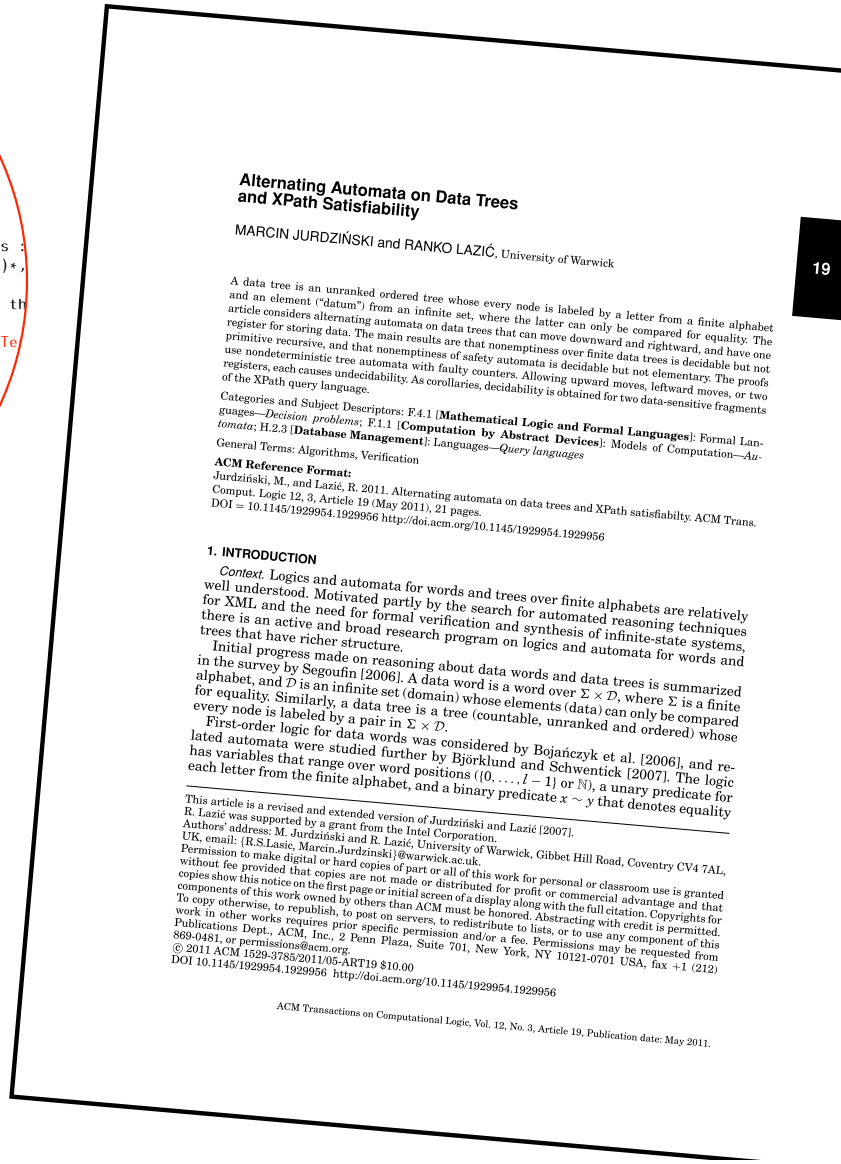
# Example: Forward XPath$^{\varepsilon}$

[Jurdziński & Lazić ′11]

Process XML files




```
(: ## Add metadata :)
declare function od-api:metadata($respon
  let $metadata := $response/json/metadata
  let $date := $response[1]/*[fn:name()="htt
  let $version := "v0.6.0"
  return element {"metadata"} {
    $metadata/node(),
    element {"od_api_xquery"} {$version},
    element {"date"} {$date}
  }
};
(: ## Create elements for optional fragment arrays :
declare function od-api:option($fragment as item()*,
  (: ### General arrays :)
  if ($fragment and $function = "arrayofstrings") th
    od-api:arrayofstrings($fragment)
  else if ($fragment and $function = "CategorizedTe
    element {"notes"} {
      for $note in $fragment/_
      return od-api:CategorizedTextList($note, "n
    }
  (: ### Lemmatron arrays :)
  else if ($fragment and $function = "headwor
    element {"results"} {
      for $result in $fragment/_
      return od-api:headwordLemmatron($
    }
  else if ($fragment/fn:name()
```



**Alternating Automata on Data Trees and XPath Satisfiability**

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

19

A data tree is an unranked ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("datum") from an infinite set, where the latter can only be compared for equality. The article considers alternating automata on data trees that can move downward and rightward, and have one register for storing data. The main results are that nonemptiness over finite data trees is decidable but not primitive recursive, and that nonemptiness of safety automata is decidable but not elementary. The proofs use nondeterministic tree automata with faulty counters. Allowing upward moves, leftward moves, or two registers, each causes undecidability. As corollaries, decidability is obtained for two data-sensitive fragments of the XPath query language.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Verification

**ACM Reference Format:**
Jurdziński, M., and Lazić, R. 2011. Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Logic 12, 3, Article 19 (May 2011), 21 pages.
DOI = 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

**1. INTRODUCTION**

*Context.* Logics 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 for formal verification and synthesis of infinite-state systems, there is an active and broad research program on logics and automata 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 Segoufin [2006]. A data word is a word over $\Sigma \times \mathcal{D}$, where $\Sigma$ is a finite alphabet, and $\mathcal{D}$ is an infinite set (domain) whose elements (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 pair in $\Sigma \times \mathcal{D}$.

First-order logic for data words was considered by Bojańczyk et al. [2006], and related automata were studied further by Björklund and Schwentick [2007]. The logic has variables that range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), a unary predicate for each letter from the finite alphabet, and a binary predicate $x \sim y$ that denotes equality

This article is a revised and extended version of Jurdziński and Lazić [2007].
R. Lazić was supported by a grant from the Intel Corporation.
Authors' address: M. Jurdziński and R. Lazić, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK, email: {R.S.Lasic, Marcin.Jurdzinski}@warwick.ac.uk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1529-3785/2011/05-ART19 $10.00
DOI 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

ACM Transactions on Computational Logic, Vol. 12, No. 3, Article 19, Publication date: May 2011.

# Example: Forward XPath$^{\varepsilon}$

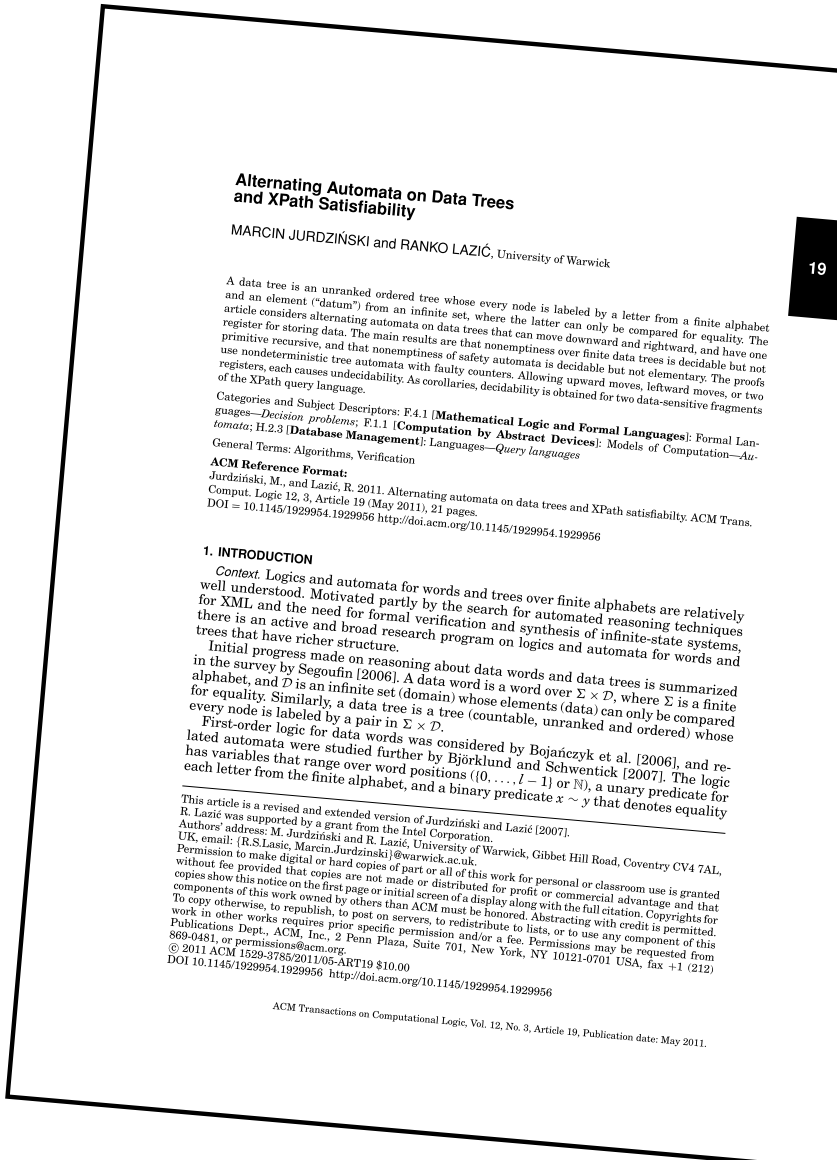
[Jurdziński & Lazić ′11]

XML document

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

XPath expression

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



**Alternating Automata on Data Trees and XPath Satisfiability**

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

19

A data tree is an unranked ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("datum") from an infinite set, where the latter can only be compared for equality. The article considers alternating automata on data trees that can move downward and rightward, and have one register for storing data. The main results are that nonemptiness over finite data trees is decidable but not primitive recursive, and that nonemptiness of safety automata is decidable but not elementary. The proofs use nondeterministic tree automata with faulty counters. Allowing upward moves, leftward moves, or two registers, each causes undecidability. As corollaries, decidability is obtained for two data-sensitive fragments of the XPath query language.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Verification

**ACM Reference Format:**
Jurdziński, M., and Lazić, R. 2011. Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Logic 12, 3, Article 19 (May 2011), 21 pages.
DOI = 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

**1. INTRODUCTION**

*Context.* Logics 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 for formal verification and synthesis of infinite-state systems, there is an active and broad research program on logics and automata 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 Segoufin [2006]. A data word is a word over $\Sigma \times \mathcal{D}$, where $\Sigma$ is a finite alphabet, and $\mathcal{D}$ is an infinite set (domain) whose elements (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 pair in $\Sigma \times \mathcal{D}$.

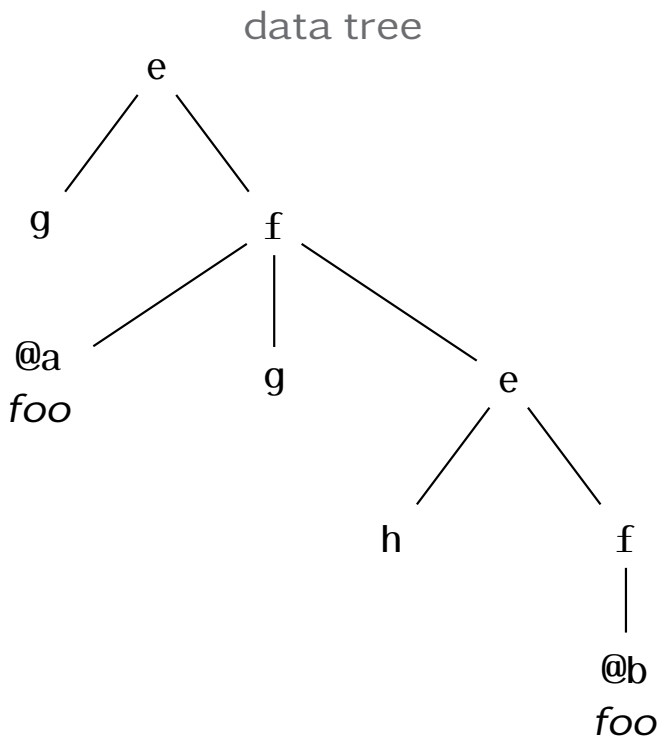
First-order logic for data words was considered by Bojańczyk et al. [2006], and related automata were studied further by Björklund and Schwentick [2007]. The logic has variables that range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), a unary predicate for each letter from the finite alphabet, and a binary predicate $x \sim y$ that denotes equality

This article is a revised and extended version of Jurdziński and Lazić [2007].
R. Lazić was supported by a grant from the Intel Corporation.
Authors' address: M. Jurdziński and R. Lazić, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK, email: {R.S.Lasic, Marcin.Jurdzinski}@warwick.ac.uk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1529-3785/2011/05-ART19 $10.00
DOI 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

ACM Transactions on Computational Logic, Vol. 12, No. 3, Article 19, Publication date: May 2011.

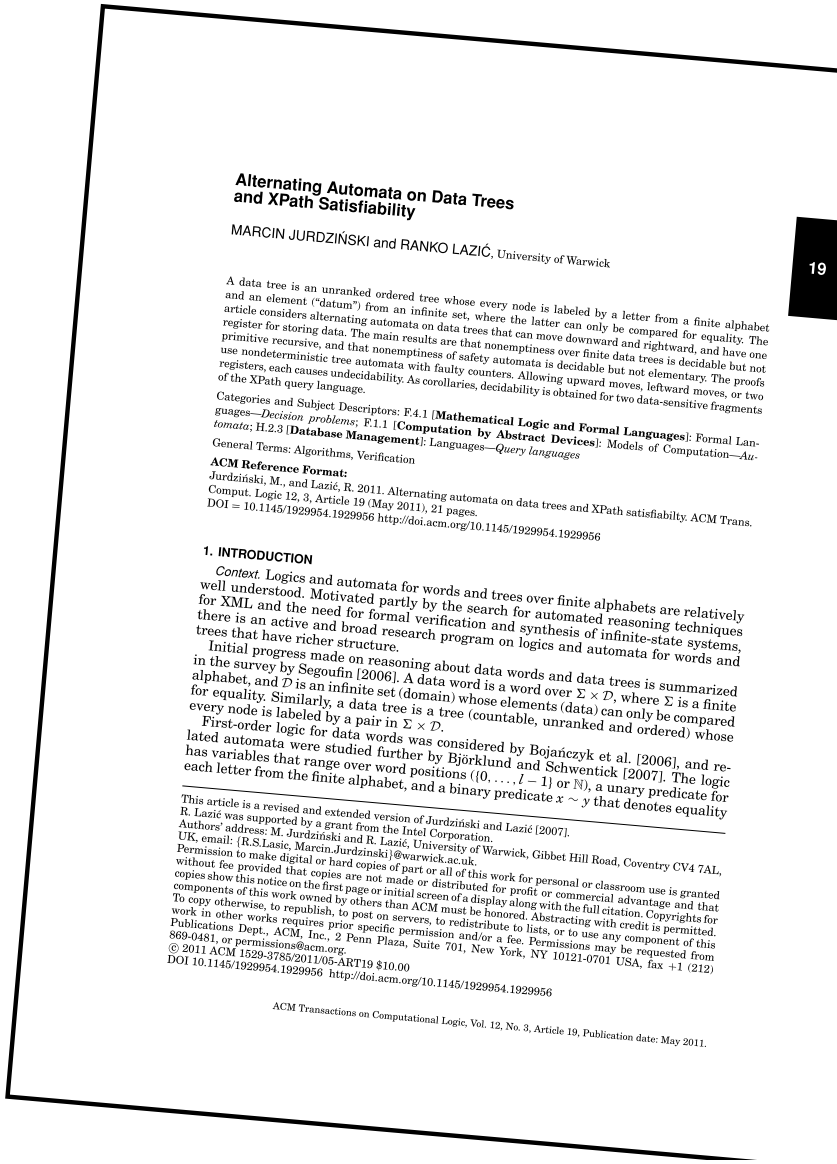
# Example: Forward XPath$^{\varepsilon}$

[Jurdziński & Lazić ′11]




XPath expression

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



**Alternating Automata on Data Trees and XPath Satisfiability**

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

19

A data tree is an unranked ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("datum") from an infinite set, where the latter can only be compared for equality. The article considers alternating automata on data trees that can move downward and rightward, and have one register for storing data. The main results are that nonemptiness over finite data trees is decidable but not primitive recursive, and that nonemptiness of safety automata is decidable but not elementary. The proofs use nondeterministic tree automata with faulty counters. Allowing upward moves, leftward moves, or two registers, each causes undecidability. As corollaries, decidability is obtained for two data-sensitive fragments of the XPath query language.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Verification

**ACM Reference Format:**
Jurdziński, M., and Lazić, R. 2011. Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Logic 12, 3, Article 19 (May 2011), 21 pages.
DOI = 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

**1. INTRODUCTION**

*Context.* Logics 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 for formal verification and synthesis of infinite-state systems, there is an active and broad research program on logics and automata 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 Segoufin [2006]. A data word is a word over $\Sigma \times \mathcal{D}$, where $\Sigma$ is a finite alphabet, and $\mathcal{D}$ is an infinite set (domain) whose elements (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 pair in $\Sigma \times \mathcal{D}$.

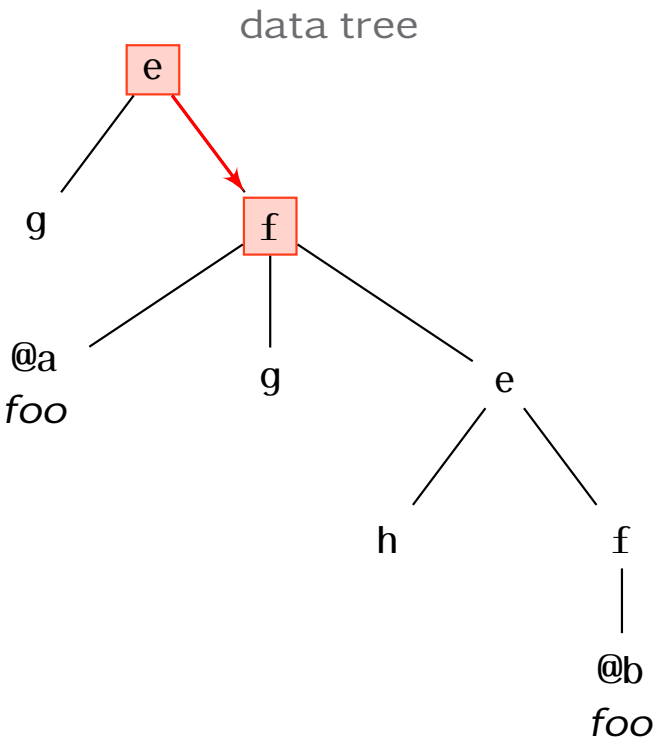
First-order logic for data words was considered by Bojańczyk et al. [2006], and related automata were studied further by Björklund and Schwentick [2007]. The logic has variables that range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), a unary predicate for each letter from the finite alphabet, and a binary predicate $x \sim y$ that denotes equality

This article is a revised and extended version of Jurdziński and Lazić [2007].
R. Lazić was supported by a grant from the Intel Corporation.
Authors' address: M. Jurdziński and R. Lazić, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK, email: {R.S.Lasic, Marcin.Jurdzinski}@warwick.ac.uk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1529-3785/2011/05-ART19 $10.00
DOI 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

ACM Transactions on Computational Logic, Vol. 12, No. 3, Article 19, Publication date: May 2011.

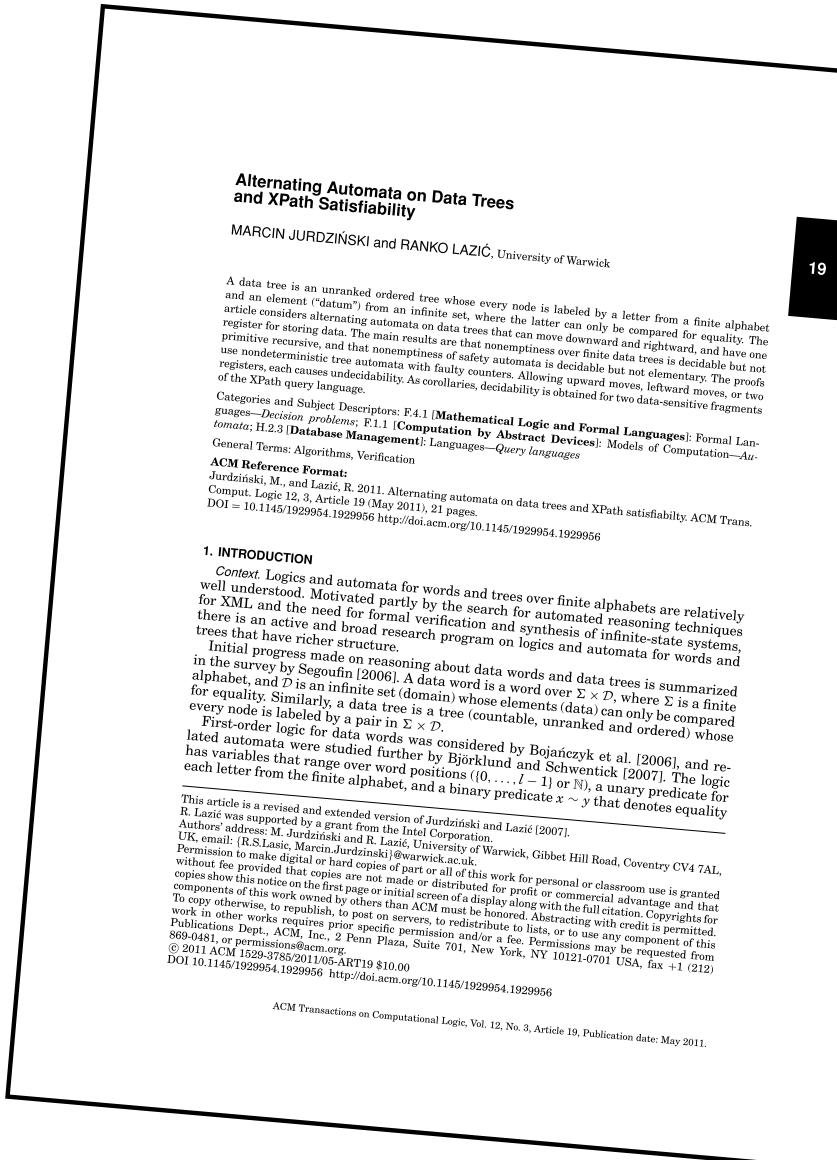
# Example: Forward XPath$^{\varepsilon}$

[Jurdziński & Lazić ′11]




XPath expression

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



**Alternating Automata on Data Trees and XPath Satisfiability**

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

19

A data tree is an unranked ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("datum") from an infinite set, where the latter can only be compared for equality. The article considers alternating automata on data trees that can move downward and rightward, and have one register for storing data. The main results are that nonemptiness over finite data trees is decidable but not primitive recursive, and that nonemptiness of safety automata is decidable but not elementary. The proofs use nondeterministic tree automata with faulty counters. Allowing upward moves, leftward moves, or two registers, each causes undecidability. As corollaries, decidability is obtained for two data-sensitive fragments of the XPath query language.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Verification

**ACM Reference Format:**
Jurdziński, M., and Lazić, R. 2011. Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Logic 12, 3, Article 19 (May 2011), 21 pages.
DOI = 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

**1. INTRODUCTION**

*Context.* Logics 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 for formal verification and synthesis of infinite-state systems, there is an active and broad research program on logics and automata 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 Segoufin [2006]. A data word is a word over $\Sigma \times \mathcal{D}$, where $\Sigma$ is a finite alphabet, and $\mathcal{D}$ is an infinite set (domain) whose elements (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 pair in $\Sigma \times \mathcal{D}$.

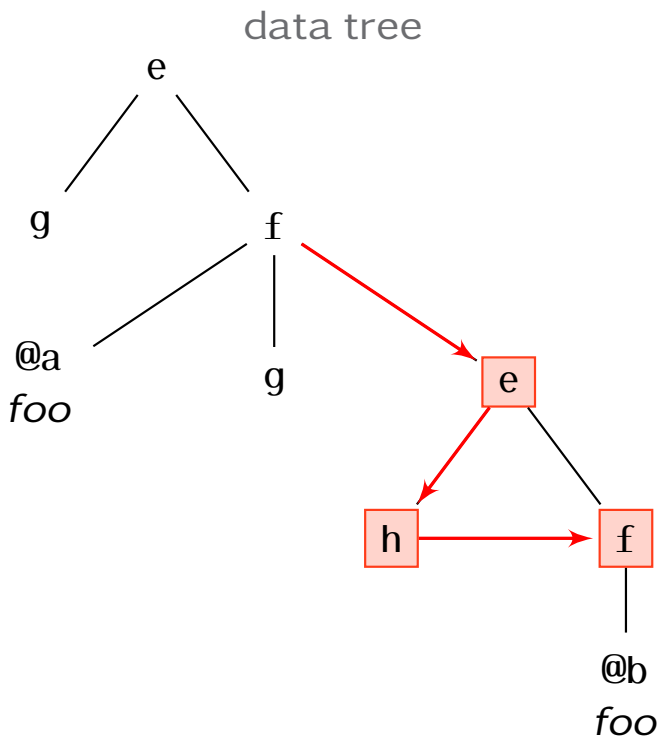
First-order logic for data words was considered by Bojańczyk et al. [2006], and related automata were studied further by Björklund and Schwentick [2007]. The logic has variables that range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), a unary predicate for each letter from the finite alphabet, and a binary predicate $x \sim y$ that denotes equality

This article is a revised and extended version of Jurdziński and Lazić [2007].
R. Lazić was supported by a grant from the Intel Corporation.
Authors' address: M. Jurdziński and R. Lazić, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK, email: {R.S.Lasic, Marcin.Jurdzinski}@warwick.ac.uk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1529-3785/2011/05-ART19 $10.00
DOI 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

ACM Transactions on Computational Logic, Vol. 12, No. 3, Article 19, Publication date: May 2011.

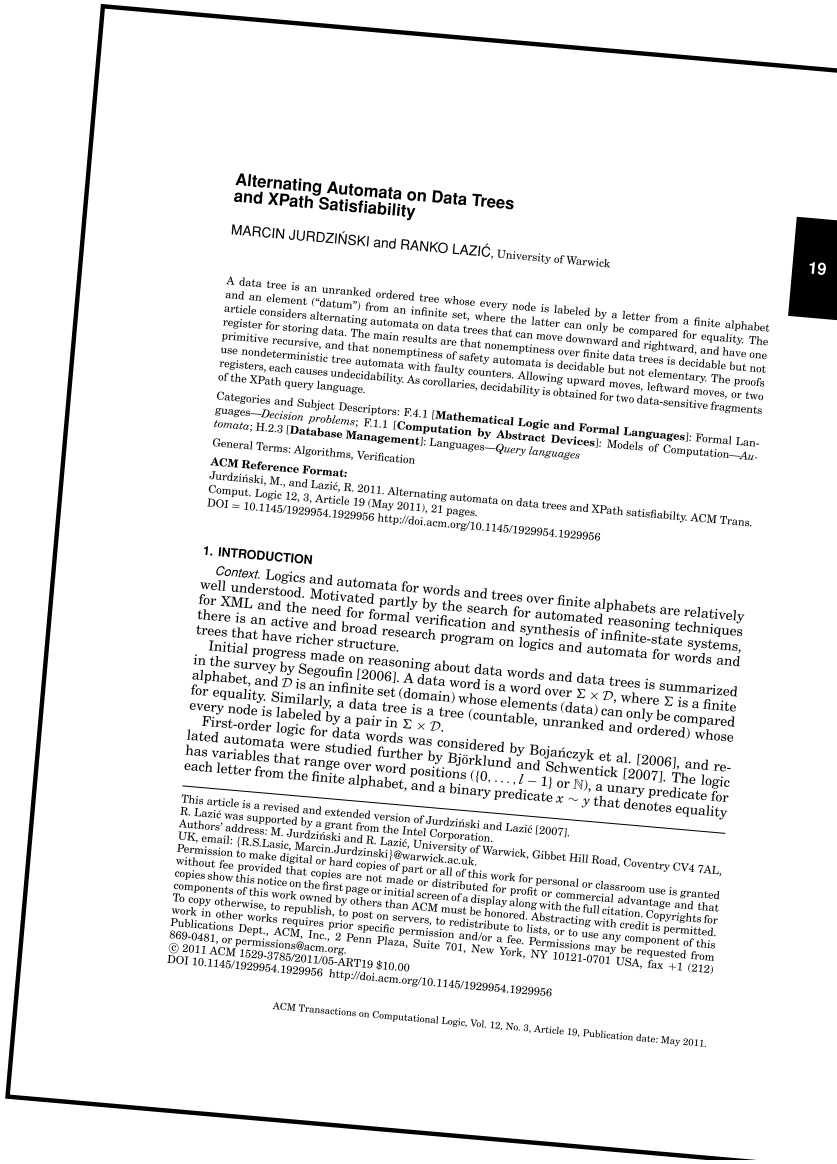
# Example: Forward XPath$^{\varepsilon}$

[Jurdziński & Lazić ′11]




XPath expression

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



**Alternating Automata on Data Trees and XPath Satisfiability**

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

19

A data tree is an unranked ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("datum") from an infinite set, where the latter can only be compared for equality. The article considers alternating automata on data trees that can move downward and rightward, and have one register for storing data. The main results are that nonemptiness over finite data trees is decidable but not primitive recursive, and that nonemptiness of safety automata is decidable but not elementary. The proofs use nondeterministic tree automata with faulty counters. Allowing upward moves, leftward moves, or two registers, each causes undecidability. As corollaries, decidability is obtained for two data-sensitive fragments of the XPath query language.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Verification

**ACM Reference Format:**
Jurdziński, M., and Lazić, R. 2011. Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Logic 12, 3, Article 19 (May 2011), 21 pages.
DOI = 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

**1. INTRODUCTION**

*Context.* Logics 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 for formal verification and synthesis of infinite-state systems, there is an active and broad research program on logics and automata 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 Segoufin [2006]. A data word is a word over $\Sigma \times \mathcal{D}$, where $\Sigma$ is a finite alphabet, and $\mathcal{D}$ is an infinite set (domain) whose elements (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 pair in $\Sigma \times \mathcal{D}$.

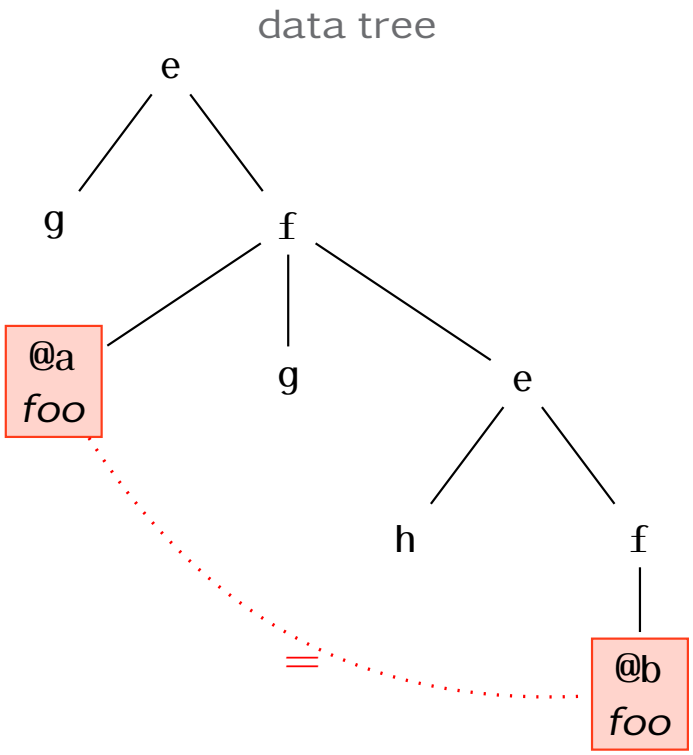
First-order logic for data words was considered by Bojańczyk et al. [2006], and related automata were studied further by Björklund and Schwentick [2007]. The logic has variables that range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), a unary predicate for each letter from the finite alphabet, and a binary predicate $x \sim y$ that denotes equality

This article is a revised and extended version of Jurdziński and Lazić [2007].
R. Lazić was supported by a grant from the Intel Corporation.
Authors' address: M. Jurdziński and R. Lazić, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK, email: {R.S.Lasic, Marcin.Jurdzinski}@warwick.ac.uk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1529-3785/2011/05-ART19 $10.00
DOI 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

ACM Transactions on Computational Logic, Vol. 12, No. 3, Article 19, Publication date: May 2011.

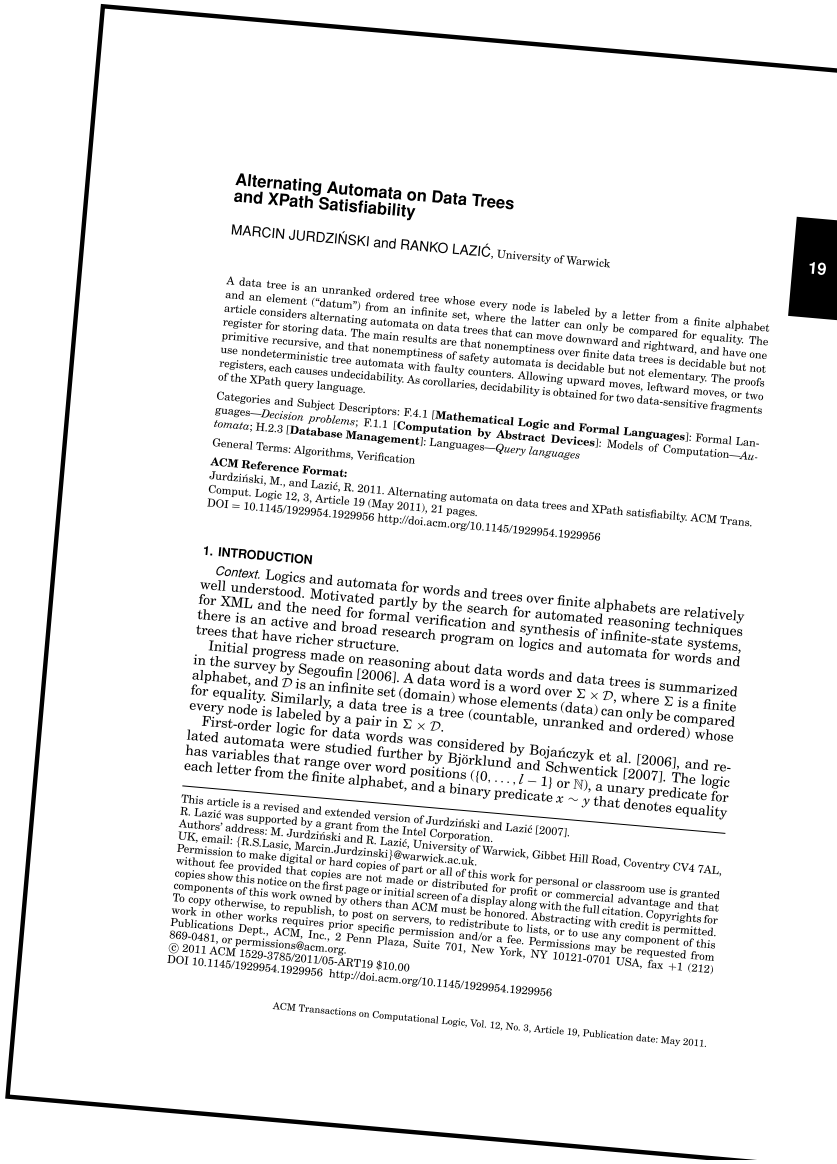
# Example: Forward XPath$^{\varepsilon}$

[Jurdziński & Lazić ′11]




XPath expression

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



**Alternating Automata on Data Trees and XPath Satisfiability**

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

19

A data tree is an unranked ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("datum") from an infinite set, where the latter can only be compared for equality. The article considers alternating automata on data trees that can move downward and rightward, and have one register for storing data. The main results are that nonemptiness over finite data trees is decidable but not primitive recursive, and that nonemptiness of safety automata is decidable but not elementary. The proofs use nondeterministic tree automata with faulty counters. Allowing upward moves, leftward moves, or two registers, each causes undecidability. As corollaries, decidability is obtained for two data-sensitive fragments of the XPath query language.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Verification

**ACM Reference Format:**
Jurdziński, M., and Lazić, R. 2011. Alternating automata on data trees and XPath satisfiability. ACM Trans. Comput. Logic 12, 3, Article 19 (May 2011), 21 pages.
DOI = 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

**1. INTRODUCTION**

*Context.* Logics 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 for formal verification and synthesis of infinite-state systems, there is an active and broad research program on logics and automata 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 Segoufin [2006]. A data word is a word over $\Sigma \times \mathcal{D}$, where $\Sigma$ is a finite alphabet, and $\mathcal{D}$ is an infinite set (domain) whose elements (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 pair in $\Sigma \times \mathcal{D}$.

First-order logic for data words was considered by Bojańczyk et al. [2006], and related automata were studied further by Björklund and Schwentick [2007]. The logic has variables that range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), a unary predicate for each letter from the finite alphabet, and a binary predicate $x \sim y$ that denotes equality

This article is a revised and extended version of Jurdziński and Lazić [2007].
R. Lazić was supported by a grant from the Intel Corporation.
Authors' address: M. Jurdziński and R. Lazić, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK, email: {R.S.Lasic, Marcin.Jurdzinski}@warwick.ac.uk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1529-3785/2011/05-ART19 $10.00
DOI 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

ACM Transactions on Computational Logic, Vol. 12, No. 3, Article 19, Publication date: May 2011.

# Example: Forward XPath$^{\varepsilon}$

[Jurdziński & Lazić ′11]

reductions

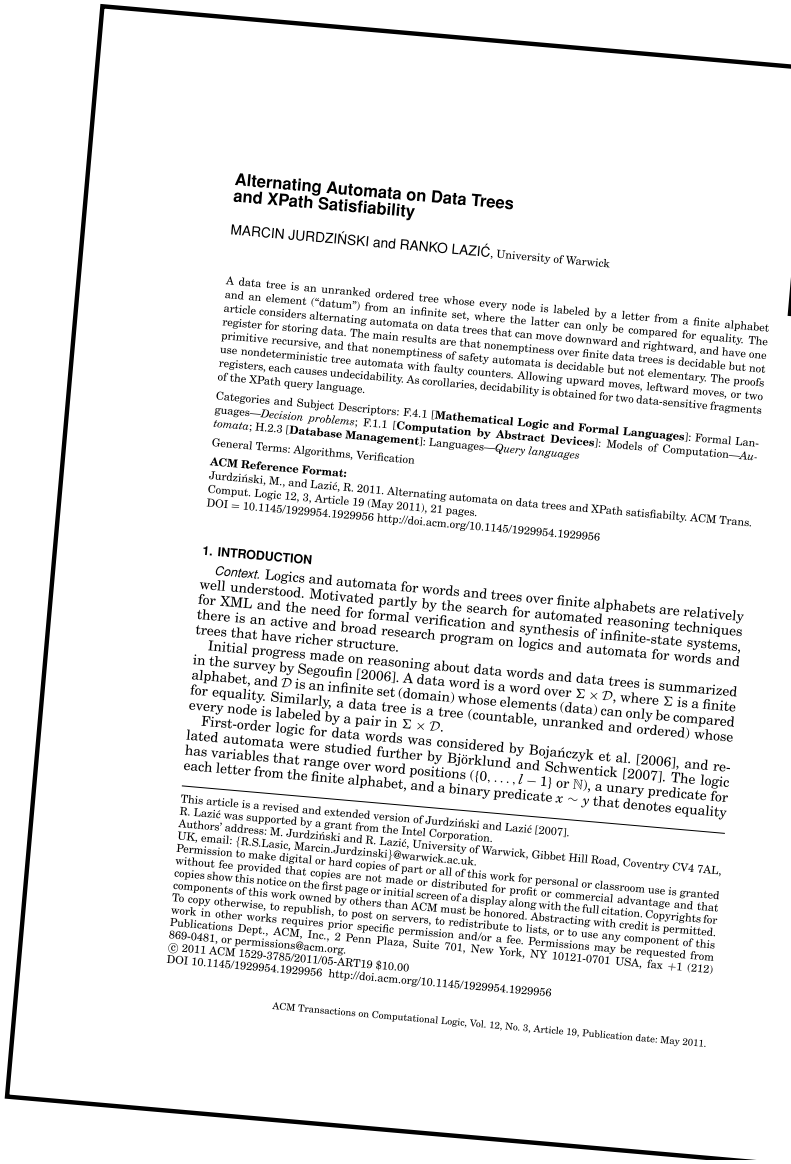
satisfiability for
Forward XPath$^{\varepsilon}$

↓

emptiness of
alternating 1-register tree automata

↓ counter abstraction

subcoverability of
incrementing tree counter machines



**Alternating Automata on Data Trees and XPath Satisfiability**

MARCIN JURDZIŃSKI and RANKO LAZIĆ, University of Warwick

A data tree is an unranked ordered tree whose every node is labeled by a letter from a finite alphabet and an element ("datum") from an infinite set, where the latter can only be compared for equality. The article considers alternating automata on data trees that can move downward and rightward, and have one register for storing data. The main results are that nonemptiness over finite data trees is decidable but not primitive recursive, and that nonemptiness of safety automata is decidable but not elementary. The proofs use nondeterministic tree automata with faulty counters. Allowing upward moves, leftward moves, or two registers, each causes undecidability. As corollaries, decidability is obtained for two data-sensitive fragments of the XPath query language.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Formal Languages—*Decision problems*; F.1.1 [**Computation by Abstract Devices**]: Models of Computation—*Automata*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms: Algorithms, Verification

**ACM Reference Format:**
Jurdziński, M., and Lazić, R. 2011. Alternating automata on data trees and XPath satisfiabilty. ACM Trans. Comput. Logic 12, 3, Article 19 (May 2011), 21 pages.
DOI = 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

**1. INTRODUCTION**

*Context.* Logics 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 for formal verification and synthesis of infinite-state systems, there is an active and broad research program on logics and automata 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 Segoufin [2006]. A data word is a word over $\Sigma \times \mathcal{D}$, where $\Sigma$ is a finite alphabet, and $\mathcal{D}$ is an infinite set (domain) whose elements (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 pair in $\Sigma \times \mathcal{D}$.

First-order logic for data words was considered by Bojańczyk et al. [2006], and related automata were studied further by Björklund and Schwentick [2007]. The logic has variables that range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), a unary predicate for each letter from the finite alphabet, and a binary predicate $x \sim y$ that denotes equality

This article is a revised and extended version of Jurdziński and Lazić [2007].
R. Lazić was supported by a grant from the Intel Corporation.
Authors' address: M. Jurdziński and R. Lazić, University of Warwick, Gibbet Hill Road, Coventry CV4 7AL, UK, email: {R.S.Lasic, Marcin.Jurdzinski}@warwick.ac.uk.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1529-3785/2011/05-ART19 $10.00
DOI 10.1145/1929954.1929956 http://doi.acm.org/10.1145/1929954.1929956

ACM Transactions on Computational Logic, Vol. 12, No. 3, Article 19, Publication date: May 2011.

# 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

**Satisfiability of XPath on data trees**

DIEGO FIGUEIRA, CNRS & LaBRI, France

This is a survey on the satisfiability problem for XPath on data trees. *Data trees* are finite trees whose every node carries a label from a finite alphabet and a data value from an infinite domain. XPath is an expressive node selecting language for XML documents, which can be abstracted as data trees. Its satisfiability problem is in general undecidable. However, various fragments of XPath have decidable satisfiability problem, these are fragments defined in terms of the navigational axes which are allowed. We explore the state of the art in terms of decidability and discuss briefly some of the algorithmic techniques involved.

**1. INTRODUCTION**

A data tree is a tree whose every node contains a label from a finite alphabet and a data value from some infinite domain. This model has been considered as a suitable abstraction for semistructured data such as XML documents: XML tags are modelled as data tree labels, and XML attributes as its data values. A simpler abstraction would only take into account the tags, resulting in a tree over a finite alphabet, a model that has been intensively studied in automata theory. However, this abstraction ignores all actual data stored in the XML document attributes. This is why there has been an interest in 'data' trees as trees that also carry data from an infinite domain. Data trees have also been considered in the realm of timed automata, program verification, and generally in systems manipulating data values. Finding decidable logics or automata models over data trees is generally a relevant question when studying data-driven systems.

More precisely, a *data tree* is an unranked ordered finite tree whose every node contains a label and a data value. Labels belong to some finite alphabet (*e.g.*, the set $\{a, b\}$), and data values to some infinite domain (*e.g.*, $\mathbb{N} = \{0, 1, 2, \ldots\}$), see Figure 1 for an example. In this paper we survey the decidability status on a family of logics based on XPath, which is arguably the most widely used XML query language. XPath is implemented in XSLT and XQuery and it is used as a constituent part of several specification and update languages. XPath is fundamentally a general purpose language for addressing, searching, and matching pieces of an XML document. It is an open standard and constitutes a World Wide Web Consortium (W3C) Recommendation [Clark and DeRose 1999].

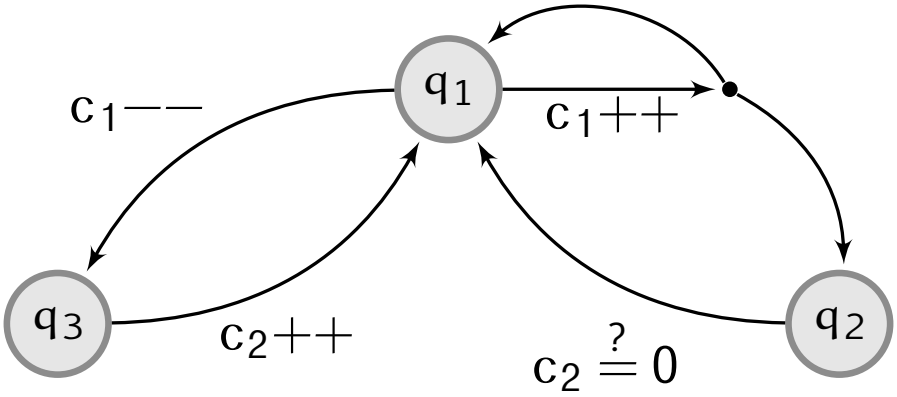
Fig. 1. A data tree over $\mathbb{A} \times \mathbb{D}$, whose alphabet and data domain are $\mathbb{A} = \{a, b\}$ and $\mathbb{D} = \mathbb{N}$.

ACM SIGLOG News 4 April 2018, Vol. 5, No. 2

# Incrementing Tree Counter Machines

[Jurdziński & Lazić '11]

Example




Incrementing Semantics

$$q_1(1,2) \leqslant q_1(2,4) \xrightarrow{c_1{-}{-}} q_3(1,4) \leqslant q_3(3,4)$$

$$q_1(0,2) \leqslant q_1(0,2) \xrightarrow{c_1{+}{+}} q_1(1,2) \leqslant q_1(2,4)$$

$$q_1(0,2) \leqslant q_1(0,2) \xrightarrow{c_1{+}{+}} q_2(1,2) \leqslant q_2(5,2)$$

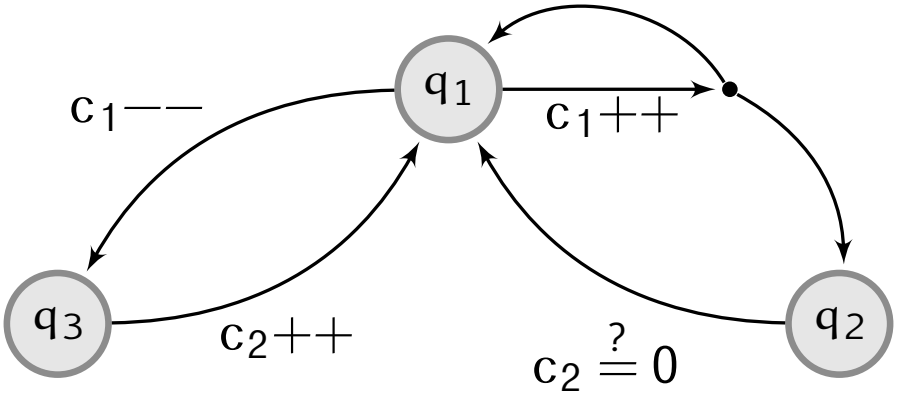
**Theorem** (Jurdziński & Lazić '11)

*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

$$q_1(1,2) \leqslant q_1(2,4) \xrightarrow{c_1{-}{-}} q_3(1,4) \leqslant q_3(3,4)$$

$$q_1(0,2) \leqslant q_1(0,2) \xrightarrow{c_1{+}{+}} q_1(1,2) \leqslant q_1(2,4)$$

$$q_1(0,2) \leqslant q_1(0,2) \xrightarrow{c_1{+}{+}} q_2(1,2) \leqslant q_2(5,2)$$

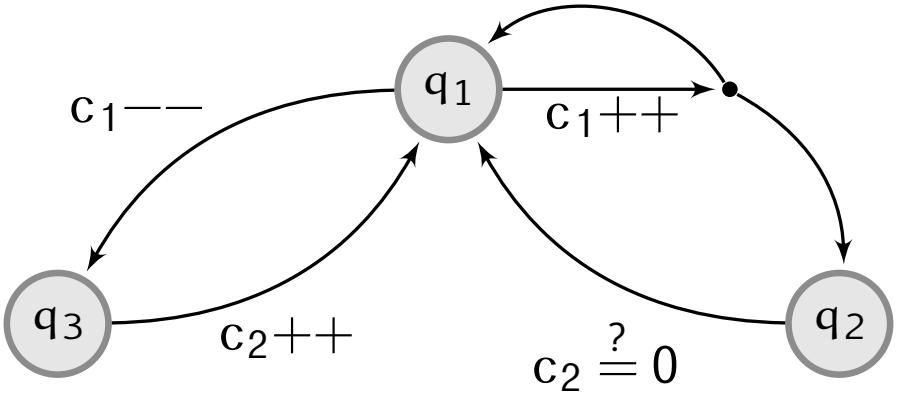
**Theorem** (Jurdziński & Lazić '11)

*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

$$q_1(1,2) \leqslant q_1(2,4) \xrightarrow{c_1{-}{-}} q_3(1,4) \leqslant q_3(3,4)$$

$$q_1(0,2) \leqslant q_1(0,2) \xrightarrow{c_1{+}{+}} q_1(1,2) \leqslant q_1(2,4)$$

$$q_1(0,2) \leqslant q_1(0,2) \xrightarrow{c_1{+}{+}} q_2(1,2) \leqslant q_2(5,2)$$

**Theorem** (Jurdziński & Lazić '11)

*Subcoverability in incrementing tree counter machines is decidable. Thus satisfiability of Forward XPath$^{\varepsilon}$ is also decidable.*

# Example: Relevance Logic $\mathbf{R}_{\to}$

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

## Example: $A \to (B \to A)$

*"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

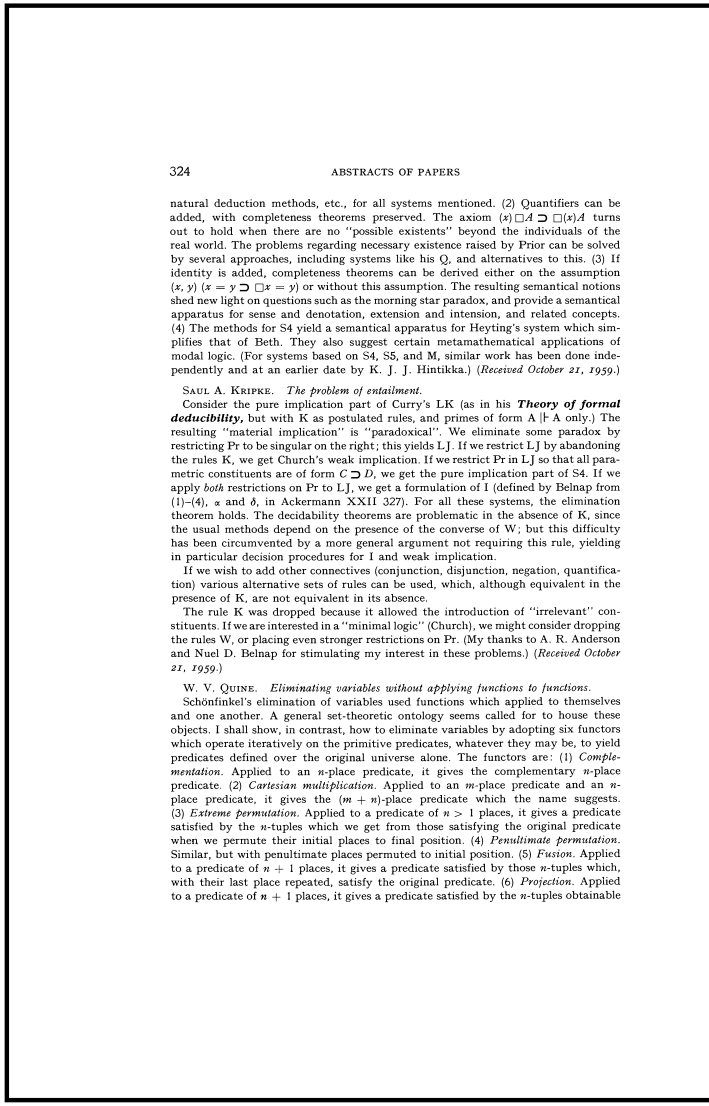
$A, B, C$ formulæ; $\Gamma, \Delta$ finite multisets of formulæ; no weakening

$$\frac{}{A \vdash A}\ (\mathrm{Id}) \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ (\mathrm{C})$$

$$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \to B \vdash C}\ (\to_{\mathrm{L}}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\to_{\mathrm{R}})$$

**Theorem** (Kripke '59)

*Provability in $\mathbf{R}_{\to}$ is decidable.*



324 ABSTRACTS OF PAPERS

natural deduction methods, etc., for all systems mentioned. (2) Quantifiers can be added, with completeness theorems preserved. The axiom $(x)\Box A \supset \Box(x)A$ turns out to hold when there are no "possible existents" beyond the individuals of the real world. The problems regarding necessary existence raised by Prior can be solved by several approaches, including systems like his Q, and alternatives to this. (3) If identity is added, completeness theorems can be derived either on the assumption $(x, y)\ (x = y \supset \Box x = y)$ or without this assumption. The resulting semantical notions shed new light on questions such as the morning star paradox, and provide a semantical apparatus for sense and denotation, extension and intension, and related concepts. (4) The methods for S4 yield a semantical apparatus for Heyting's system which simplifies that of Beth. They also suggest certain metamathematical applications of modal logic. (For systems based on S4, S5, and M, similar work has been done independently and at an earlier date by K. J. J. Hintikka.) (*Received October 21, 1959.*)

Saul A. Kripke. *The problem of entailment.*

Consider the pure implication part of Curry's LK (as in his ***Theory of formal deducibility,*** but with K as postulated rules, and primes of form A ⊢ A only.) The resulting "material implication" is "paradoxical". We eliminate some paradox by restricting Pr to be singular on the right; this yields LJ. If we restrict LJ by abandoning the rules K, we get Church's weak implication. If we restrict Pr in LJ so that all parametric constituents are of form $C \supset D$, we get the pure implication part of S4. If we apply *both* restrictions on Pr to LJ, we get a formulation of I (defined by Belnap from (1)–(4), $\alpha$ and $\delta$, in Ackermann XXII 327). For all these systems, the elimination theorem holds. The decidability theorems are problematic in the absence of K, since the usual methods depend on the presence of the converse of W; but this difficulty has been circumvented by a more general argument not requiring this rule, yielding in particular decision procedures for I and weak implication.

If we wish to add other connectives (conjunction, disjunction, negation, quantification) various alternative sets of rules can be used, which, although equivalent in the presence of K, are not equivalent in its absence.

The rule K was dropped because it allowed the introduction of "irrelevant" constituents. If we are interested in a "minimal logic" (Church), we might consider dropping the rules W, or placing even stronger restrictions on Pr. (My thanks to A. R. Anderson and Nuel D. Belnap for stimulating my interest in these problems.) (*Received October 21, 1959.*)

W. V. Quine. *Eliminating variables without applying functions to functions.*

Schönfinkel's elimination of variables used functions which applied to themselves and one another. A general set-theoretic ontology seems called for to house these objects. I shall show, in contrast, how to eliminate variables by adopting six functors which operate iteratively on the primitive predicates, whatever they may be, to yield predicates defined over the original universe alone. The functors are: (1) *Complementation.* Applied to an $n$-place predicate, it gives the complementary $n$-place predicate. (2) *Cartesian multiplication.* Applied to an $m$-place predicate and an $n$-place predicate, it gives the $(m + n)$-place predicate which the name suggests. (3) *Extreme permutation.* Applied to a predicate of $n > 1$ places, it gives a predicate satisfied by the $n$-tuples which we get from those satisfying the original predicate when we permute their initial places to final position. (4) *Penultimate permutation.* Similar, but with penultimate places permuted to initial position. (5) *Fusion.* Applied to a predicate of $n + 1$ places, it gives a predicate satisfied by those $n$-tuples which, with their last place repeated, satisfy the original predicate. (6) *Projection.* Applied to a predicate of $n + 1$ places, it gives a predicate satisfied by the $n$-tuples obtainable

# Example: Relevance Logic $\mathbf{R}_{\to}$

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

## Example: $A \to (B \to A)$

*"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

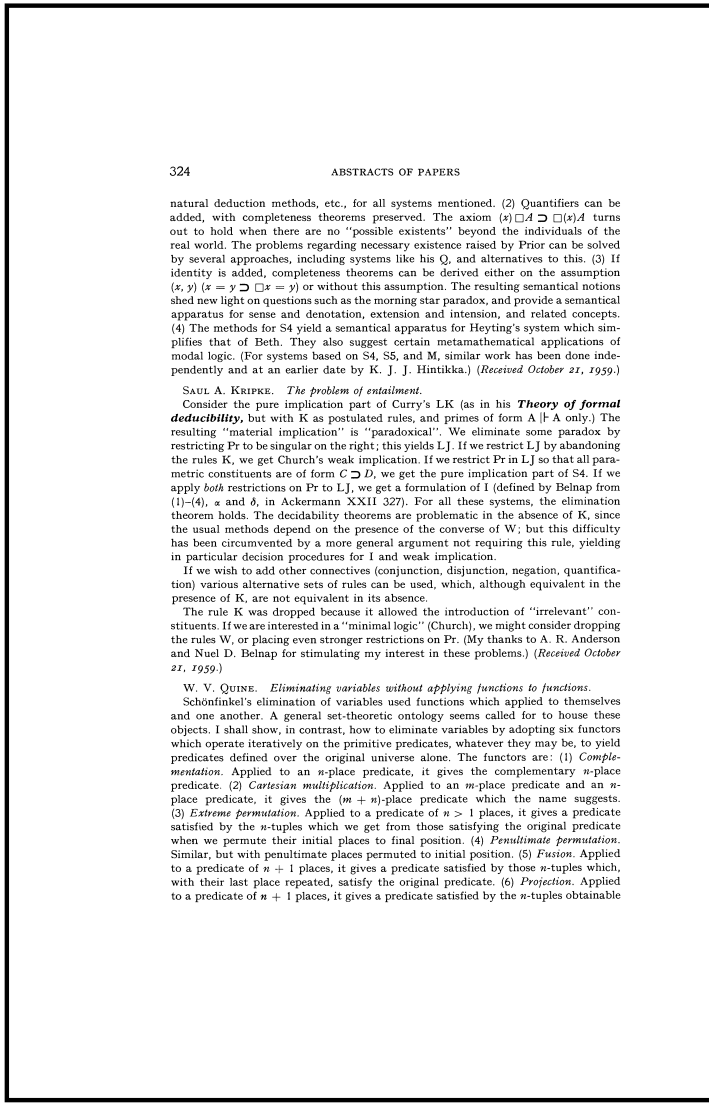
$A, B, C$ formulæ; $\Gamma, \Delta$ finite multisets of formulæ;
no weakening

$$\frac{}{A \vdash A}\ (\mathsf{Id}) \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ (\mathsf{C})$$

$$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \to B \vdash C}\ (\to_{\mathsf{L}}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\to_{\mathsf{R}})$$

**Theorem** (Kripke '59)

*Provability in $\mathbf{R}_{\to}$ is decidable.*



324 ABSTRACTS OF PAPERS

natural deduction methods, etc., for all systems mentioned. (2) Quantifiers can be added, with completeness theorems preserved. The axiom $(x)\Box A \supset \Box(x)A$ turns out to hold when there are no "possible existents" beyond the individuals of the real world. The problems regarding necessary existence raised by Prior can be solved by several approaches, including systems like his Q, and alternatives to this. (3) If identity is added, completeness theorems can be derived either on the assumption $(x, y)\ (x = y \supset \Box x = y)$ or without this assumption. The resulting semantical notions shed new light on questions such as the morning star paradox, and provide a semantical apparatus for sense and denotation, extension and intension, and related concepts. (4) The methods for S4 yield a semantical apparatus for Heyting's system which simplifies that of Beth. They also suggest certain metamathematical applications of modal logic. (For systems based on S4, S5, and M, similar work has been done independently and at an earlier date by K. J. J. Hintikka.) (*Received October 21, 1959.*)

Saul A. Kripke. *The problem of entailment.*

Consider the pure implication part of Curry's LK (as in his ***Theory of formal deducibility,*** but with K as postulated rules, and primes of form A ⊢ A only.) The resulting "material implication" is "paradoxical". We eliminate some paradox by restricting Pr to be singular on the right; this yields LJ. If we restrict LJ by abandoning the rules K, we get Church's weak implication. If we restrict Pr in LJ so that all parametric constituents are of form $C \supset D$, we get the pure implication part of S4. If we apply *both* restrictions on Pr to LJ, we get a formulation of I (defined by Belnap from (1)–(4), $\alpha$ and $\delta$, in Ackermann XXII 327). For all these systems, the elimination theorem holds. The decidability theorems are problematic in the absence of K, since the usual methods depend on the presence of the converse of W; but this difficulty has been circumvented by a more general argument not requiring this rule, yielding in particular decision procedures for I and weak implication.

If we wish to add other connectives (conjunction, disjunction, negation, quantification) various alternative sets of rules can be used, which, although equivalent in the presence of K, are not equivalent in its absence.

The rule K was dropped because it allowed the introduction of "irrelevant" constituents. If we are interested in a "minimal logic" (Church), we might consider dropping the rules W, or placing even stronger restrictions on Pr. (My thanks to A. R. Anderson and Nuel D. Belnap for stimulating my interest in these problems.) (*Received October 21, 1959.*)

W. V. Quine. *Eliminating variables without applying functions to functions.*

Schönfinkel's elimination of variables used functions which applied to themselves and one another. A general set-theoretic ontology seems called for to house these objects. I shall show, in contrast, how to eliminate variables by adopting six functors which operate iteratively on the primitive predicates, whatever they may be, to yield predicates defined over the original universe alone. The functors are: (1) *Complementation.* Applied to an $n$-place predicate, it gives the complementary $n$-place predicate. (2) *Cartesian multiplication.* Applied to an $m$-place predicate and an $n$-place predicate, it gives the $(m + n)$-place predicate which the name suggests. (3) *Extreme permutation.* Applied to a predicate of $n > 1$ places, it gives a predicate satisfied by the $n$-tuples which we get from those satisfying the original predicate when we permute their initial places to final position. (4) *Penultimate permutation.* Similar, but with penultimate places permuted to initial position. (5) *Fusion.* Applied to a predicate of $n + 1$ places, it gives a predicate satisfied by those $n$-tuples which, with their last place repeated, satisfy the original predicate. (6) *Projection.* Applied to a predicate of $n + 1$ places, it gives a predicate satisfied by the $n$-tuples obtainable

# Example: Relevance Logic $\mathbf{R}_{\to}$

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

## Example: $A \to (B \to A)$

*"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

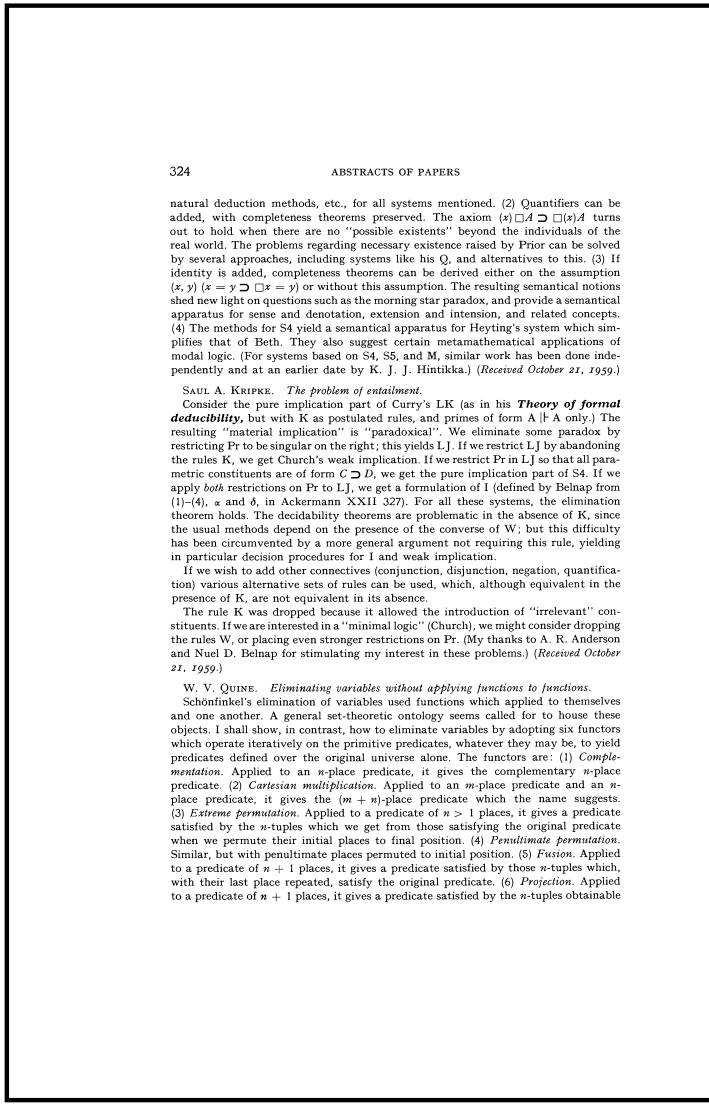
$A, B, C$ formulæ; $\Gamma, \Delta$ finite multisets of formulæ;
no weakening

$$\frac{}{A \vdash A}\ (\mathsf{Id}) \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ (\mathsf{C})$$

$$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \to B \vdash C}\ (\to_{\mathsf{L}}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\to_{\mathsf{R}})$$

**Theorem** (Kripke '59)

*Provability in $\mathbf{R}_{\to}$ is decidable.*



324 ABSTRACTS OF PAPERS

natural deduction methods, etc., for all systems mentioned. (2) Quantifiers can be added, with completeness theorems preserved. The axiom $(x)\Box A \supset \Box(x)A$ turns out to hold when there are no "possible existents" beyond the individuals of the real world. The problems regarding necessary existence raised by Prior can be solved by several approaches, including systems like his Q, and alternatives to this. (3) If identity is added, completeness theorems can be derived either on the assumption $(x, y)\,(x = y \supset \Box x = y)$ or without this assumption. The resulting semantical notions shed new light on questions such as the morning star paradox, and provide a semantical apparatus for sense and denotation, extension and intension, and related concepts. (4) The methods for S4 yield a semantical apparatus for Heyting's system which simplifies that of Beth. They also suggest certain metamathematical applications of modal logic. (For systems based on S4, S5, and M, similar work has been done independently and at an earlier date by K. J. J. Hintikka.) (*Received October 21, 1959.*)

Saul A. Kripke. *The problem of entailment.*

Consider the pure implication part of Curry's LK (as in his ***Theory of formal deducibility,*** but with K as postulated rules, and primes of form A ⊢ A only.) The resulting "material implication" is "paradoxical". We eliminate some paradox by restricting Pr to be singular on the right; this yields LJ. If we restrict LJ by abandoning the rules K, we get Church's weak implication. If we restrict Pr in LJ so that all parametric constituents are of form $C \supset D$, we get the pure implication part of S4. If we apply *both* restrictions on Pr to LJ, we get a formulation of I (defined by Belnap from (1)–(4), $\alpha$ and $\delta$, in Ackermann XXII 327). For all these systems, the elimination theorem holds. The decidability theorems are problematic in the absence of K, since the usual methods depend on the presence of the converse of W; but this difficulty has been circumvented by a more general argument not requiring this rule, yielding in particular decision procedures for I and weak implication.

If we wish to add other connectives (conjunction, disjunction, negation, quantification) various alternative sets of rules can be used, which, although equivalent in the presence of K, are not equivalent in its absence.

The rule K was dropped because it allowed the introduction of "irrelevant" constituents. If we are interested in a "minimal logic" (Church), we might consider dropping the rules W, or placing even stronger restrictions on Pr. (My thanks to A. R. Anderson and Nuel D. Belnap for stimulating my interest in these problems.) (*Received October 21, 1959.*)

W. V. Quine. *Eliminating variables without applying functions to functions.*

Schönfinkel's elimination of variables used functions which applied to themselves and one another. A general set-theoretic ontology seems called for to house these objects. I shall show, in contrast, how to eliminate variables by adopting six functors which operate iteratively on the primitive predicates, whatever they may be, to yield predicates defined over the original universe alone. The functors are: (1) *Complementation.* Applied to an $n$-place predicate, it gives the complementary $n$-place predicate. (2) *Cartesian multiplication.* Applied to an $m$-place predicate and an $n$-place predicate, it gives the $(m + n)$-place predicate which the name suggests. (3) *Extreme permutation.* Applied to a predicate of $n > 1$ places, it gives a predicate satisfied by the $n$-tuples which we get from those satisfying the original predicate when we permute their initial places to final position. (4) *Penultimate permutation.* Similar, but with penultimate places permuted to initial position. (5) *Fusion.* Applied to a predicate of $n + 1$ places, it gives a predicate satisfied by those $n$-tuples which, with their last place repeated, satisfy the original predicate. (6) *Projection.* Applied to a predicate of $n + 1$ places, it gives a predicate satisfied by the $n$-tuples obtainable

# Example: Relevance Logic LR+

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

## Gentzen-style sequent calculus

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

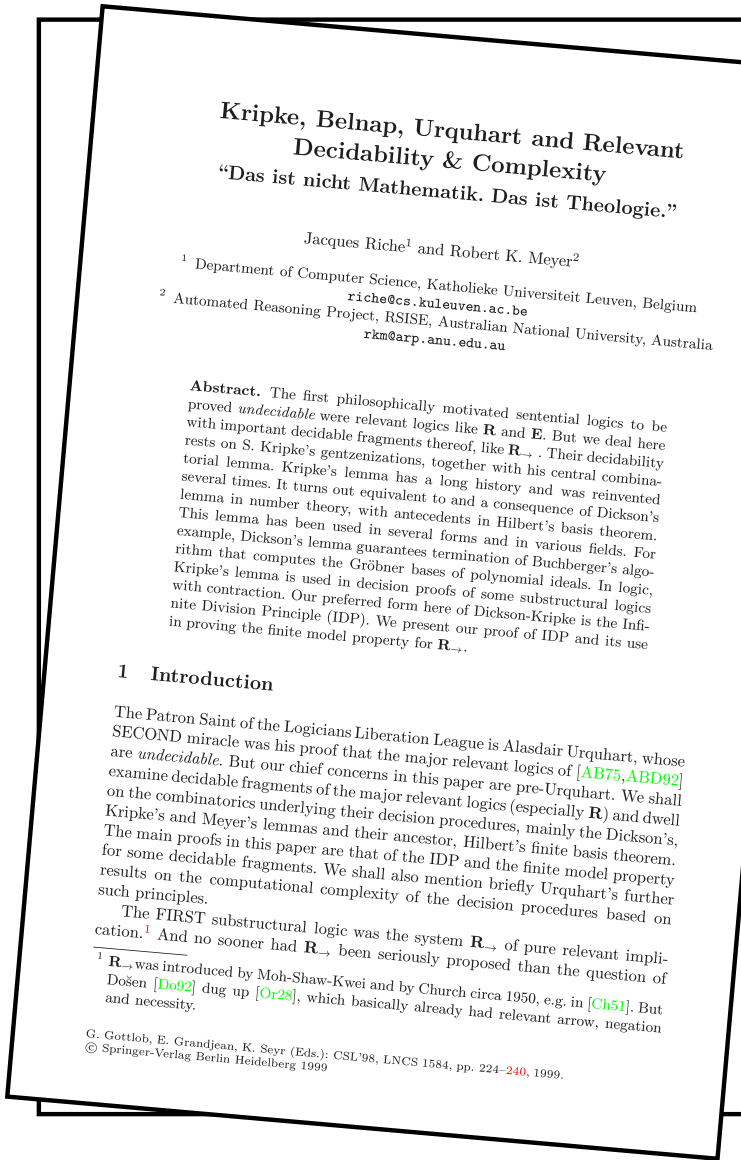
$$\frac{}{A \vdash A}\ (\mathsf{Id}) \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ (\mathsf{C})$$

$$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \to B \vdash C}\ (\to_{\mathsf{L}}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\to_{\mathsf{R}})$$

Thus $\to$ behaves like intuitionistic linear implication $\multimap$ with added contraction. 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.*



**Kripke, Belnap, Urquhart and Relevant Decidability & Complexity**

**"Das ist nicht Mathematik. Das ist Theologie."**

Jacques Riche[1] and Robert K. Meyer[2]

[1] Department of Computer Science, Katholieke Universiteit Leuven, Belgium
riche@cs.kuleuven.ac.be

[2] Automated Reasoning Project, RSISE, Australian National University, Australia
rkm@arp.anu.edu.au

**Abstract.** The first philosophically motivated sentential logics to be proved *undecidable* were relevant logics like **R** and **E**. But we deal here with important decidable fragments thereof, like $\mathbf{R}_{\to}$. Their decidability rests on S. Kripke's gentzenizations, together with his central combinatorial lemma. Kripke's lemma has a long history and was reinvented several times. It turns out equivalent to and a consequence of Dickson's lemma in number theory, with antecedents in Hilbert's basis theorem. This lemma has been used in several forms and in various fields. For example, Dickson's lemma guarantees termination of Buchberger's algorithm that computes the Gröbner bases of polynomial ideals. In logic, Kripke's lemma is used in decision proofs of some substructural logics with contraction. Our preferred form here of Dickson-Kripke is the Infinite Division Principle (IDP). We present our proof of IDP and its use in proving the finite model property for $\mathbf{R}_{\to}$.

**1 Introduction**

The Patron Saint of the Logicians Liberation League is Alasdair Urquhart, whose SECOND miracle was his proof that the major relevant logics of [AB75,ABD92] are *undecidable*. But our chief concerns in this paper are pre-Urquhart. We shall examine decidable fragments of the major relevant logics (especially **R**) and dwell on the combinatorics underlying their decision procedures, mainly the Dickson's, Kripke's and Meyer's lemmas and their ancestor, Hilbert's finite basis theorem. The main proofs in this paper are that of the IDP and the finite model property for some decidable fragments. We shall also mention briefly Urquhart's further results on the computational complexity of the decision procedures based on such principles.

The FIRST substructural logic was the system $\mathbf{R}_{\to}$ of pure relevant implication.[1] And no sooner had $\mathbf{R}_{\to}$ been seriously proposed than the question of

[1] $\mathbf{R}_{\to}$ was introduced by Moh-Shaw-Kwei and by Church circa 1950, e.g. in [Ch51]. But Došen [Do92] dug up [Or28], which basically already had relevant arrow, negation and necessity.

G. Gottlob, E. Grandjean, K. Seyr (Eds.): CSL'98, LNCS 1584, pp. 224–240, 1999.
© Springer-Verlag Berlin Heidelberg 1999

# Example: Relevance Logic LR+

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

## Gentzen-style sequent calculus

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

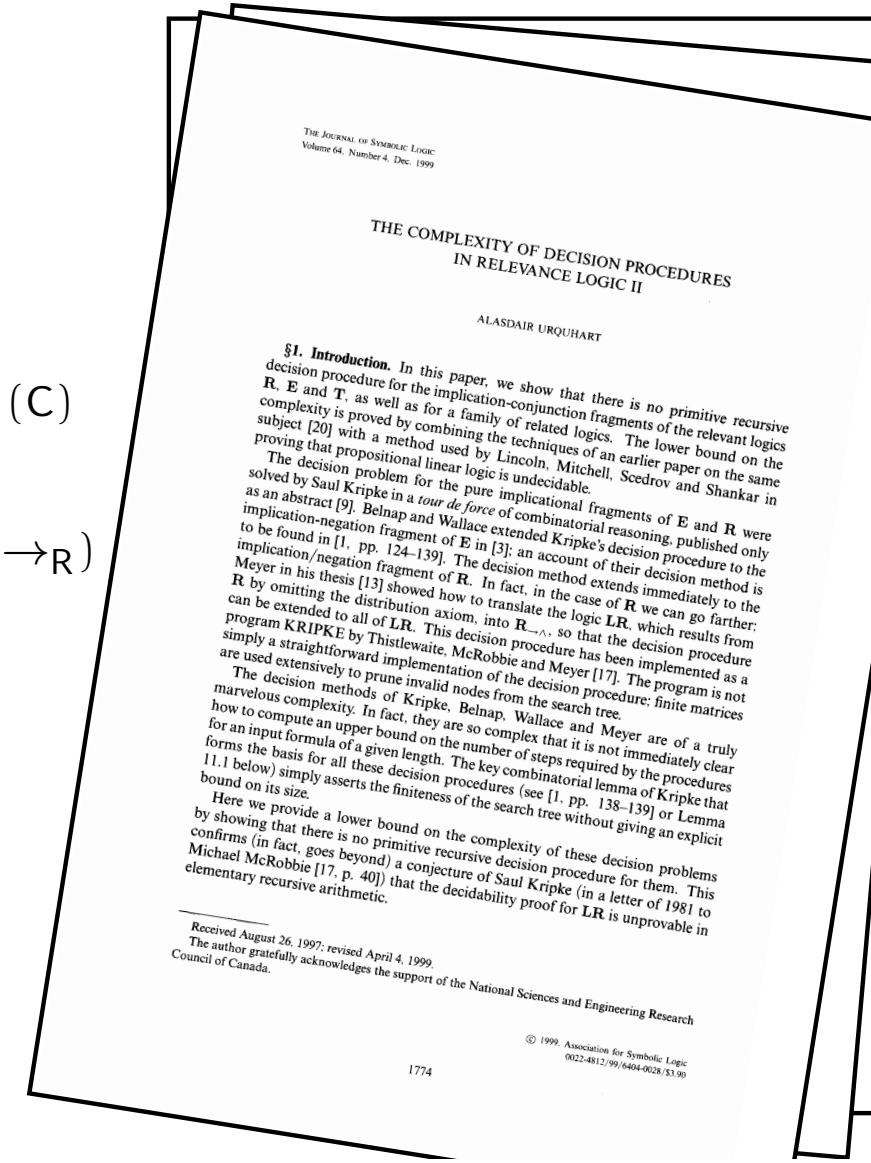
$$\frac{}{A \vdash A}\ (\mathsf{Id}) \qquad \frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ (\mathsf{C})$$

$$\frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \to B \vdash C}\ (\to_{\mathsf{L}}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ (\to_{\mathsf{R}})$$

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

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

*Provability in* $\mathbf{LR}+$ *is decidable.*



The Journal of Symbolic Logic
Volume 64, Number 4, Dec. 1999

THE COMPLEXITY OF DECISION PROCEDURES IN RELEVANCE LOGIC II

ALASDAIR URQUHART

**§1. Introduction.** In this paper, we show that there is no primitive recursive decision procedure for the implication-conjunction fragments of the relevant logics **R**, **E** and **T**, as well as for a family of related logics. The lower bound on the complexity is proved by combining the techniques of an earlier paper on the same subject [20] with a method used by Lincoln, Mitchell, Scedrov and Shankar in proving that propositional linear logic is undecidable.

The decision problem for the pure implicational fragments of **E** and **R** were solved by Saul Kripke in a *tour de force* of combinatorial reasoning, published only as an abstract [9]. Belnap and Wallace extended Kripke's decision procedure to the implication-negation fragment of **E** in [3]; an account of their decision method is to be found in [1, pp. 124–139]. The decision method extends immediately to the implication/negation fragment of **R**. In fact, in the case of **R** we can go farther: Meyer in his thesis [13] showed how to translate the logic **LR**, which results from **R** by omitting the distribution axiom, into $\mathbf{R}_{\to\wedge}$, so that the decision procedure can be extended to all of **LR**. This decision procedure has been implemented as a program KRIPKE by Thistlewaite, McRobbie and Meyer [17]. The program is not simply a straightforward implementation of the decision procedure; finite matrices are used extensively to prune invalid nodes from the search tree.

The decision methods of Kripke, Belnap, Wallace and Meyer are of a truly marvelous complexity. In fact, they are so complex that it is not immediately clear how to compute an upper bound on the number of steps required by the procedures for an input formula of a given length. The key combinatorial lemma of Kripke that forms the basis for all these decision procedures (see [1, pp. 138–139] or Lemma 11.1 below) simply asserts the finiteness of the search tree without giving an explicit bound on its size.

Here we provide a lower bound on the complexity of these decision problems by showing that there is no primitive recursive decision procedure for them. This confirms (in fact, goes beyond) a conjecture of Saul Kripke (in a letter of 1981 to Michael McRobbie [17, p. 40]) that the decidability proof for **LR** is unprovable in elementary recursive arithmetic.

Received August 26, 1997; revised April 4, 1999.
The author gratefully acknowledges the support of the National Sciences and Engineering Research Council of Canada.

# 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 $\Gamma \vdash A$

$$\dfrac{\overline{B \vdash B} \quad \cdots \quad \overline{C \vdash C}}{\vdots}$$

$$\Gamma \vdash A$$

we only need to work with formulæ from the finite set

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

- For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\ulcorner \Delta \vdash B \urcorner \stackrel{\text{def}}{=} (\mathrm{dom}(\Delta), \mathbf{v}_\Delta, B)$ of

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

where $\mathbf{v}_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

- $(X, \leqslant)$ as the product of $(\mathbb{P}(\Sigma), =)$, $(\mathbb{N}^{|\Sigma|}, \leqslant)$, and $(\Sigma, =)$ is a wqo by Dickson's Lemma
- $\ulcorner \Delta \vdash B \urcorner \leqslant \ulcorner \Delta' \vdash B' \urcorner$ 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 $\ulcorner \Gamma \vdash A \urcorner$ and leaves in $\{\ulcorner B \vdash B \urcorner \mid B \in \Sigma\}$

# 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 $\Gamma \vdash A$

$$\dfrac{\overline{B \vdash B} \quad \cdots \quad \overline{C \vdash C}}{\Gamma \vdash A}$$

we only need to work with formulæ from the finite set

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

- For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\ulcorner \Delta \vdash B \urcorner \stackrel{\text{def}}{=} (\mathrm{dom}(\Delta), \mathbf{v}_\Delta, B)$ of

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

where $\mathbf{v}_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

- $(X, \leqslant)$ as the product of $(\mathbb{P}(\Sigma), =)$, $(\mathbb{N}^{|\Sigma|}, \leqslant)$, and $(\Sigma, =)$ is a wqo by Dickson's Lemma
- $\ulcorner \Delta \vdash B \urcorner \leqslant \ulcorner \Delta' \vdash B' \urcorner$ 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 $\ulcorner \Gamma \vdash A \urcorner$ and leaves in $\{\ulcorner B \vdash B \urcorner \mid B \in \Sigma\}$

# 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 $\Gamma \vdash A$

$$\dfrac{\overline{B \vdash B} \quad \cdots \quad \overline{C \vdash C}}{\Gamma \vdash A}$$

we only need to work with formulæ from the finite set

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

- For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\ulcorner \Delta \vdash B \urcorner \stackrel{\text{def}}{=} (\mathrm{dom}(\Delta), \mathbf{v}_\Delta, B)$ of

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

where $\mathbf{v}_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

- $(X, \leqslant)$ as the product of $(\mathbb{P}(\Sigma), =)$, $(\mathbb{N}^{|\Sigma|}, \leqslant)$, and $(\Sigma, =)$ is a wqo by Dickson's Lemma
- $\ulcorner \Delta \vdash B \urcorner \leqslant \ulcorner \Delta' \vdash B' \urcorner$ 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 $\ulcorner \Gamma \vdash A \urcorner$ and leaves in $\{\ulcorner B \vdash B \urcorner \mid B \in \Sigma\}$

# 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 $\Gamma \vdash A$

$$\dfrac{\overline{B \vdash B} \quad \cdots \quad \overline{C \vdash C}}{\Gamma \vdash A}$$

we only need to work with formulæ from the finite set

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

- For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\ulcorner \Delta \vdash B \urcorner \stackrel{\text{def}}{=} (\mathrm{dom}(\Delta), \mathbf{v}_\Delta, B)$ of

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

where $\mathbf{v}_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

- $(X, \leqslant)$ as the product of $(\mathbb{P}(\Sigma), =)$, $(\mathbb{N}^{|\Sigma|}, \leqslant)$, and $(\Sigma, =)$ is a wqo by Dickson's Lemma
- $\ulcorner \Delta \vdash B \urcorner \leqslant \ulcorner \Delta' \vdash B' \urcorner$ 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 $\ulcorner \Gamma \vdash A \urcorner$ and leaves in $\{\ulcorner B \vdash B \urcorner \mid B \in \Sigma\}$

# 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 $\Gamma \vdash A$

$$\dfrac{\overline{B \vdash B} \quad \cdots \quad \overline{C \vdash C}}{\Gamma \vdash A}$$

we only need to work with formulæ from the finite set

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

- For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\ulcorner \Delta \vdash B \urcorner \stackrel{\text{def}}{=} (\mathrm{dom}(\Delta), \mathbf{v}_\Delta, B)$ of

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

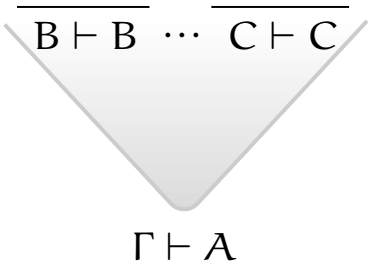
where $\mathbf{v}_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

- $(X, \leqslant)$ as the product of $(\mathbb{P}(\Sigma), =)$, $(\mathbb{N}^{|\Sigma|}, \leqslant)$, and $(\Sigma, =)$ is a wqo by Dickson's Lemma
- $\ulcorner \Delta \vdash B \urcorner \leqslant \ulcorner \Delta' \vdash B' \urcorner$ 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 $\ulcorner \Gamma \vdash A \urcorner$ and leaves in $\{\ulcorner B \vdash B \urcorner \mid B \in \Sigma\}$

# 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 $\Gamma \vdash A$




we only need to work with formulæ from the finite set

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

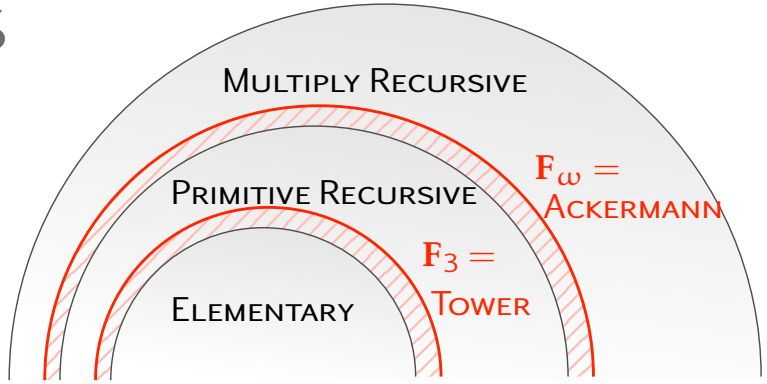
- For a sequent $\Delta \vdash B$ built over $\Sigma$, associate an element $\ulcorner \Delta \vdash B \urcorner \stackrel{\text{def}}{=} (\mathrm{dom}(\Delta), \mathbf{v}_\Delta, B)$ of

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

where $\mathbf{v}_\Delta(C)$ is the number of occurrences of $C$ in the multiset $\Delta$

- $(X, \leqslant)$ as the product of $(\mathbb{P}(\Sigma), =)$, $(\mathbb{N}^{|\Sigma|}, \leqslant)$, and $(\Sigma, =)$ is a wqo by Dickson's Lemma
- $\ulcorner \Delta \vdash B \urcorner \leqslant \ulcorner \Delta' \vdash B' \urcorner$ 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 $\ulcorner \Gamma \vdash A \urcorner$ and leaves in $\{\ulcorner B \vdash B \urcorner \mid B \in \Sigma\}$

# Complexity Lower Bounds




**Theorem** (Demri & Lazic '09, Figueira & Segoufin, '09)

*Satisfiability of Forward XPath$^\varepsilon$ expressions is $\mathbf{F}_\omega$-hard.*

**Theorem** (Urquhart '99)

*Provability in $\mathbf{LR}+$ is $\mathbf{F}_\omega$-hard.*

# Lifting wqos

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

$$S \subseteq_{\mathbb{H}} S' \overset{\text{def}}{\Leftrightarrow} \forall x \in S \,.\, \exists x' \in S' \,.\, x \leqslant x'$$

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

$$S \subseteq_{\mathbb{S}} S' \overset{\text{def}}{\Leftrightarrow} \forall x' \in S' \,.\, \exists x \in S \,.\, x \leqslant x'$$

# Lifting wqos

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

$$S \subseteq_{\mathbb{H}} S' \overset{\text{def}}{\Leftrightarrow} \forall x \in S \,.\, \exists x' \in S' \,.\, x \leqslant x'$$

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

$$S \subseteq_{\mathbb{S}} S' \overset{\text{def}}{\Leftrightarrow} \forall x' \in S' \,.\, \exists x \in S \,.\, x \leqslant x'$$

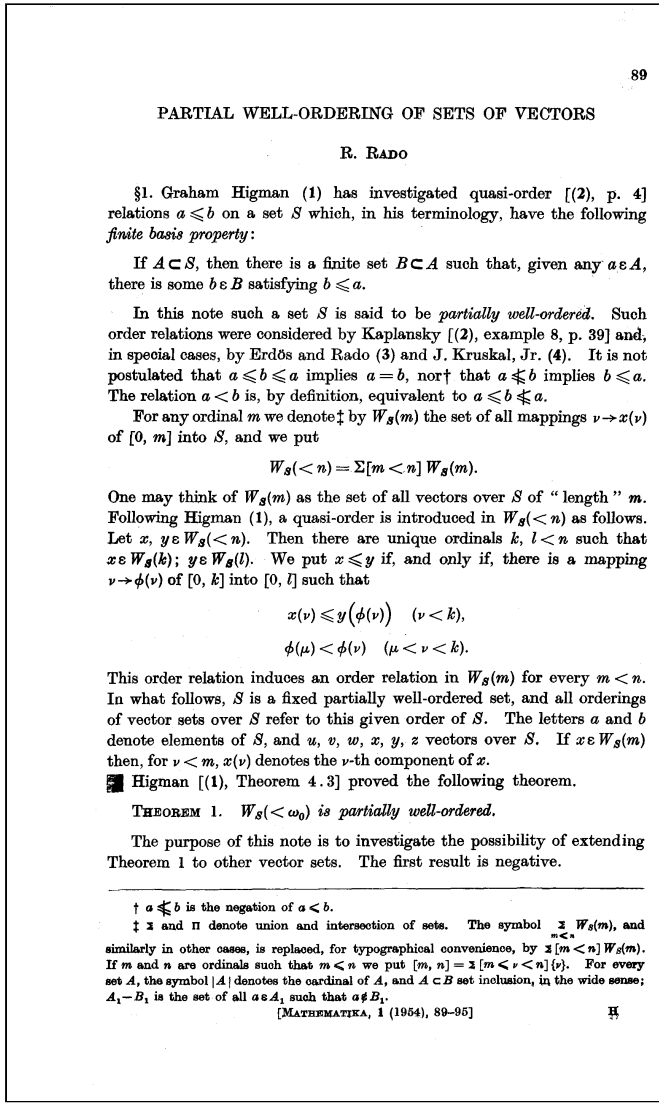
# Lifting wqos

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

$$S \subseteq_{\mathbb{H}} S' \overset{\text{def}}{\Leftrightarrow} \forall x \in S \,.\, \exists x' \in S' \,.\, x \leqslant x'$$

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

$$S \subseteq_{\mathbb{S}} S' \overset{\text{def}}{\Leftrightarrow} \forall x' \in S' \,.\, \exists x \in S \,.\, x \leqslant x'$$



89

PARTIAL WELL-ORDERING OF SETS OF VECTORS

R. Rado

§1. Graham Higman (1) has investigated quasi-order [(2), p. 4] relations $a \leqslant b$ on a set $S$ which, in his terminology, have the following *finite basis property*:

If $A \subset S$, then there is a finite set $B \subset A$ such that, given any $a \varepsilon A$, there is some $b \varepsilon B$ satisfying $b \leqslant a$.

In this note such a set $S$ is said to be *partially well-ordered*. Such order relations were considered by Kaplansky [(2), example 8, p. 39] and, in special cases, by Erdös and Rado (3) and J. Kruskal, Jr. (4). It is not postulated that $a \leqslant b \leqslant a$ implies $a = b$, nor† that $a \nleqslant b$ implies $b \leqslant a$. The relation $a < b$ is, by definition, equivalent to $a \leqslant b \nleqslant a$.

For any ordinal $m$ we denote‡ by $W_S(m)$ the set of all mappings $\nu \to x(\nu)$ of $[0, m]$ into $S$, and we put

$$W_S(< n) = \Sigma[m < n]\, W_S(m).$$

One may think of $W_S(m)$ as the set of all vectors over $S$ of "length" $m$. Following Higman (1), a quasi-order is introduced in $W_S(< n)$ as follows. Let $x, y \varepsilon W_S(< n)$. Then there are unique ordinals $k, l < n$ such that $x \varepsilon W_S(k)$; $y \varepsilon W_S(l)$. We put $x \leqslant y$ if, and only if, there is a mapping $\nu \to \phi(\nu)$ of $[0, k]$ into $[0, l]$ such that

$$x(\nu) \leqslant y\big(\phi(\nu)\big) \quad (\nu < k),$$

$$\phi(\mu) < \phi(\nu) \quad (\mu < \nu < k).$$

This order relation induces an order relation in $W_S(m)$ for every $m < n$. In what follows, $S$ is a fixed partially well-ordered set, and all orderings of vector sets over $S$ refer to this given order of $S$. The letters $a$ and $b$ denote elements of $S$, and $u, v, w, x, y, z$ vectors over $S$. If $x \varepsilon W_S(m)$ then, for $\nu < m$, $x(\nu)$ denotes the $\nu$-th component of $x$.

Higman [(1), Theorem 4.3] proved the following theorem.

Theorem 1. *$W_S(< \omega_0)$ is partially well-ordered.*

The purpose of this note is to investigate the possibility of extending Theorem 1 to other vector sets. The first result is negative.

† $a \nleqslant b$ is the negation of $a < b$.

‡ $\Sigma$ and $\Pi$ denote union and intersection of sets. The symbol $\sum_{m<n} W_S(m)$, and similarly in other cases, is replaced, for typographical convenience, by $\Sigma[m < n]\, W_S(m)$. If $m$ and $n$ are ordinals such that $m < n$ we put $[m, n] = \Sigma[m < \nu < n]\{\nu\}$. For every set $A$, the symbol $|A|$ denotes the cardinal of $A$, and $A \subset B$ set inclusion, in the wide sense; $A_1 - B_1$ is the set of all $a \varepsilon A_1$ such that $a \notin B_1$.

[Mathematika, 1 (1954), 89–95] H

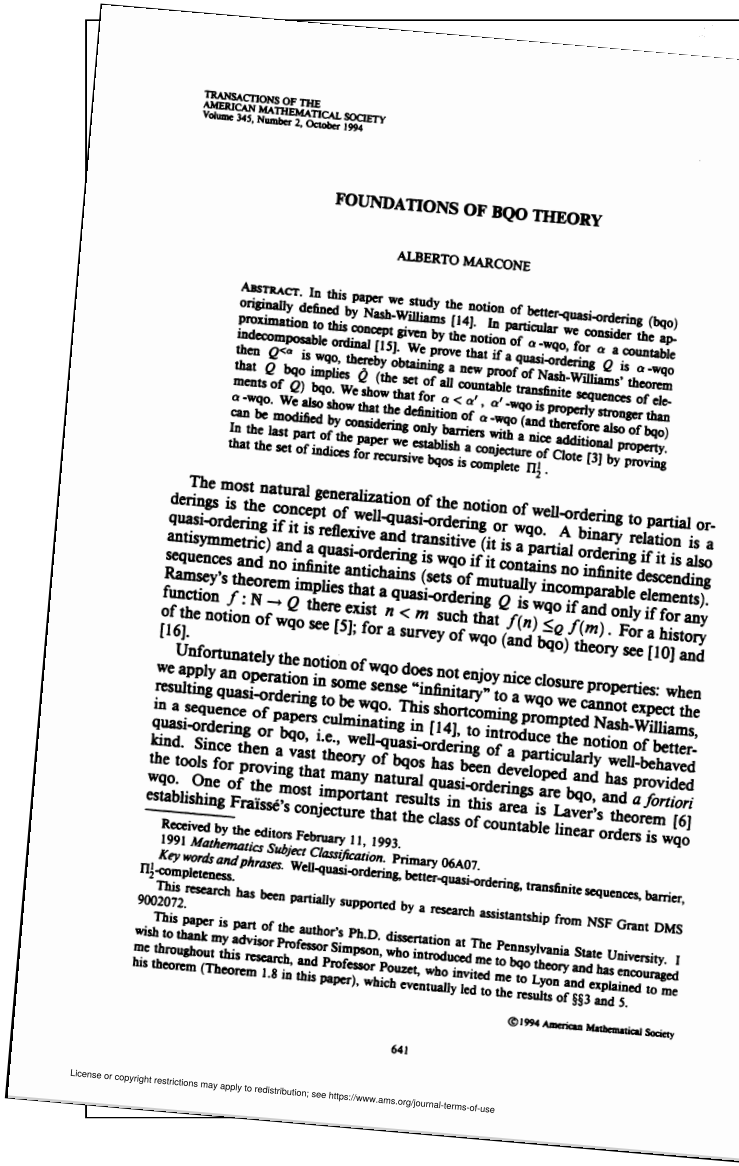
# Lifting wqos

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

$$S \subseteq_{\mathbb{H}} S' \overset{\text{def}}{\Leftrightarrow} \forall x \in S \,.\, \exists x' \in S' \,.\, x \leqslant x'$$

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

$$S \subseteq_{\mathbb{S}} S' \overset{\text{def}}{\Leftrightarrow} \forall x' \in S' \,.\, \exists x \in S \,.\, x \leqslant x'$$



TRANSACTIONS OF THE
AMERICAN MATHEMATICAL SOCIETY
Volume 345, Number 2, October 1994

FOUNDATIONS OF BQO THEORY

ALBERTO MARCONE

ABSTRACT. In this paper we study the notion of better-quasi-ordering (bqo) originally defined by Nash-Williams [14]. In particular we consider the approximation to this concept given by the notion of $\alpha$-wqo, for $\alpha$ a countable indecomposable ordinal [15]. We prove that if a quasi-ordering $Q$ is $\alpha$-wqo then $Q^{<\alpha}$ is wqo, thereby obtaining a new proof of Nash-Williams' theorem that $Q$ bqo implies $\hat{Q}$ (the set of all countable transfinite sequences of elements of $Q$) bqo. We show that for $\alpha < \alpha'$, $\alpha'$-wqo is properly stronger than $\alpha$-wqo. We also show that the definition of $\alpha$-wqo (and therefore also of bqo) can be modified by considering only barriers with a nice additional property. In the last part of the paper we establish a conjecture of Clote [3] by proving that the set of indices for recursive bqos is complete $\Pi^1_2$.

The most natural generalization of the notion of well-ordering to partial orderings is the concept of well-quasi-ordering or wqo. A binary relation is a quasi-ordering if it is reflexive and transitive (it is a partial ordering if it is also antisymmetric) and a quasi-ordering is wqo if it contains no infinite descending sequences and no infinite antichains (sets of mutually incomparable elements). Ramsey's theorem implies that a quasi-ordering $Q$ is wqo if and only if for any function $f : \mathbb{N} \to Q$ there exist $n < m$ such that $f(n) \leq_Q f(m)$. For a history of the notion of wqo see [5]; for a survey of wqo (and bqo) theory see [10] and [16].

Unfortunately the notion of wqo does not enjoy nice closure properties: when we apply an operation in some sense "infinitary" to a wqo we cannot expect the resulting quasi-ordering to be wqo. This shortcoming prompted Nash-Williams, in a sequence of papers culminating in [14], to introduce the notion of better-quasi-ordering or bqo, i.e., well-quasi-ordering of a particularly well-behaved kind. Since then a vast theory of bqos has been developed and has provided the tools for proving that many natural quasi-orderings are bqo, and *a fortiori* wqo. One of the most important results in this area is Laver's theorem [6] establishing Fraïssé's conjecture that the class of countable linear orders is wqo

Received by the editors February 11, 1993.
1991 *Mathematics Subject Classification*. Primary 06A07.
*Key words and phrases*. Well-quasi-ordering, better-quasi-ordering, transfinite sequences, barrier, $\Pi^1_2$-completeness.
This research has been partially supported by a research assistantship from NSF Grant DMS 9002072.
This paper is part of the author's Ph.D. dissertation at The Pennsylvania State University. I wish to thank my advisor Professor Simpson, who introduced me to bqo theory and has encouraged me throughout this research, and Professor Pouzet, who invited me to Lyon and explained to me his theorem (Theorem 1.8 in this paper), which eventually led to the results of §§3 and 5.

# Lifting wqos

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

$$S \subseteq_{\mathbb{H}} S' \overset{\text{def}}{\Leftrightarrow} \forall x \in S \,.\, \exists x' \in S' \,.\, x \leqslant x'$$

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

$$S \subseteq_{\mathbb{S}} S' \overset{\text{def}}{\Leftrightarrow} \forall x' \in S' \,.\, \exists x \in S \,.\, x \leqslant x'$$

ELSEVIER

Information Processing Letters 72 (1999) 155–160

A note on well quasi-orderings for powersets

Petr Jančar [1]

*Department of Computer Science, FEI, Technical University of Ostrava (VŠB), 17. listopadu 15, CZ-708 33 Ostrava-Poruba, Czech Republic*

Received 1 June 1999; received in revised form 5 November 1999
Communicated by L.A. Hemaspaandra

**Abstract**

This note characterizes those quasi-orderings $(A, \preceq)$ for which $(\mathcal{P}(A), \sqsubseteq)$ are well quasi-orderings, where $B_1 \sqsubseteq B_2$ iff $(\forall y \in B_2)(\exists x \in B_1)$: $x \preceq y$ (for $B_1, B_2 \subseteq A$). It turns out that they are those which do not contain the "Rado structure", hence are $\omega^2$-well quasi-orderings in other words. A motivation for the question has come from the area of verification of infinite-state systems, where the usefulness of well quasi-orderings has already been recognized. This note suggests that finer notions might be useful as well. In particular, $\omega^2$-well quasi-orderings illuminate a specific problem related to termination of a reachability algorithm, which has been touched on by Abdulla and Jonsson. © 1999 Elsevier Science B.V. All rights reserved.

*Keywords:* Theory of computation; Well quasi-orderings; Powersets

**1. Introduction**

When studying general reachability algorithms used in (symbolic) verification methods, Abdulla and Jonsson [2,3] touched on the problem whether (or when) termination of an algorithm using "constraints" guarantees termination of a (more efficient) implementation which uses sets of constraints; in particular, they mention the problem for "regions" and "zones" (a zone is a concise representation of a finite set of regions). In fact, the termination is tightly related to the existence of a certain well quasi-ordering, and the relevant question can be formulated in general terms as follows.

$(A, \preceq)$ is a *quasi-ordering*, *qo* for short, iff $\preceq$ is a reflexive and transitive relation on the set $A$. $(A, \preceq)$ is a *well quasi-ordering*, *wqo*, iff there is no bad sequence of elements of $A$, where a *bad sequence* is an infinite sequence $a_1, a_2, a_3, \ldots$ s.t. $\forall i, j$: $i < j \Rightarrow a_i \not\preceq a_j$. The question is whether (or when) $(A, \preceq)$ being wqo guarantees that $(\mathcal{P}(A), \sqsubseteq)$ is wqo, where $\mathcal{P}(A)$ denotes the set of subsets of $A$ and $\sqsubseteq$ is defined as follows: $B_1 \sqsubseteq B_2$ iff $(\forall y \in B_2)(\exists x \in B_1)$: $x \preceq y$.

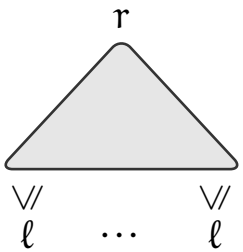
**Remark.** For a qo $(A, \preceq)$, we denote by $Eq(\preceq)$ the equivalence $\{(x, y) \in A \times A \mid x \preceq y \wedge y \preceq x\}$; the set $A$ is thus partitioned into $Eq(\preceq)$-classes. Given now a bad sequence $B_0, B_1, B_2, \ldots$ in $(\mathcal{P}(A), \sqsubseteq)$, we can replace each $B_i$ by $B'_i$ which contains just one element out of each $Eq(\preceq)$-class which has nonempty intersection with $B_i$, and then take $B''_i$ as the set of the minimal elements of $B'_i$. Obviously, $B''_0, B''_1, B''_2, \ldots$ keeps the bad sequence property, and if $(A, \preceq)$ is wqo then each $B''_i$ is *finite*. Hence our question for $\mathcal{P}(A)$ can be reduced to the question for $\mathcal{P}_{fin}(A)$ (meaning *finite* subsets).

[1] Email: Petr.Jancar@vsb.cz. The author is supported by the Grant Agency of the Czech Republic, Grant No. 201/97/0456.

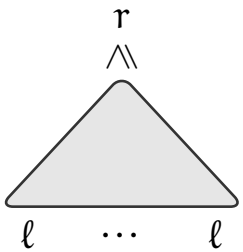

PII: S0020-0190(99)00149-0

# Compatibility for Branching Systems

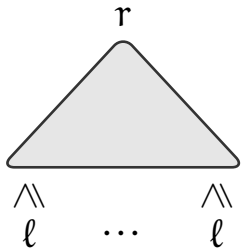
leaf coverability



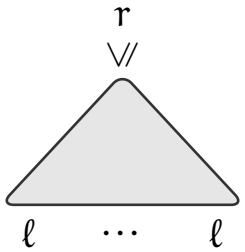

root coverability




leaf subcoverability




root subcoverability




t.-d. compatibility

$\forall x, x', S$

$x'$

$\rotatebox{90}{$\geq$}$

$x \longrightarrow S$

b.-up compatibility

$\forall x, x', S$

$x'$

$\rotatebox{90}{$\geq$}$

$x \longrightarrow S$

t.-d. downward compatibility

$\forall x, x', S$

$x \longrightarrow S$

$\rotatebox{90}{$\geq$}$

$x'$

b.-up downward compatibility

$\forall x, x', S$

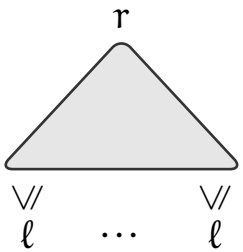
$x \longrightarrow S$

$\rotatebox{90}{$\geq$}$

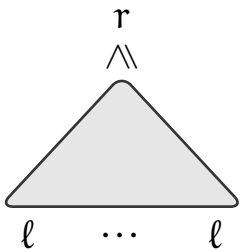
$x'$

# Compatibility for Branching Systems

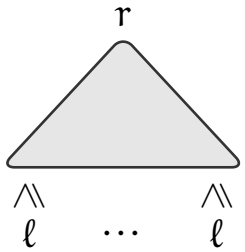
leaf coverability



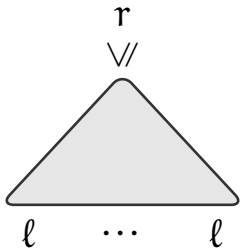

root coverability



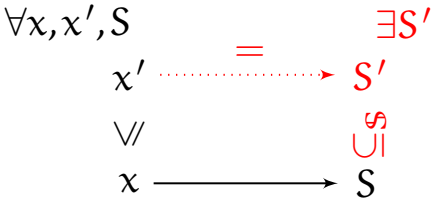

leaf subcoverability




root subcoverability




t.-d. compatibility




b.-up compatibility



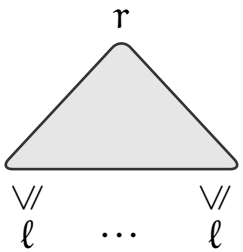

t.-d. downward compatibility



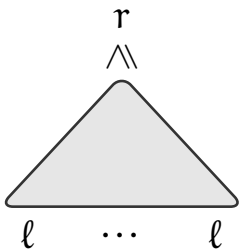

b.-up downward compatibility

# Compatibility for Branching Systems

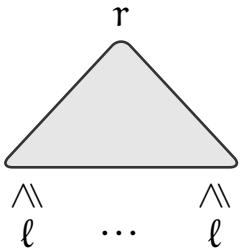
leaf coverability



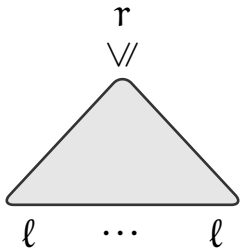

root coverability



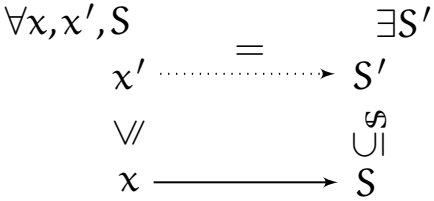

leaf subcoverability




root subcoverability




t.-d. compatibility




b.-up compatibility



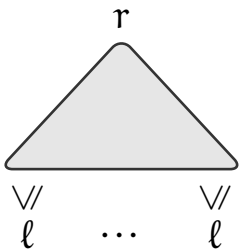

t.-d. downward compatibility



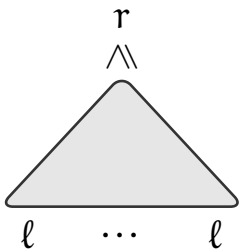

b.-up downward compatibility

# Compatibility for Branching Systems

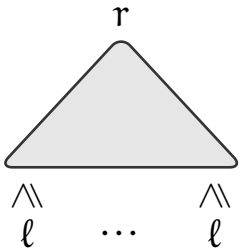
leaf coverability



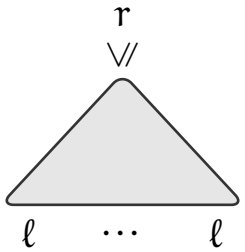

root coverability




leaf subcoverability




root subcoverability




t.-d. compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] & S' & \\ x \arrow[u, phantom, "\rotatebox{90}{$\leq$}"] \arrow[r] & S \arrow[u, phantom, "\rotatebox{90}{$\subseteq$}"] & \end{tikzcd}$$

b.-up compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] & S' & \\ x \arrow[u, phantom, "\rotatebox{90}{$\leq$}"] \arrow[r] & S \arrow[u, phantom, "\rotatebox{90}{$\subseteq$}"] & \end{tikzcd}$$

t.-d. downward compatibility

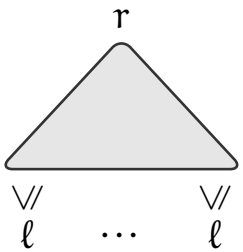
$$\begin{tikzcd} \forall x, x', S & \\ x \arrow[r] & S \\ x' \arrow[u, phantom, "\rotatebox{90}{$\leq$}"] & \end{tikzcd}$$

b.-up downward compatibility

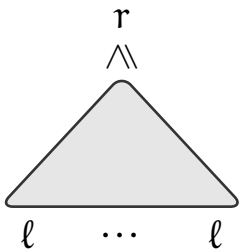
$$\begin{tikzcd} \forall x, x', S & \\ x \arrow[r] & S \\ x' \arrow[u, phantom, "\rotatebox{90}{$\leq$}"] & \end{tikzcd}$$

# Compatibility for Branching Systems

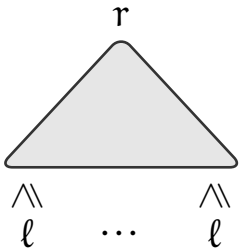
leaf coverability



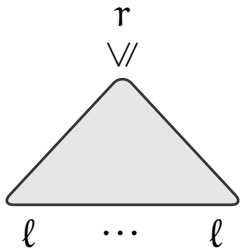

root coverability




leaf subcoverability




root subcoverability




t.-d. compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] \arrow[d, phantom, "\rotatebox{90}{$\leq$}"] & S' \arrow[d, phantom, "\subseteq_{\mathbb{S}}"] \\ x \arrow[r] & S \end{tikzcd}$$

b.-up compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] \arrow[d, phantom, "\rotatebox{90}{$\leq$}"] & S' \arrow[d, phantom, "\subseteq_{\mathbb{H}}"] \\ x \arrow[r] & S \end{tikzcd}$$

t.-d. downward compatibility

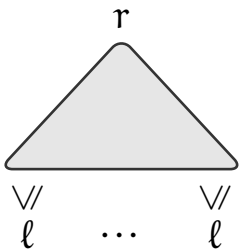
$$\begin{tikzcd} \forall x, x', S & \\ x \arrow[r] \arrow[d, phantom, "\rotatebox{90}{$\leq$}"] & S \\ x' & \end{tikzcd}$$

b.-up downward compatibility

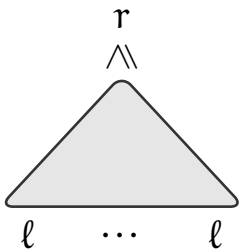
$$\begin{tikzcd} \forall x, x', S & \\ x \arrow[r] \arrow[d, phantom, "\rotatebox{90}{$\leq$}"] & S \\ x' & \end{tikzcd}$$

# Compatibility for Branching Systems

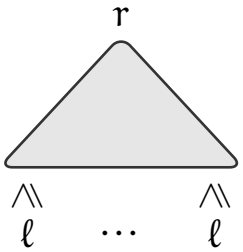
leaf coverability



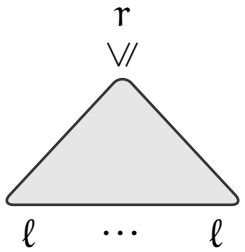

root coverability




leaf subcoverability




root subcoverability




t.-d. compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] & S' & \\ x \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r] & S \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] & \end{tikzcd}$$

b.-up compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] & S' & \\ x \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r] & S \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] & \end{tikzcd}$$

t.-d. downward compatibility

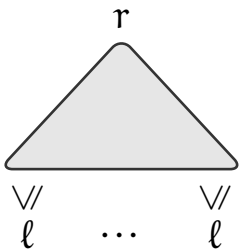
$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x \arrow[r] & S & \\ x' \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r, dotted, "="] & S' \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] & \end{tikzcd}$$

b.-up downward compatibility

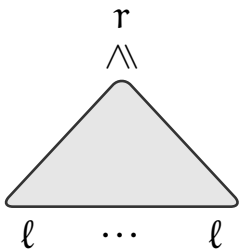
$$\begin{tikzcd} \forall x, x', S & \\ x \arrow[r] & S \\ x' \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] & \end{tikzcd}$$

# Compatibility for Branching Systems

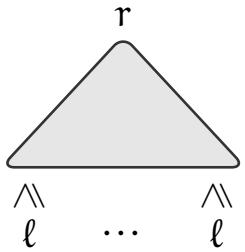
leaf coverability



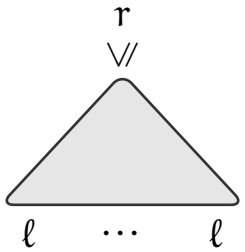

root coverability




leaf subcoverability




root subcoverability




t.-d. compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] & S' & \\ x \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r] & S \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] & \end{tikzcd}$$

b.-up compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] & S' & \\ x \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r] & S \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] & \end{tikzcd}$$

t.-d. downward compatibility

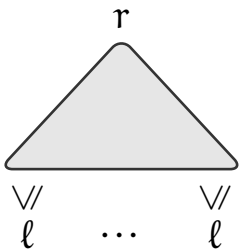
$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x \arrow[r] & S & \\ x' \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r, dotted, "="] & S' \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] & \end{tikzcd}$$

b.-up downward compatibility

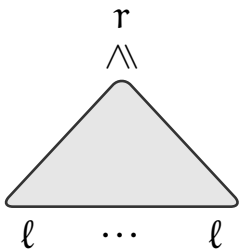
$$\begin{tikzcd} \forall x, x', S & \\ x \arrow[r] & S \\ x' \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] & \end{tikzcd}$$

# Compatibility for Branching Systems

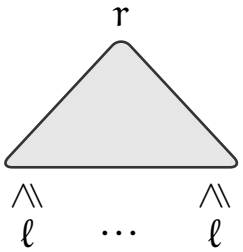
leaf coverability



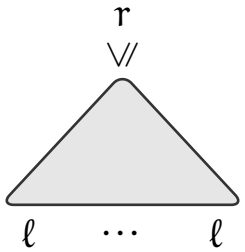

root coverability




leaf subcoverability




root subcoverability




t.-d. compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] \arrow[d, phantom, "\rotatebox{90}{$\leq$}"] & S' \arrow[d, phantom, "\rotatebox{90}{$\subseteq$}"] \\ x \arrow[r] & S \end{tikzcd}$$

b.-up compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] \arrow[d, phantom, "\rotatebox{90}{$\leq$}"] & S' \arrow[d, phantom, "\rotatebox{90}{$\subseteq$}"] \\ x \arrow[r] & S \end{tikzcd}$$

t.-d. downward compatibility

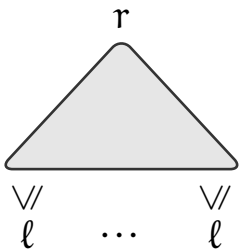
$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x \arrow[r] \arrow[d, phantom, "\rotatebox{90}{$\leq$}"] & S \arrow[d, phantom, "\rotatebox{90}{$\subseteq$}"] \\ x' \arrow[r, dotted, "="] & S' \end{tikzcd}$$

b.-up downward compatibility

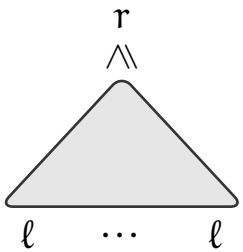
$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x \arrow[r] \arrow[d, phantom, "\rotatebox{90}{$\leq$}"] & S \arrow[d, phantom, "\rotatebox{90}{$\subseteq$}"] \\ x' \arrow[r, dotted, "="] & S' \end{tikzcd}$$

# Compatibility for Branching Systems

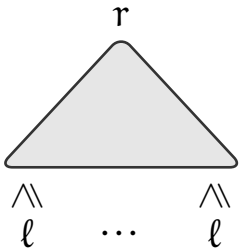
leaf coverability



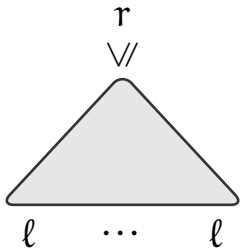

root coverability




leaf subcoverability




root subcoverability




t.-d. compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] & S' \\ x \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r] & S \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] \end{tikzcd}$$

b.-up compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x' \arrow[r, dotted, "="] & S' \\ x \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r] & S \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] \end{tikzcd}$$

t.-d. downward compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x \arrow[r] & S \\ x' \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r, dotted, "="] & S' \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] \end{tikzcd}$$

b.-up downward compatibility

$$\begin{tikzcd} \forall x, x', S & & \exists S' \\ x \arrow[r] & S \\ x' \arrow[u, phantom, "\rotatebox{-90}{$\leq$}"] \arrow[r, dotted, "="] & S' \arrow[u, phantom, "\rotatebox{-90}{$\subseteq$}"] \end{tikzcd}$$

# Lifting WSTS

[Lazić & S. '21]

## For leaf (sub)coverability

$$S \rightrightarrows T \quad \overset{\text{def}}{\Leftrightarrow} \quad T = \bigcup_{x \in S} T_x \text{ where } \forall x \in S \,.\, x \xrightarrow{=} T_x$$

- $\{r\} \rightrightarrows^* S$ whenever $S$ is a cut of an execution tree 
- leaf coverability of $\ell$ from $r$ $\Leftrightarrow$ $\{r\} \rightrightarrows^* S \supseteq_{\mathbb{S}} \{\ell\}$ 
- leaf subcoverability of $\ell$ from $r$ $\Leftrightarrow$ $\{r\} \rightrightarrows^* S \subseteq_{\mathbb{H}} \{\ell\}$ 

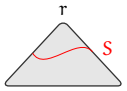
# Lifting WSTS

[Lazić & S. '21]

## For leaf (sub)coverability

$$S \rightrightarrows T \quad \overset{\text{def}}{\Leftrightarrow} \quad T = \bigcup_{x \in S} T_x \text{ where } \forall x \in S \,.\, x \overset{=}{\to} T_x$$

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




- leaf coverability of $\ell$ from $r$ $\Leftrightarrow$ $\{r\} \rightrightarrows^* S \supseteq_{\mathbb{S}} \{\ell\}$



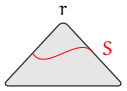

- leaf subcoverability of $\ell$ from $r$ $\Leftrightarrow$ $\{r\} \rightrightarrows^* S \subseteq_{\mathbb{H}} \{\ell\}$

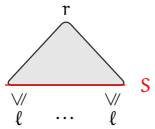
# Lifting WSTS

[Lazić & S. '21]

## For leaf (sub)coverability

$$S \rightrightarrows T \quad \overset{\text{def}}{\Leftrightarrow} \quad T = \bigcup_{x \in S} T_x \text{ where } \forall x \in S \,.\, x \overset{=}{\to} T_x$$

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

- leaf coverability of $\ell$ from $r$ $\Leftrightarrow$ $\{r\} \rightrightarrows^* S \supseteq_{\mathbb{S}} \{\ell\}$ 

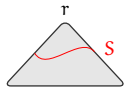
- leaf subcoverability of $\ell$ from $r$ $\Leftrightarrow$ $\{r\} \rightrightarrows^* S \subseteq_{\mathbb{H}} \{\ell\}$ 

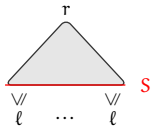
# Lifting WSTS

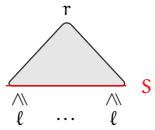
[Lazić & S. '21]

## For leaf (sub)coverability

$$S \rightrightarrows T \quad \overset{\text{def}}{\Leftrightarrow} \quad T = \bigcup_{x \in S} T_x \text{ where } \forall x \in S \,.\, x \xrightarrow{=} T_x$$

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

- leaf coverability of $\ell$ from $r$ $\Leftrightarrow$ $\{r\} \rightrightarrows^* S \supseteq_{\mathbb{S}} \{\ell\}$ 

- leaf subcoverability of $\ell$ from $r$ $\Leftrightarrow$ $\{r\} \rightrightarrows^* S \subseteq_{\mathbb{H}} \{\ell\}$ 

# Top-Down Downward Compatibility

**Lemma**

*An ordered branching system $(X, \leqslant, \to)$ has top-down downward compatibility if and only if $(\mathbb{P}(X), \subseteq_{\mathbb{H}}, \rightrightarrows)$ has downward compatibility.*

In $(X, \leqslant, \to)$:

$$\begin{tikzcd}
\forall x, x', S & & \exists S' \\
x \arrow[r] \arrow[d, phantom, "\leqslant"] & S \arrow[d, phantom, "\subseteq_{\mathbb{H}}"] & \\
x' \arrow[r, dotted, "="] & S' &
\end{tikzcd}$$

In $(\mathbb{P}_{\text{fin}}(X), \subseteq_{\mathbb{H}}, \rightrightarrows)$:

$$\begin{tikzcd}
\forall S, S', T & & \exists T' \\
S \arrow[r, shift left] \arrow[r, shift right] \arrow[d, phantom, "\subseteq_{\mathbb{H}}"] & T \arrow[d, phantom, "\subseteq_{\mathbb{H}}"] & \\
S' \arrow[r, dotted, shift left] \arrow[r, dotted, shift right] \arrow[r, phantom, "="] & T' &
\end{tikzcd}$$

# Top-Down Downward Compatibility

**Lemma**

*An ordered branching system $(X, \leqslant, \rightarrow)$ has top-down downward compatibility if and only if $(\mathbb{P}(X), \subseteq_{\mathbb{H}}, \rightrightarrows)$ has downward compatibility.*

In $(X, \leqslant, \rightarrow)$:

$$\begin{tikzcd}
\forall x, x', S & & \exists S' \\
x \arrow[r] \arrow[d, phantom, "\leqslant"] & S \arrow[d, phantom, "\subseteq_{\mathbb{H}}"] & \\
x' \arrow[r, dotted, "="] & S' &
\end{tikzcd}$$

In $(\mathbb{P}_{\text{fin}}(X), \subseteq_{\mathbb{H}}, \rightrightarrows)$:

$$\begin{tikzcd}
\forall S, S', T & & \exists T' \\
S \arrow[r, shift left] \arrow[r, shift right] \arrow[d, phantom, "\subseteq_{\mathbb{H}}"] & T \arrow[d, phantom, "\subseteq_{\mathbb{H}}"] & \\
S' \arrow[r, dotted, shift left] \arrow[r, dotted, shift right] \arrow[r, phantom, "="] & T' &
\end{tikzcd}$$

# Top-Down Downward Compatibility

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

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

$$\frac{D \vdash A \qquad D, B \vdash C}{D, D, A \to B \vdash C}\ (\to_L)$$

$\ulcorner D, A \to B \vdash C \urcorner \leqslant \ulcorner D, D, A \to B \vdash C \urcorner$

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

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

where $[\Gamma, \Delta, A \to B]$ is any multiset of formulæ embeddable into $\Gamma, \Delta, A \to B$ and with the same domain

# Top-Down Downward Compatibility

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

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

$$\frac{D \vdash A \qquad D, B \vdash C}{D, D, A \to B \vdash C}\ (\to_L)$$

$\ulcorner D, A \to B \vdash C \urcorner \leqslant \ulcorner D, D, A \to B \vdash C \urcorner$

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

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

where $[\Gamma, \Delta, A \to B]$ is any multiset of formulæ embeddable into $\Gamma, \Delta, A \to B$ and with the same domain

# Top-Down Downward Compatibility

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

$$\frac{D \vdash A \qquad D, B \vdash C}{D, D, A \to B \vdash C}\ (\to_L)$$

$\ulcorner D, A \to B \vdash C \urcorner \leqslant \ulcorner D, D, A \to B \vdash C \urcorner$

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

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

where $[\Gamma, \Delta, A \to B]$ is any multiset of formulæ embeddable into $\Gamma, \Delta, A \to B$ and with the same domain

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

# Top-Down Downward Compatibility

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

$$\frac{D \vdash A \qquad D, B \vdash C}{D, D, A \to B \vdash C}\ (\to_L)$$

$\ulcorner D, A \to B \vdash C \urcorner \leqslant \ulcorner D, D, A \to B \vdash C \urcorner$

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

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

where $[\Gamma, \Delta, A \to B]$ is any multiset of formulæ embeddable into $\Gamma, \Delta, A \to B$ and with the same domain

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

# Length Function Theorems for Powersets

[Abriola, Figueira & Senno '15; Balasubramanian '20]

**Theorem** (Balasubramanian '20)

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

- $L_{g,X}(n) \leqslant F_{\omega^{d-1}+1}(n)$ *for the Hoare quasi-ordering* $\subseteq_{\mathbb{H}}$,
- $L_{g,X}(n) \leqslant F_{\omega^{d-1}.2^d+1}(n)$ *for the Smyth quasi-ordering* $\subseteq_{\mathbb{S}}$.

Yields $\mathbf{F}_{\omega^\omega}$ upper bounds for

- satisfiability of Forward XPath$^{\varepsilon}$ and
- provability in $\mathbf{R}_{\to,\wedge}$.

Theoretical Computer Science 603 (2015) 3–22

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Linearizing well quasi-orders and bounding the length of bad sequences

Sergio Abriola [a], Santiago Figueira [a,b,*], Gabriel Senno [a]

[a] Departamento de Computación, FCEN, Universidad de Buenos Aires, Argentina
[b] CONICET, Argentina

ARTICLE INFO

*Article history:*
Received 11 February 2013
Accepted 27 November 2014
Available online 15 July 2015

*Keywords:*
Well quasi-order
Controlled bad sequence
Lexicographic ordering
Product ordering
Multiset ordering
Majoring ordering
Fast Growing Hierarchy

ABSTRACT

We study the length functions of controlled bad sequences over some wel
(wqo's) and classify them in the Fast Growing Hierarchy. We develop a
contained study of the length of bad sequences over the disjoint product in
Lemma), which leads to recently discovered upper bounds but through a simp
We also give a tight upper bound for the length of controlled decreasing
multisets of $\mathbb{N}^n$ with the underlying lexicographic ordering, and use it to g
bound for the length of controlled bad sequences in the majoring orderi
underlying disjoint product ordering. We apply this last result to attain comp
bounds for the emptiness problem of RTCA and ATRA automata.
For the case of the product and majoring wqo's the idea is to *linearize* bad se
to transform a *bad* sequence over a wqo into a *decreasing* one over a well-orde
upper bounds can be more easily handled.



**1. Well quasi-orders and bad sequences**

A quasi-order is a binary relation $\leq$ over a given set $A$ that is reflexive and transitive. A sequence $\mathbf{x} = x_0, x_1, \ldots$
elements of $A$ is called *good* if there are $i < j$ such that $x_i \leq x_j$. A sequence is *bad* if it is not good. A *well quasi-orde*
is a quasi-order where all infinite sequences are good, or, equivalently, all bad sequences are finite.
The theory of well quasi-orderings was initially developed by Higman [1] (under the name of "finite basis pro
and by Erdös and Rado in an unpublished manuscript, although some early evidence of the theory had already appe
a work by Neumann [2]. Further developments were [3–5], and more recently [6]. Wqo's have become a key ingrec
a great number of results related with decidability, finiteness, and regularity results that appear in areas such as te
tion proofs for rewriting systems [7,8], their extensions [9,10], complexity upper bounds [11,12], well-structured tra
systems [13–16], etc.
From the analysis of a termination proof of a given algorithm $\mathcal{S}$, whose correctness is grounded in the analysi
certain wqo, one may extract a computational complexity upper bound for $\mathcal{S}$. Roughly, the idea is that any sequer
successive configurations of $\mathcal{S}$ (with a given input) is transformed into a bad sequence in the wqo. Thus, having an
bound for the length of the bad sequence entails an upper bound for the number of steps that the algorithm nee
terminate.

* Corresponding author.
*E-mail address:* santiago@dc.uba.ar (S. Figueira).

http://dx.doi.org/10.1016/j.tcs.2015.07.012
0304-3975/

# Length Function Theorems for Powersets

[Abriola, Figueira & Senno '15; Balasubramanian '20]

**Theorem** (Balasubramanian '20)

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

- $L_{g,X}(n) \leqslant F_{\omega^{d-1}+1}(n)$ *for the Hoare quasi-ordering* $\subseteq_{\mathbb{H}}$,
- $L_{g,X}(n) \leqslant F_{\omega^{d-1}.2^d+1}(n)$ *for the Smyth quasi-ordering* $\subseteq_{\mathbb{S}}$.

Yields $\mathbf{F}_{\omega^\omega}$ upper bounds for

- satisfiability of Forward XPath$^{\varepsilon}$ and
- provability in $\mathbf{R}_{\rightarrow,\wedge}$.

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

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

**Abstract**

We provide upper and lower bounds for the length of controlled bad sequences over the majoring and the minoring orderings of finite sets of $\mathbb{N}^d$. The results are obtained by bounding the length of such sequences by functions from the Cichon hierarchy. This allows us to translate these results to bounds over the fast-growing complexity classes.

The obtained bounds are proven to be tight for the majoring ordering, which solves a problem left open by Abriola, Figueira and Senno (Theor. Comp. Sci, Vol. 603). Finally, we use the results on controlled bad sequences to prove upper bounds for the emptiness problem of some classes of automata.

**CCS Concepts:** • **Theory of computation → Complexity classes**; *Program verification.*

**Keywords:** well-quasi orders, controlled bad sequences, majoring and minoring ordering

**ACM Reference Format:**
A. R. Balasubramanian. 2020. Complexity of controlled bad sequences over finite sets of $\mathbb{N}^d$. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3373718.3394753

**1 Introduction**

A well-quasi order (wqo) over a set $A$ is a reflexive and transitive relation $\leq_A$ such that every infinite sequence $x_0, x_1, x_2, \ldots$ over $A$ has an increasing pair $x_i \leq_A x_j$ with $i < j$. A normed wqo (nwqo) is a wqo $(A, \leq_A)$ which has a *norm* function $|\cdot| : A \to \mathbb{N}$ such that the pre-image of $n$ under $|\cdot|$ is finite for every $n$. A sequence over a wqo is called a bad sequence

*Work done when the author was an intern at LSV, ENS-Saclay Paris

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*LICS '20, July 8–11, 2020, Saarbrücken, Germany*
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7104-9/20/07…$15.00
https://doi.org/10.1145/3373718.3394753

if it contains no increasing pair. Hence, all bad over a well-quasi order are necessarily finite.

Well-quasi orders are an important tool in log natorics and computer science as evidenced by ti cations in term-rewriting systems [8], algorithm and verification of infinite state systems [1, 2, 12, well-quasi orders form the backbone for the ubiquit *structured transition systems* (wsts) [1, 12], whose cov problem is shown to be decidable thanks to well-quas

In recent years, significant effort has been put in to stand the complexity of the coverability procedure fo ous well-structured transition systems (See [4, 6, 11, and also [19] for a catalogue of many problems). The ke behind proving upper bounds for the coverability algor is the following: For a given class of wsts, the running of the coverability procedure for that class can be bour by the length of *controlled bad sequences* of the underly normed well-quasi order (See definition 2.4 for a formal nition of controlled bad sequences). Intuitively, for a fu tion $g$ and a number $n$, a sequence $x_0, x_1, x_2, \ldots, x_l$ is calle $(g, n)$-controlled bad sequence if $|x_0| \leq n, |x_1| \leq g(n), |x_2|$ $g(g(n))$ and so on. A simple application of Konig's lemma ca be used to show that for every $n$, there is a $(g, n)$-controlle bad sequence of maximum length. Hence, for every functio $g$ we can define a *length function* which maps a number $n$ to the length of the longest $(g, n)$-controlled bad sequence.

The main observation made in [4, 6, 11, 17, 19, 20] is that, for various classes of well-structured systems, an upper bound on the running time of the coverability procedure could be obtained by bounding the length function of some specific $g$ over the underlying wqo of that class. Motivated by this, upper bounds on the length of controlled bad sequences have been obtained for various well-quasi orders: The product ordering over $\mathbb{N}^d$ ([4]), the lexicographic ordering over $\mathbb{N}^d$ ([11]), the multiset ordering over multisets of $\mathbb{N}^d$ ([4]), the subword ordering over words [20] and the linear ordering over ordinals [18], to name a few. Using these results, time bounds have been established for the following problems (See [19] for a more detailed overview): coverability of lossy counter machines, coverability and termination of lossy channel systems, coverability of unordered data nets, emptiness of alternating 1-register and 1-clock automata, the

# Length Function Theorems for Powersets

[Abriola, Figueira & Senno '15; Balasubramanian '20]

**Theorem** (Balasubramanian '20)

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

- $L_{g,X}(n) \leqslant F_{\omega^{d-1}+1}(n)$ *for the Hoare quasi-ordering* $\subseteq_{\mathbb{H}}$,
- $L_{g,X}(n) \leqslant F_{\omega^{d-1}.2^d+1}(n)$ *for the Smyth quasi-ordering* $\subseteq_{\mathbb{S}}$.

Yields $\mathbf{F}_{\omega^\omega}$ upper bounds for

- satisfiability of Forward XPath$^{\varepsilon}$ and
- provability in $\mathbf{R}_{\rightarrow,\wedge}$.

Complexity of controlled bad sequences over sets of $\mathbb{N}^d$

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

Abstract

We provide upper and lower bounds for the length of controlled bad sequences over the majoring and the minoring orderings of finite sets of $\mathbb{N}^d$. The results are obtained by bounding the length of such sequences by functions from the Cichon hierarchy. This allows us to translate these results to bounds over the fast-growing complexity classes.

The obtained bounds are proven to be tight for the majoring ordering, which solves a problem left open by Abriola, Figueira and Senno (Theor. Comp. Sci, Vol. 603). Finally, we use the results on controlled bad sequences to prove upper bounds for the emptiness problem of some classes of automata.

CCS Concepts: • Theory of computation → Complexity classes; Program verification.

Keywords: well-quasi orders, controlled bad sequences, majoring and minoring ordering

ACM Reference Format:
A. R. Balasubramanian. 2020. Complexity of controlled bad sequences over finite sets of $\mathbb{N}^d$. In Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20), July 8–11, 2020, Saarbrücken, Germany. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3373718.3394753

1 Introduction

A well-quasi order (wqo) over a set $A$ is a reflexive and transitive relation $\leq_A$ such that every infinite sequence $x_0, x_1, x_2, \ldots$ over $A$ has an increasing pair $x_i \leq_A x_j$ with $i < j$. A normed wqo (nwqo) is a wqo $(A, \leq_A)$ which has a *norm* function $|\cdot| : A \to \mathbb{N}$ such that the pre-image of $n$ under $|\cdot|$ is finite for every $n$. A sequence over a wqo is called a bad sequence if it contains no increasing pair. Hence, all bad over a well-quasi order are necessarily finite.

Well-quasi orders are an important tool in log natorics and computer science as evidenced by ti cations in term-rewriting systems [8], algorithm and verification of infinite state systems [1, 2, 12, well-quasi orders form the backbone for the ubiquit *structured transition systems* (wsts) [1, 12], whose cov

In recent years, significant effort has been put in to stand the complexity of the coverability procedure fo ous well-structured transition systems (See [4, 6, 11, and also [19] for a catalogue of many problems). The ke behind proving upper bounds for the coverability algo is the following: For a given class of wsts, the running of the coverability procedure for that class can be bour by the length of *controlled bad sequences* of the underly normed well-quasi order (See definition 2.4 for a formal nition of controlled bad sequences). Intuitively, for a fu tion $g$ and a number $n$, a sequence $x_0, x_1, x_2, \ldots, x_l$ is calle $(g, n)$-controlled bad sequence if $|x_0| \leq n, |x_1| \leq g(n), |x_2|$ $g(g(n))$ and so on. A simple application of Konig's lemma ca be used to show that for every $n$, there is a $(g, n)$-controlle bad sequence of maximum length. Hence, for every functio $g$ we can define a *length function* which maps a number $n$ to the length of the longest $(g, n)$-controlled bad sequence.

The main observation made in [4, 6, 11, 17, 19, 20] is that, for various classes of well-structured systems, an upper bound on the running time of the coverability procedure could be obtained by bounding the length function of some specific $g$ over the underlying wqo of that class. Motivated by this, upper bounds on the length of controlled bad sequences have been obtained for various well-quasi orders: The product ordering over $\mathbb{N}^d$ ([4]), the lexicographic ordering over $\mathbb{N}^d$ ([11]), the multiset ordering over multisets of $\mathbb{N}^d$ ([4]), the subword ordering over words [20] and the linear ordering over ordinals [18], to name a few. Using these results, time bounds have been established for the following problems (See [19] for a more detailed overview): coverability of lossy counter machines, coverability and termination of lossy channel systems, coverability of unordered data nets, emptiness of alternating 1-register and 1-clock automata, the

*Work done when the author was an intern at LSV, ENS-Saclay Paris

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
LICS '20, July 8–11, 2020, Saarbrücken, Germany
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7104-9/20/07…$15.00
https://doi.org/10.1145/3373718.3394753

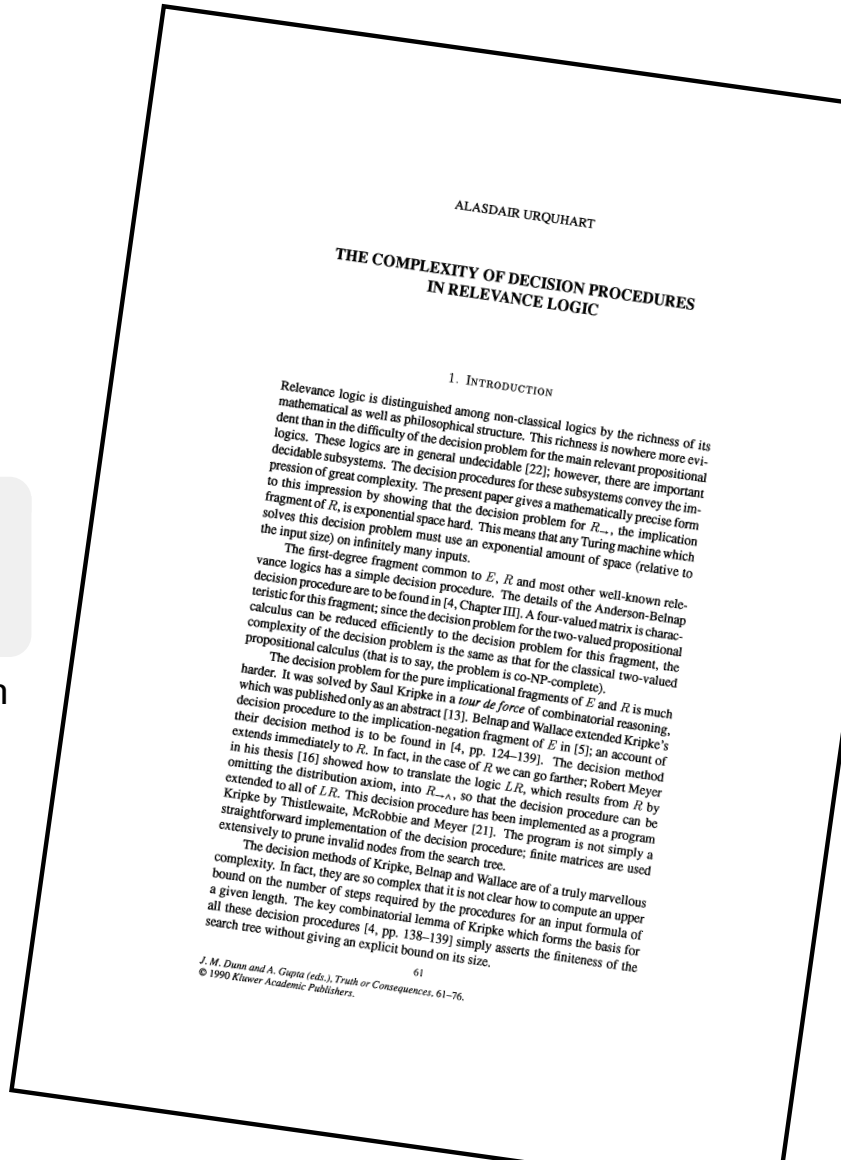
# Complexity Upper Bounds

[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]



ALASDAIR URQUHART

THE COMPLEXITY OF DECISION PROCEDURES IN RELEVANCE LOGIC

1. Introduction

Relevance logic is distinguished among non-classical logics by the richness of its mathematical as well as philosophical structure. This richness is nowhere more evident than in the difficulty of the decision problem for the main relevant propositional logics. These logics are in general undecidable [22]; however, there are important decidable subsystems. The decision procedures for these subsystems convey the impression of great complexity. The present paper gives a mathematically precise form to this impression by showing that the decision problem for $R_{\to}$, the implication fragment of $R$, is exponential space hard. This means that any Turing machine which solves this decision problem must use an exponential amount of space (relative to the input size) on infinitely many inputs.

The first-degree fragment common to $E$, $R$ and most other well-known relevance logics has a simple decision procedure. The details of the Anderson-Belnap decision procedure are to be found in [4, Chapter III]. A four-valued matrix is characteristic for this fragment; since the decision problem for the two-valued propositional calculus can be reduced efficiently to the decision problem for this fragment, the complexity of the decision problem is the same as that for the classical two-valued propositional calculus (that is to say, the problem is co-NP-complete).

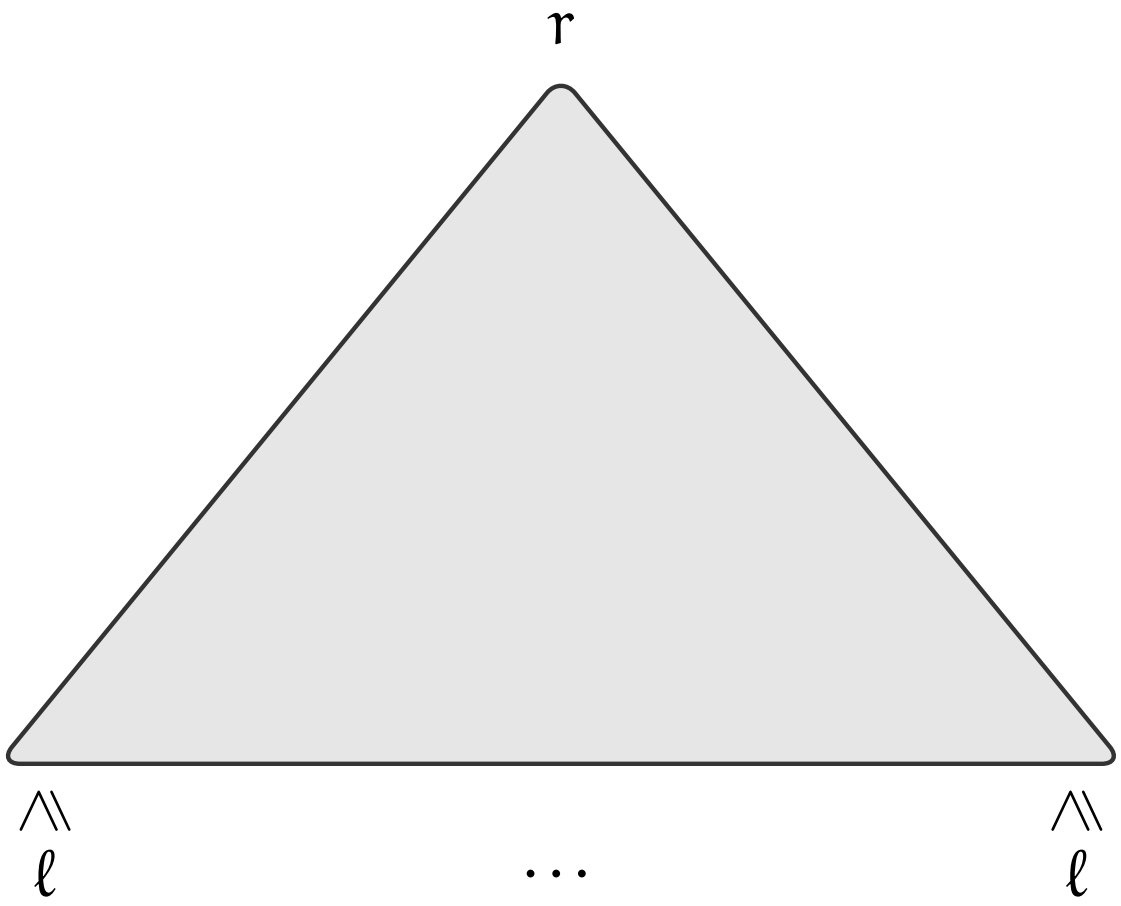
The decision problem for the pure implicational fragments of $E$ and $R$ is much harder. It was solved by Saul Kripke in a *tour de force* of combinatorial reasoning, which was published only as an abstract [13]. Belnap and Wallace extended Kripke's decision procedure to the implication-negation fragment of $E$ in [5]; an account of their decision method is to be found in [4, pp. 124–139]. The decision method extends immediately to $R$. In fact, in the case of $R$ we can go farther; Robert Meyer in his thesis [16] showed how to translate the logic $LR$, which results from $R$ by omitting the distribution axiom, into $R_{\to\wedge}$, so that the decision procedure can be extended to all of $LR$. This decision procedure has been implemented as a program Kripke by Thistlewaite, McRobbie and Meyer [21]. The program is not simply a straightforward implementation of the decision procedure; finite matrices are used extensively to prune invalid nodes from the search tree.

The decision methods of Kripke, Belnap and Wallace are of a truly marvellous complexity. In fact, they are so complex that it is not clear how to compute an upper bound on the number of steps required by the procedures for an input formula of a given length. The key combinatorial lemma of Kripke which forms the basis for all these decision procedures [4, pp. 138–139] simply asserts the finiteness of the search tree without giving an explicit bound on its size.

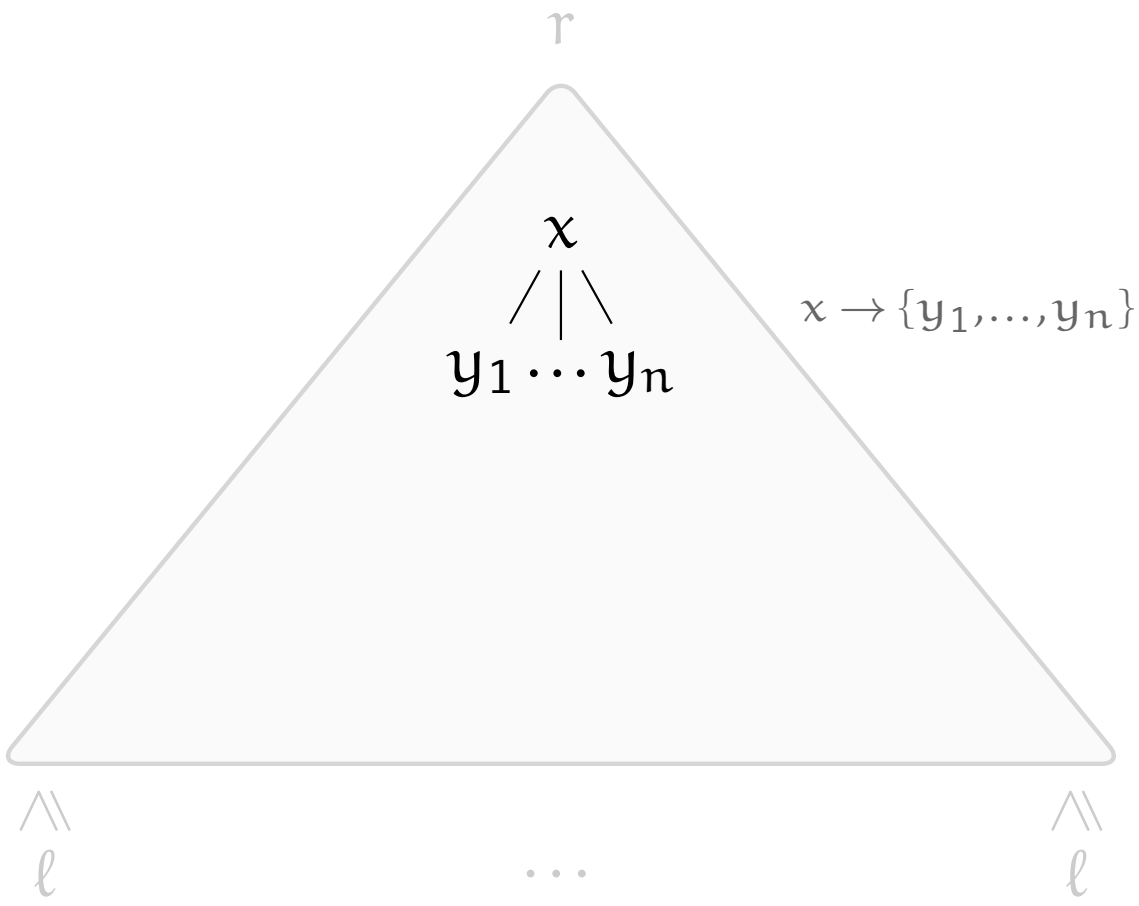
61

J. M. Dunn and A. Gupta (eds.), Truth or Consequences, 61–76.
© 1990 Kluwer Academic Publishers.

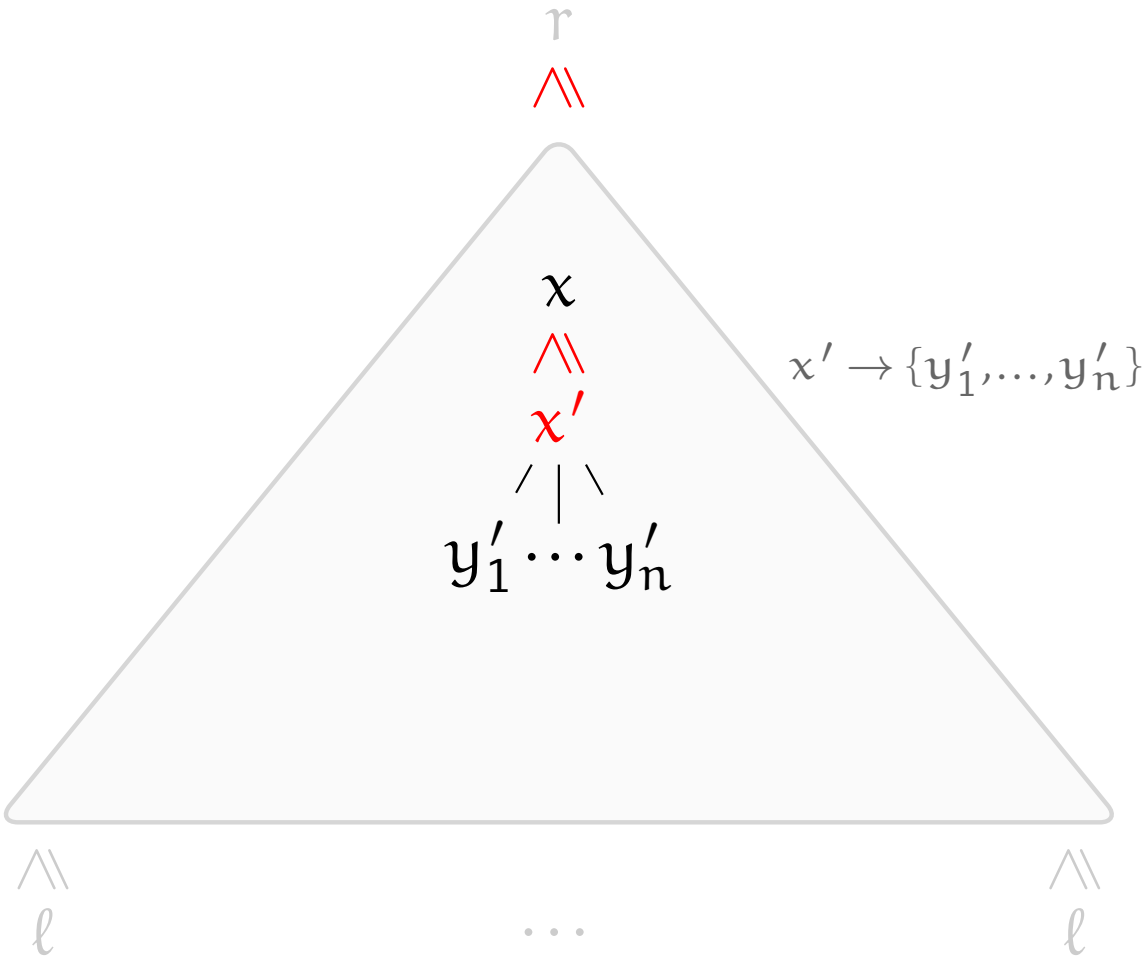
# Witnesses



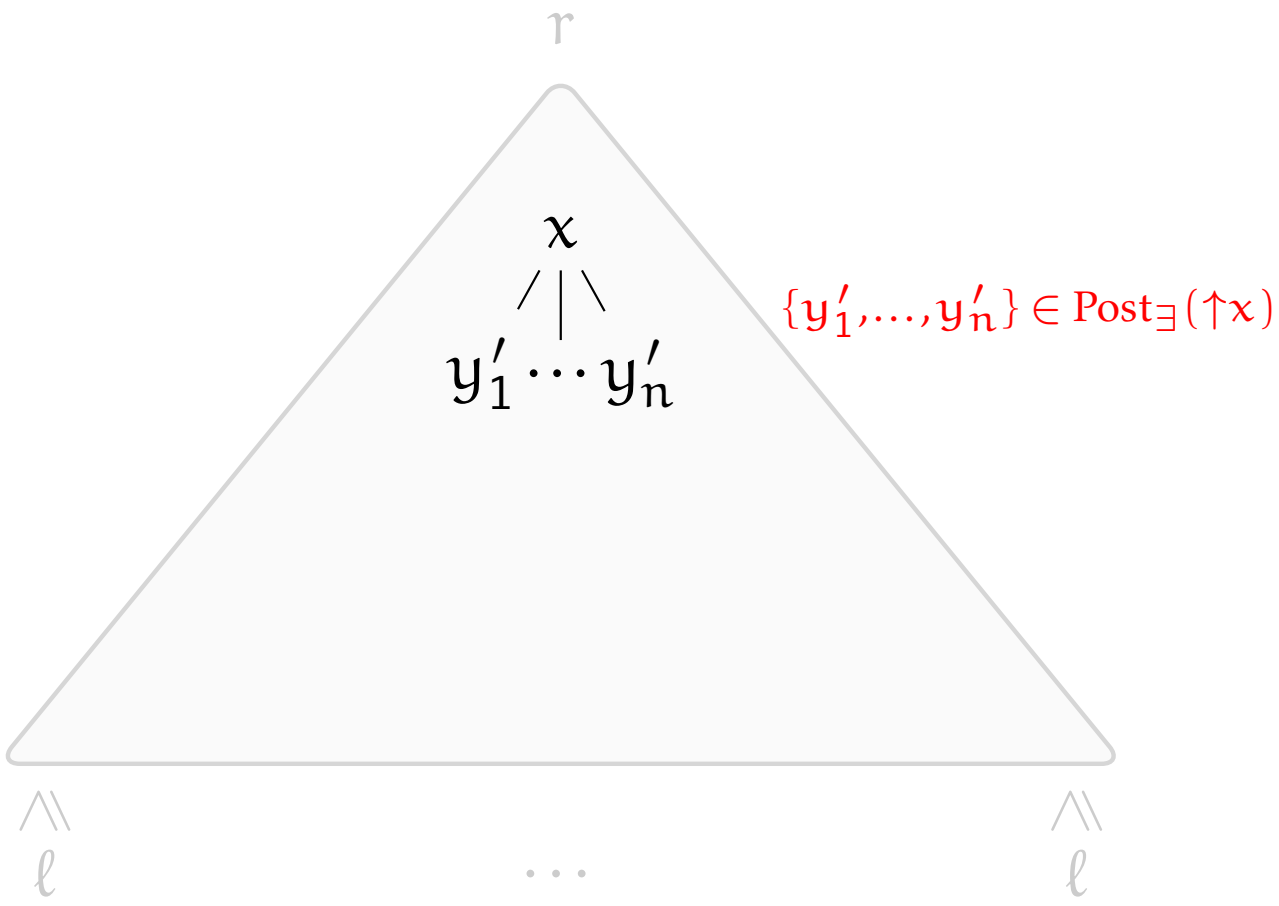
# Witnesses



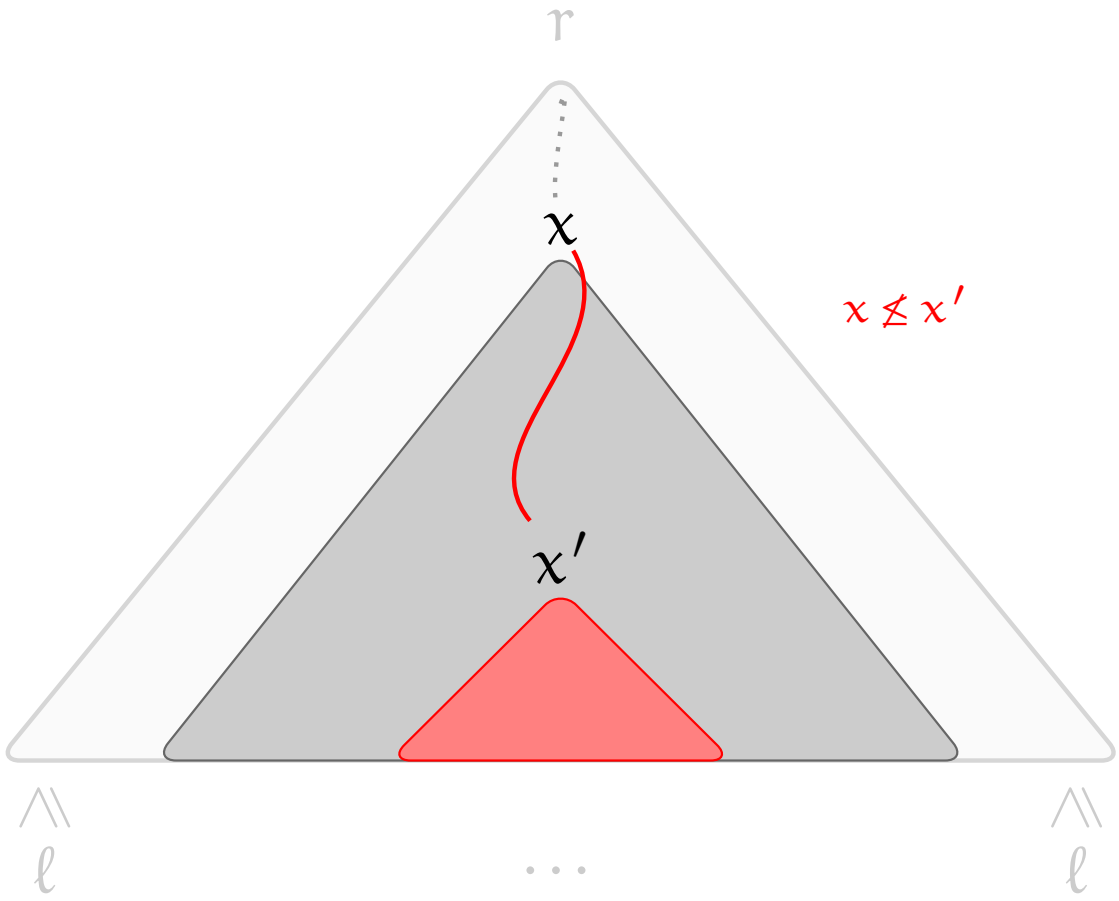
# Pseudo Witnesses



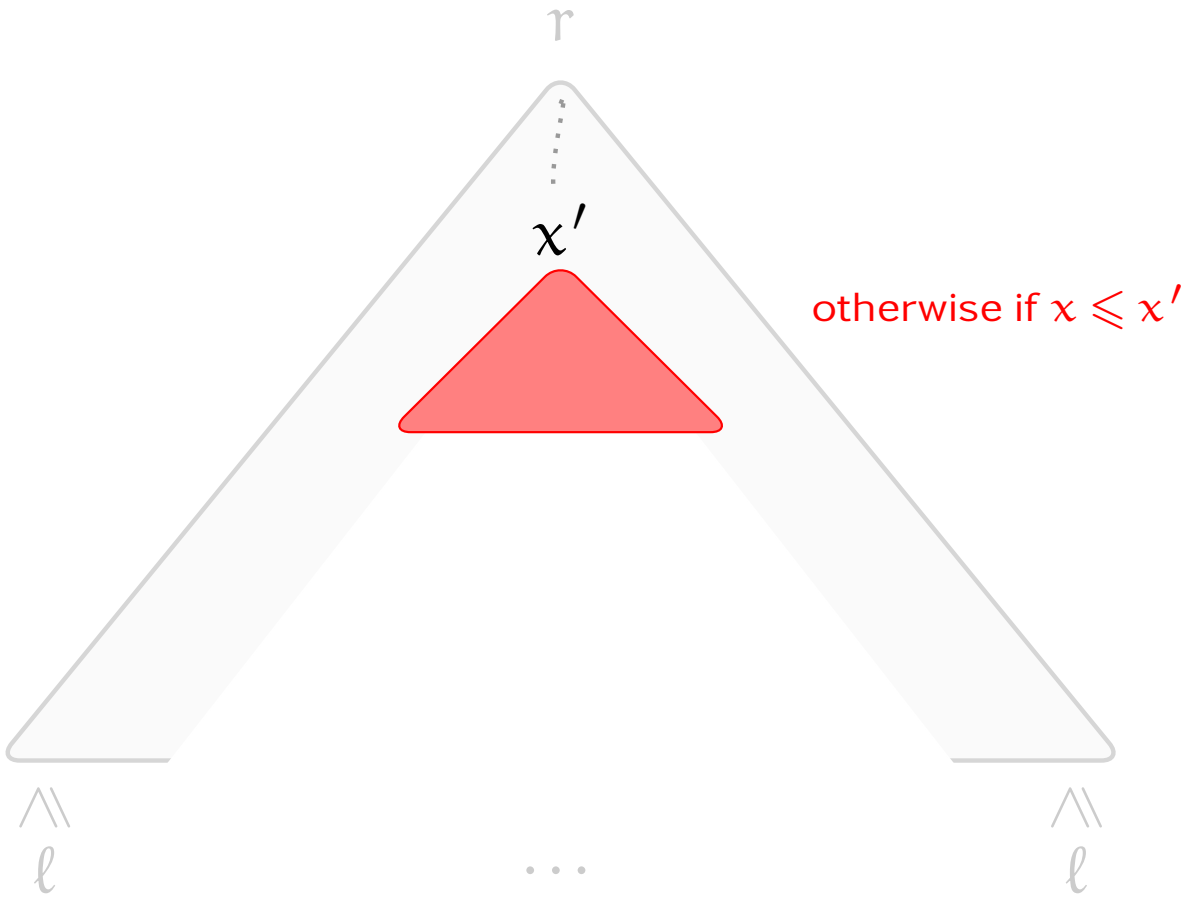
# Pseudo Witnesses



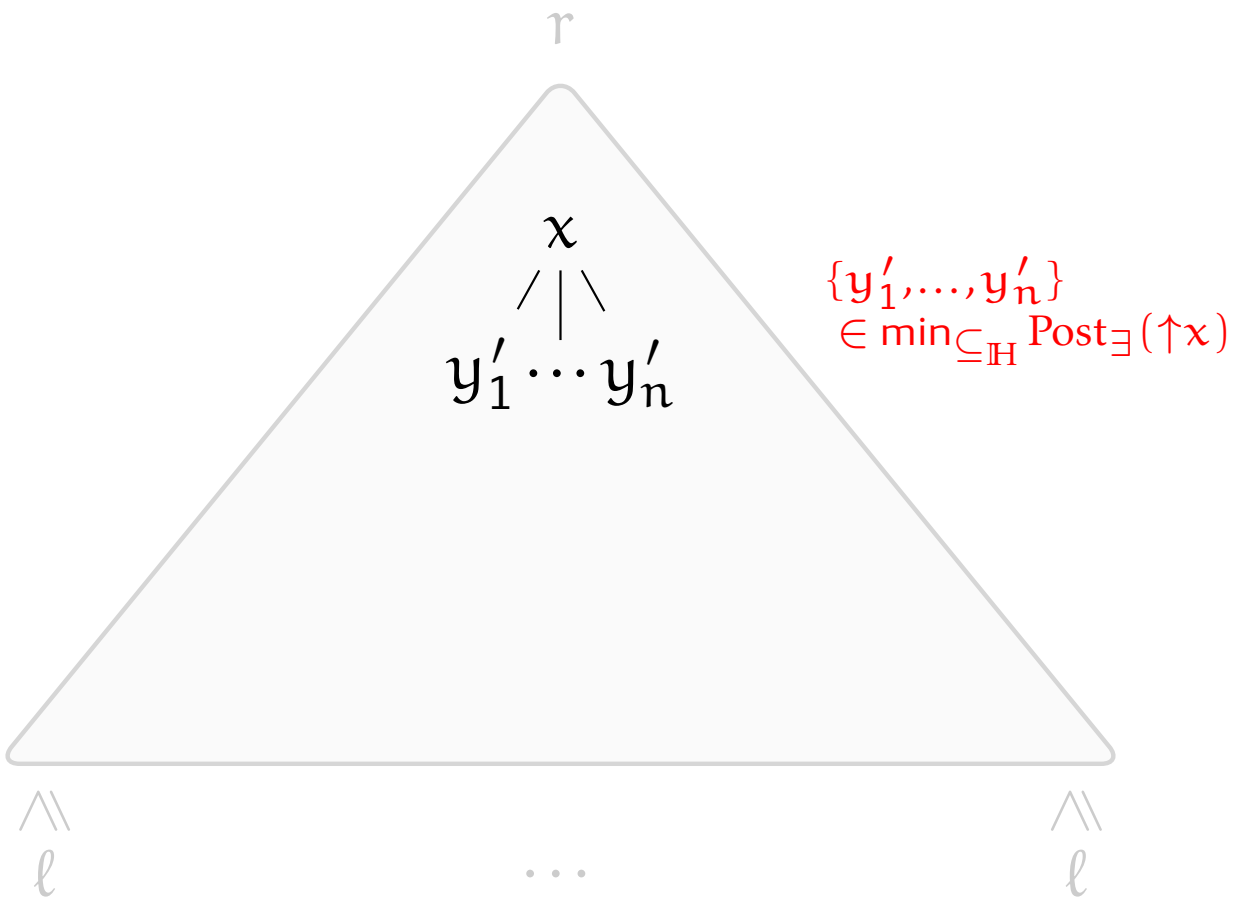
# Irredundant Pseudo Witnesses



# Irredundant Pseudo Witnesses



# Irredundant Pseudo Witnesses



# Irredundant Pseudo Witnesses

- there exists an irredundant pseudo witness iff subcoverability holds
- the sequences $r = x_0, x_1, \ldots, x_m \leqslant \ell$ along the branches are bad
- because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq_{\mathbb{H}}} \mathrm{Post}_{\exists}(\uparrow x_i)$, we may be able to control those sequences
- also need to control the "and-branching degree", e.g. if $x \to S$ then $|S| \leqslant f(\|x\|)$ 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 $r = x_0, x_1, \ldots, x_m \leqslant \ell$ along the branches are bad
- because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq_{\mathbb{H}}} \mathrm{Post}_{\exists}(\uparrow x_i)$, we may be able to control those sequences
- also need to control the "and-branching degree", e.g. if $x \to S$ then $|S| \leqslant f(\|x\|)$ 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 $r = x_0, x_1, \ldots, x_m \leqslant \ell$ along the branches are bad
- because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq_{\mathbb{H}}} \mathrm{Post}_{\exists}(\uparrow x_i)$, we may be able to control those sequences
- also need to control the "and-branching degree", e.g. if $x \to S$ then $|S| \leqslant f(\|x\|)$ 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 $r = x_0, x_1, \ldots, x_m \leqslant \ell$ along the branches are bad
- because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq_{\mathbb{H}}} \mathrm{Post}_{\exists}(\uparrow x_i)$, we may be able to control those sequences
- also need to control the "and-branching degree", e.g. if $x \to S$ then $|S| \leqslant f(\|x\|)$ 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 $r = x_0, x_1, \ldots, x_m \leqslant \ell$ along the branches are bad
- because $x_{i+1}$ is in some $S_{i+1} \in \min_{\subseteq_{\mathbb{H}}} \mathrm{Post}_{\exists}(\uparrow x_i)$, we may be able to control those sequences
- also need to control the "and-branching degree", e.g. if $x \to S$ then $|S| \leqslant f(\|x\|)$ for some reasonable $f$
- length function theorems then allow to bound the height and degree of irredundant pseudo witnesses, thus their size

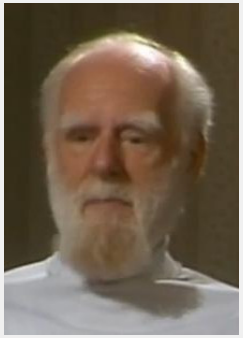
# Concluding Remarks

- 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

# Examples of WQOs

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

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



$$a_1 \cdots a_m \sqsubseteq b_1 \cdots b_n \overset{\text{def}}{\Leftrightarrow} \begin{cases} \exists 1 \leqslant i_1 < \cdots < i_m \leqslant n, \\ \bigwedge_{j=1}^{m} a_j \leqslant_X b_{i_j} \ . \end{cases}$$

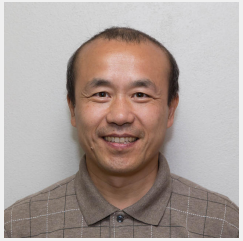
**Example**

$$aba \sqsubseteq baaacabbab$$

# Examples of WQOs

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

*For all* $k$, $(\mathrm{Graphs} \setminus \uparrow P_k, \subseteq)$ *and* $(\mathrm{Graphs} \setminus \uparrow P_k, \subseteq_i)$ *are wqo.*



## Non-Examples