On the Complexity of VAS Reachability

Sylvain Schmitz

based on joint works with D. Figueira, S. Figueira, J. Leroux, and Ph. Schnoebelen

Séminaire LIGM

Outline

well-quasi-orders (wqo):

- proving algorithm termination

a toolbox for wqo complexity

- upper bounds
- lower bounds
- complexity classes

this talk: focus on one problem

- reachability in vector addition systems

Outline

well-quasi-orders (wqo):

- proving algorithm termination

a toolbox for wqo complexity

- upper bounds
- lower bounds
- complexity classes

this talk: focus on one problem

- reachability in vector addition systems

Outline

well-quasi-orders (wqo):

- proving algorithm termination

a toolbox for wqo complexity

- upper bounds
- lower bounds
- complexity classes

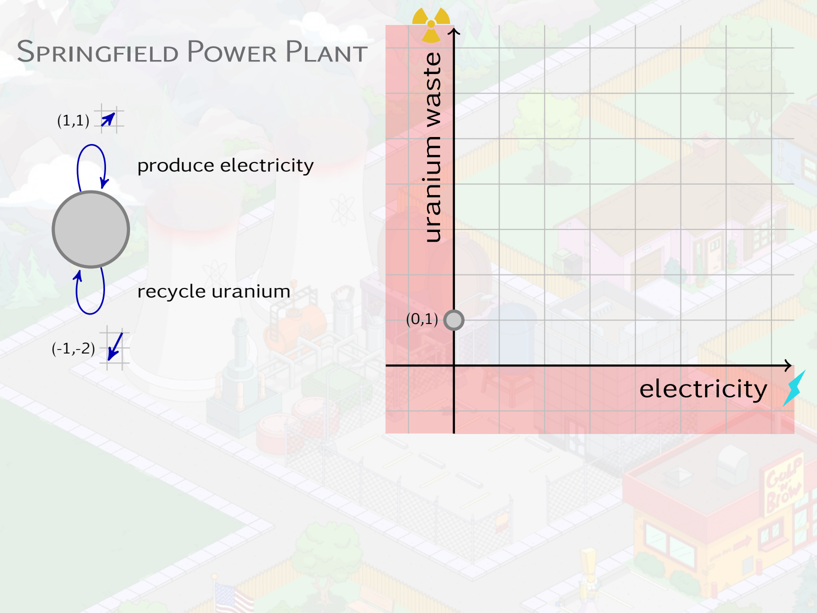
this talk: focus on one problem

- reachability in vector addition systems

Vector Addition Systems

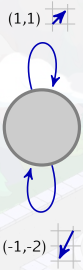
Vector Addition Systems

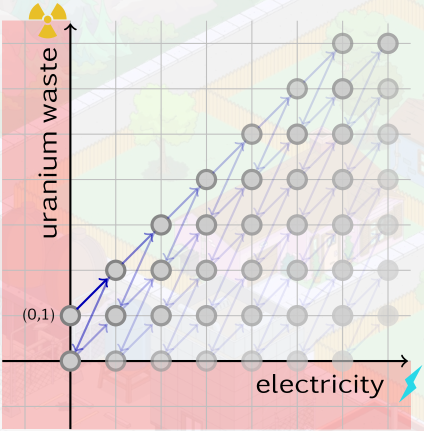
Vector Addition Systems



Vector Addition Systems

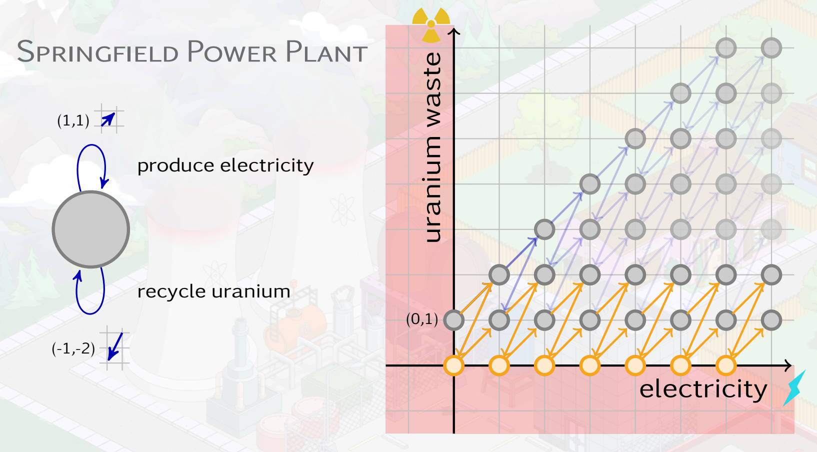
Springfield Power Plant





Can we produce unbounded electricity with no left-over uranium waste?

Vector Addition Systems



Can we produce unbounded electricity with no left-over uranium waste? Yes, $(\infty,0)$ is reachable

Importance of the Problem

Reachability Problem

input: *a vector addition system and two configurations* **source** *and* **target**

question: **source** $\rightarrow^*$ **target***?*

Importance of the Problem

Discrete Resources

- modelling: items, money, energy, molecules, ...
- distributed computing: active threads in thread pool
- data: isomorphism types in data logics and data-centric systems

Importance of the Problem

Central Decision Problem [S.'16]

Large number of problems interreducible with reachability in vector addition systems

The Complexity of Reachability in Vector Addition Systems

SYLVAIN SCHMITZ, LSV, ENS Cachan & CNRS & INRIA, Université Paris-Saclay

The program of the 30th Symposium on Logic in Computer Science held in 2015 in Kyoto included two contributions on the computational complexity of the *reachability problem* for vector addition systems: Blondin, Finkel, Göller, Haase, and McKenzie [2015] attacked the problem by providing the first tight complexity bounds in the case of dimension 2 systems with states, while Leroux and Schmitz [2015] proved the first complexity upper bound in the general case. The purpose of this column is to present the main ideas behind these two results, and more generally survey the current state of affairs.

1. INTRODUCTION

Vector addition systems with states (VASS), or equivalently Petri nets, find a wide range of applications in the modelling of concurrent, chemical, biological, or business processes. Maybe more importantly for this column, their algorithmics, and in particular the decidability of their *reachability problem* [Mayr 1981; Kosaraju 1982; Lambert 1992; Leroux 2011], is the cornerstone of many decidability results in logic, automata, verification, etc.—see Section 5 for a few examples.

In spite of its importance, fairly little is known about the computational complexity of the reachability problem. Regarding the general case, the inclusive surveys on the complexity of decision problems on VASS by Esparza and Nielsen [1994] and Esparza [1998] could only point to the EXPSPACE lower bound of Lipton [1976] and to the fact that the running time of the known algorithms is not primitive recursive: *no* complexity upper bound was known, besides decidability first proven in 1981 by Mayr. When turning to restricted versions of the problem, the 2-dimensional case was only known to be in 2-EXP [Howell, Rosier, Huynh, and Yen 1986] and NP-hard [Rosier and Yen 1986].

This state of affairs has very recently improved with two articles:

— Leroux and Schmitz [2015] have shown that reachability has a 'cubic Ackermann' upper bound, i.e. is in F_{ω^3}, by analysing the complexity of the classical algorithm developed and refined by Mayr [1981], Kosaraju [1982], and Lambert [1992]. Here, F_{ω^3} is a non primitive-recursive complexity class, but among the lower multiply-recursive ones. The main ingredients for this analysis are the fast-growing complexity bounds for termination proofs by well-quasi-orders and ordinal ranking functions from [Figueira et al. 2011; Schmitz 2014].

— Blondin, Finkel, Göller, Haase, and McKenzie [2015] have shown that reachability in 2-dimensional VASS is PSPACE-complete by a careful analysis of the complexity of the 'flattenings' of Leroux and Sutre [2004] for the upper bound, and by applying results on bounded one-counter automata by Fearnley and Jurdziński [2015] ...

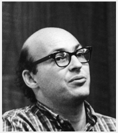
The main focus of the column is the complexity of the al... and Lambert [1992]. Section 3 presents it in ...

January 2016, Vol. 3, No. 1

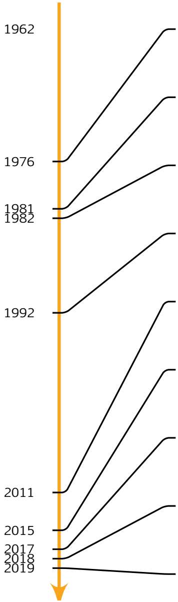
Importance of the Problem

Theorem (Minsky'67)

Reachability is undecidable in 2-dimensional Minsky machines (vector addition systems with zero tests).



Importance of the Problem




R. J. Lipton: EXPSPACE lower bound


E. W. Mayr: decidability by decomposition


S. R. Kosaraju: decidability by decomposition


J.-L. Lambert: decidability by decomposition


J. Leroux: decidability by Presburger inductive invariants

Leroux & S.: cubic Ackermann upper bound ($\mathbf{F}_{\omega^3}$)

S.: quadratic Ackermann upper bound ($\mathbf{F}_{\omega^2}$)



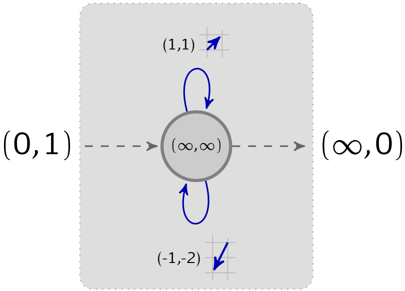



W. Czerwinski, S. Lasota, R. Lazić, J. Leroux
F. Mazowiecki: Tower lower bound ($\mathbf{F}_3$)

J. Leroux & S.: Ackermann upper bound ($\mathbf{F}_\omega$)

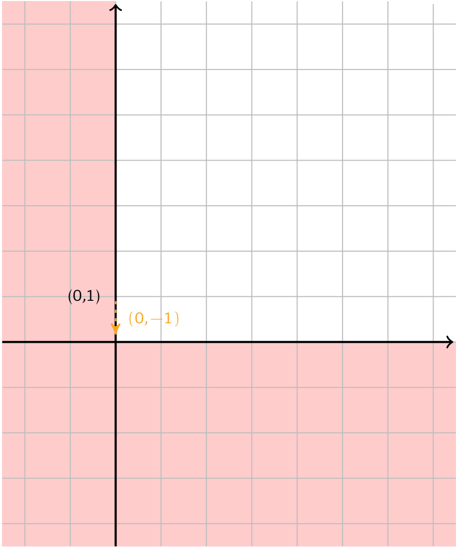
"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]



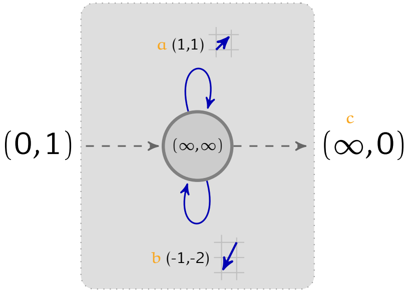
"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]



“Simple Runs” (Θ Condition)

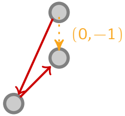
[Mayr’81, Kosaraju’82, Lambert’92]



Characteristic System

$$0 + 1 \cdot a - 1 \cdot b = c$$
$$1 + 1 \cdot a - 2 \cdot b = 0$$

Solution Path

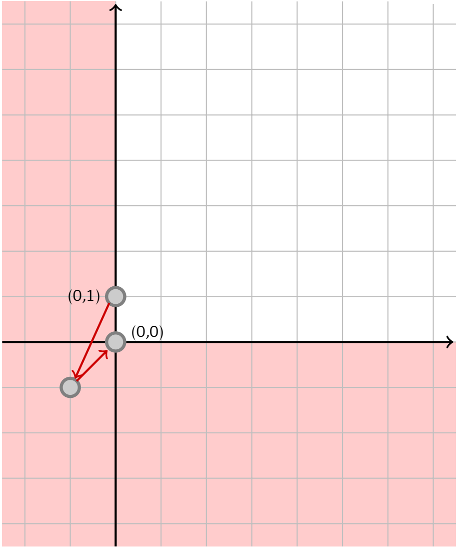


"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]

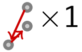
solution path

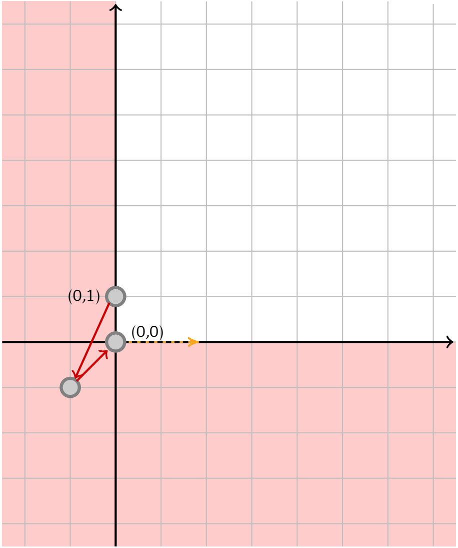
$\times 1$



"Simple Runs" (Θ Condition)

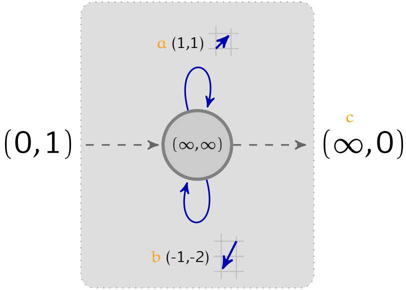
[Mayr'81, Kosaraju'82, Lambert'92]





“Simple Runs” (Θ Condition)

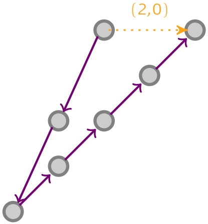
[Mayr’81, Kosaraju’82, Lambert’92]



Homogeneous System

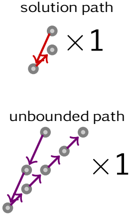
$$1\cdot a - 1\cdot b = c$$
$$1\cdot a - 2\cdot b = 0$$
$$a, b, c > 0$$

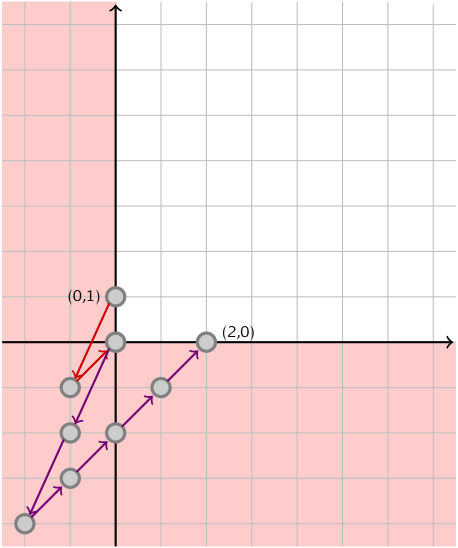
Unbounded Path



"Simple Runs" (Θ Condition)

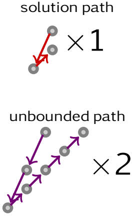
[Mayr'81, Kosaraju'82, Lambert'92]

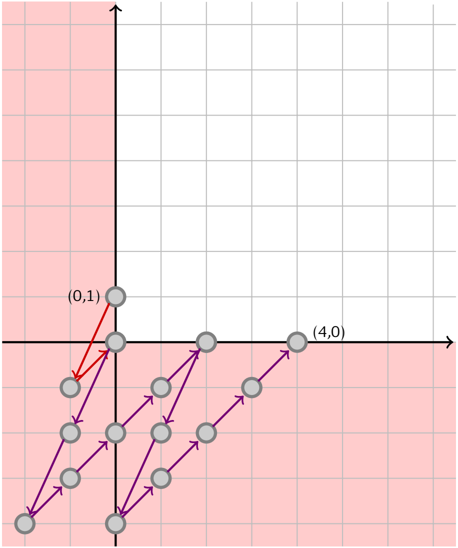




"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]





“Simple Runs” (Θ Condition)

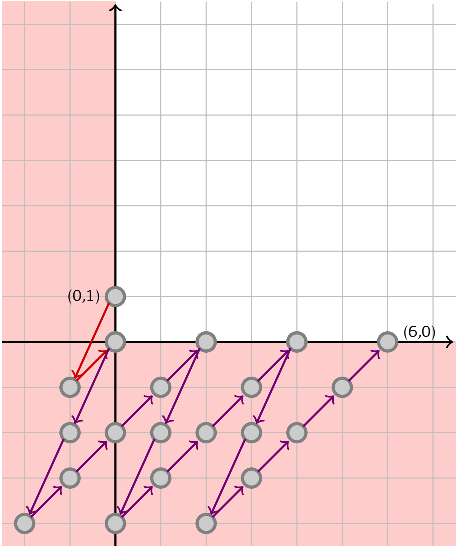
[Mayr’81, Kosaraju’82, Lambert’92]

solution path

$\times 1$

unbounded path

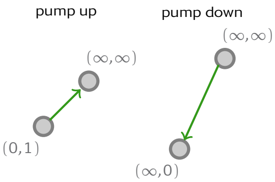
$\times 3$



"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]

Pumpable Paths



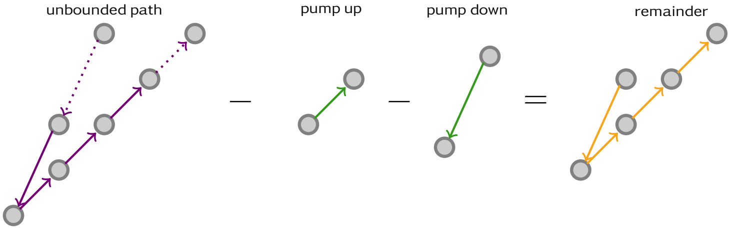
classically: uses *coverability trees* [Karp & Miller'69]

in [Leroux & S.'19] *Rackoff*-style witnesses

"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]

Pumpable Paths

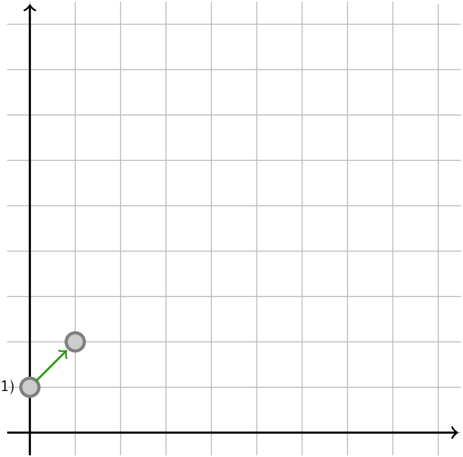


"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]

pump up

 $\times 1$

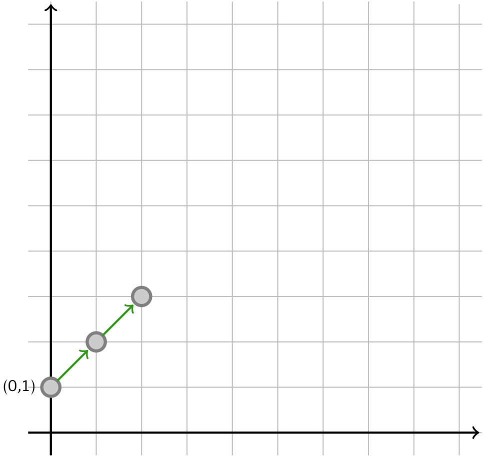


“Simple Runs” (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]

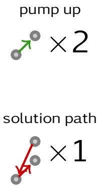
pump up

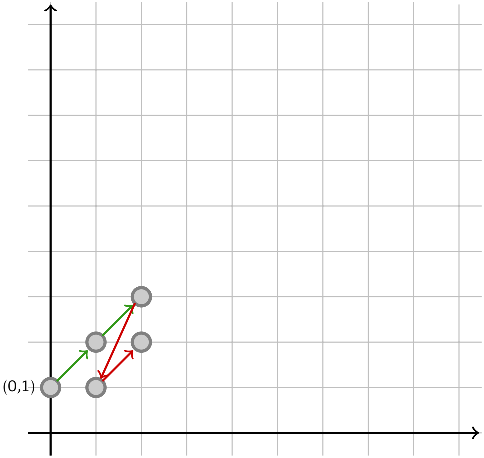
 $\times 2$



"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]





"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]

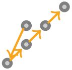
pump up

$\times 2$

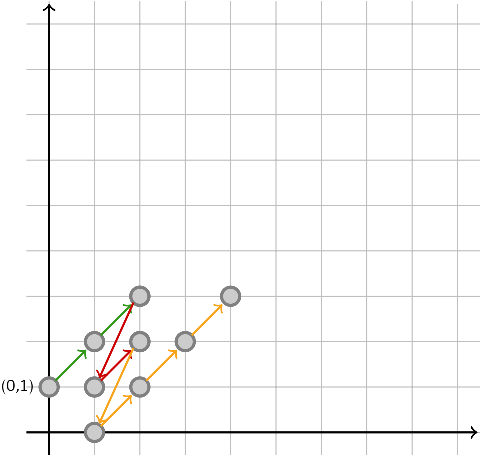
solution path

$\times 1$

remainder

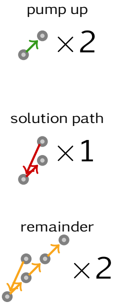


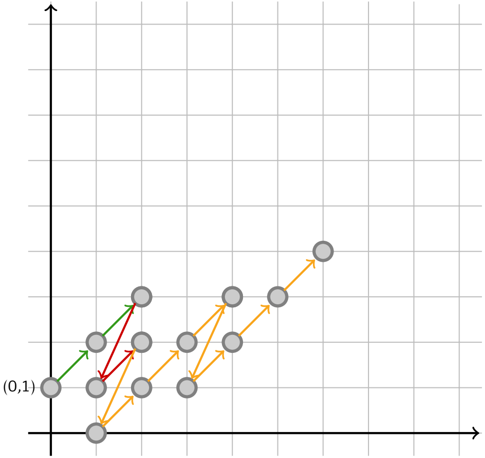
$\times 1$



"Simple Runs" (Θ Condition)

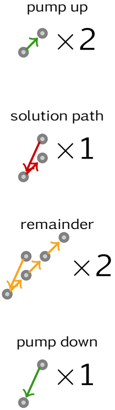
[Mayr'81, Kosaraju'82, Lambert'92]

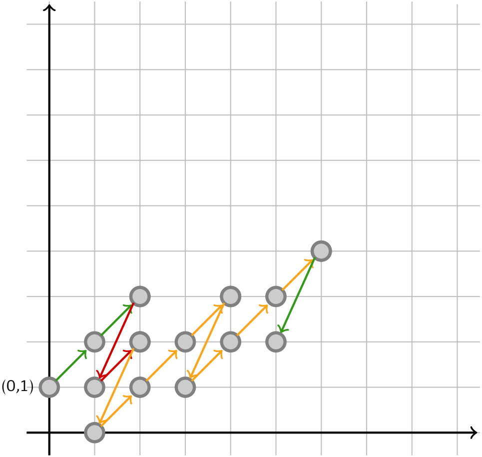




"Simple Runs" (Θ Condition)

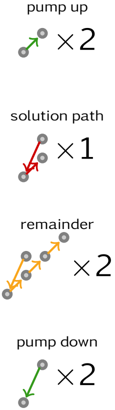
[Mayr'81, Kosaraju'82, Lambert'92]

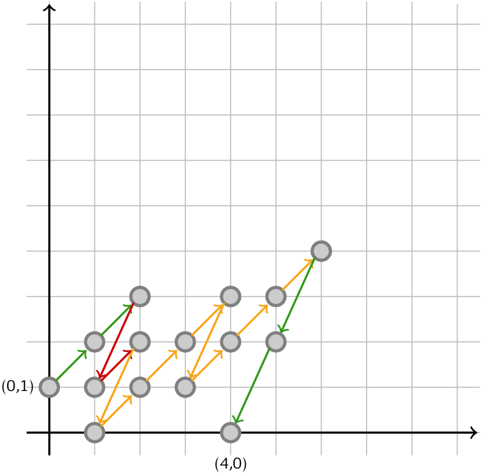




“Simple Runs” (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]





“Simple Runs” (Θ Condition)

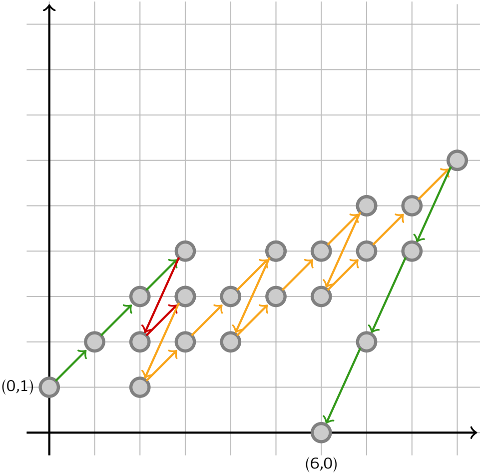
[Mayr’81, Kosaraju’82, Lambert’92]

pump up $\times 3$

solution path $\times 1$

remainder $\times 3$

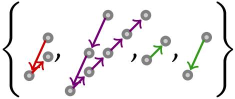
pump down $\times 3$



Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92]

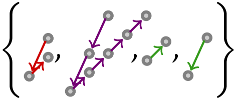
can we build a "simple run"?



Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92]

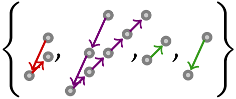
can we build a "simple run"? yes

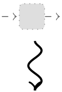


DECOMPOSITION ALGORITHM

[Mayr'81, Kosaraju'82, Lambert'92]

can we build a "simple run"? no





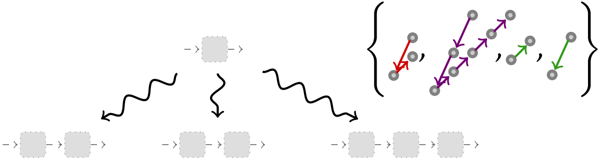
☹

decompose

Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92]

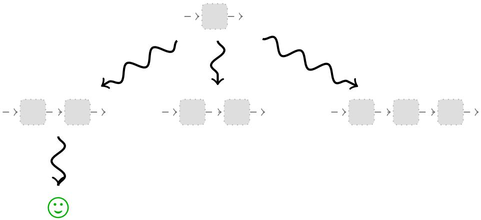
can we build a "simple run"? no



decompose

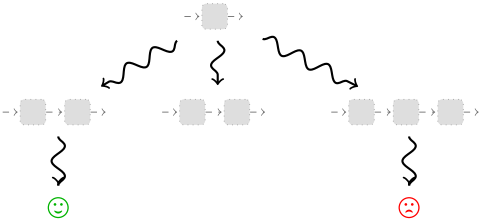
Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92]



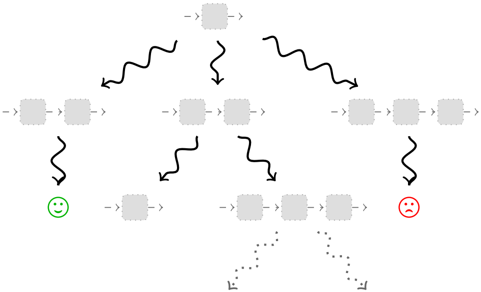
Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92]



Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92]



Termination

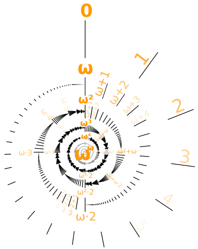
"Finally the checker has to verify that the process comes to an end. Here again he should be assisted by the programmer giving a further definite assertion to be verified. This may take the form of a quantity which is asserted to decrease continually and vanish when the machine stops."

[Turing'49]

Termination

"Finally the checker has to verify that the process comes to an end. Here again he should be assisted by the programmer giving a further definite assertion to be verified. This may take the form of a quantity which is asserted to decrease continually and vanish when the machine stops. To the pure mathematician it is natural to give an ordinal number."

[Turing'49]



Termination of the Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92, Leroux & S.'19]

Ranking Function

ω^ω (ω^{d+1} in dim. d)

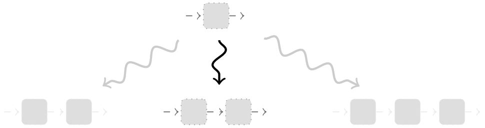
$\vee$

α_0

Termination of the Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92, Leroux & S.'19]

Ranking Function



ω^ω (ω^{d+1} in dim. d)

$\vee$

α_0

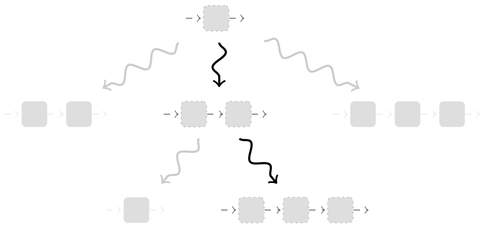
$\vee$

α_1

Termination of the Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92, Leroux & S.'19]

Ranking Function



ω^ω (ω^{d+1} in dim. d)

$\vee$

α_0

$\vee$

α_1

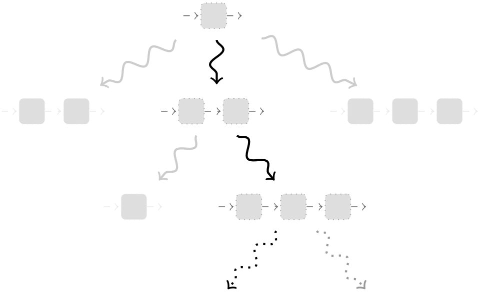
$\vee$

α_2

Termination of the Decomposition Algorithm

[Mayr'81, Kosaraju'82, Lambert'92, Leroux & S.'19]

Ranking Function



$$\omega^\omega \ (\omega^{d+1} \text{ in dim. } d) > \alpha_0 > \alpha_1 > \alpha_2 > \vdots$$

Upper Bounds

How to bound the running time of algorithms with ordinal-based termination proofs?

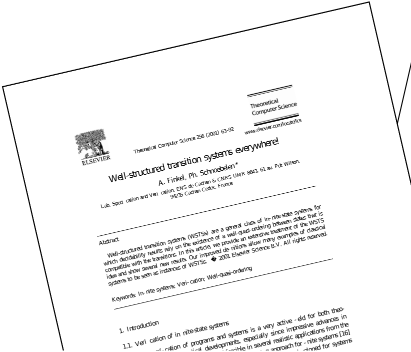
Upper Bounds

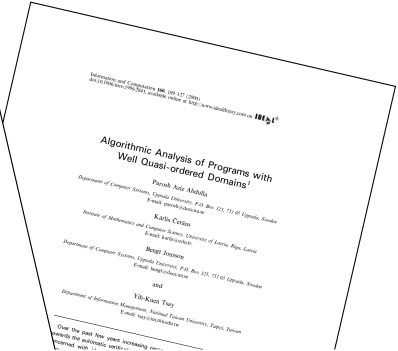
How to bound the running time of algorithms with wqo-based termination proofs?

Upper Bounds

How to bound the running time of algorithms with wqo-based termination proofs?

wqos ubiquitous in infinite-state verification

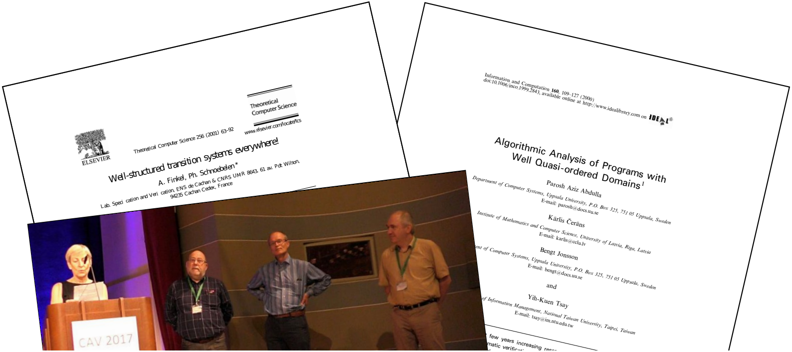




Upper Bounds

How to bound the running time of algorithms with wqo-based termination proofs?

wqos ubiquitous in infinite-state verification



Theoretical Computer Science 256 (2001) 63–92

Theoretical Computer Science

www.elsevier.com/locate/tcs

ELSEVIER

Well-structured transition systems everywhere!

A. Finkel, Ph. Schnoebelen*

Lab. Speci cation and Veri cation, ENS de Cachan & CNRS UMR 8643, 61 av. Pdt Wilson, 94235 Cachan Cedex, France

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

...ent of Computer Systems, Uppsala University, P.O. Box 325, 751 05 Uppsala, Sweden
E-mail: bengt@docs.uu.se

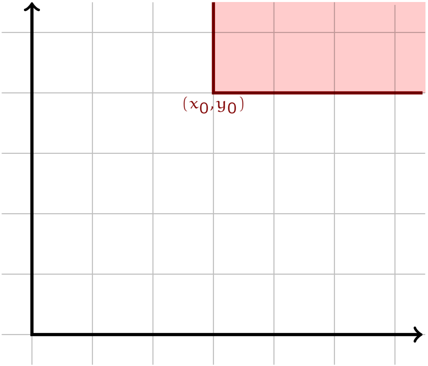
and

Yih-Kuen Tsay

...of Information Management, National Taiwan University, Taipei, Taiwan
E-mail: tsay@im.ntu.edu.tw

A One-Player Game

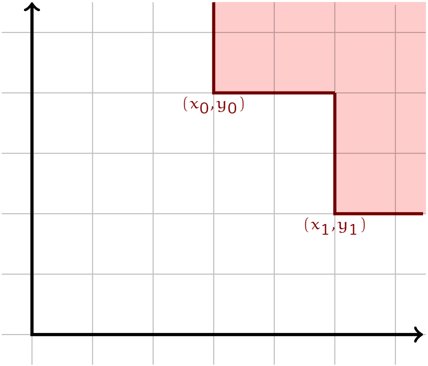
- over $\mathbb{Q}_{\geqslant 0} \times \mathbb{Q}_{\geqslant 0}$
- given initially (x_0, y_0)
- Eloise plays (x_j, y_j) s.t. $\forall 0 \leqslant i < j, x_i > x_j$ or $y_i > y_j$



- Can Eloise win, i.e. play indefinitely?
- If not, how long can she last?

A One-Player Game

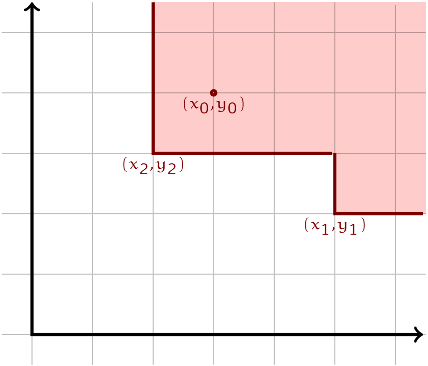
- over $\mathbb{Q}_{\geqslant 0} \times \mathbb{Q}_{\geqslant 0}$
- given initially (x_0, y_0)
- Eloise plays (x_j, y_j) s.t. $\forall 0 \leqslant i < j, x_i > x_j$ or $y_i > y_j$



- Can Eloise win, i.e. play indefinitely?
- If not, how long can she last?

A One-Player Game

- over $\mathbb{Q}_{\geqslant 0} \times \mathbb{Q}_{\geqslant 0}$
- given initially (x_0, y_0)
- Eloise plays (x_j, y_j) s.t. $\forall 0 \leqslant i < j, x_i > x_j$ or $y_i > y_j$

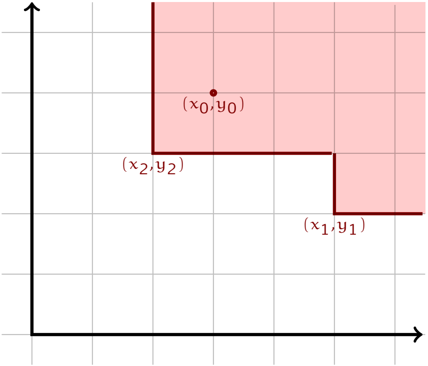


- Can Eloise win, i.e. play indefinitely?
- If not, how long can she last?

If $(x_0, y_0) \neq (0,0)$, then choosing $(x_j, y_j) = (\frac{x_0}{2^j}, \frac{y_0}{2^j})$ wins.

A One-Player Game

- over $\mathbb{N} \times \mathbb{N}$
- given initially (x_0, y_0)
- Eloise plays (x_j, y_j) s.t. $\forall 0 \leqslant i < j, x_i > x_j$ or $y_i > y_j$



- Can Eloise win, i.e. play indefinitely?
- If not, how long can she last?

Assume there exists an infinite sequence $(x_j, y_j)_j$ of moves over $\mathbb{N}^2$.

Assume there exists an infinite sequence $(x_j, y_j)_j$ of moves over $\mathbb{N}^2$. Consider the pairs of indices $i < j$: color (i, j)

purple if $x_i > x_j$ but $y_i \leqslant y_j$,

red if $x_i > x_j$ and $y_i > y_j$,

orange if $y_i > y_j$ but $x_i \leqslant x_j$.

$(3,4)$ $(5,2)$ $(2,3)$ $\cdots$

Assume there exists an infinite sequence $(x_j, y_j)_j$ of moves over $\mathbb{N}^2$. Consider the pairs of indices $i < j$: color (i,j)

purple if $x_i > x_j$ but $y_i \leqslant y_j$,

red if $x_i > x_j$ and $y_i > y_j$,

orange if $y_i > y_j$ but $x_i \leqslant x_j$.

$(3,4) \qquad (5,2) \qquad (2,3) \qquad \cdots$

By the infinite Ramsey Theorem, there exists an infinite monochromatic subset of indices.

Assume there exists an infinite sequence $(x_j, y_j)_j$ of moves over $\mathbb{N}^2$. Consider the pairs of indices $i < j$: color (i, j)

purple if $x_i > x_j$ but $y_i \leqslant y_j$,

red if $x_i > x_j$ and $y_i > y_j$,

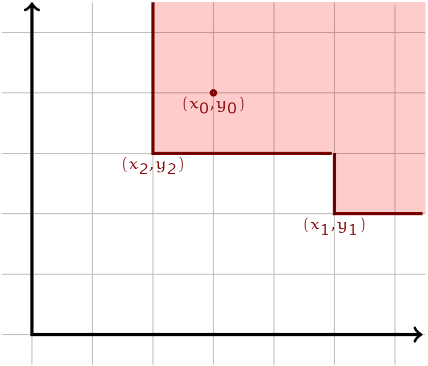
orange if $y_i > y_j$ but $x_i \leqslant x_j$.

$(3,4) \quad (5,2) \quad (2,3) \quad \cdots$

By the infinite Ramsey Theorem, there exists an infinite monochromatic subset of indices. In all cases, it implies the existence of an infinite decreasing sequence in $\mathbb{N}$, a contradiction.

A One-Player Game

- over $\mathbb{N} \times \mathbb{N}$
- given initially (x_0, y_0)
- Eloise plays (x_j, y_j) s.t. $\forall 0 \leqslant i < j, x_i > x_j$ or $y_i > y_j$



- Can Eloise win, i.e. play indefinitely?
- If not, how long can she last?

Bad Sequences

Over a qo $(X, \leqslant)$

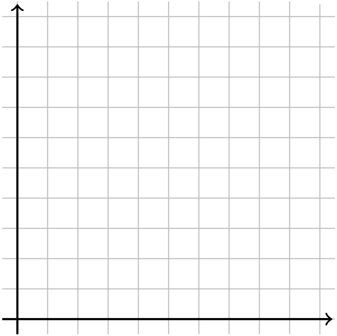
- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $(X, \leqslant)$ wqo iff all bad sequences are finite
-

Bad Sequences

Over a qo $(X, \leqslant)$

$x_0, x_1, \dots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$

$(X, \leqslant)$ wqo iff all bad sequences are finite



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

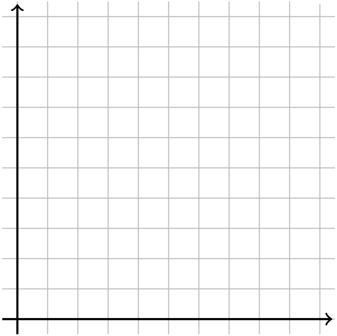
Bad Sequences

Over a qo $(X, \leqslant)$

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

$(X, \leqslant)$ wqo iff all bad sequences are finite

but can be of arbitrary length



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

Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $(X, \leqslant)$ wqo iff all bad sequences are finite
- controlled by $g\colon \mathbb{N} \to \mathbb{N}$ monotone and inflationary and $n_0 \in \mathbb{N}$ if $\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$
  [Cichoń & Tahhan Bittar'98]

Controlled Bad Sequences

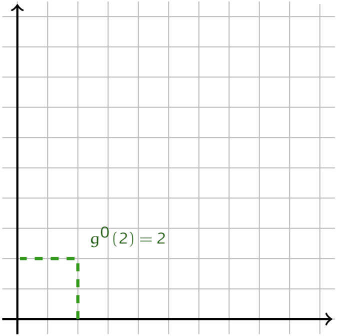
Over a qo $(X, \leqslant)$ with norm $\|\cdot\|$

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

$(X, \leqslant)$ wqo iff all bad sequences are finite

controlled by $g\colon \mathbb{N} \to \mathbb{N}$ monotone and inflationary and $n_0 \in \mathbb{N}$ if $\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$

[Cichoń & Tahhan Bittar'98]



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

Controlled Bad Sequences

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

- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $(X, \leqslant)$ wqo iff all bad sequences are finite
- controlled by $g : \mathbb{N} \to \mathbb{N}$ monotone and inflationary and $n_0 \in \mathbb{N}$ if $\forall i \,.\, \|x_i\| \leqslant g^i(n_0)$

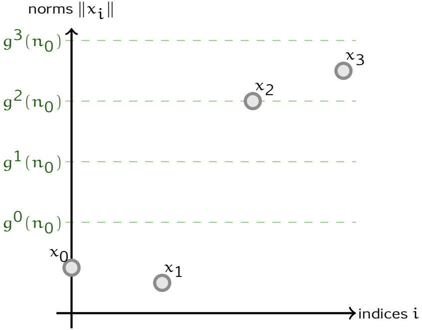
[Cichoń & Tahhan Bittar'98]

Proposition

Over $(X, \leqslant)$, assuming $\forall n \{x \in X \mid \|x\| \leqslant n\}$ finite, (g, n_0)-controlled bad sequences have a maximal length, noted $L_{g,X}(n_0)$.

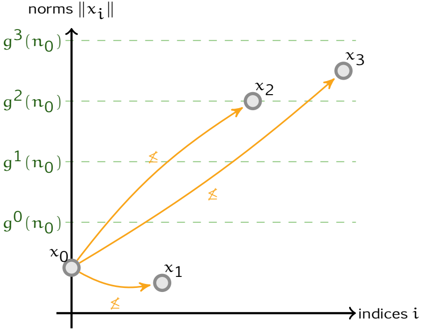
Descent Equation

(g,n_0)-controlled bad sequence $x_0,x_1,x_2,x_3,\ldots$ over a wqo $(X,\leqslant)$:



Descent Equation

(g,n_0)-controlled bad sequence $x_0,x_1,x_2,x_3,\ldots$ over a wqo $(X,\leqslant)$:

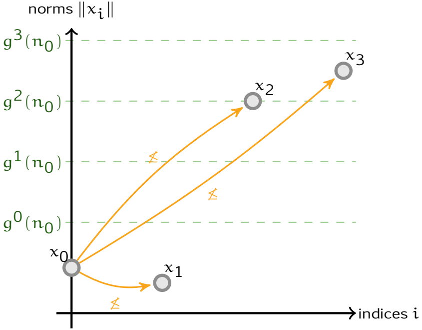


over the suffix $x_1,x_2,x_3,\ldots,\forall i>0$,

$$x_0 \not\leqslant x_i$$

Descent Equation

(g, n_0)-controlled bad sequence $x_0, x_1, x_2, x_3, \ldots$ over a wqo $(X, \leqslant)$:

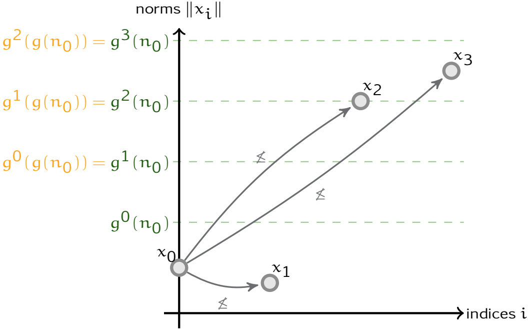


over the suffix $x_1, x_2, x_3, \ldots, \forall i > 0$,

$$x_i \in X \setminus \uparrow x_0 \stackrel{\text{def}}{=} \{x \in X \mid x_0 \not\leq x\}$$

Descent Equation

(g, n_0)-controlled bad sequence $x_0, x_1, x_2, x_3, \ldots$ over a wqo $(X, \leqslant)$:



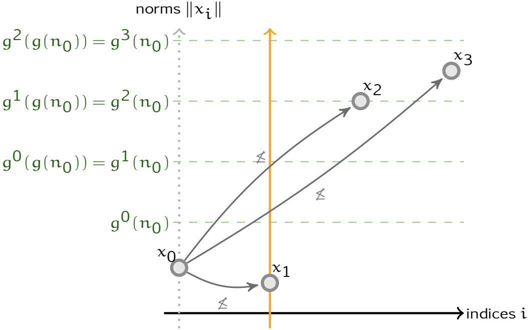
over the suffix $x_1, x_2, x_3, \ldots, \forall i > 0$,

$$x_i \in X \backslash \uparrow x_0 \stackrel{\text{def}}{=} \{x \in X \mid x_0 \not\leqslant x\}$$

$$\|x_i\| \leqslant g^{i-1}(g(n_0))$$

Descent Equation

(g,n_0)-controlled bad sequence $x_0, x_1, x_2, x_3, \ldots$ over a wqo $(X, \leqslant)$:



over the suffix $x_1, x_2, x_3, \ldots, \forall i > 0$,

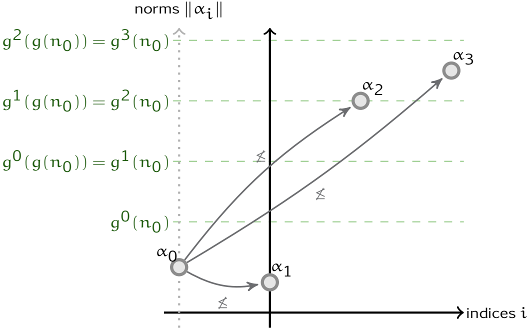
$$x_i \in X \backslash {\uparrow} x_0 \stackrel{\text{def}}{=} \{x \in X \mid x_0 \not\leqslant x\}$$

$$\|x_i\| \leqslant g^{i-1}(g(n_0))$$

$$L_{g,X}(n_0) = \max_{x_0 \in X, \|x_0\| \leqslant n_0} 1 + L_{g, X \backslash \uparrow x_0}(g(n_0))$$

Descent Equation

(g,n_0)-controlled bad sequence $\alpha_0,\alpha_1,\alpha_2,\alpha_3,\ldots$ over an ordinal α:



over the suffix $\alpha_1,\alpha_2,\alpha_3,\ldots,\forall i>0$,

$$\alpha_i \in \alpha_0 \stackrel{\text{def}}{=} \{\beta \in \alpha \mid \beta \not\geq \alpha_0\}$$

$$\|\alpha_i\| \leqslant g^{i-1}(g(n_0))$$

$$L_{g,\alpha}(n_0) = \max_{\alpha_0\in\alpha,\|\alpha_0\|\leqslant n_0} 1 + L_{g,\alpha_0}(g(n_0))$$

The Case of Ordinals

[S.'14]

$$L_{g,\alpha}(n_0) = \max_{\alpha_0 \in \alpha, \|\alpha_0\| \leqslant n_0} 1 + L_{g,\alpha_0}(g(n_0))$$

The Case of Ordinals

[S.'14]

$$L_{g,\alpha}(n_0) = \max_{\alpha_0 \in \alpha, \|\alpha_0\| \leqslant n_0} 1 + L_{g,\alpha_0}(g(n_0))$$

The Case of Ordinals

[S.'14]

For a suitable norm function, there is a "maximising" ordinal $P_{n_0}(\alpha)$:

$$L_{g,0}(n_0) = 0 \qquad L_{g,\alpha}(n_0) = 1 + L_{g,P_{n_0}(\alpha)}(g(n_0))$$

These functions form the Cichón hierarchy.

The Case of Ordinals

[S.'14]

For a suitable norm function, there is a "maximising" ordinal $P_{n_0}(\alpha)$:

$$L_{g,0}(n_0) = 0 \qquad L_{g,\alpha}(n_0) = 1 + L_{g,P_{n_0}(\alpha)}(g(n_0))$$

These functions form the Cichón hierarchy.

Relating Norm and Length

[Cichoń & Tahhan Bittar'98]

Recall the definition of the Cichoń Hierarchy:

$$L_{g,0}(x) \stackrel{\text{def}}{=} 0 \qquad Lg,\alpha(x) \stackrel{\text{def}}{=} 1 + L_{g,P_x(\alpha)}(g(x)) \text{ for } \alpha > 0$$

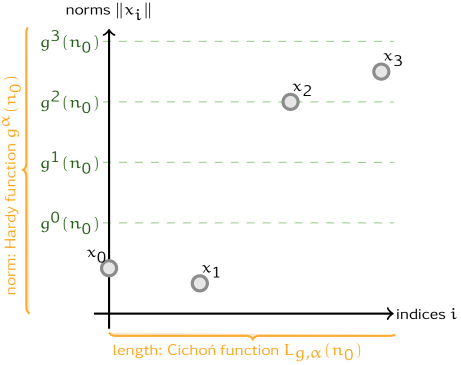
Definition (Hardy Hierarchy)

For $g : \mathbb{N} \to \mathbb{N}$, define $(g^\alpha : \mathbb{N} \to \mathbb{N})_\alpha$ by

$$g^0(x) \stackrel{\text{def}}{=} x \qquad g^\alpha(x) \stackrel{\text{def}}{=} g^{P_x(\alpha)}(g(x)) \text{ for } \alpha > 0$$

Relating Norm and Length

[Cichoń & Tahhan Bittar'98]

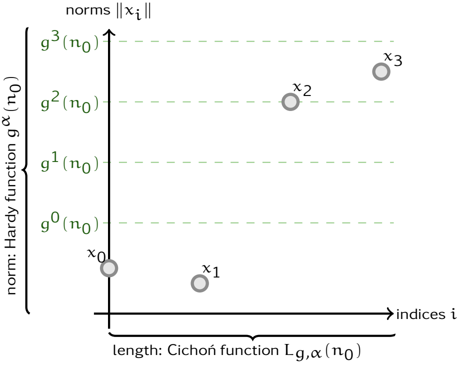


$$g^\alpha(x) = g^{L_{g,\alpha}(x)}(x)$$

$$g^\alpha(x) \geqslant L_{g,\alpha}(x) + x$$

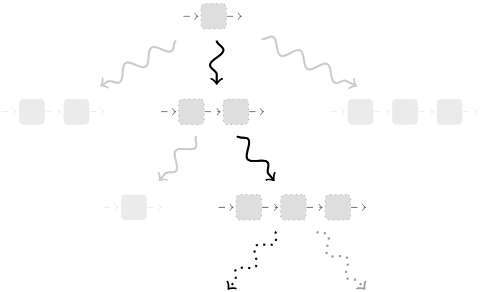
RELATING NORM AND LENGTH

[Cichoń & Tahhan Bittar'98]



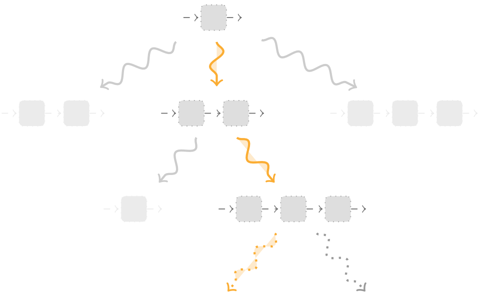
$$g^\alpha(x) = g^{L_{g,\alpha}(x)}(x)$$
$$g^\alpha(x) \geqslant L_{g,\alpha}(x) + x$$

The Length of Decomposition Branches



$$\alpha_0 > \alpha_1 > \alpha_2 > \cdots$$

The Length of Decomposition Branches

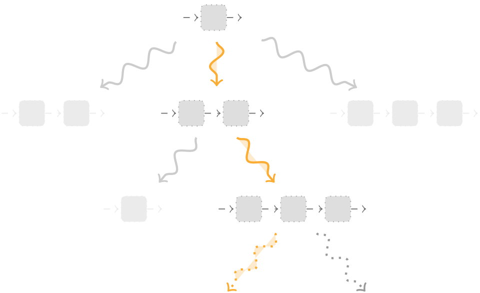


$$\alpha_0 > \alpha_1 > \alpha_2 > \vdots$$

Consequence of (Leroux & S.'19)

An elementary control g and n the size of the reachability instance fit. Thus the decomposition algorithm runs in $\mathit{SPACE}(g^{\omega^\omega}(n))$, and $\mathit{SPACE}(g^{\omega^{d+1}}(n))$ in fixed dimension d.

The Length of Decomposition Branches



$$\begin{array}{c} \alpha_0 \\ \vee \\ \alpha_1 \\ \vee \\ \alpha_2 \\ \vee \\ \vdots \end{array}$$

Consequence of (Leroux & S.'19)

An elementary control g and n the size of the reachability instance fit. Thus the decomposition algorithm runs in SPACE$(g^{\omega^\omega}(n))$, and SPACE$(g^{\omega^{d+1}}(n))$ in fixed dimension d.

Restating the Result

“$\mathsf{SPACE}\left(g^{\omega^{d+1}}(n)\right)$” is unreadable!

Restating the Result

Hardy hierarchy with base function $H(x) \stackrel{\text{def}}{=} x+1$:

$$H^0(x) = x$$

$$H^k(x) = \overbrace{H \circ \cdots \circ H}^{k \text{ times}}(x) \qquad = x+k$$

$$H^\omega(x) = H^{x+1}(x) = \overbrace{H \circ \cdots \circ H}^{x+1 \text{ times}}(x) \qquad = 2x+1$$

$$H^{\omega^2}(x) = H^{\omega\cdot(x+1)} = \overbrace{H^\omega \circ \cdots \circ H^\omega}^{x+1 \text{ times}}(x) \qquad \approx 2^x$$

$$H^{\omega^3}(x) = H^{\omega^2\cdot(x+1)} = \overbrace{H^{\omega^2} \circ \cdots \circ H^{\omega^2}}^{x+1 \text{ times}}(x) \qquad \approx \mathrm{tower}(x)$$

$$\vdots$$

$$H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x) \qquad \approx \mathrm{ack}(x)$$

Restating the Result

Hardy hierarchy with base function $H(x) \stackrel{\text{def}}{=} x+1$:

$$H^0(x) = x$$

$$H^k(x) = \overbrace{H \circ \cdots \circ H}^{k \text{ times}}(x) \qquad = x + k$$

$$H^\omega(x) = H^{x+1}(x) = \overbrace{H \circ \cdots \circ H}^{x+1 \text{ times}}(x) \qquad = 2x+1$$

$$H^{\omega^2}(x) = H^{\omega\cdot(x+1)} = \overbrace{H^\omega \circ \cdots \circ H^\omega}^{x+1 \text{ times}}(x) \qquad \approx 2^x$$

$$H^{\omega^3}(x) = H^{\omega^2\cdot(x+1)} = \overbrace{H^{\omega^2} \circ \cdots \circ H^{\omega^2}}^{x+1 \text{ times}}(x) \qquad \approx \mathrm{tower}(x)$$

$$\vdots$$

$$H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x) \qquad \approx \mathrm{ack}(x)$$

Restating the Result

Hardy hierarchy with base function $H(x) \stackrel{\text{def}}{=} x+1$:

$$H^0(x) = x$$

$$H^k(x) = \overbrace{H \circ \cdots \circ H}^{k \text{ times}}(x) \qquad = x + k$$

$$H^\omega(x) = H^{x+1}(x) = \overbrace{H \circ \cdots \circ H}^{x+1 \text{ times}}(x) \qquad = 2x+1$$

$$H^{\omega^2}(x) = H^{\omega\cdot(x+1)} = \overbrace{H^\omega \circ \cdots \circ H^\omega}^{x+1 \text{ times}}(x) \qquad \approx 2^x$$

$$H^{\omega^3}(x) = H^{\omega^2\cdot(x+1)} = \overbrace{H^{\omega^2} \circ \cdots \circ H^{\omega^2}}^{x+1 \text{ times}}(x) \qquad \approx \text{tower}(x)$$

$$\vdots$$

$$H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x) \qquad \approx \text{ack}(x)$$

Restating the Result

Hardy hierarchy with base function $H(x) \stackrel{\text{def}}{=} x+1$:

$$H^0(x) = x$$

$$H^k(x) = \overbrace{H \circ \cdots \circ H}^{k \text{ times}}(x) \qquad = x + k$$

$$H^\omega(x) = H^{x+1}(x) = \overbrace{H \circ \cdots \circ H}^{x+1 \text{ times}}(x) \qquad = 2x+1$$

$$H^{\omega^2}(x) = H^{\omega\cdot(x+1)} = \overbrace{H^\omega \circ \cdots \circ H^\omega}^{x+1 \text{ times}}(x) \qquad \approx 2^x$$

$$H^{\omega^3}(x) = H^{\omega^2\cdot(x+1)} = \overbrace{H^{\omega^2} \circ \cdots \circ H^{\omega^2}}^{x+1 \text{ times}}(x) \qquad \approx \mathrm{tower}(x)$$

$$\vdots$$

$$H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x) \qquad \approx \mathrm{ack}(x)$$

Restating the Result

Hardy hierarchy with base function $H(x) \stackrel{\text{def}}{=} x+1$:

$$H^0(x) = x$$

$$H^k(x) = \overbrace{H \circ \cdots \circ H}^{k \text{ times}}(x) \qquad = x + k$$

$$H^\omega(x) = H^{x+1}(x) = \overbrace{H \circ \cdots \circ H}^{x+1 \text{ times}}(x) \qquad = 2x + 1$$

$$H^{\omega^2}(x) = H^{\omega\cdot(x+1)} = \overbrace{H^\omega \circ \cdots \circ H^\omega}^{x+1 \text{ times}}(x) \qquad \approx 2^x$$

$$H^{\omega^3}(x) = H^{\omega^2\cdot(x+1)} = \overbrace{H^{\omega^2} \circ \cdots \circ H^{\omega^2}}^{x+1 \text{ times}}(x) \qquad \approx \mathrm{tower}(x)$$

$$\vdots$$

$$H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x) \qquad \approx \mathrm{ack}(x)$$

Restating the Result

Hardy hierarchy with base function $H(x) \stackrel{\text{def}}{=} x+1$:

$$H^0(x) = x$$

$$H^k(x) = \overbrace{H \circ \cdots \circ H}^{k \text{ times}}(x) \qquad = x+k$$

$$H^\omega(x) = H^{x+1}(x) = \overbrace{H \circ \cdots \circ H}^{x+1 \text{ times}}(x) \qquad = 2x+1$$

$$H^{\omega^2}(x) = H^{\omega\cdot(x+1)} = \overbrace{H^\omega \circ \cdots \circ H^\omega}^{x+1 \text{ times}}(x) \qquad \approx 2^x$$

$$H^{\omega^3}(x) = H^{\omega^2\cdot(x+1)} = \overbrace{H^{\omega^2} \circ \cdots \circ H^{\omega^2}}^{x+1 \text{ times}}(x) \qquad \approx \mathrm{tower}(x)$$

$$\vdots$$

$$H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x) \qquad \approx \mathrm{ack}(x)$$

Restating the Result

Define coarse-grained classes:

$$\mathscr{F}_{<\alpha} \stackrel{\text{def}}{=} \bigcup_{\beta<\omega^\alpha} \mathsf{FDTIME}(H^\beta(n))$$

$$\mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{f\in\mathscr{F}_{<\alpha}} \mathsf{DTIME}(H^{\omega^\alpha}(f(n)))$$

Consequence of (S.'16, Thm. 4.4)

VAS Reachability is in $\mathbf{F}_\omega$, and in $\mathbf{F}_{d+4}$ in fixed dimension d.

Restating the Result

Define coarse-grained classes:

$$\mathscr{F}_{<\alpha} \stackrel{\text{def}}{=} \bigcup_{\beta<\omega^\alpha} \mathsf{FDTIME}(H^\beta(n))$$

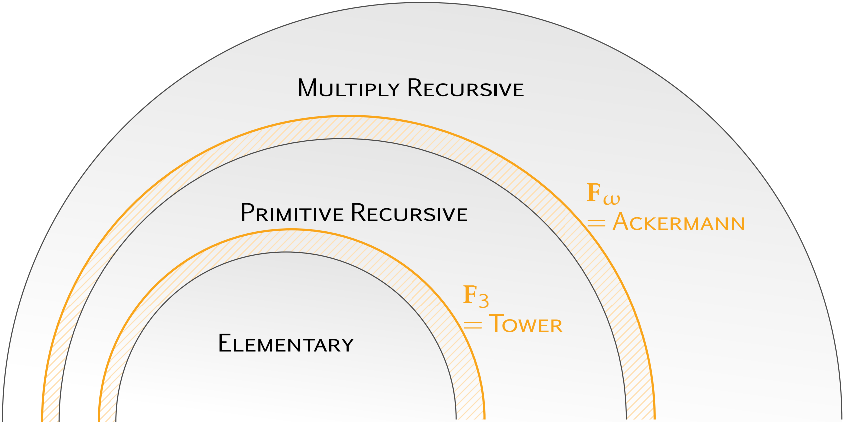
$$\mathbf{F}_\alpha \stackrel{\text{def}}{=} \bigcup_{f\in\mathscr{F}_{<\alpha}} \mathsf{DTIME}(H^{\omega^\alpha}(f(n)))$$

Consequence of (S.'16, Thm. 4.4)

VAS Reachability is in $\mathbf{F}_\omega$, and in $\mathbf{F}_{d+4}$ in fixed dimension d.

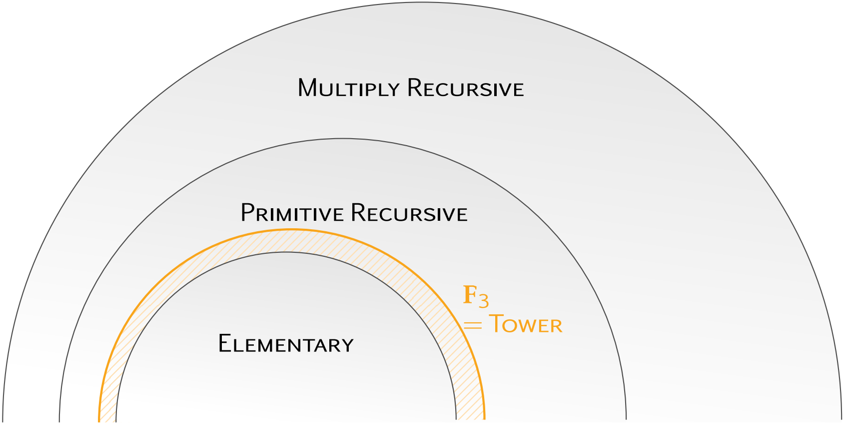
Complexity Classes Beyond Elementary

[S.'16]



Complexity Classes Beyond Elementary

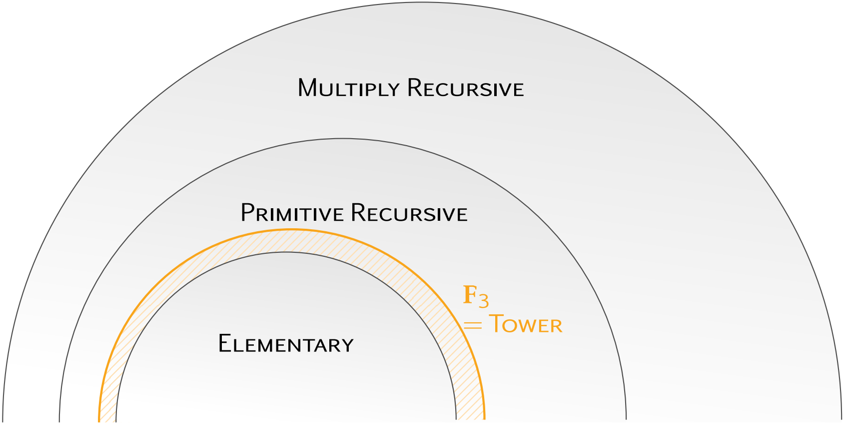
[S.'16]



$$\mathbf{F}_3 \stackrel{\text{def}}{=} \bigcup_{e \text{ elementary}} \mathrm{DTime}\big(\mathrm{tower}(e(n))\big)$$

Complexity Classes Beyond Elementary

[S.'16]

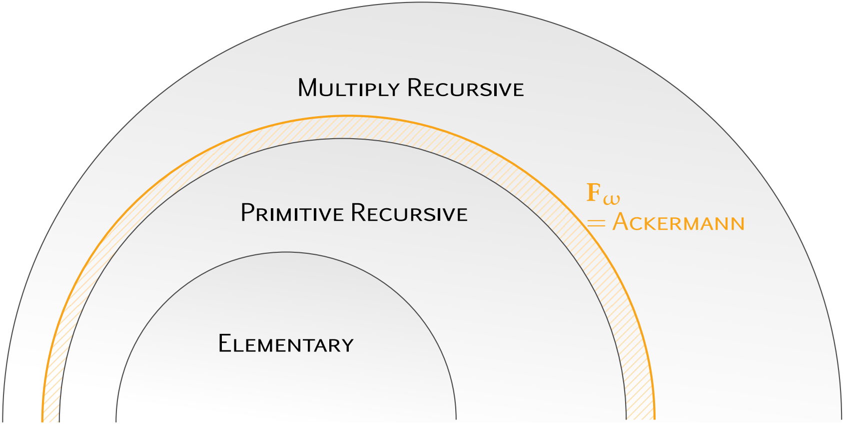


Examples of Tower-Complete Problems:

- satisfiability of first-order logic on words [Meyer'75]
- β-equivalence of simply typed λ terms [Statman'79]
- model-checking higher-order recursion schemes [Ong'06]

Complexity Classes Beyond Elementary

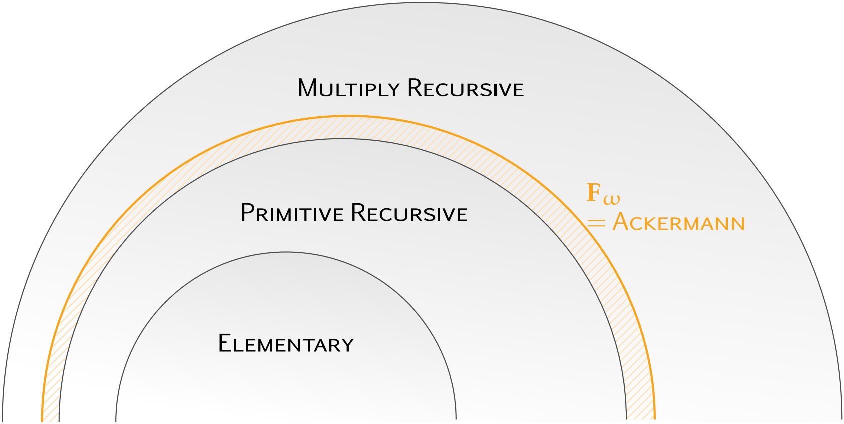
[S.'16]



$$\mathbf{F}_\omega \stackrel{\text{def}}{=} \bigcup_{p \text{ primitive recursive}} \mathrm{DTime}\big(\mathrm{ack}(p(n))\big)$$

Complexity Classes Beyond Elementary

[S.'16]

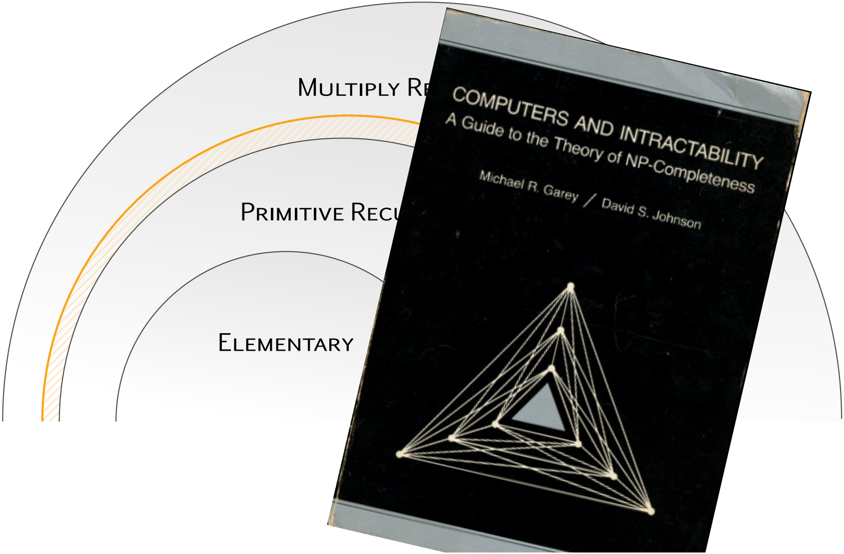


Examples of Ackermann-Complete Problems:

- reachability in lossy Minsky machines [Urquhart'98, Schnoebelen'02]
- satisfiability of safety Metric Temporal Logic [Lazić et al.'16]
- satisfiability of Vertical XPath [Figueira and Segoufin'17]

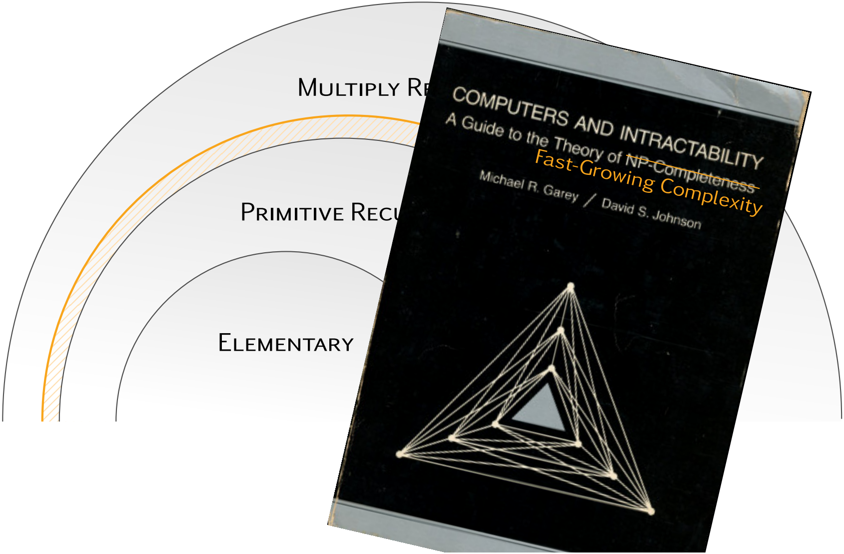
Complexity Classes Beyond Elementary

[S.'16]



Complexity Classes Beyond Elementary

[S.'16]



A Related Problem

labelled VAS transitions carry labels from some alphabet

$L(\mathcal{V}, \mathbf{source}, \mathbf{target})$ the language of labels in runs from **source** to **target**

$\downarrow L$ the set of scattered subwords of the words in the language L

Downwards Language Inclusion Problem

input: *two labelled VAS $\mathcal{V}$ and $\mathcal{V}'$ and configurations* $\mathbf{source}, \mathbf{target}, \mathbf{source}', \mathbf{target}'$

question: $\downarrow L(\mathcal{V}, \mathbf{source}, \mathbf{target}) \subseteq \downarrow L(\mathcal{V}', \mathbf{source}', \mathbf{target}')$?

A Related Problem

labelled VAS transitions carry labels from some alphabet

$L(\mathcal{V}, \mathbf{source}, \mathbf{target})$ the language of labels in runs from **source** to **target**

$\downarrow L$ the set of scattered subwords of the words in the language L

Downwards Language Inclusion Problem

input: *two labelled VAS $\mathcal{V}$ and $\mathcal{V}'$ and configurations* $\mathbf{source}, \mathbf{target}, \mathbf{source}', \mathbf{target}'$

question: $\downarrow L(\mathcal{V}, \mathbf{source}, \mathbf{target}) \subseteq \downarrow L(\mathcal{V}', \mathbf{source}', \mathbf{target}')$?

A Related Problem

Downwards Language Inclusion Problem

input: *two labelled VAS $\mathcal{V}$ and $\mathcal{V}'$ and configurations* $\mathbf{source}, \mathbf{target}, \mathbf{source}', \mathbf{target}'$

question: $\downarrow L(\mathcal{V}, \mathbf{source}, \mathbf{target}) \subseteq \downarrow L(\mathcal{V}', \mathbf{source}', \mathbf{target}')$?

Theorem (Habermehl, Meyer & Wimmel'10)

Given a labelled VAS $\mathcal{V}$ and configurations $\mathbf{source}$ *and* $\mathbf{target}$ *and its decomposition, one can construct a finite automaton for* $\downarrow L(\mathcal{V}, \mathbf{source}, \mathbf{target})$ *in polynomial time.*

Corollary

The Downwards Language Inclusion is in Ackermann.

A Related Problem

Downwards Language Inclusion Problem

input: *two labelled VAS $\mathcal{V}$ and $\mathcal{V}'$ and configurations* $\mathbf{source}, \mathbf{target}, \mathbf{source}', \mathbf{target}'$

question: $\downarrow L(\mathcal{V}, \mathbf{source}, \mathbf{target}) \subseteq \downarrow L(\mathcal{V}', \mathbf{source}', \mathbf{target}')$?

Theorem (Habermehl, Meyer & Wimmel'10)

Given a labelled VAS $\mathcal{V}$ and configurations $\mathbf{source}$ *and* $\mathbf{target}$ *and its decomposition, one can construct a finite automaton for* $\downarrow L(\mathcal{V}, \mathbf{source}, \mathbf{target})$ *in polynomial time.*

Corollary

The Downwards Language Inclusion is in Ackermann.

A Related Problem

Downwards Language Inclusion Problem

input: *two labelled VAS* $\mathcal{V}$ *and* $\mathcal{V}'$ *and configurations* $\mathbf{source}, \mathbf{target}, \mathbf{source}', \mathbf{target}'$

question: $\downarrow L(\mathcal{V}, \mathbf{source}, \mathbf{target}) \subseteq \downarrow L(\mathcal{V}', \mathbf{source}', \mathbf{target}')$?

Corollary

The Downwards Language Inclusion is in Ackermann.

Theorem (Zetzsche'16)

The Downwards Language Inclusion is Ackermann*-hard.*

Summary

well-quasi-orders (wqo):

- proving algorithm termination

a toolbox for wqo-based complexity

- upper bounds: length function theorems (for ordinals, Dickson's Lemma, Higman's Lemma, and combinations)
- lower bounds
- complexity classes: $(\mathbf{F}_\alpha)_\alpha$

this talk: focus on one problem

- reachability in vector addition systems in $\mathbf{F}_\omega$

Perspectives

1. complexity gap for VAS reachability
   - Tower-hard [Czerwinski et al.'18]
   - decomposition algorithm: requires $\mathbf{F}_\omega$ = Ackermann time, because downward language inclusion is $\mathbf{F}_\omega$-hard [Zetzsche'16]
2. reachability in VAS extensions
   - decidable in VAS with hierarchical zero tests [Reinhardt'08]
   - what about
     - branching VAS
     - unordered data Petri nets
     - pushdown VAS

Perspectives

1. complexity gap for VAS reachability
   - Tower-hard [Czerwinski et al.'18]
   - decomposition algorithm: requires $\mathbf{F}_\omega = $ Ackermann time, because downward language inclusion is $\mathbf{F}_\omega$-hard [Zetzsche'16]
2. reachability in VAS extensions
   - decidable in VAS with hierarchical zero tests [Reinhardt'08]
   - what about
     - branching VAS
     - unordered data Petri nets
     - pushdown VAS

PERSPECTIVES

1. complexity gap for VAS reachability
   - TOWER-hard [Czerwinski et al.'18]
   - decomposition algorithm: requires $\mathbf{F}_\omega = \textsc{Ackermann}$ time, because downward language inclusion is $\mathbf{F}_\omega$-hard [Zetzsche'16]
2. reachability in VAS extensions
   - decidable in VAS with hierarchical zero tests [Reinhardt'08]
   - what about
     - branching VAS
     - unordered data Petri nets
     - pushdown VAS

Demystifying Reachability in Vector Addition Systems

[Leroux & S.'15]

Ideal Decomposition Theorem

The Decomposition Algorithm computes the ideal decomposition of the set of runs from source to target.

Upper Bound Theorem

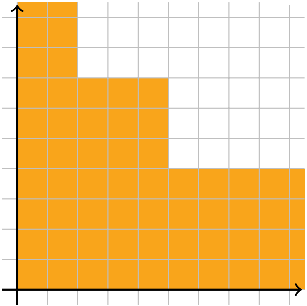
Reachability in vector addition systems is in cubic Ackermann.

Ideals of Well-Quasi-Orders $(X, \leqslant)$

- Canonical decompositions
  [Bonnet'75]
  if $D \subseteq X$ is $\downarrow$-closed, then

$$D = I_1 \cup \cdots \cup I_n$$

  for (maximal) ideals $I_1, \ldots, I_n$



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

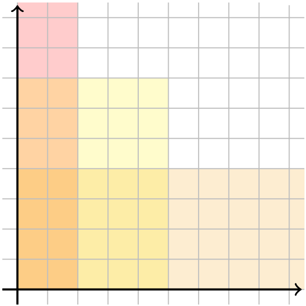
$$D = (\{0, \ldots, 2\} \times \mathbb{N}) \cup (\{0, \ldots, 5\} \times \{0, \ldots, 7\}) \cup (\mathbb{N} \times \{0, \ldots, 4\})$$

Ideals of Well-Quasi-Orders $(X, \leqslant)$

- Canonical decompositions
  [Bonnet'75]
  if $D \subseteq X$ is $\downarrow$-closed, then

  $$D = I_1 \cup \cdots \cup I_n$$

  for (maximal) ideals $I_1, \ldots, I_n$

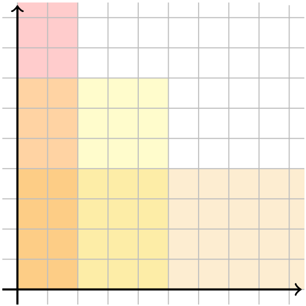


Example (over $\mathbb{N}^2$)

$$D = (\{0,\ldots,2\} \times \mathbb{N}) \cup (\{0,\ldots,5\} \times \{0,\ldots,7\}) \cup (\mathbb{N} \times \{0,\ldots,4\})$$

Ideals of Well-Quasi-Orders $(X, \leqslant)$

- Canonical decompositions
  [Bonnet'75]
  if $D \subseteq X$ is $\downarrow$-closed, then
  $$D = I_1 \cup \cdots \cup I_n$$
  for (maximal) ideals $I_1, \ldots, I_n$
- Effective representations
  [Goubault-Larrecq et al.'17]



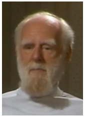
Example (over $\mathbb{N}^2$)

$$D = [\![(2,\infty)]\!] \cup [\![(5,7)]\!] \cup [\![(\infty,4)]\!]$$

Decomposition Theorem

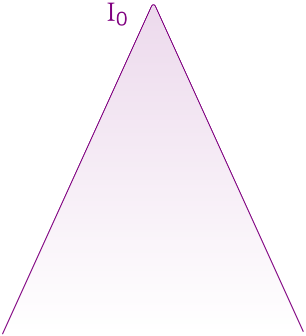
Well-Quasi-Order on Runs

combination of Dickson's and Higman's lemmata



Syntax

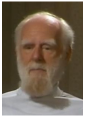
Semantics



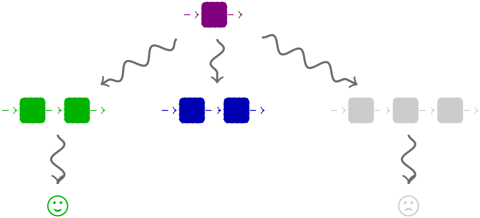
Decomposition Theorem

Well-Quasi-Order on Runs

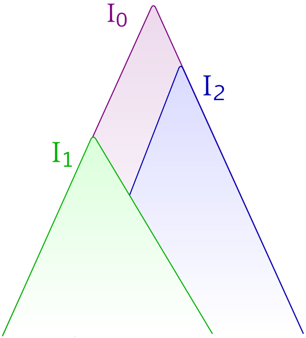
combination of Dickson's and Higman's lemmata



Syntax



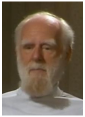
Semantics



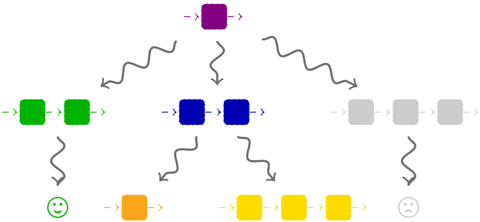
Decomposition Theorem

Well-Quasi-Order on Runs

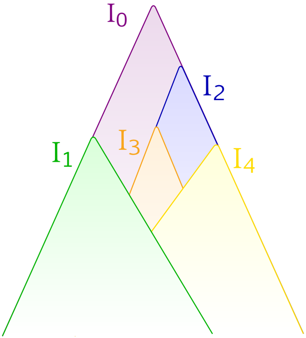
combination of Dickson's and Higman's lemmata



Syntax



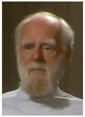
Semantics



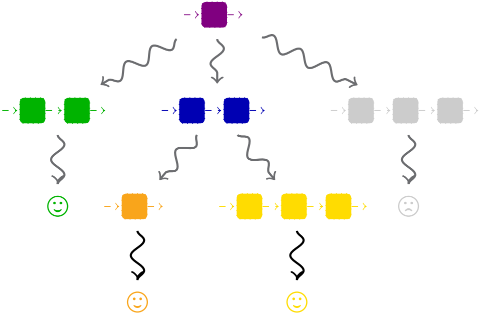
Decomposition Theorem

Well-Quasi-Order on Runs

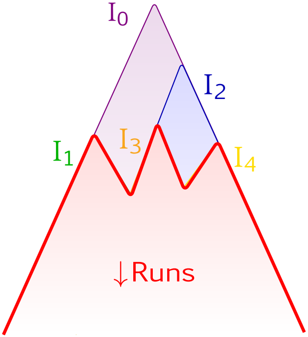
combination of Dickson's and Higman's lemmata



Syntax

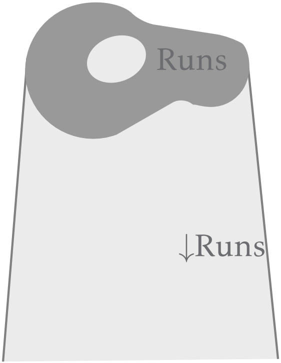


Semantics



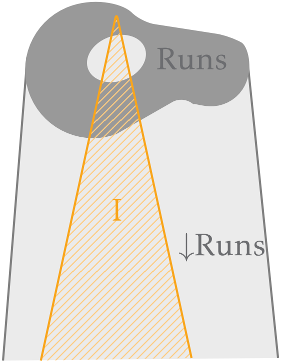
Adherence Membership

- I is adherent to Runs if $I \subseteq {\downarrow}(I \cap \mathrm{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



Adherence Membership

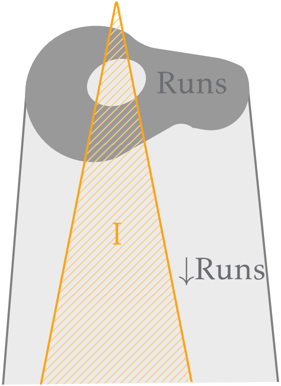
- I is adherent to Runs if $I \subseteq \downarrow(I \cap \text{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I adherent

Adherence Membership

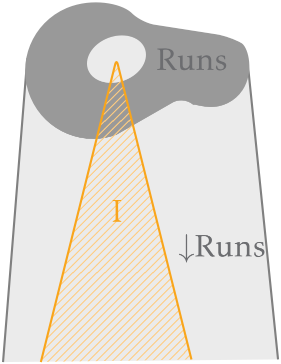
- I is adherent to Runs if $I \subseteq \downarrow(I \cap \text{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I not adherent

Adherence Membership

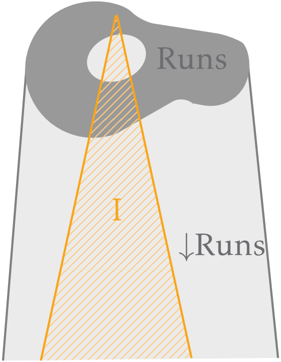
- I is adherent to Runs if $I \subseteq \downarrow(I \cap \mathrm{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I not adherent

Adherence Membership

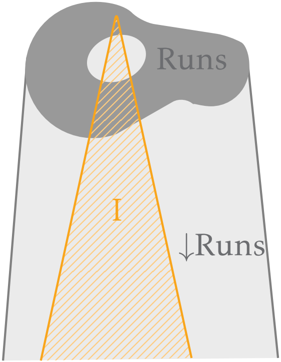
- I is adherent to Runs if $I \subseteq \downarrow(I \cap \text{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I adherent

Adherence Membership

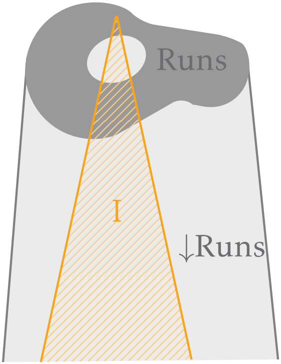
- I is adherent to Runs if $\mathrm{I} \subseteq \downarrow(\mathrm{I} \cap \mathrm{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I adherent

Adherence Membership

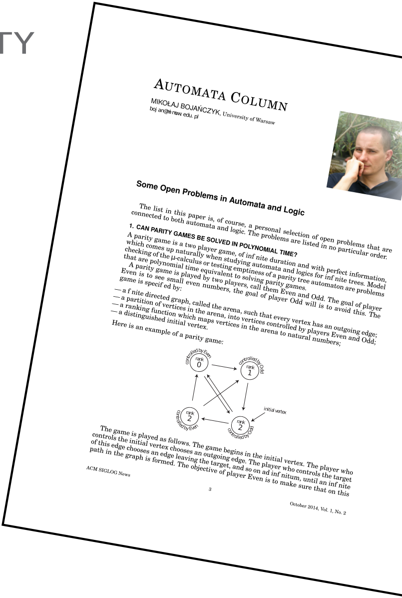
- I is adherent to Runs if $\mathrm{I} \subseteq \downarrow(\mathrm{I} \cap \mathrm{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I adherent

Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:



Automata Column

MIKOŁAJ BOJAŃCZYK, University of Warsaw
boj an@mimuw.edu.pl

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

A parity game is a two player game, of inf nite duration and with perfect information, which comes up naturally when studying automata and logics for inf nite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specif ed by:

— a f nite directed graph, called the arena, such that every vertex has an outgoing edge;
— a partition of vertices in the arena, into vertices controlled by players Even and Odd;
— a ranking function which maps vertices in the arena to natural numbers;
— a distinguished initial vertex.

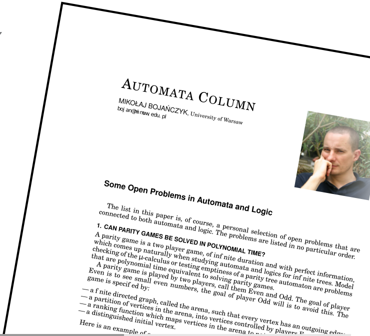
Here is an example of a parity game:

The game is played as follows. The game begins in the initial vertex. The player who controls the initial vertex chooses an outgoing edge. The player who controls the target of this edge chooses an edge leaving the target, and so on ad inf nitum, until an inf nite path in the graph is formed. The objective of player Even is to make sure that on this

ACM SIGLOG News 3 October 2014, Vol. 1, No. 2

Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:


Automata Column

MIKOŁAJ BOJAŃCZYK, University of Warsaw

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

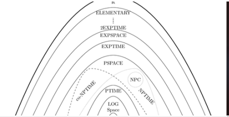
A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specified by:

— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players E…
— a ranking function which maps vertices in the arena to …
— a distinguished initial vertex.

Here is an example of …

Complexity Theory

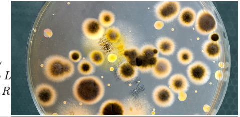


Distributed Computing

Proof Theory

$$\dfrac{\dfrac{}{X \vdash X}\,\mathrm{ax} \qquad \dfrac{\dfrac{\overline{0 \vdash Y, Z}\;0L}{\vdash Y, 0 \multimap Z}\multimap R}{}}{\dfrac{X \vdash Y, X \otimes (0 \multimap Z)}{\dfrac{\vdash X \multimap Y, X \otimes (0 \multimap Z)}{\dfrac{(X \multimap Y) \multimap 0 \vdash X \otimes (0 \multimap Z)}{\vdash ((X \multimap Y) \multimap 0) \multimap (X \otimes (0 \multimap Z))}\multimap R}\multimap L \quad \overline{0 \vdash}\;0L}\multimap R}\otimes R$$

Computational Biology



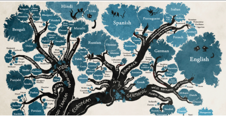
Programming Languages

```
fun append (xs, ys) =
    if null xs
    then ys
    else (hd xs):: append (tl xs, ys)

fun map (f, xs) =
    case xs of
        [] => []
      | x :: xs' => (f x)::(map (f, xs'))

val a = map (increment, [4,8,12,16])
val b = map (hd, [[8,6],[7,5],[3,0,9]])
```

Database Theory

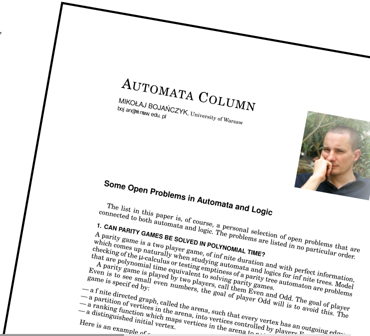
Security

Computational Linguistics



Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:



Automata Column

MIKOŁAJ BOJAŃCZYK, University of Warsaw

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

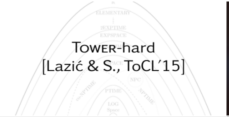
A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specified by:

— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players E…
— a ranking function which maps vertices in the arena to …
— a distinguished initial vertex.

Here is an example of …

Complexity Theory



Tower-hard [Lazić & S., ToCL'15]

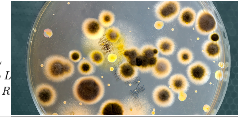
Distributed Computing

Proof Theory

$$\dfrac{\dfrac{}{X \vdash X}\,\text{ax} \qquad \dfrac{\dfrac{\dfrac{}{0 \vdash Y, Z}\,0L}{\vdash Y, 0 \multimap Z}\multimap R}{}}{\dfrac{X \vdash Y, X \otimes (0 \multimap Z)}{\dfrac{\vdash X \multimap Y, X \otimes (0 \multimap Z)}{}\multimap R}}\otimes R \qquad \dfrac{}{0 \vdash}\,0L$$

$$\dfrac{(X \multimap Y) \multimap 0 \vdash X \otimes (0 \multimap Z)}{\vdash ((X \multimap Y) \multimap 0) \multimap (X \otimes (0 \multimap Z))}\multimap R \quad (\multimap L)$$

Computational Biology



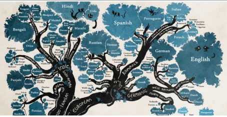
Programming Languages

```
fun append (xs, ys) =
    if null xs
    then ys
    else (hd xs):: append (tl xs, ys)

fun map (f, xs) =
    case xs of
        [] => []
      | x :: xs' => (f x)::(map (f, xs'))

val a = map (increment, [4,8,12,16])
val b = map (hd, [[8,6],[7,5],[3,0,9]])
```

Database Theory

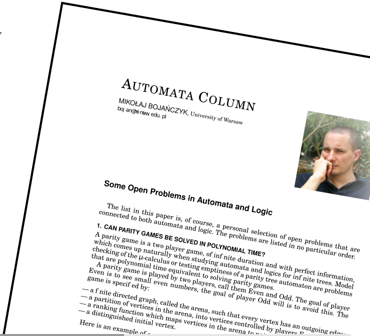
Security

Computational Linguistics



Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:


Automata Column

MIKOŁAJ BOJAŃCZYK, University of Warsaw

bojan@mimuw.edu.pl

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

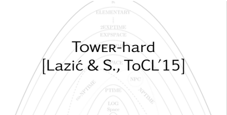
A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

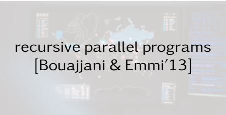
A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specified by:

— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players E…
— a ranking function which maps vertices in the arena to …
— a distinguished initial vertex.

Here is an example of …

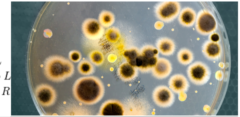
Complexity Theory	Distributed Computing	Proof Theory	Computational Biology


Tower-hard
[Lazić & S., ToCL'15]


recursive parallel programs
[Bouajjani & Emmi'13]

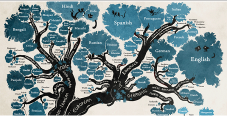
$$\dfrac{\dfrac{}{X \vdash X}\,\mathrm{ax} \qquad \dfrac{\dfrac{}{0 \vdash Y, Z}\,0L}{\vdash Y, 0 \multimap Z}\multimap R}{\dfrac{X \vdash Y, X \otimes (0 \multimap Z)}{\dfrac{\vdash X \multimap Y, X \otimes (0 \multimap Z)}{}\multimap R}}\otimes R$$

$$\dfrac{\dfrac{\vdash X \multimap Y, X \otimes (0 \multimap Z) \qquad \dfrac{}{0 \vdash}\,0L}{(X \multimap Y) \multimap 0 \vdash X \otimes (0 \multimap Z)}\multimap L}{\vdash ((X \multimap Y) \multimap 0) \multimap (X \otimes (0 \multimap Z))}\multimap R$$



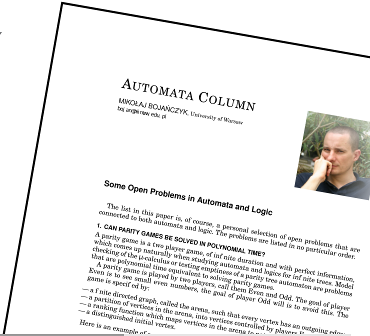
Programming Languages	Database Theory	Security	Computational Linguistics

```
fun append (xs, ys) =
    if null xs
    then ys
    else (hd xs):: append (tl xs, ys)

fun map (f, xs) =
    case xs of
        [] => []
      | x :: xs' => (f x)::(map (f, xs'))

val a = map (increment, [4,8,12,16])
val b = map (hd, [[8,6],[7,5],[3,0,9]])
```



Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:


Automata Column

Mikołaj Bojańczyk, University of Warsaw

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specified by:

— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players E...
— a ranking function which maps vertices in the arena to n...
— a distinguished initial vertex.

Here is an example of ...

Complexity Theory

Tower-hard
[Lazić & S., ToCL'15]

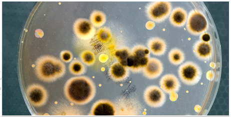
Distributed Computing

recursive parallel programs
[Bouajjani & Emmi'13]

Proof Theory

linear and relevance logics
[de Groote et al.'04
Lazić & S., ToCL'15
S., JSL'16]

Computational Biology



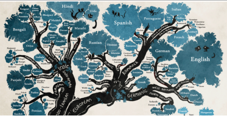
Programming Languages

```
fun append (xs, ys) =
    if null xs
    then ys
    else (hd xs):: append (tl xs, ys)

fun map (f, xs) =
    case xs of
       [] => []
     | x :: xs' => (f x)::(map (f, xs'))

val a = map (increment, [4,8,12,16])
val b = map (hd, [[8,6],[7,5],[3,0,9]])
```

Database Theory

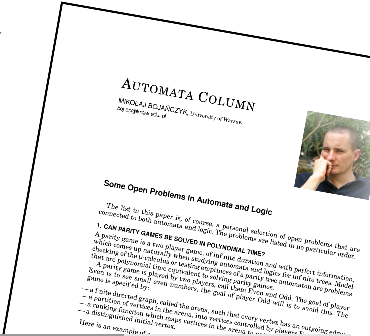
Security

Computational Linguistics



Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:


AUTOMATA COLUMN

MIKOŁAJ BOJAŃCZYK, University of Warsaw

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

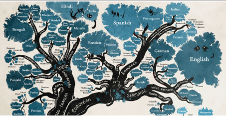
1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specified by:

— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players
— a ranking function which maps vertices in the arena to
— a distinguished initial vertex.

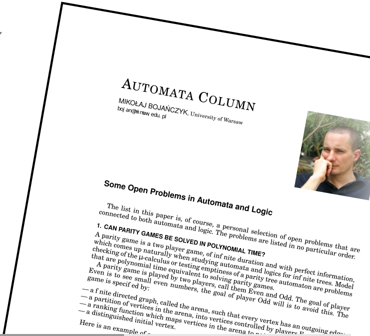
Here is an example of

Complexity Theory	Distributed Computing	Proof Theory	Computational Biology
Tower-hard [Lazić & S., ToCL'15]	recursive parallel programs [Bouajjani & Emmi'13]	linear and relevance logics [de Groote et al.'04 Lazić & S., ToCL'15 S., JSL'16]	population protocols [Bertrand et al.'17]
Programming Languages	**Database Theory**	**Security**	**Computational Linguistics**
see code below			

```
fun append (xs, ys) =
    if null xs
    then ys
    else (hd xs):: append (tl xs, ys)

fun map (f, xs) =
    case xs of
       [] => []
     | x :: xs' => (f x)::(map (f, xs'))

val a = map (increment, [4,8,12,16])
val b = map (hd, [[8,6],[7,5],[3,0,9]])
```

Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:


AUTOMATA COLUMN

MIKOŁAJ BOJAŃCZYK, University of Warsaw
bojan@mimuw.edu.pl

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

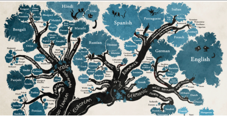
1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specified by:

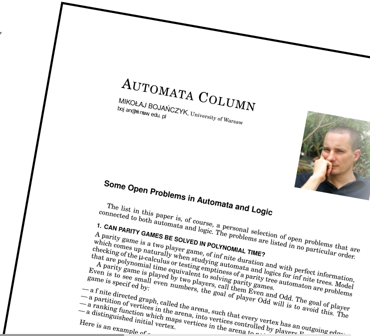
— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players E...
— a ranking function which maps vertices in the arena to n...
— a distinguished initial vertex.

Here is an example of ...

Complexity Theory	Distributed Computing	Proof Theory	Computational Biology
TOWER-hard [Lazić & S., ToCL'15]	recursive parallel programs [Bouajjani & Emmi'13]	linear and relevance logics [de Groote et al.'04 Lazić & S., ToCL'15 S., JSL'16]	population protocols [Bertrand et al.'17]
Programming Languages	**Database Theory**	**Security**	**Computational Linguistics**
observational equivalence [Cotton-Barratt et al.'17]			

Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:


Automata Column
MIKOŁAJ BOJAŃCZYK, University of Warsaw
bojan@mimuw.edu.pl

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

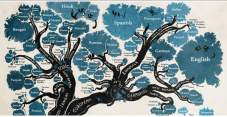
1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specified by:

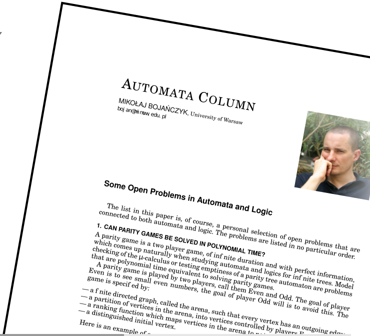
— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players
— a ranking function which maps vertices in the arena to
— a distinguished initial vertex.

Here is an example of

Complexity Theory	Distributed Computing	Proof Theory	Computational Biology
Tower-hard [Lazić & S., ToCL'15]	recursive parallel programs [Bouajjani & Emmi'13]	linear and relevance logics [de Groote et al.'04 Lazić & S., ToCL'15 S., JSL'16]	population protocols [Bertrand et al.'17]
Programming Languages	Database Theory	Security	Computational Linguistics
observational equivalence [Cotton-Barratt et al.'17]	data logics [Bojańczyk et al.'09, Abriola et al.'17]		

Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:


Automata Column

Mikołaj Bojańczyk, University of Warsaw

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

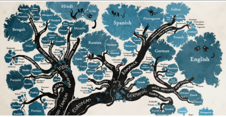
1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specified by:

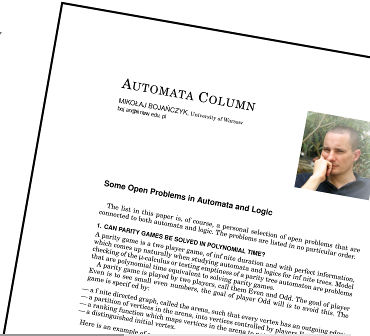
— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players
— a ranking function which maps vertices in the arena to
— a distinguished initial vertex.

Here is an example of

Complexity Theory	Distributed Computing	Proof Theory	Computational Biology
Tower-hard [Lazić & S., ToCL'15]	recursive parallel programs [Bouajjani & Emmi'13]	linear and relevance logics [de Groote et al.'04 Lazić & S., ToCL'15 S., JSL'16]	population protocols [Bertrand et al.'17]
Programming Languages	**Database Theory**	**Security**	**Computational Linguistics**
observational equivalence [Cotton-Barratt et al.'17]	data logics [Bojańczyk et al.'09, Abriola et al.'17]	security protocols [Verma & Goubault-Larrecq'05]	

Branching VAS Reachability

- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:



Automata Column

Mikołaj Bojańczyk, University of Warsaw

Some Open Problems in Automata and Logic

The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.

1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving parity games.

A parity game is played by two players, call them Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd will is to avoid this. The game is specified by:

— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players E...
— a ranking function which maps vertices in the arena to ...
— a distinguished initial vertex.

Here is an example of ...

Complexity Theory	Distributed Computing	Proof Theory	Computational Biology
Tower-hard [Lazić & S., ToCL'15]	recursive parallel programs [Bouajjani & Emmi'13]	linear and relevance logics [de Groote et al.'04 Lazić & S., ToCL'15 S., JSL'16]	population protocols [Bertrand et al.'17]
Programming Languages	**Database Theory**	**Security**	**Computational Linguistics**
observational equivalence [Cotton-Barratt et al.'17]	data logics [Bojańczyk et al.'09, Abriola et al.'17]	security protocols [Verma & Goubault-Larrecq'05]	dominance grammars [Rambow'94; S., ACL'10] minimalist syntax [Salvati'10]