Algorithmic Complexity of Well-Quasi-Orders

Sylvain Schmitz

Habilitation Defense
November 27, 2017

Outline

well-quasi-orders (wqo):

- proving algorithm termination

thesis: 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

thesis: 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

thesis: a toolbox for wqo complexity

- upper bounds
- lower bounds
- complexity classes

this talk: focus on one problem

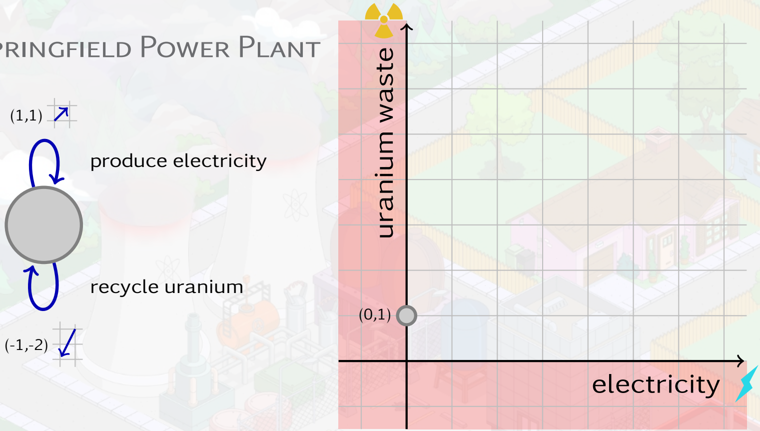
- reachability in vector addition systems

Vector Addition Systems

Vector Addition Systems

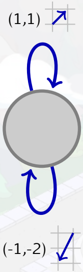
Vector Addition Systems

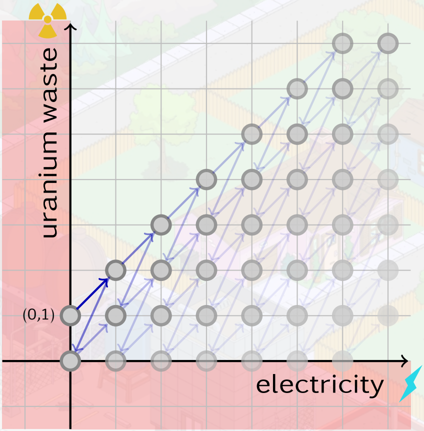
Springfield Power Plant



Vector Addition Systems

Springfield Power Plant

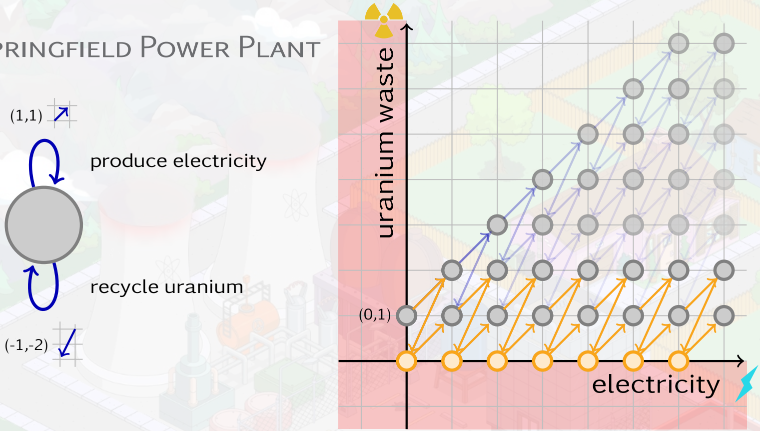




Can we produce unbounded electricity with no left-over uranium waste?

Vector Addition Systems

Springfield Power Plant



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***?*

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

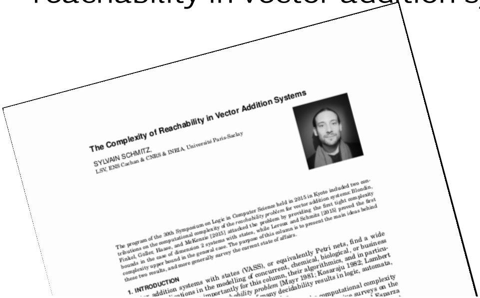
Reachability Problem

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

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

Central Decision Problem [invited survey S., SIGLOG'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, Goller, 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

... addition systems with states (VASS), or equivalently Petri nets, find a wide
... in the modelling of concurrent, chemical, biological, or business
... importantly for this column, their algorithmics, and in particu-
... *ability problem* [Mayr 1981; Kosaraju 1982; Lambert
... many decidability results in logic, automata,
... computational complexity
... surveys on the
... Esparza

Importance of the Problem

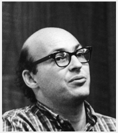
Reachability Problem

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

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

Theorem (Minsky'67)

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



Importance of the Problem

- 1962  C. A. Petri: Petri nets
- 1969   R. M. Karp & R. E. Miller: coverability trees
- 1976  R. J. Lipton: EXPSPACE lower bound
- 1979   J. E. Hopcroft & J.-J. Pansiot: dim. $\geqslant 3$ not definable in Presburger arithmetic
- 1981  E. W. Mayr: decidability by decomposition
- 1982  S. R. Kosaraju: decidability by decomposition
- 1992  J.-L. Lambert: decidability by decomposition
- 2011  J. Leroux: decidability by Presburger inductive invariants
- 2015 this talk: Leroux & S., LICS'15

Demystifying Reachability in Vector Addition Systems

[Leroux & S., LICS'15]

Upper Bound Theorem

Reachability in vector addition systems is in cubic Ackermann.

Ideal Decomposition Theorem

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

Demystifying Reachability in Vector Addition Systems

[Leroux & S., LICS'15; S., 2017]

Upper Bound Theorem

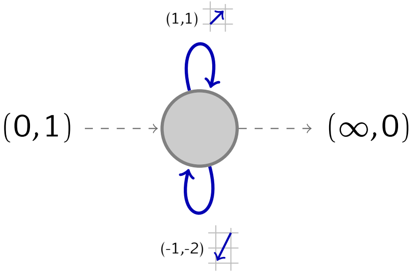
Reachability in vector addition systems is in quadratic Ackermann.

Ideal Decomposition Theorem

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

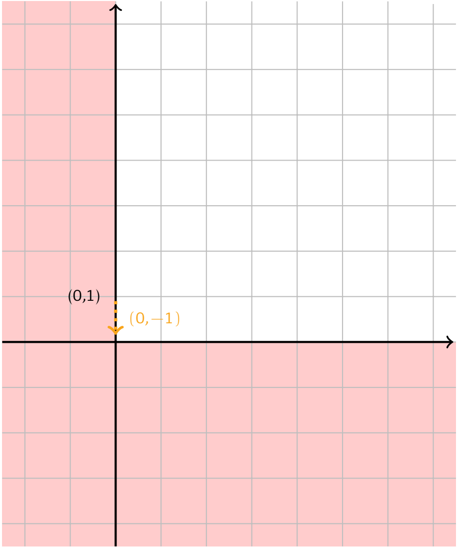
Decomposition Algorithm

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



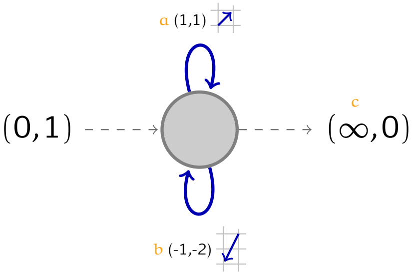
Decomposition Algorithm

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



Decomposition Algorithm

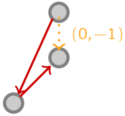
[Mayr'81, Kosaraju'82, Lambert'92]



Equations

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

Solution Path

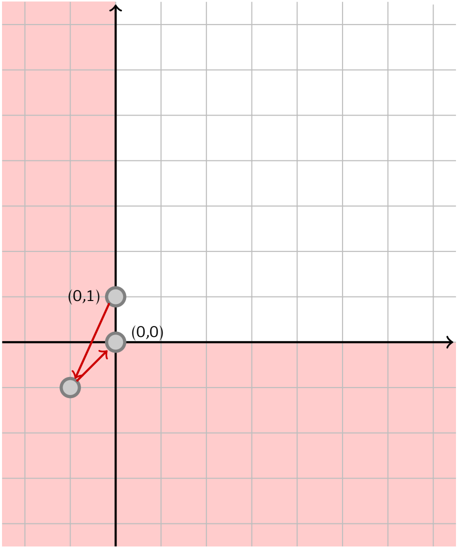


Decomposition Algorithm

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

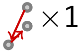
solution path

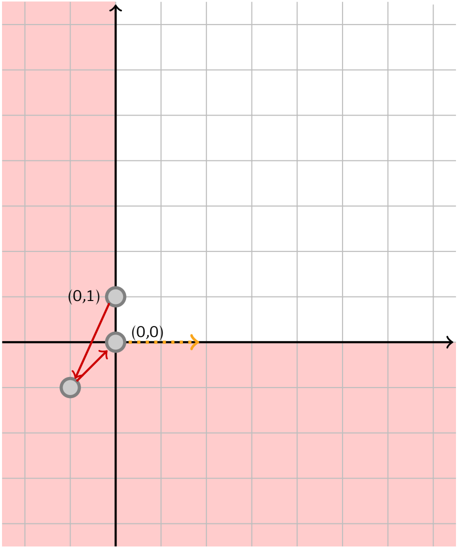

$\times 1$



Decomposition Algorithm

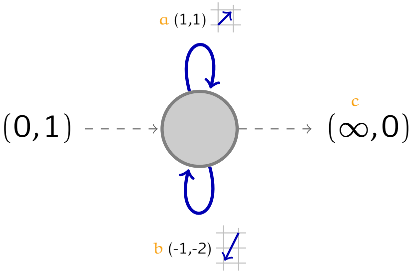
[Mayr'81, Kosaraju'82, Lambert'92]





Decomposition Algorithm

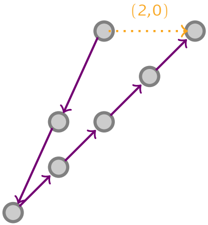
[Mayr'81, Kosaraju'82, Lambert'92]



Equations

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

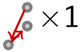
Unbounded Path



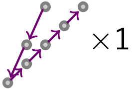
Decomposition Algorithm

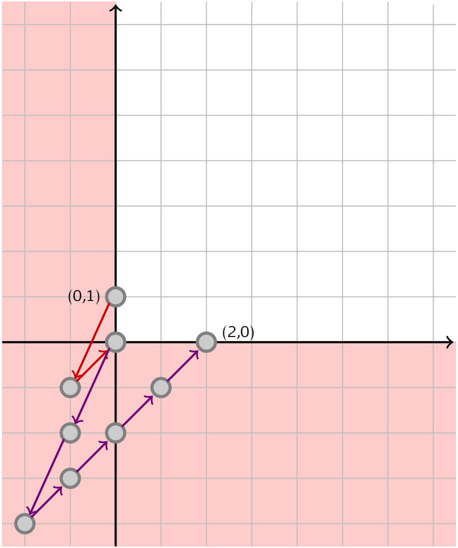
[Mayr'81, Kosaraju'82, Lambert'92]

solution path



unbounded path





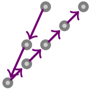
Decomposition Algorithm

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

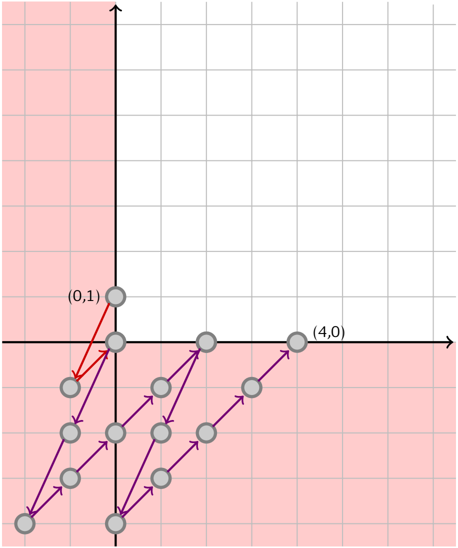
solution path

$\times 1$

unbounded path

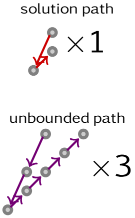


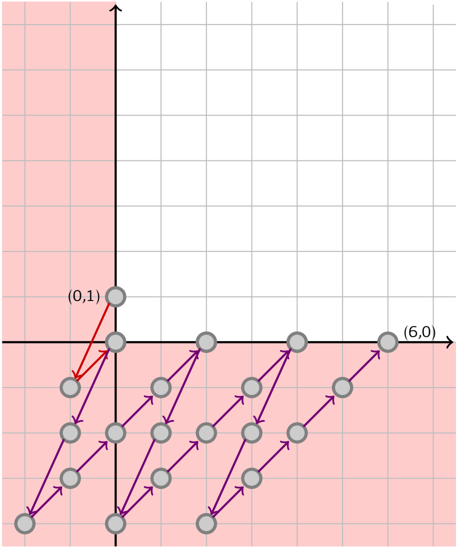
$\times 2$



Decomposition Algorithm

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

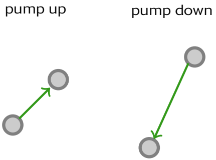




Decomposition Algorithm

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

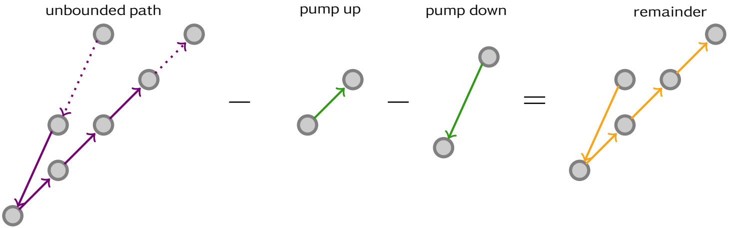
Pumpable Paths



Decomposition Algorithm

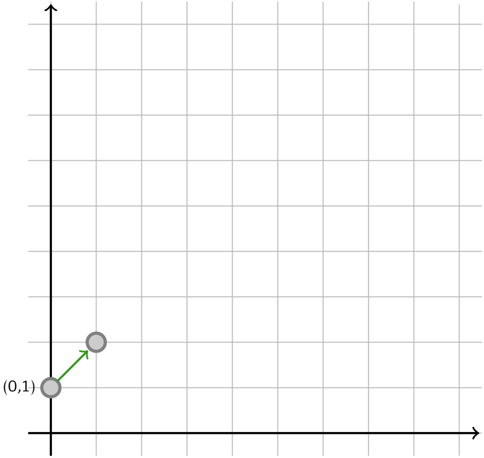
[Mayr'81, Kosaraju'82, Lambert'92]

Pumpable Paths



Decomposition Algorithm

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

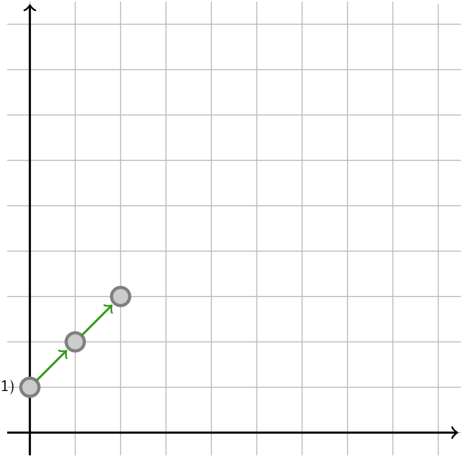


Decomposition Algorithm

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

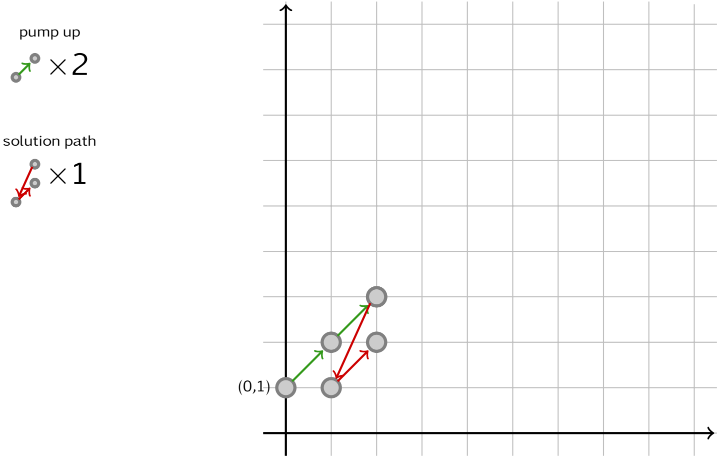
pump up

$\times 2$



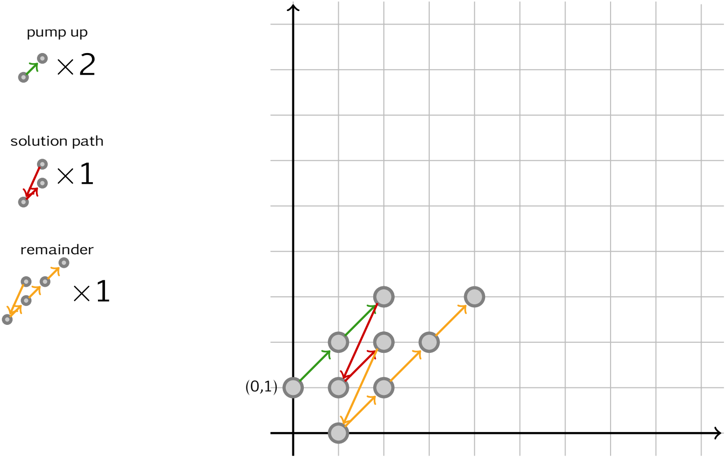
Decomposition Algorithm

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



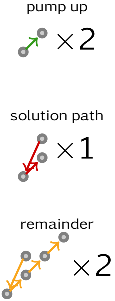
Decomposition Algorithm

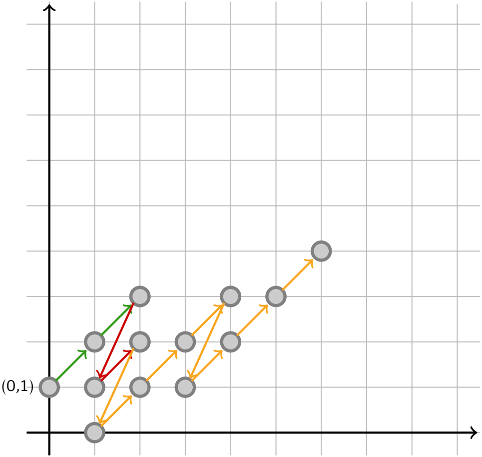
[Mayr'81, Kosaraju'82, Lambert'92]



Decomposition Algorithm

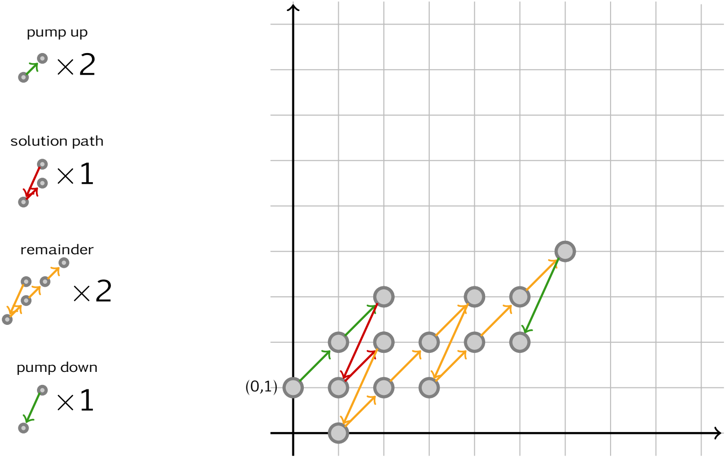
[Mayr'81, Kosaraju'82, Lambert'92]





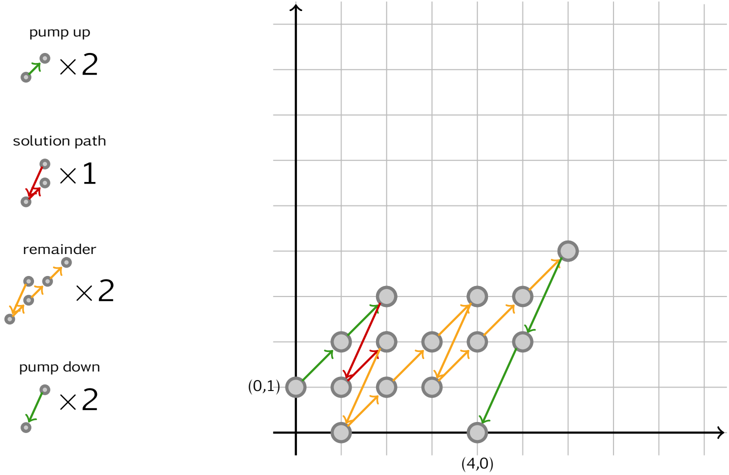
Decomposition Algorithm

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



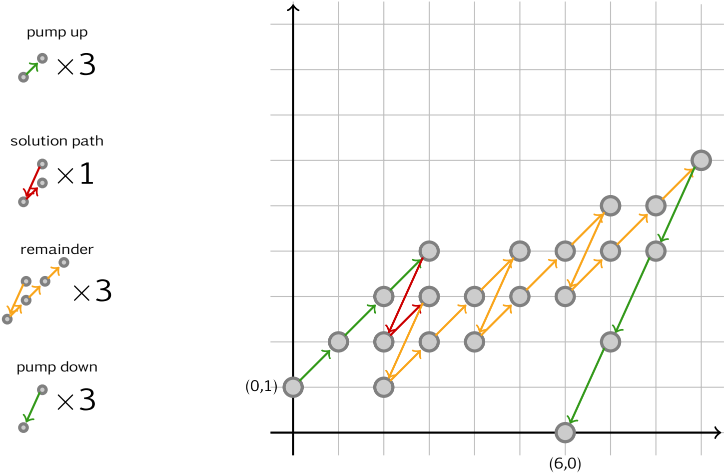
Decomposition Algorithm

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



Decomposition Algorithm

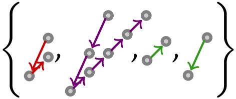
[Mayr'81, Kosaraju'82, Lambert'92]



Decomposition Algorithm

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

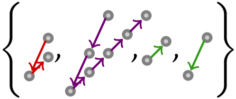
can we build a simple run?



Decomposition Algorithm

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

can we build a simple run?



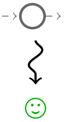
"Θ Condition"

in ExpSpace

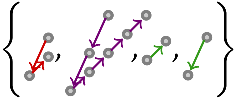
[e.g. Rackoff'78, Demri'13, Blockelet & S., MFCS'11]

Decomposition Algorithm

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



can we build a simple run? yes



"Θ Condition"

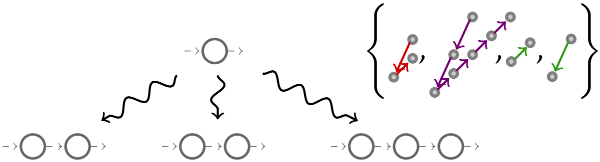
in ExpSpace

[e.g. Rackoff'78, Demri'13, Blockelet & S., MFCS'11]

Decomposition Algorithm

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

can we build a simple run? no

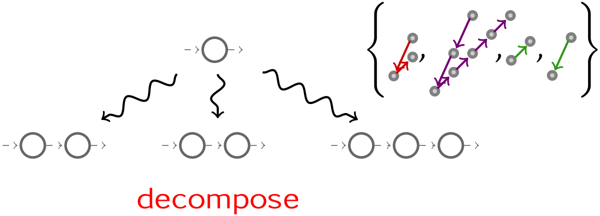


decompose

Decomposition Algorithm

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

can we build a simple run? no

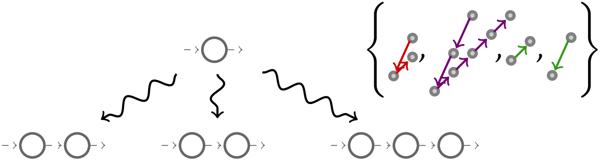


uses *coverability trees* [Karp & Miller'69]

Decomposition Algorithm

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

can we build a simple run? no



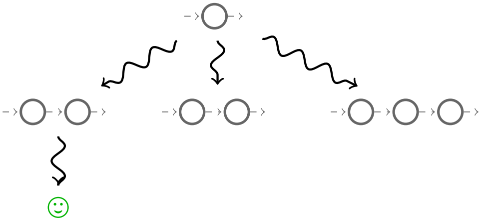
decompose

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

which use *Dickson's Lemma* [Dickson, 1913]

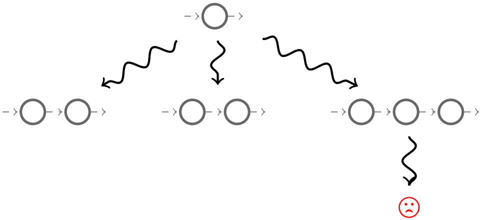
Decomposition Algorithm

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



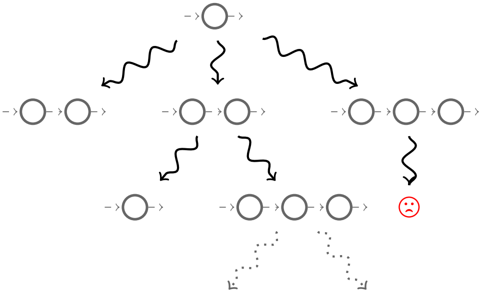
Decomposition Algorithm

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



Decomposition Algorithm

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



Termination

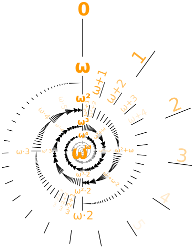
"Finally the checker has to verify that the process comes to an end. Here again he should be assisted by the programmer giving a further definite assertion to be verified. This may take the form of a quantity which is asserted to decrease continually and vanish when the machine stops."

[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]

Ranking Function

$$\omega^{\omega^2}$$

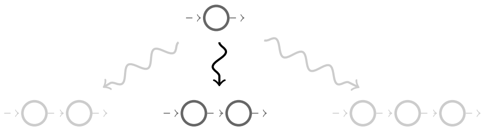
$$\vee$$

$$\alpha_0$$

Termination of the Decomposition Algorithm

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

Ranking Function



ω^{ω^2}

$\vee$

α_0

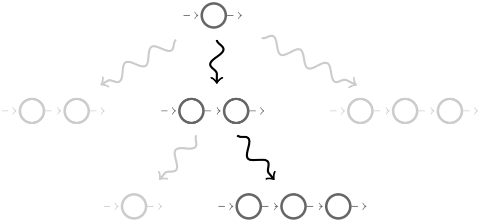
$\vee$

α_1

Termination of the Decomposition Algorithm

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

Ranking Function

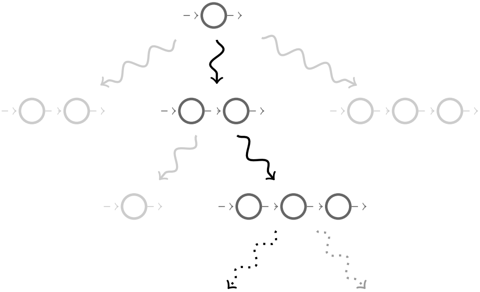


$$\omega^{\omega^2} > \alpha_0 > \alpha_1 > \alpha_2$$

Termination of the Decomposition Algorithm

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

Ranking Function



$$\omega^{\omega^2} > \alpha_0 > \alpha_1 > \alpha_2 > \vdots$$

Demystifying Reachability in Vector Addition Systems

[Leroux & S., LICS′15; S., 2017]

Upper Bound Theorem

Reachability in vector addition systems is in quadratic Ackermann.

Ideal Decomposition Theorem

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

Upper Bounds

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

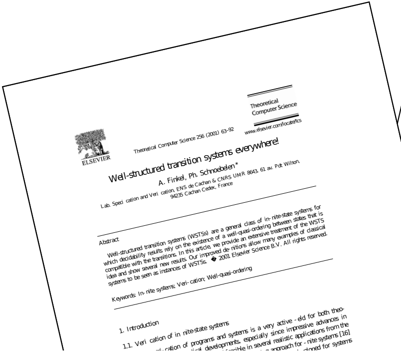
Upper Bounds

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



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

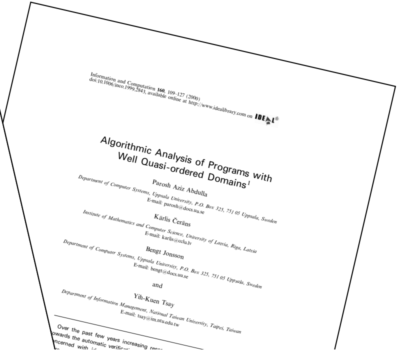
Abstract
Well-structured transition systems (WSTSs) are a general class of in-nite-state systems for which decidability results rely on the existence of a well-quasi-ordering between states that is compatible with the transitions. In this article, we provide an extensive treatment of the WSTS idea and 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

Veri cation of programs and systems is a very active -eld for both theo-



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]

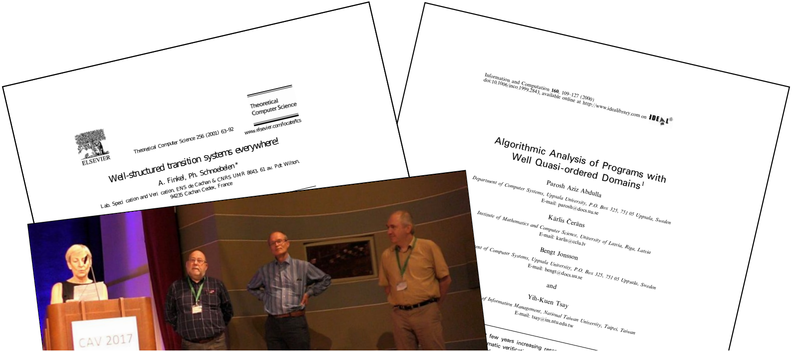
Parosh Aziz Abdulla
Department of Computer Systems, Uppsala University, P.O. Box 325, 751 05 Uppsala, Sweden
E-mail: parosh@docs.uu.se
Kārlis Čerāns
Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia
E-mail: karlis@cclu.lv
Bengt Jonsson
Department of Computer Systems, Uppsala 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

Upper Bounds

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

wqos ubiquitous in infinite-state verification



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

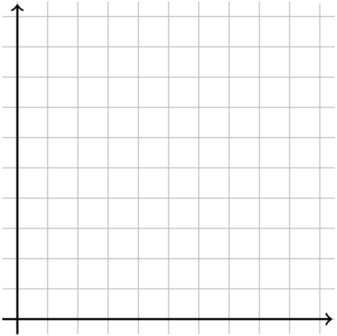
Bad Sequences

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 $\mathbb{N}^2$)

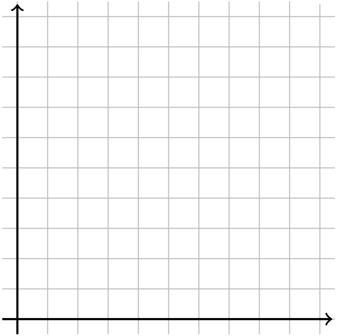
Bad Sequences

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

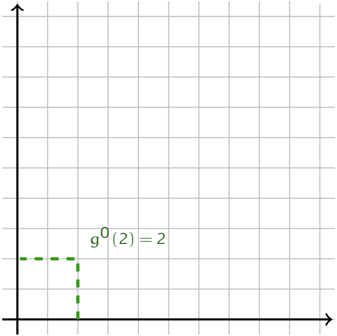
Over a qo $(X, \leqslant)$ with norm $\|\cdot\|$

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

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

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

[Cichoń & Tahhan Bittar'98]



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

Controlled Bad Sequences

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

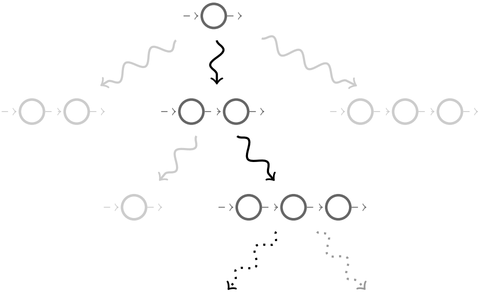
- $x_0, x_1, \ldots$ is bad if $\forall i < j \,.\, x_i \not\leqslant x_j$
- $(X, \leqslant)$ wqo iff all bad sequences are finite
- controlled by $g : \mathbb{N} \to \mathbb{N}$ and $n \in \mathbb{N}$ if $\forall i \,.\, \|x_i\| \leqslant g^i(n)$

[Cichoń & Tahhan Bittar'98]

Proposition

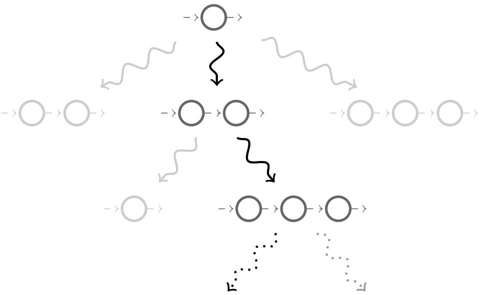
Assuming $\{x \in X \mid \|x\| \leqslant n\}$ finite $\forall n$, controlled bad sequences have bounded length.

The Length of Descending Sequences



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

The Length of Descending Sequences

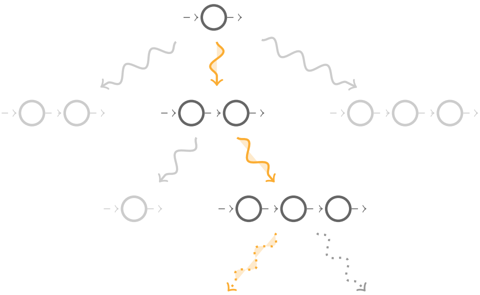


$\alpha_0 > \alpha_1 > \alpha_2 > \vdots$

Length Function Theorem (for Ordinals [invited talk S., RP'14])

Descending sequences over ω^{ω^2} controlled by Ackermannian functions are of at most quadratic Ackermannian length.

The Length of Descending Sequences

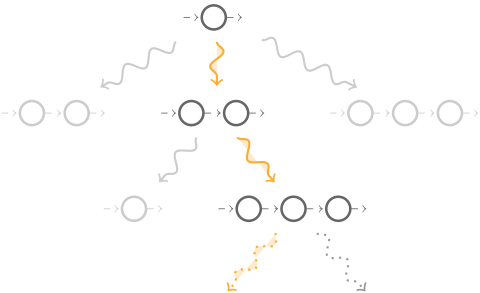


$$\alpha_0 > \alpha_1 > \alpha_2 > \cdots$$

Length Function Theorem (for Ordinals [invited talk S., RP'14]**)**

Descending sequences over ω^{ω^2} controlled by Ackermannian functions are of at most quadratic Ackermannian length.

The Length of Bad Sequences



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

Length Function Theorem (for Dickson's Lemma [Figueira, Figueira, S. & Schnoebelen, LICS'11])

Bad sequences over $\mathbb{N}^d$ controlled by primitive recursive functions are of at most Ackermannian length.

Fast-Growing Functions

Ackermann Function

$$A(1,n) = 2n$$
$$A(2,n) = 2^n$$
$$A(3,n) = \mathrm{tower}(n) \stackrel{\text{def}}{=} 2^{\cdot^{\cdot^{\cdot^{2}}}} \Big\} n \text{ times}$$
$$\vdots$$

- $\mathrm{ackermann}(n) \stackrel{\text{def}}{=} A(n,n)$ not primitive recursive
- quadratic Ackermann function F_{ω^2}: 3-arguments variant

Fast-Growing Functions

Ackermann Function

$$A(1,n) = 2n$$
$$A(2,n) = 2^n$$
$$A(3,n) = \text{tower}(n) \stackrel{\text{def}}{=} 2^{\cdot^{\cdot^{\cdot^{2}}}} \Big\} n \text{ times}$$
$$\vdots$$

- $\text{ackermann}(n) \stackrel{\text{def}}{=} A(n,n)$ not primitive recursive
- quadratic Ackermann function F_{ω^2}: 3-arguments variant

Fast-Growing Functions

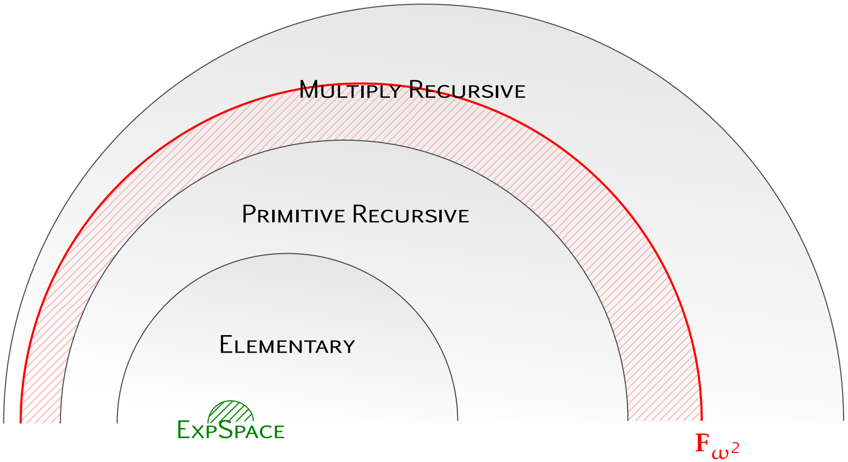
Ackermann Function

$$A(1,n) = 2n$$
$$A(2,n) = 2^n$$
$$A(3,n) = \text{tower}(n) \overset{\text{def}}{=} 2^{\cdot^{\cdot^{\cdot^{2}}}}\Big\}n \text{ times}$$
$$\vdots$$

- $\text{ackermann}(n) \overset{\text{def}}{=} A(n,n)$ not primitive recursive
- quadratic Ackermann function F_{ω^2}: 3-arguments variant

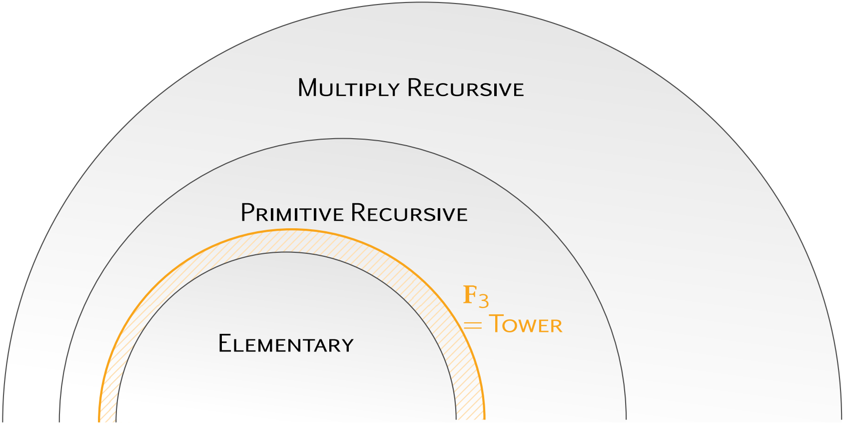
Complexity Classes Beyond Elementary

[S., ToCT'16]



Complexity Classes Beyond Elementary

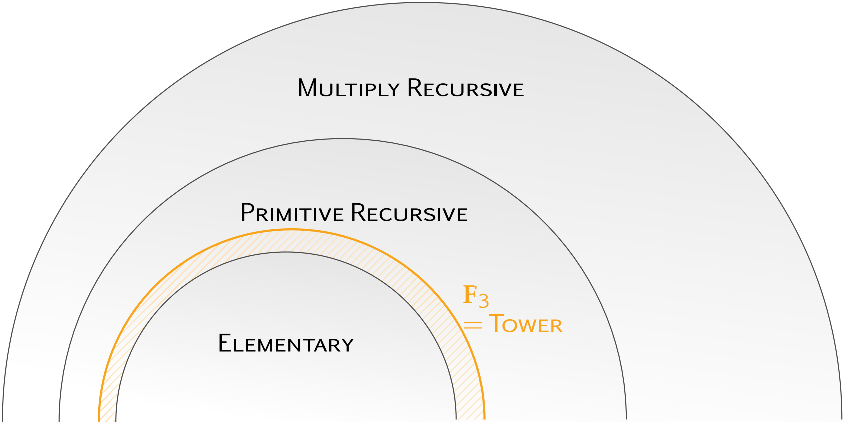
[S., ToCT'16]



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

Complexity Classes Beyond Elementary

[S., ToCT'16]

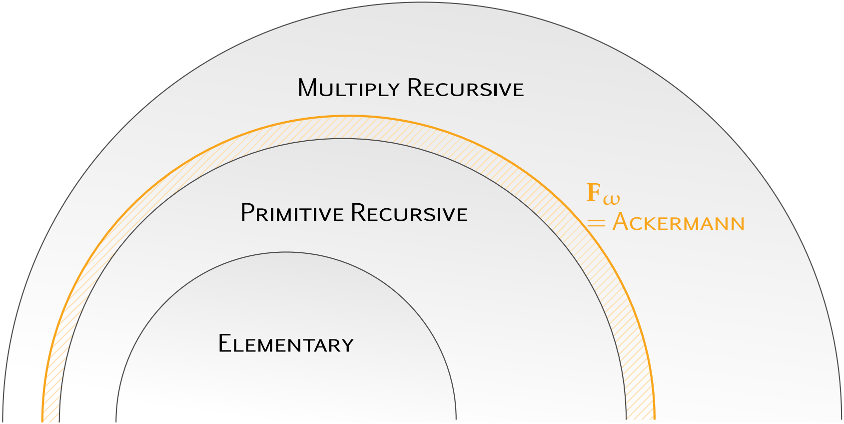


Examples of Tower-Complete Problems:

- 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

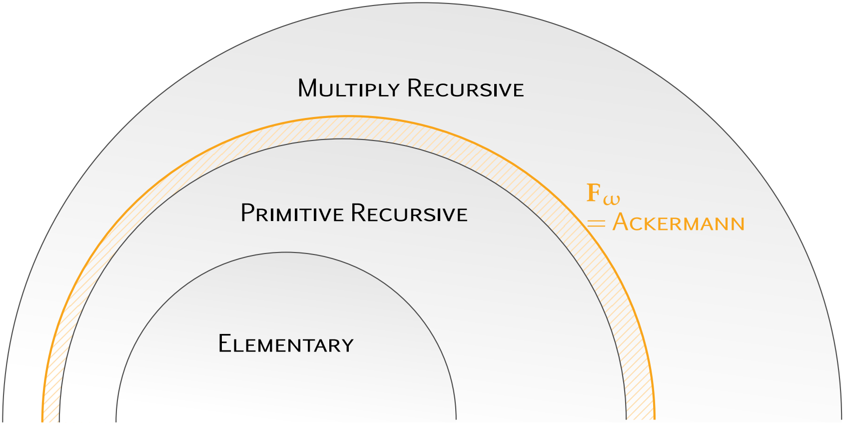
[S., ToCT'16]



$$\mathbf{F}_\omega \stackrel{\text{def}}{=} \bigcup_{p \text{ primitive recursive}} \text{DTime}\left(\text{ackermann}(p(n))\right)$$

Complexity Classes Beyond Elementary

[S., ToCT'16]

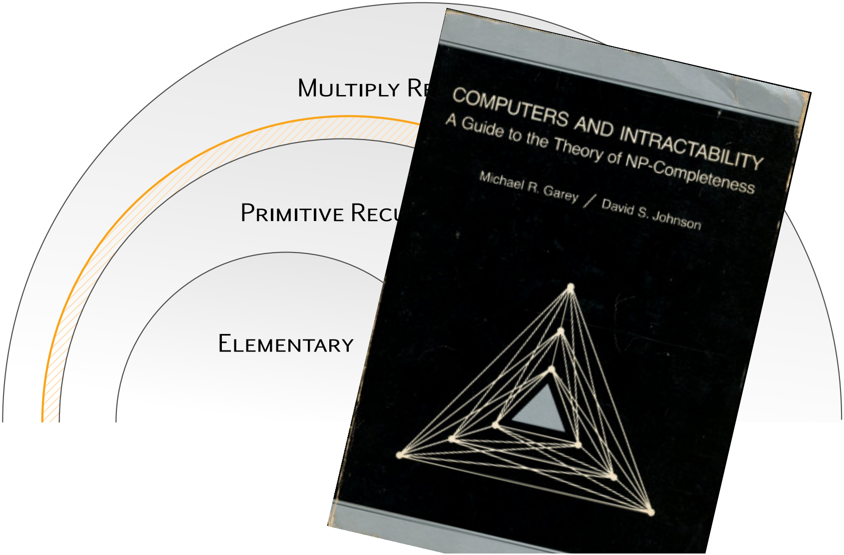


Examples of Ackermann-Complete Problems:

- reachability in lossy Minsky machines [Urquhart'98, Schnoebelen'02]
- satisfiability of safety Metric Temporal Logic [Lazić et al.'16]
- satisfiability of Forward XPath [Figueira'12]

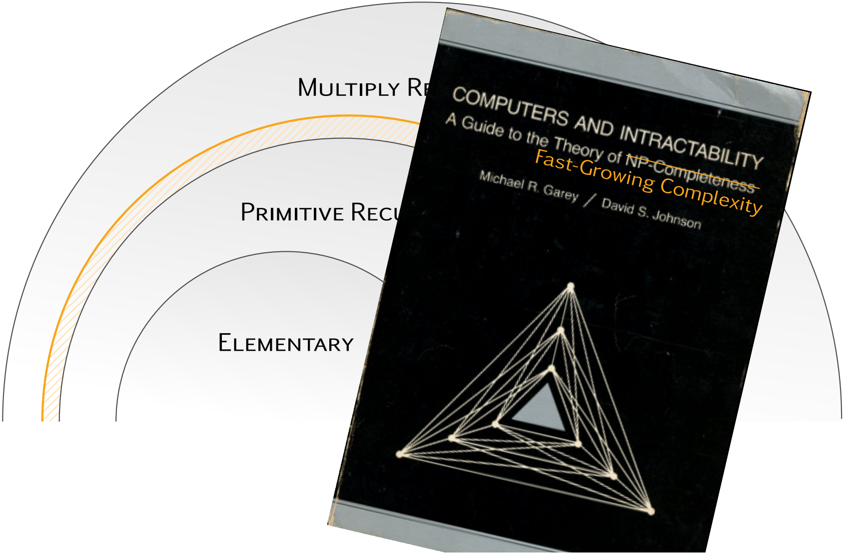
Complexity Classes Beyond Elementary

[S., ToCT'16]



Complexity Classes Beyond Elementary

[S., ToCT'16]



Summary

well-quasi-orders (wqo):

- proving algorithm termination

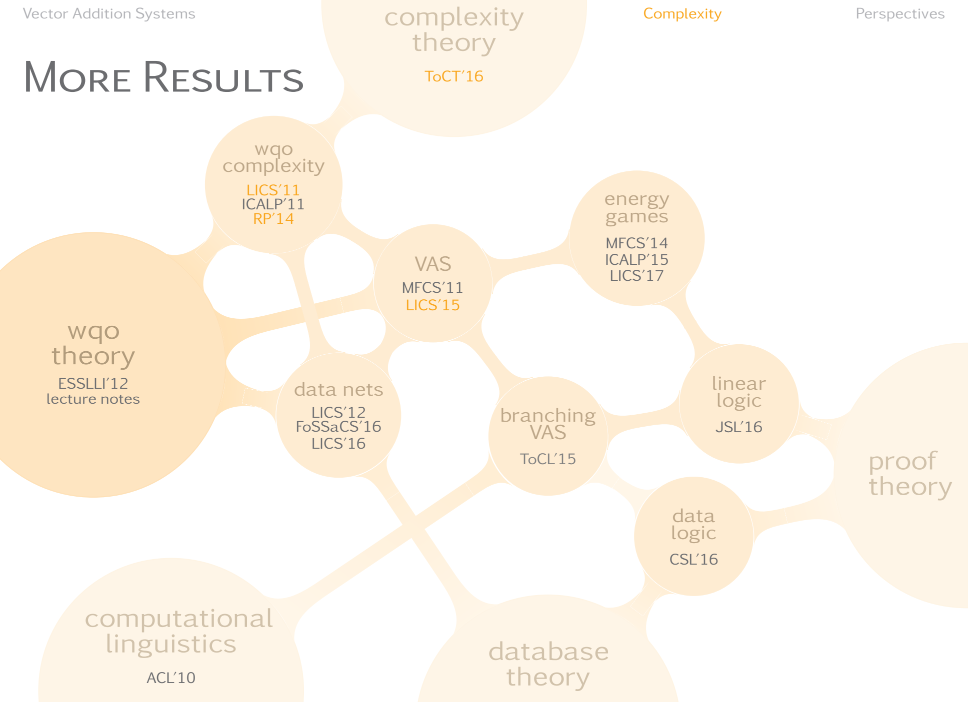
thesis: a toolbox for wqo complexity

- 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 $\mathbf{F}_{\omega^2}$

More Results



Perspectives

1. complexity gap for VAS reachability
   - ExpSpace-hard [Lipton'76]
   - decomposition algorithm: at least $\mathbf{F}_\omega$ (Ackermannian) time
2. parameterisations for counter systems
   - the dimension is the main source of complexity
   - find better parameters with tight bounds? [Kristiansen & Niggl'04]
3. beyond wqos: FAC qos, Noetherian spaces [Goubault-Larrecq'06]
   - complexity?
4. 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
   - ExpSpace-hard [Lipton'76]
   - decomposition algorithm: at least $\mathbf{F}_\omega$ (Ackermannian) time

2. parameterisations for counter systems
   - the dimension is the main source of complexity
   - find better parameters with tight bounds? [Kristiansen & Niggl'04]

3. beyond wqos: FAC qos, Noetherian spaces [Goubault-Larrecq'06]
   - complexity?

4. 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
   - ExpSpace-hard [Lipton'76]
   - decomposition algorithm: at least $\mathbf{F}_\omega$ (Ackermannian) time
2. parameterisations for counter systems
   - the dimension is the main source of complexity
   - find better parameters with tight bounds? [Kristiansen & Niggl'04]
3. beyond wqos: FAC qos, Noetherian spaces [Goubault-Larrecq'06]
   - complexity?
4. 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
   - ExpSpace-hard [Lipton'76]
   - decomposition algorithm: at least $\mathbf{F}_\omega$ (Ackermannian) time
2. parameterisations for counter systems
   - the dimension is the main source of complexity
   - find better parameters with tight bounds? [Kristiansen & Niggl'04]
3. beyond wqos: FAC qos, Noetherian spaces [Goubault-Larrecq'06]
   - complexity?
4. 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
   - ExpSpace-hard [Lipton'76]
   - decomposition algorithm: at least $\mathbf{F}_\omega$ (Ackermannian) time
2. parameterisations for counter systems
   - the dimension is the main source of complexity
   - find better parameters with tight bounds? [Kristiansen & Niggl'04]
3. beyond wqos: FAC qos, Noetherian spaces [Goubault-Larrecq'06]
   - complexity?
4. 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., LICS'15; S., 2017]

Upper Bound Theorem

Reachability in vector addition systems is in quadratic Ackermann.

Ideal Decomposition Theorem

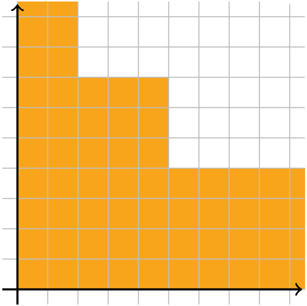
The Decomposition Algorithm computes the ideal decomposition of the set of runs from source to target.

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 $\mathbb{N}^2$)

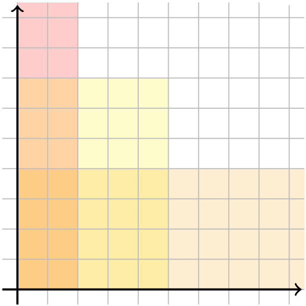
$$D = (\{0, \ldots, 2\} \times \mathbb{N}) \cup (\{0, \ldots, 5\} \times \{0, \ldots, 7\}) \cup (\mathbb{N} \times \{0, \ldots, 4\})$$

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 $I_1, \ldots, I_n$

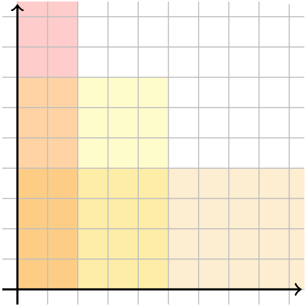


Example (over $\mathbb{N}^2$)

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

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]



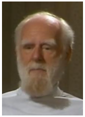
Example (over $\mathbb{N}^2$)

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

Decomposition Theorem

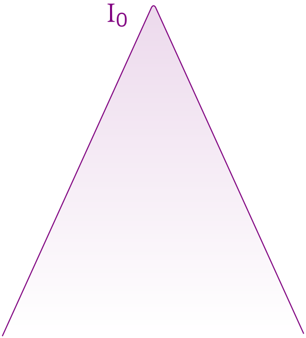
Well-Quasi-Order on Runs

combination of Dickson's and Higman's lemmata



Syntax

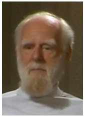
Semantics



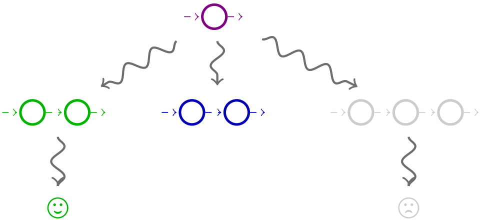
Decomposition Theorem

Well-Quasi-Order on Runs

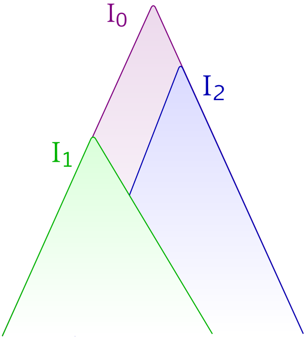
combination of Dickson's and Higman's lemmata



Syntax



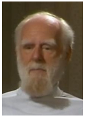
Semantics



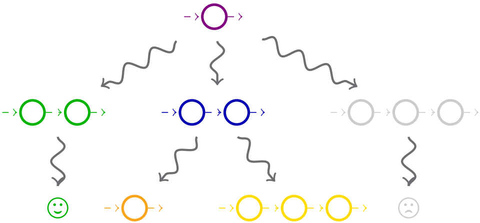
Decomposition Theorem

Well-Quasi-Order on Runs

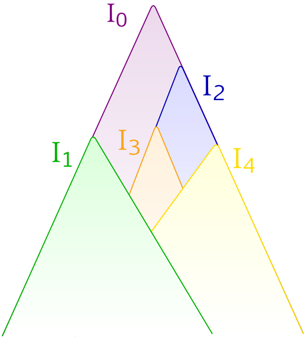
combination of Dickson's and Higman's lemmata



Syntax



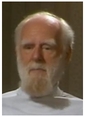
Semantics



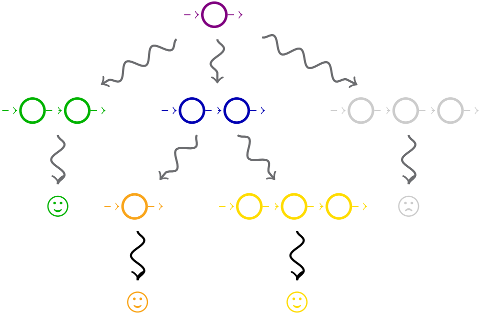
Decomposition Theorem

Well-Quasi-Order on Runs

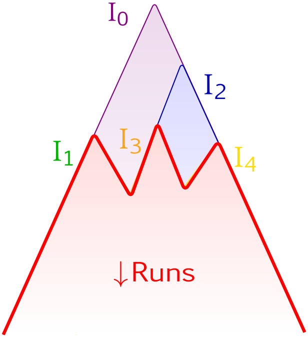
combination of Dickson's and Higman's lemmata



Syntax

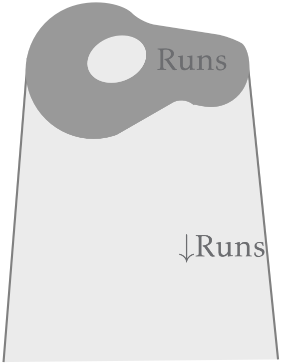


Semantics



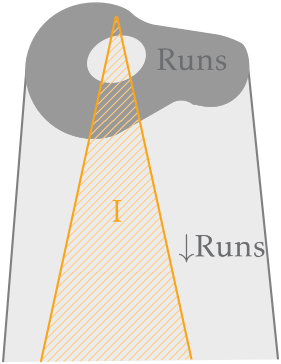
Adherence Membership

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



Adherence Membership

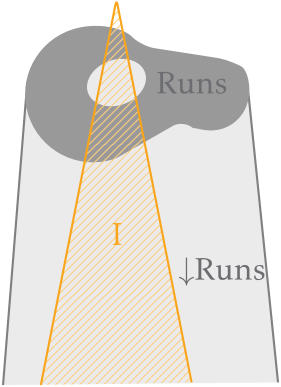
- I is adherent to Runs if $I \subseteq \downarrow(I \cap \text{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I adherent

Adherence Membership

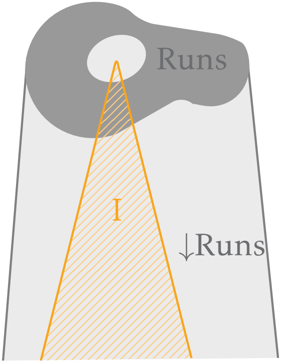
- I is adherent to Runs if $I \subseteq \downarrow(I \cap \mathrm{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I not adherent

Adherence Membership

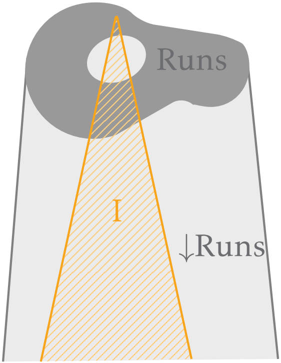
- I is adherent to Runs if $I \subseteq {\downarrow}(I \cap \text{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I not adherent

Adherence Membership

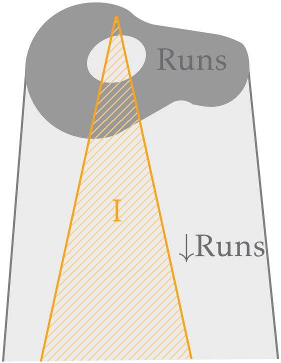
- I is adherent to Runs if $I \subseteq \downarrow(I \cap \text{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I adherent

Adherence Membership

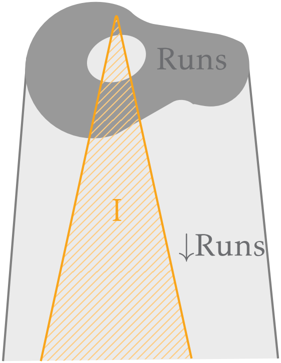
- I is adherent to Runs if $I \subseteq \downarrow(I \cap \text{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I adherent

Adherence Membership

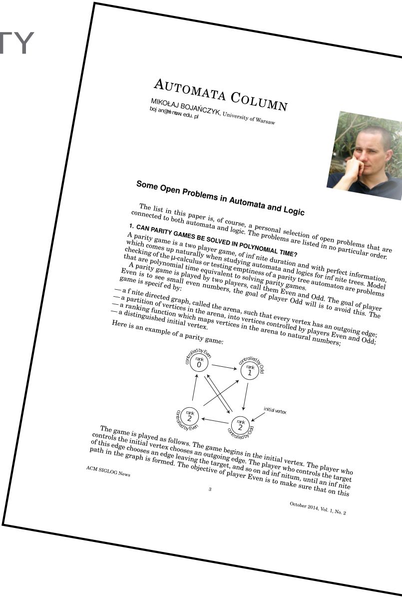
- I is adherent to Runs if $I \subseteq \downarrow(I \cap \text{Runs})$
- semantic equivalent to Θ condition
- undecidable for arbitrary ideals
- decidable for the ideals arising in the decomposition algorithm



I adherent

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 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.

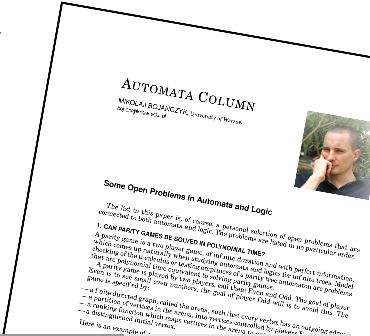
Here is an example of a parity game:

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

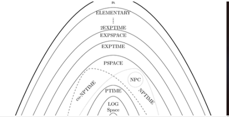
ACM SIGLOG News 3 October 2014, Vol. 1, No. 2

Branching VAS Reachability

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

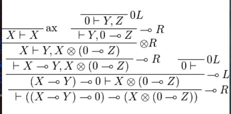

Automata Column
MIKOŁAJ BOJAŃCZYK, University of Warsaw
Some Open Problems in Automata and Logic
The list in this paper is, of course, a personal selection of open problems that are connected to both automata and logic. The problems are listed in no particular order.
1. CAN PARITY GAMES BE SOLVED IN 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
— a ranking function which maps vertices in the arena to
— a distinguished initial vertex.
Here is an example of

Complexity Theory

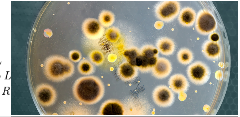


Distributed Computing

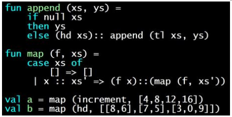
Proof Theory



Computational Biology



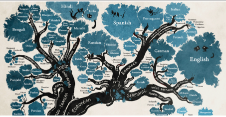
Programming Languages



Database Theory

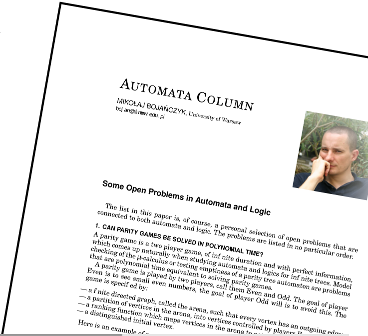
Security

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?

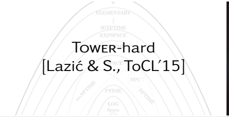
A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving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

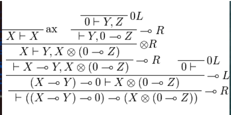
Here is an example of ...

Complexity Theory

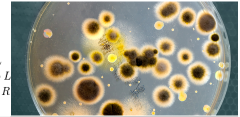


Distributed Computing

Proof Theory



Computational Biology



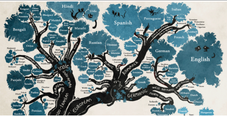
Programming Languages

```
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]])
```

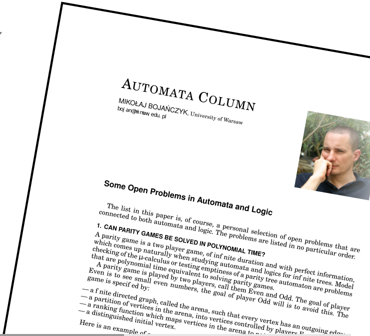
Database Theory

Security

Computational Linguistics



Branching VAS Reachability


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?

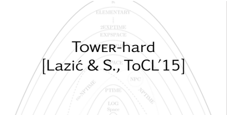
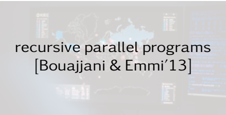
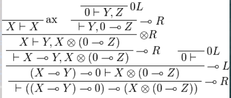
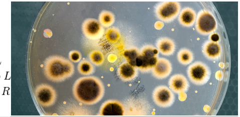
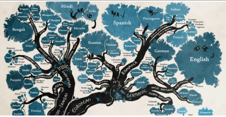
A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving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

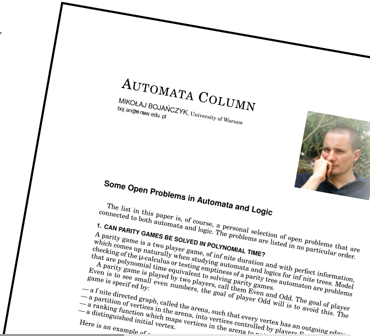
- important open problem [Bojańczyk'14]
- incorrect decidability proof in [Bimbó'15]
- application domains:

Complexity Theory	Distributed Computing	Proof Theory	Computational Biology
 Tower-hard [Lazić & S., ToCL'15]	 recursive parallel programs [Bouajjani & Emmi'13]		
Programming Languages	**Database Theory**	**Security**	**Computational Linguistics**
(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.

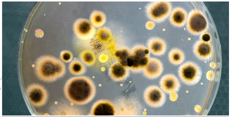
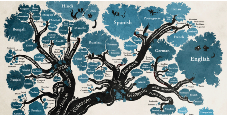
1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

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.

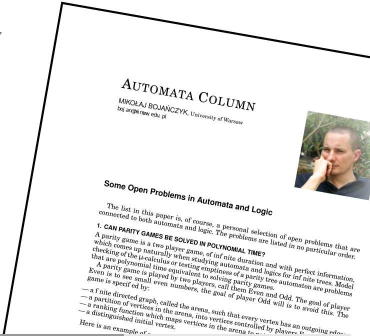
Here is an example of …

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]	
Programming Languages	**Database Theory**	**Security**	**Computational Linguistics**
(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

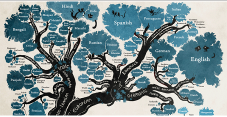
MIKOŁAJ BOJAŃCZYK, University of Warsaw

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?

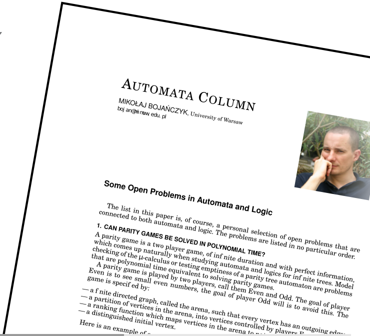
Complexity Theory	Distributed Computing	Proof Theory	Computational Biology
Tower-hard [Lazić & S., ToCL'15]	recursive parallel programs [Bouajjani & Emmi'13]	linear and relevance logics [de Groote et al.'04 Lazić & S., ToCL'15 S., JSL'16]	population protocols [Bertrand et al.'17]

Programming Languages	Database Theory	Security	Computational Linguistics
(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.

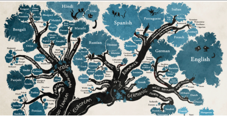
1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

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
— a ranking function which maps vertices in the arena to
— a distinguished initial vertex.

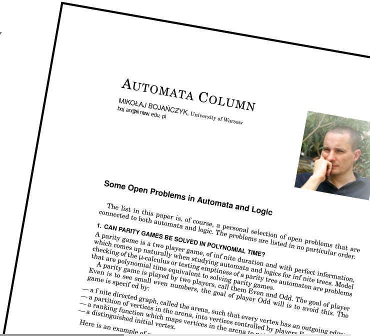
Here is an example of

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]			

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?

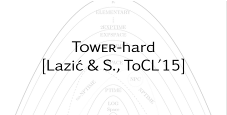
A parity game is a two player game, of infinite duration and with perfect information, which comes up naturally when studying automata and logics for infinite trees. Model checking of the μ-calculus or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to solving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

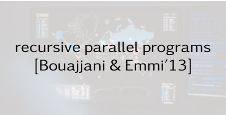
Here is an example of ...

Complexity Theory



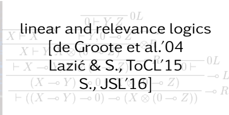
Tower-hard [Lazić & S., ToCL'15]

Distributed Computing



recursive parallel programs [Bouajjani & Emmi'13]

Proof Theory

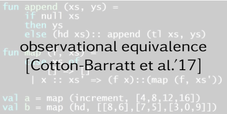


linear and relevance logics [de Groote et al.'04, Lazić & S., ToCL'15, S., JSL'16]

Computational Biology

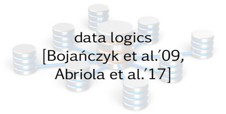
population protocols [Bertrand et al.'17]

Programming Languages



observational equivalence [Cotton-Barratt et al.'17]

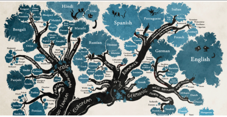
Database Theory



data logics [Bojańczyk et al.'09, Abriola et al.'17]

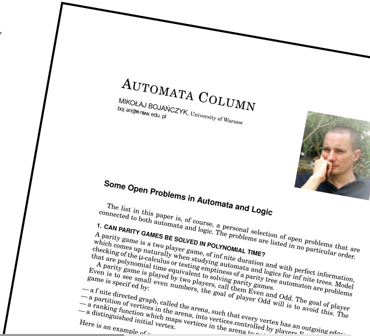
Security

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.

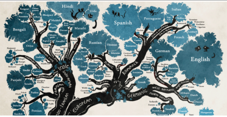
1. CAN PARITY GAMES BE SOLVED IN POLYNOMIAL TIME?

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:

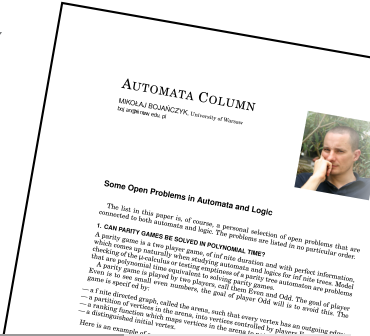
— a finite directed graph, called the arena, such that every vertex has an outgoing edge
— a partition of vertices in the arena, into vertices controlled by players E...
— a ranking function which maps vertices in the arena to n...
— a distinguished initial vertex.

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]	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:



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]

Summary

- well-quasi-orders
  ubiquitous in termination proofs
- complexity toolbox
  upper & lower bounds, fast-growing complexity classes
- application
  VAS reachability

Perspectives

1. complexity gap
   for VAS reachability
2. parameterisations
   for counter systems
3. beyond wqos
   FAC orders, Noetherian spaces
4. reachability in VAS extensions

Thank you!