Algorithmic Complexity of Well-Quasi-Orders

Sylvain Schmitz

Habilitation Defense
November 27, 2017
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
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

- Produce electricity
- Recycle uranium

(1,1)
(-1,-2)
(0,1)

Electricity

Uranium waste
Can we produce unbounded electricity with no leftover 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 →* 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 →* target?

**CENTRAL DECISION PROBLEM** [invited survey S., SIGLOG’16]

Large number of problems interreducible with reachability in vector addition systems
IMPORTANCE OF THE PROBLEM

**Reachability Problem**

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

question: source →* 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: \( \dim \geq 3 \)
not definable in Presburger arithmetic

1981
E. W. Mayr: decidability by decomposition

1982
J.-L. Lambert: decidability by decomposition

1992
S. R. Kosaraju: decidability by decomposition

2011

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 \textit{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

\[
\begin{align*}
0 + 1 \cdot a - 1 \cdot b &= c \\
1 + 1 \cdot a - 2 \cdot b &= 0
\end{align*}
\]

Solution Path
DECOMPOSITION ALGORITHM

[Mayr’81, Kosaraju’82, Lambert’92]
DECOMPOSITION ALGORITHM

[Mayr’81, Kosaraju’82, Lambert’92]
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]

\[ \begin{align*}
    1 \cdot a - 1 \cdot b &= c \\
    1 \cdot a - 2 \cdot b &= 0 \\
    a, b, c &> 0
\end{align*} \]

**Equations**

**Unbounded Path**
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]
**DECOMPOSITION ALGORITHM**

[Mayr‘81, Kosaraju‘82, Lambert‘92]

- Solution path $\times 1$
- Unbounded path $\times 3$
DECOMPOSITION ALGORITHM

[Mayr’81, Kosaraju’82, Lambert’92]

PUMPABLE PATHS

pump up  pump down
**DECOMPOSITION ALGORITHM**

[Mayr‘81, Kosaraju‘82, Lambert‘92]

**PUMPABLE PATHS**

unbounded path

---

pump up

---

pump down

= remainder
DECOMPOSITION ALGORITHM

[Mayr’81, Kosaraju’82, Lambert’92]
DECOMPOSITION ALGORITHM

[Mayr‘81, Kosaraju‘82, Lambert‘92]
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]

- Pump up: \( \times 2 \)
- Solution path: \( \times 1 \)
DECOMPOSITION ALGORITHM

[Mayr’81, Kosaraju’82, Lambert’92]
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]

- **pump up** \( \times 2 \)
- **solution path** \( \times 1 \)
- **remainder** \( \times 2 \)
DECOMPOSITION ALGORITHM

[Mayr’81, Kosaraju’82, Lambert’92]

- **Pump up**: \( \times 2 \)
- **Solution path**: \( \times 1 \)
- **Remainder**: \( \times 2 \)
- **Pump down**: \( \times 1 \)
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]

- **Pump up**: $\times 2$
- **Solution path**: $\times 1$
- **Remainder**: $\times 2$
- **Pump down**: $\times 2$
**DECOMPOSITION ALGORITHM**

[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?

\[ \Theta \text{ in ExpSpace} \]

Uses coverability trees [Karp & Miller'69], which use Dickson's Lemma [Dickson, 1913].

\[ \{ 6/23, 1/23, \ldots \} \]
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]

"Θ CONDITION"

in **ExpSpace**

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

can we build a simple run?

\[
\begin{align*}
&\rightarrow \circ \rightarrow \\
&\left\{ \text{, , ,} \right\}
\end{align*}
\]
**Decomposition Algorithm**

[Mayr’81, Kosaraju’82, Lambert’92]

- Can we build a simple run? **Yes**

\[ \Theta \] 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

\[
\begin{align*}
\text{decompose} & \\
\text{uses coverability trees} & \text{[Karp & Miller’69]} 
\end{align*}
\]
**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]
DECISION 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]
“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

\[ \omega \omega^2 \lor \alpha_0 \]
Termination of the Decomposition Algorithm

[Mayr’81, Kosaraju’82, Lambert’92]

Ranking Function

\[ \omega \omega^2 \]
\[ \lor \]
\[ \alpha_0 \]
\[ \lor \]
\[ \alpha_1 \]
Termination of the Decomposition Algorithm

[Mayr’81, Kosaraju’82, Lambert’92]

Ranking Function

\[ \omega^2 \lor \alpha_0 \lor \alpha_1 \lor \alpha_2 \]
Termination of the Decomposition Algorithm

[Mayr’81, Kosaraju’82, Lambert’92]

Ranking Function

\( \omega \omega^2 \)

\( \lor \)

\( \alpha_0 \)

\( \lor \)

\( \alpha_1 \)

\( \lor \)

\( \alpha_2 \)

\( \lor \)

\( \cdots \)
**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.
How to bound the running time of algorithms with ordinal-based termination proofs?
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
How to bound the running time of algorithms with \textit{wqo}-based termination proofs?

\textit{wqos ubiquitous in infinite-state verification}
**Bad Sequences**

Over a qo $(X, \leq)$

- $x_0, x_1, \ldots$ is **bad** if $\forall i < j. x_i \nleq x_j$
- $(X, \leq)$ wqo iff all bad sequences are **finite**
- but can be of arbitrary length
**Bad Sequences**

Over a qo $(X, \leq)$

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

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

but can be of arbitrary length

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

Over a qo $(X, \preceq)$

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

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

but can be of arbitrary length

---

**Example (over $\mathbb{N}^2$)**
**CONTROLLED BAD SEQUENCES**

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

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

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

controlled by \(g : \mathbb{N} \to \mathbb{N}\) and \(n \in \mathbb{N}\) if \(\forall i. \|x_i\| \leq g^i(n)\)

[Cichoń & Tahhan Bittar’98]

**EXAMPLE** (over \(\mathbb{N}^2\) with \(n = 2\) and \(g(n) = n + 1\))
**Controlled Bad Sequences**

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

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

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

- **controlled** by $g : \mathbb{N} \rightarrow \mathbb{N}$ and $n \in \mathbb{N}$ if $\forall i . \| x_i \| \leq g^i(n)$

[Cichoń & Tahhan Bittar’98]

**Proposition**

Assuming $\{ x \in X \mid \| x \| \leq n \}$ finite $\forall n$, **controlled bad sequences** have **bounded length**.
**The Length of Descending Sequences**

\[ \alpha_0 \lor \alpha_1 \lor \alpha_2 \lor \cdots \]

Length Function Theorem (for Ordinals)

Descending sequences over \(\omega^2\) controlled by Ackermannian functions are of at most quadratic Ackermannian length.
The Length of Descending Sequences

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

Descending sequences over $\omega^\omega$ controlled by Ackermannian functions are of at most quadratic Ackermannian length.
The Length of Descending Sequences

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

Descending sequences over $\omega^2$ controlled by Ackermannian functions are of at most quadratic Ackermannian length.
The Length of Bad Sequences

**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

\[
\begin{align*}
A(1, n) &= 2n \\
A(2, n) &= 2^n \\
A(3, n) &= \text{tower}(n) = 2 \cdot 2 \cdots 2 \text{, } n \text{ times} \\
& \quad \vdots \\
\text{ackermann}(n) &= A(n, n) \text{ not primitive recursive} \\
\text{quadratic Ackermann function } F_{\omega^2}: 3\text{-arguments variant}
\end{align*}
\]
FAST-GROWING FUNCTIONS

ACKERMANN FUNCTION

\[ A(1,n) = 2n \]
\[ A(2,n) = 2^n \]
\[ A(3,n) = \text{tower}(n) \overset{\text{def}}{=} 2 \cdot 2^2 \text{ \{} n \text{ times} \]

\[ \vdots \]

- \text{ackermann}(n) \overset{\text{def}}{=} A(n,n) \text{ not primitive recursive}

- \text{quadratic Ackermann function } F_{\omega^2} : 3\text{-arguments variant}
FAST-GROWING FUNCTIONS

ACKERMANN FUNCTION

\[ A(1, n) = 2n \]
\[ A(2, n) = 2^n \]
\[ A(3, n) = \text{tower}(n) \overset{\text{def}}{=} 2 \cdot 2 \cdot \ldots \cdot 2 \text{ } n \text{ times} \]

\[ \vdots \]

- \text{ackermann}(n) \overset{\text{def}}{=} A(n, n) \text{ not primitive recursive}

- \textbf{quadratic Ackermann function} \( F_{\omega^2} \): 3-arguments variant
Complexity Classes Beyond Elementary

[S., ToCT’16]
**Complexity Classes Beyond Elementary**

[S., ToCT’16]

Mathematical expressions:

- \( F_3 \) defined as the union of DTIME(
multiply recursive functions)

\[
F_3 \overset{\text{def}}{=} \bigcup_{e \text{ elementary}} \text{DTIME}\left(\text{tower}(e(n))\right)
\]
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]

*Multify Recursive*

*Primitive Recursive*

*Elementary*

\[ F_\omega \overset{\text{def}}{=} \bigcup_{p \text{ primitive recursive}} \text{DTIME} (\text{ackermann}(p(n))) \]

\[ F_\omega = \text{ACKERMANN} \]
**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]

- **Elementary**
- **Primitive Recursive**
- **Multiply Recursive**

**Fast-Growing Complexity**

Fast-Growing Complexity in

- **$F_2$**
- **$F_{\omega}$**
- **$F_{\omega^2}$**

$F_{\omega^2}$ def $= \bigcup_{p \in F_{\omega}}$ DTime$(F_{\omega}(p(n)))$
**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: \((F_\alpha)_\alpha\)

this talk: focus on one problem
- reachability in vector addition systems
  in \(F_{\omega^2}\)
Perspectives

1. Complexity gap for VAS reachability
   ▶ \textsc{ExpSpace}-hard \cite{Lipton'76}
   ▶ Decomposition algorithm: at least $F_\omega$ (Ackermannian) time

2. Parameterisations for counter systems
   ▶ The dimension is the main source of complexity
   ▶ Find better parameters with tight bounds? \cite{Kristiansen & Niggl'04}

3. Beyond WQOS: FAC qos, Noetherian spaces \cite{Goubault-Larrecq'06}
   ▶ Complexity?

4. Reachability in VAS extensions
   ▶ Decidable in VAS with hierarchical zero tests \cite{Reinhardt'08}
   ▶ What about
     ▶ Branching VAS
     ▶ Unordered data Petri nets
     ▶ Pushdown VAS
**PERSPECTIVES**

1. **complexity gap for VAS reachability**
   - \(\text{ExpSpace-hard} \ [\text{Lipton’76}]\)
   - decomposition algorithm: at least \(F_\omega\) (Ackermannian) time

2. **parameterisations for counter systems**
   - the dimension is the main source of complexity
   - find better parameters with tight bounds? \[\text{Kristiansen & Niggl’04}\]

3. **beyond wqos: FAC qos, Noetherian spaces** \[\text{Goubault-Larrecq’06}\]
   - complexity?

4. **reachability in VAS extensions**
   - decidable in VAS with hierarchical zero tests \[\text{Reinhardt’08}\]
   - 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 $F_\omega$ (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
   ▶ \textsc{ExpSpace}-hard \cite{Lipton'76}
   ▶ decomposition algorithm: at least $\mathbb{F}_\omega$ (Ackermannian) time

2. parameterisations for counter systems
   ▶ the dimension is the main source of complexity
   ▶ find better parameters with tight bounds? \cite{Kristiansen & Niggl’04}

3. beyond wqos: FAC qos, Noetherian spaces \cite{Goubault-Larrecq’06}
   ▶ complexity?

4. reachability in VAS extensions
   ▶ decidable in VAS with hierarchical zero tests \cite{Reinhardt’08}
   ▶ what about
     ▶ branching VAS
     ▶ unordered data Petri nets
     ▶ pushdown VAS
PERSPECTIVES

1. complexity gap for VAS reachability
   ▶ \textsc{ExpSpace}-hard [Lipton’76]
   ▶ decomposition algorithm: at least $F_\omega$ (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

[Geroux & 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 \((X, \leq)\)

- Canonical decompositions
  \[\text{[Bonnet'75]}\]
  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 \((X, \leq)\)

- Canonical decompositions [Bonnet’75]
  
  \[
  \text{if } D \subseteq X \text{ is } \downarrow\text{-closed, then} \]
  \[
  D = I_1 \cup \cdots \cup I_n
  \]
  
  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** \((X, \preceq)\)

- 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 \(\mathbb{N}^2\))**

\[
D = \left[ (2, \infty) \right] \cup \left[ (5, 7) \right] \cup \left[ (\infty, 4) \right]
\]
**Decomposition Theorem**

**Well-Quasi-Order on Runs**

combination of Dickson’s and Higman’s lemmata
**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
**Decomposition Theorem**

**Well-Quasi-Order on Runs**

combination of Dickson’s and Higman’s lemmata
**Adherence Membership**

- I is adherent to Runs if $I \subseteq \downarrow (I \cap \text{Runs})$
- Semantic equivalent to $\Theta$ 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 \( \Theta \) 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 $\Theta$ 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 $\Theta$ condition

- undecidable for arbitrary ideals

- decidable for the ideals arising in the decomposition algorithm
**Adherence Membership**

- $I$ is adherent to $\text{Runs}$ if $I \subseteq \downarrow(I \cap \text{Runs})$
- Semantic equivalent to $\Theta$ 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 $\Theta$ 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 $\Theta$ condition
- Undecidable for arbitrary ideals
- Decidable for the ideals arising in the decomposition algorithm
**Branching VAS Reachability**

- important open problem [Bojańczyk’14]

- incorrect decidability proof in [Bimbó’15]

- application domains:
  - Complexity Theory
  - Distributed Computing
  - Computational Biology
  - Proof Theory
  - Database Theory
  - Programming Languages
  - Security
  - Computational Linguistics
Branching VAS Reachability

- important open problem [Bojańczyk’14]
- incorrect decidability proof in [Bimbó’15]
- application domains:
Branching VAS Reachability

- important open problem [Bojańczyk’14]

- incorrect decidability proof in [Bimbó’15]

- application domains:
  - Complexity Theory
  - Distributed Computing
  - Proof Theory
  - Computational Biology
  - Database Theory
  - Security
  - Programming Languages
  - Computational Linguistics

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. Modeling checking of the propositional or first-order logic of a parity tree automaton are problems that are polynomial-time equivalent to solving parity games.
   A parity game is played by two players, called Even and Odd. The goal of player Even is to see small even numbers, the goal of player Odd is to avoid that. The game is played on a tree, in which
   — a finite directed graph, called the arena, such that every vertex has an outgoing edge,
   — a ranking function which maps vertices in the arena to natural numbers,
   — a distinguished initial vertex.

   Here is an example of a parity game:

   \[
   \begin{align*}
   X &\rightarrow Y, X \otimes (0 \rightarrow Z) &\rightarrow R \\
   Y &\rightarrow 0, Z \rightarrow R &\rightarrow L \\
   X &\rightarrow X, X \otimes (0 \rightarrow Z) &\rightarrow R
   \end{align*}
   \]

   fun append (xs, ys) =
   | null xs => 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:**
  - **Complexity Theory**
  - **Distributed Computing**
  - **Proof Theory**
  - **Computational Biology**
  - **Programming Languages**
  - **Database Theory**
  - **Security**
  - **Computational Linguistics**

### Complexity Theory

- Tower-hard
- [Lazić & S., ToCL’15]

### Distributed Computing

- recursive parallel programs
- [Bouajjani & Emmi’13]

### Proof Theory

- \( X + X \xrightarrow{\text{ax}} \vdash Y, Z \xrightarrow{0L} R \)
- \( X + Y, X \otimes (0 \rightarrow Z) \xrightarrow{\ominus R} R \)
- \( X \rightarrow Y, X \otimes (0 \rightarrow Z) \xrightarrow{\ominus R} R \)
- \( \vdash \vdash (X \rightarrow Y) \rightarrow 0 \leftarrow X \otimes (0 \rightarrow Z) \)
- \( \vdash ((X \rightarrow Y) \rightarrow 0) \leftarrow (X \otimes (0 \rightarrow Z)) \rightarrow R \)

### Computational Biology

- Automata Column

### Some Open Problems in Automata and Logic

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 state systems.
   - Modeling checking of the protocol or testing compliance of a parity tree automaton are problems that are polynomial time approximable to winning parity games.
   - A parity game is played by two players, until one of them wins and the other loses. The goal of player even is to win as many moves as possible; the goal of player odd is to avoid this. The game is won by:
   - a finite directed graph, called the arena, such that every vertex has an outgoing edge
   - a reaching function which maps vertices to the arena
   - a distinguished initial vertex
   - Here is an example of a

```haskell
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:
  - Complexity Theory
  - Distributed Computing
  - Proof Theory
  - Computational Biology
  - Linear and relevance logics [de Groote et al.’04]
  - Recursive parallel programs [Bouajjani & Emmi’13]
  - 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:

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

  Database Theory
  - `val a = map (increment, [4,8,12,16])`

  Security
  - `val b = map (hd, [[8,6],[7,5],[3,0,9]])`

  Computational Linguistics
**Branching VAS Reachability**

- **important open problem** [Bojańczyk’14]
- **incorrect decidability proof in** [Bimbó’15]
- **application domains:**
  - Complexity Theory
  - Distributed Computing
  - Proof Theory
  - Computational Biology
  - Programming Languages
  - Database Theory
  - Security
  - Computational Linguistics

---

**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]

**Computational Biology**

- population protocols
  - [Bertrand et al.’17]

---

**Programming Languages**

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

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

---

**Security**

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

```
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:
  - 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:

<table>
<thead>
<tr>
<th>Complexity Theory</th>
<th>Distributed Computing</th>
<th>Proof Theory</th>
<th>Computational Biology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tower-hard</td>
<td>recursive parallel programs</td>
<td>linear and relevance logics</td>
<td>population protocols</td>
</tr>
<tr>
<td>[Lazić &amp; S., ToCL’15]</td>
<td>[Bouajjani &amp; Emmi’13]</td>
<td>[de Groote et al.’04, Lazić &amp; S., ToCL’15, S., JSL’16]</td>
<td>[Bertrand et al.’17]</td>
</tr>
</tbody>
</table>

Programming Languages

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

observational equivalence
[Colton-Barratt et al.’17]

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

Database Theory

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

Security

security protocols
[Verma & Goubault-Larrecq’05]

Computational Linguistics
Branching VAS Reachability

- important open problem [Bojańczyk’14]
- incorrect decidability proof in [Bimbó’15]
- application domains:
  
  **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, 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**
  - security protocols [Verma & Goubault-Larrecq’05]

  **Computational Linguistics**
  - 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!