Algorithmic Complexity of Well-Quasi-Orders

Sylvain Schmitz

Séminaire Automates, IRIF

February 9, 2018
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
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**

**Springfield Power Plant**

- Produce electricity
- Recycle uranium

Vector (1,1) and (0,1): Uranium waste and electricity coordinates.
Can we produce unbounded electricity with no left-over uranium waste?
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 $\rightarrow^*$ 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
- **1969**: R. J. Lipton: \( \text{EXPSPACE} \) lower bound
- **1976**: J. E. Hopcroft & J.-J. Pansiot: \( \text{dim.} \geq 3 \) not definable in Presburger arithmetic
- **1979**: E. W. Mayr: decidability by decomposition
- **1981**: S. R. Kosaraju: decidability by decomposition
- **1982**: J.-L. Lambert: decidability by decomposition
- **1992**: J. Leroux: decidability by Presburger inductive invariants
- **2011**: 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**

\[
0 + 1 \cdot a - 1 \cdot b = c
\]

\[
1 + 1 \cdot a - 2 \cdot b = 0
\]

**Solution Path**

\[
(0, 1) \rightarrow \rightarrow \rightarrow (\infty, 0)
\]
DECOMPOSITION ALGORITHM

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

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

[Mayr’81, Kosaraju’82, Lambert’92]

\[ a = (1,1), \quad b = (-1,-2), \quad c = (\infty,0) \]

EQUATIONS

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

UNBOUNDED PATH

(2,0) → (∞,0)

(0,1) → (c) → (∞,0)
DECOMPOSITION ALGORITHM

[Mayr’81, Kosaraju’82, Lambert’92]

- Solution path: \( \times 1 \)
- Unbounded path: \( \times 1 \)
**DECOMPOSITION ALGORITHM**

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

[Mayr’81, Kosaraju’82, Lambert’92]
**Decomposition Algorithm**

[Mayr’81, Kosaraju’82, Lambert’92]

**Pumpable Paths**

![Diagram of pumpable paths](image-url)
**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]

pump up

\[
\times 1
\]

(0,1)
**DECOMPOSITION ALGORITHM**

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

[Mayr’81, Kosaraju’82, Lambert’92]

pump up

$\times 2$

solution path

$\times 1$
DECOMPOSITION ALGORITHM

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

[Mayr’81, Kosaraju’82, Lambert’92]

- **pump up** × 2
- **solution path** × 1
- **remainder** × 2

Graph showing the decomposition algorithm with vectors and arrows pointing to different points on the grid.
**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]
DECOMPOSITION ALGORITHM

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

[Mayr’81, Kosaraju’82, Lambert’92]

Can we build a simple run?

\[ \Theta \text{Condition} \]

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

Decompose uses coverability trees [Karp & Miller’69] which use Dickson’s Lemma [Dickson, 1913].
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]

Can we build a simple run?

\[
\Theta \ \text{Condition}
\]

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

\[ \Theta \text{ Condition} \in \text{ExpSpace} \]  
  - e.g. Rackoff ’78, Demri’13, Blockelet & S., MFCS’11

- uses coverability trees  
  - [Karp & Miller’69]

- which use Dickson’s Lemma  
  - [Dickson, 1913]
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]

Can we build a simple run? **no**

Decompose

*uses coverability trees* [Karp & Miller’69]
**DECOMPOSITION ALGORITHM**

[Mayr’81, Kosaraju’82, Lambert’92]

---

**can we build a simple run?**  
```plaintext
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]
“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

\[ \omega \omega^2 \]

\[ \vee \]

\[ \alpha_0 \]

\[ \vee \]

\[ \alpha_1 \]

\[ \vee \]

\[ \alpha_2 \]
Termination of the Decomposition Algorithm

Ranking Function

\[ \omega \omega^2 \]

\[ \land \land \land \]

\[ \alpha_0 \land \alpha_1 \land \alpha_2 \land \ldots \]
**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 \textit{wqo}-based termination proofs?
How to bound the running time of algorithms with \textit{wqo}-based termination proofs?

\textit{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 \(\forall i < j. x_i \not\leq x_j\)

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

- but can be of arbitrary length
**Bad Sequences**

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

\(x_0, x_1, ...\) 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 $q_0$ $(X, \leq)$

$x_0, x_1, \ldots \text{ is bad if } \forall i < j \cdot x_i \nleq x_j$

$(X, \leq)$ 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 $\text{norm } \| \cdot \|$  
$x_0, x_1, \ldots$ is bad if $\forall i < j . x_i \not\preceq x_j$

$(X, \leq)$ 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

\[ \alpha_0 \lor \alpha_1 \lor \alpha_2 \ldots \]

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 Descending Sequences

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

Descending sequences over $\omega \omega^2$ 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 $\omega^2$ controlled by Ackermannian functions are of at most quadratic Ackermannian length.
THE LENGTH OF BAD SEQUENCES

\[ \alpha_0 \lor \alpha_1 \lor \alpha_2 \lor \ldots \]

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) \equiv 2 \cdot 2 \cdot \ldots \cdot 2 \text{ \(n\) times} \\
&\vdots
\end{align*}
\]

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

- Quadratic Ackermann function \(F_{\omega^2} \): 3-arguments variant
**Fast-Growing Functions**

**Ackermann Function**

\[
\begin{align*}
A(1,n) &= 2n \\
A(2,n) &= 2^n \\
A(3,n) &= \text{tower}(n) \overset{\text{def}}{=} 2 \cdot 2 \cdots 2 \} \text{n times} \\
& \quad \vdots \\
\end{align*}
\]

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

- Quadratic Ackermann function \( F_{\omega^2} \): 3-arguments variant
**FAST-GROWING FUNCTIONS**

**ACKERMANN FUNCTION**

\[
\begin{align*}
A(1,n) &= 2n \\
A(2,n) &= 2^n \\
A(3,n) &= \text{tower}(n) = 2 \cdot \ldots \cdot 2 \text{ } n \text{ times} \\
\vdots
\end{align*}
\]

- \text{ackermann}(n) \overset{\text{def}}{=} A(n,n) \text{ not primitive recursive}
- \text{quadratic Ackermann} function \( F_{\omega^2} \): 3-arguments variant
Complexity Classes Beyond Elementary

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

[S., ToCT’16]

\( \mathbb{F}_3 \overset{\text{def}}{=} \bigcup_{e \text{ elementary}} \text{DTIME}(\text{tower}(e(n))) \)
**Complexity Classes Beyond Elementary**

[S., ToCT’16]

**Examples of Tower-Complete Problems:**
- Satisfiability of first-order logic on words [Meyer’75]
- $\beta$-equivalence of simply typed $\lambda$ terms [Statman’79]
- Model-checking higher-order recursion schemes [Ong’06]
Complexity Classes Beyond Elementary

[S., ToCT'16]

\[ F_\omega \overset{\text{def}}{=} \bigcup_{p \text{ primitive recursive}} \text{DTime} (\text{ackermann}(p(n))) \]
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 Vertical XPath [Figueira and Segoufin’17]
**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: \((F_\alpha)_\alpha\)

this talk: focus on one problem

- reachability in vector addition systems in \(F_{\omega^2}\)
MORE RESULTS

- **wqo theory**
  - ESSLII’12 lecture notes

- **wqo complexity**
  - LICS’11
  - ICALP’11
  - RP’14

- **VAS**
  - MFCS’11
  - LICS’15

- **data nets**
  - LICS’12
  - FoSSaCS’16
  - LICS’16

- **energy games**
  - MFCS’14
  - ICALP’15
  - LICS’17

- **branching VAS**
  - ToCL’15

- **linear logic**
  - JSL’16

- **data logic**
  - CSL’16

- **proof theory**

- **complexity theory**
  - ToCT’16

- **database theory**

- **computational linguistics**
  - ACL’10
PERSPECTIVES

1. complexity gap for VAS reachability
   - \textbf{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
   - 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
   - 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 [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**
   - \( \text{ExpSpace-hard} \) [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

[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 $(X, \preceq)$

- Canonical decompositions [Bonnet’75]
  
  If $D \subseteq X$ is $\downarrow$-closed, then
  
  $$D = I_1 \cup \cdots \cup I_n$$
  
  for (maximal) ideals $I_1, \ldots, I_n$

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
  
  \([\text{Bonnet'75}]\)
  
  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**
  
  \([\text{Bonnet'75}]\)

  If \(D \subseteq X\) is \(\downarrow\)-closed, then

  \[
  D = I_1 \cup \cdots \cup I_n
  \]

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

- **Effective representations**
  
  \([\text{Goubault-Larrecq et al.'17}]\)

**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**

\[ \rightarrow O \rightarrow \]

**Semantics**

\[ I_0 \]
**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

I₀

I₁

I₂

I₃

I₄
**Decomposition Theorem**

**Well-Quasi-Order on Runs**
combination of Dickson’s and Higman’s lemmata

**Syntax**

**Semantics**

\[ \downarrow \text{Runs} \]
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 $\text{Runs}$ 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
**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
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
  - Programming Languages
  - Database Theory
  - Security
  - 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 existential fragments of a parity tree automaton are problems that are polynomial time solvable in even parity games.
   - A parity game is played by two players, called Odd and Even. 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 ranking function which maps vertices in the arena to natural numbers

---

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
  - Programming Languages
  - Database Theory
  - 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
  - Programming Languages
  - Database Theory
  - Security
  - 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. Model checking of the fragment of existential second-order logic or the parity tree automaton is problems that are polynomial time equivalent to winning parity games.

2. 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 them. The game is won by
   - a finite directed graph, called the arena, such that every vertex has an outgoing edge,
   - a ranking function which maps vertices to the arena to a natural number,
   - a distinguished initial vertex.

Here is an example of a parity game:

```
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
  - Programming Languages
  - Database Theory
  - Security
  - 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 logic for infinite games. Model checking of the parity or winning condition of a parity tree automaton is a problem that is polynomial time equivalent to winning 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 will be to avoid this. The game is won by player Even if we can construct a directed graph, called the arena, such that every vertex has an outgoing edge, a reaching function which maps vertices to the arena to itself, and a distinguished initial vertex.

Here is an example of a parity game:

```
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
  - Programming Languages
  - Database Theory
  - Security
  - 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 logic for infinite trees. Model-checking of the propositional or temporal fragments of a parity tree automaton are problems that are polynomial-time equivalent to winning 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 will is to avoid this. The game is won if the existence result is used by:
     - 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 numbers,
     - a distinguished initial vertex.
   - Here is an example of an arena with a single vertex:

```
<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
```

- linear and relevance logics [de Groote et al.’04]
- recursive parallel programs [Bouajjani & Emmi’13]
- observational equivalence [Cotton-Barratt et al.’17]
- population protocols [Bertrand et al.’17]
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, de Groote et al.’16, Bouajjani &amp; Emmi’13]</td>
<td>[Bertrand et al.’17]</td>
</tr>
</tbody>
</table>

Programming Languages

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

Database Theory

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

Security

- observational equivalence [Cotton-Barratt et al.’17]
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, Z]
    - observational equivalence [Cotton-Barratt et al.’17]
  - Programming Languages
    - fun append (xs, ys) =
      | null xs    then ys
      | else (hd xs) :: append (tl xs, ys)
  - Database Theory
    - data logics [Bojańczyk et al.’09, Abriola et al.’17]
  - Security
    - security protocols [Verma & Goubault-Larrecq’05]
  - Computational Biology
    - population protocols [Bertrand et al.’17]
  - Computational Linguistics
    - 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]</td>
<td>[Bertrand et al.’17]</td>
</tr>
<tr>
<td>Programming Languages</td>
<td>Database Theory</td>
<td>Security</td>
<td>Computational Linguistics</td>
</tr>
<tr>
<td>fun append (xs, ys) = if null xs then ys else (hd xs): append tl xs, ys</td>
<td>data logics</td>
<td>security protocols</td>
<td>dominance grammars</td>
</tr>
<tr>
<td>observational equivalence [Cotton-Barratt et al.’17]</td>
<td>[Bojańczyk et al.’09, Abriola et al.’17]</td>
<td>[Verma &amp; Goubault-Larrecq’05]</td>
<td>[Rambow’94; S., ACL’10]</td>
</tr>
<tr>
<td>val a = map (increment, [4,8,12,16])</td>
<td>val b = map (hd, [[8,6],[7,5],[3,0,9]])</td>
<td>minimalist syntax [Salvati’10]</td>
<td></td>
</tr>
</tbody>
</table>
**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!