On the Complexity of VAS Reachability

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

Séminaire LIGM
Outline

well-quasi-orders (wqo):

- proving algorithm termination

a toolbox for wqo complexity

- upper bounds

- lower bounds

- complexity classes

this talk: focus on one problem

- reachability in vector addition systems
Outline

well-quasi-orders (wqo):
  ▶ proving algorithm termination

a toolbox for wqo complexity
  ▶ upper bounds
  ▶ lower bounds
  ▶ complexity classes

this talk: focus on one problem
  ▶ reachability in vector addition systems
OUTLINE

well-quasi-orders (wqo):
  ▶ proving algorithm termination

a toolbox for wqo complexity
  ▶ upper bounds
  ▶ lower bounds
  ▶ complexity classes

this talk: focus on one problem
  ▶ reachability in vector addition systems
Vector Addition Systems
Vector Addition Systems
**Vector Addition Systems**

**Springfield Power Plant**

- (1,1) produce electricity
- (-1,-2) recycle uranium
- (0,1) uranium waste

**electricity**
Can we produce unbounded electricity with no leftover uranium waste?
Vector Addition Systems

Springfield Power Plant

(1,1) → produce electricity
(0,1) → recycle uranium
(-1,-2) → electricity

Can we produce unbounded electricity with no leftover 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?
IMPORTANCE OF THE PROBLEM

DISCRETE RESOURCES

- modelling: items, money, energy, molecules, …
- distributed computing: active threads in thread pool
- data: isomorphism types in data logics and data-centric systems
IMPORTANCE OF THE PROBLEM

CENTRAL DECISION PROBLEM [S.’16]
Large number of problems interreducible with reachability in vector addition systems
IMPORTANCE OF THE PROBLEM

THEOREM (Minsky’67)

Reachability is undecidable in 2-dimensional Minsky machines (vector addition systems with zero tests).
**Importance of the Problem**

1962
- R. J. Lipton: \textit{EXPSPACE} lower bound

1976
- 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
- Leroux & S.: cubic Ackermann upper bound ($F_{\omega^3}$)

2015
- S.: quadratic Ackermann upper bound ($F_{\omega^2}$)

2017
- W. Czerwinski, S. Lasota, R. Lazić, J. Leroux, F. Mazowiecki: tower lower bound ($F_3$)

2018
- J. Leroux & S.: Ackermann upper bound ($F_\omega$)
“Simple Runs” (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]
“Simple Runs” (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]
"Simple Runs" (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]

Characterestic System

\[ 0 + 1 \cdot a - 1 \cdot b = c \]
\[ 1 + 1 \cdot a - 2 \cdot b = 0 \]
"Simple Runs" (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]
“Simple Runs” (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]
“Simple Runs” (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]

Homogeneous System

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

Unbounded Path

(2,0)
“Simple Runs” (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]
"Simple Runs" (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]
“Simple Runs” (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]
**Simple Runs** (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]

**Pumpable Paths**

**“Simple Runs” (Θ Condition)**

[Mayr’81, Kosaraju’82, Lambert’92]

**Pumpable Paths**

unbounded path — pump up — pump down = remainder

classically: uses coverability trees

[Karp & Miller’69]

[Leroux & S.’19]

Rackoff-style witnesses
“Simple Runs” ($\Theta$ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]
**“Simple Runs” (Θ Condition)**

[Mayr’81, Kosaraju’82, Lambert’92]

- **pump up**
- \( \times 2 \)

(0,1)
“Simple Runs” (Θ Condition)

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

pump up

× 2

solution path

× 1
“Simple Runs” (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]
"Simple Runs" ($\Theta$ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]
"Simple Runs" (Θ Condition)

[Mayr'81, Kosaraju'82, Lambert'92]
“**Simple Runs**” (Θ Condition)

[Mayr’81, Kosaraju’82, Lambert’92]

- **Pump Up**: $\times 2$
- **Solution Path**: $\times 1$
- **Remainder**: $\times 2$
- **Pump Down**: $\times 2$

Graph showing points $(0,1)$ and $(4,0)$ connected by arrows indicating the direction and magnitude of each operation.
**“Simple Runs” (Θ Condition)**

[Mayr’81, Kosaraju’82, Lambert’92]

- Pump up: \( \times 3 \)
- Solution path: \( \times 1 \)
- Remainder: \( \times 3 \)
- Pump down: \( \times 3 \)

Diagram showing a grid with arrows indicating the movements corresponding to the runs.
**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"?  

\[
\{ \text{yes} \}
\]
**Decomposition Algorithm**

[Mayr’81, Kosaraju’82, Lambert’92]

Can we build a “simple run”? **No**

decompose
**Decomposition Algorithm**

[Mayr’81, Kosaraju’82, Lambert’92]

Can we build a “simple run”? **No**

decompose
Decomposition Algorithm

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

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

[Mayr‘81, Kosaraju‘82, Lambert‘92]
“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, Leroux & S.’19]

RANKING FUNCTION

\[ \omega^\omega (\omega^{d+1} \text{ in dim. } d) \]

\[ \lor \alpha_0 \]
TERMINATION OF THE DECOMPOSITION ALGORITHM

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

RANKING FUNCTION

\[ \omega^\omega (\omega^{d+1} \text{ in dim. } d) \]

\[ \wedge \]

\[ \alpha_0 \]

\[ \wedge \]

\[ \alpha_1 \]
**Termination of the Decomposition Algorithm**

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

**Ranking Function**

\[
\omega^\omega (\omega^{d+1} \text{ in dim. } d)
\]

\[
\alpha_0 \\
\alpha_1 \\
\alpha_2
\]
Termination of the Decomposition Algorithm

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

Ranking Function

\[ \omega^\omega \ (\omega^{d+1} \text{ in dim. } d) \]

\[ \land \]

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

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}
A One-Player Game

- over $\mathbb{Q}_{\geq 0} \times \mathbb{Q}_{\geq 0}$
- given initially $(x_0, y_0)$
- Eloise plays $(x_j, y_j)$ s.t. $\forall 0 \leq i < j$, $x_i > x_j$ or $y_i > y_j$
- Can Eloise win, i.e. play indefinitely?
- If not, how long can she last?
A One-Player Game

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

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

- over $\mathbb{Q}_{\geq 0} \times \mathbb{Q}_{\geq 0}$
- given initially $(x_0, y_0)$
- Eloise plays $(x_j, y_j)$ s.t. $\forall 0 \leq i < j$, $x_i > x_j$ or $y_i > y_j$
- Can Eloise win, i.e. play indefinitely?
- If not, how long can she last?
If \((x_0, y_0) \neq (0,0)\), then choosing \((x_j, y_j) = \left( \frac{x_0}{2^j}, \frac{y_0}{2^j} \right)\) wins.
A One-Player Game

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

- Can Eloise win, i.e. play indefinitely?

- If not, how long can she last?
Assume there exists an infinite sequence \((x_j, y_j)\) of moves over \(\mathbb{N}^2\).
Assume there exists an infinite sequence \((x_j, y_j)\) of moves over \(\mathbb{N}^2\). Consider the pairs of indices \(i < j\): color \((i, j)\)

- **purple** if \(x_i > x_j\) but \(y_i \leq y_j\),
- **red** if \(x_i > x_j\) and \(y_i > y_j\),
- **orange** if \(y_i > y_j\) but \(x_i \leq x_j\).

\((3,4)\)  
\((5,2)\)  
\((2,3)\)  
\(\ldots\)
Assume there exists an infinite sequence \((x_j, y_j)\) of moves over \(\mathbb{N}^2\). Consider the pairs of indices \(i < j\): color \((i,j)\)

- **purple** if \(x_i > x_j\) but \(y_i \leq y_j\),
- **red** if \(x_i > x_j\) and \(y_i > y_j\),
- **orange** if \(y_i > y_j\) but \(x_i \leq x_j\).

\[(3,4) \quad (5,2) \quad (2,3) \quad \ldots\]

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

- **purple** if \(x_i > x_j\) but \(y_i \leq y_j\),
- **red** if \(x_i > x_j\) and \(y_i > y_j\),
- **orange** if \(y_i > y_j\) but \(x_i \leq x_j\).

\[(3,4) \quad (5,2) \quad (2,3) \quad \ldots \]

By the infinite Ramsey Theorem, there exists an infinite monochromatic subset of indices. In all cases, it implies the existence of an infinite decreasing sequence in \(\mathbb{N}\), a contradiction.
A One-Player Game

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

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

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

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

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

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

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

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

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

but can be of arbitrary length

Example (over \(\mathbb{N}^2\))
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 \(g: \mathbb{N} \to \mathbb{N}\) monotone and inflationary and \(n_0 \in \mathbb{N}\) if \(\forall i. \|x_i\| \leq g^i(n_0)\)

[Cichoń & Tahhan Bittar’98]
Controlled Bad Sequences

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

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

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

controlled by \(g : \mathbb{N} \to \mathbb{N}\)

monotone and inflationary and \(n_0 \in \mathbb{N}\) if \(\forall i. \|x_i\| \leq g^i(n_0)\)

[Cichon & Tahhan Bittar’98]

Example (over \(\mathbb{N}^2\) with \(n_0 = 2\) 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 \(g: \mathbb{N} \rightarrow \mathbb{N}\)
  monotone and inflationary and
  \(n_0 \in \mathbb{N}\) if \(\forall i . \|x_i\| \leq g^i(n_0)\)

[Cichoń & Tahhan Bittar’98]

**Proposition**

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

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

\[
\begin{align*}
g^3(n_0) \
g^2(n_0) \
g^1(n_0) \
g^0(n_0)
\end{align*}
\]

\[
\begin{align*}
go^{\infty} \
\bigcup_{x \in X, \|x\| \leq n_0} \
\bigcup_{x \in X, \|x\| \leq g^0(n_0)} \
\bigcup_{x \in X, \|x\| \leq g^1(n_0)} \
\bigcup_{x \in X, \|x\| \leq g^2(n_0)} \
\bigcup_{x \in X, \|x\| \leq g^3(n_0)}
\end{align*}
\]
**Descent Equation**

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

\[
\begin{align*}
\|x_i\| & \leq g^{i-1}(n_0) \\
& \quad \text{over the suffix } x_1, x_2, x_3, \ldots, \forall i > 0,
\end{align*}
\]

\(x_0 \not\preceq x_i\)
**Descent Equation**

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

\[
\begin{align*}
\|x_i\| &\leq g^{i-1}(n_0) \\
L_{g,X}(n_0) &\leq \max_{x_0 \in X, \|x_0\| \leq n_0} 1 + \frac{1}{2} \left(1 - \frac{1}{L_{g,X}(n_0)}\right)\]
\end{align*}
\]

over the suffix \(x_1, x_2, x_3, \ldots, \forall i > 0, \quad x_i \in X \setminus \uparrow x_0 \overset{\text{def}}{=} \{x \in X \mid x_0 \preceq x\}\)
Descent Equation

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

\[
g^2(g(n_0)) = g^3(n_0) \\
g^1(g(n_0)) = g^2(n_0) \\
g^0(g(n_0)) = g^1(n_0) \\
g^0(n_0) \\
x_0 \leq x_1 \leq x_2 \leq x_3 \\
\|x_i\| \leq g^{i-1}(g(n_0))
\]

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

\(x_i \in X \setminus \uparrow x_0 \overset{\text{def}}{=} \{ x \in X \mid x_0 \not\leq x \} \)
**Descent Equation**

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

\[
\begin{align*}
g^2(g(n_0)) &= g^3(n_0) \\
g^1(g(n_0)) &= g^2(n_0) \\
g^0(g(n_0)) &= g^1(n_0) \\
g^0(n_0) \\
x_0 &\leq x_1 & x_2 &\leq x_3
\end{align*}
\]

over the suffix \(x_1, x_2, x_3, \ldots, \forall i > 0, \quad x_i \in X\setminus\uparrow x_0 \overset{\text{def}}{=} \{x \in X \mid x_0 \not\leq x\}\)

\[
\|x_i\| \leq g^{i-1}(g(n_0))
\]

\[
L_{g,X}(n_0) = \max_{x_0 \in X, \|x_0\| \leq n_0} 1 + L_{g,X\setminus\uparrow x_0}(g(n_0))
\]
Descent Equation

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

\[
g^2(g(n_0)) = g^3(n_0) \]
\[
g^1(g(n_0)) = g^2(n_0) \]
\[
g^0(g(n_0)) = g^1(n_0) \]
\[
g^0(n_0) \]

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

\(\alpha_i \in \alpha_0 \overset{\text{def}}{=} \{\beta \in \alpha | \beta \not\succ \alpha_0\}\)

\[
\|\alpha_i\| \leq g^{i-1}(g(n_0))
\]

\[
L_{g,\alpha}(n_0) = \max_{\alpha_0 \in \alpha, \|\alpha_0\| \leq n_0} 1 + L_{g,\alpha_0}(g(n_0))
\]
The Case of Ordinals

[S.’14]

$$L_{g,\alpha}(n_0) = \max_{\alpha_0 \in \alpha, \|\alpha_0\| \leq n_0} 1 + L_{g,\alpha_0}(g(n_0))$$
**The Case of Ordinals**

[S.’14]

\[ L_{g,\alpha}(n_0) = \max_{\alpha_0 \in \alpha, \|\alpha_0\| \leq n_0} 1 + L_{g,\alpha_0}(g(n_0)) \]
The Case of Ordinals

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

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

These functions form the Cichón hierarchy.
The Case of Ordinals

[S.’14]

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

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

These functions form the Cichón hierarchy.
Recall the definition of the Cichoń Hierarchy:

\[ L_{g,0}(x) \overset{\text{def}}{=} 0 \quad L_{g,\alpha}(x) \overset{\text{def}}{=} 1 + L_{g,P_x(\alpha)}(g(x)) \text{ for } \alpha > 0 \]

**Definition** (Hardy Hierarchy)

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

\[ g^0(x) \overset{\text{def}}{=} x \quad g^\alpha(x) \overset{\text{def}}{=} g^{P_x(\alpha)}(g(x)) \text{ for } \alpha > 0 \]
**RELATING NORM AND LENGTH**

[Cichoń & Tahhan Bittar’98]

\[ g^\alpha(x) = g^{Lg,\alpha}(x)(x) \]

\[ g^\alpha(x) \geq Lg,\alpha(x) + x \]
**RELATING NORM AND LENGTH**

[Cichoń & Tahhan Bittar'98]

\[ g^\alpha(x) = g^{L_{g,\alpha}}(x) \quad \text{(x)} \]

\[ g^\alpha(x) \geq L_{g,\alpha}(x) + x \]
The Length of Decomposition Branches

\[ \alpha_0 \lor \alpha_1 \lor \alpha_2 \lor \ldots \]
The Length of Decomposition Branches

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

Consequence of (Leroux & S.’19)

An elementary control \( g \) and \( n \) the size of the reachability instance fit. Thus the decomposition algorithm runs in

\[ \text{SPACE}(g^{\omega \omega}(n)), \text{ and SPACE}(g^{\omega d+1}(n))) \]

in fixed dimension \( d \).
The Length of Decomposition Branches

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

Consequence of (Leroux & S.’19)

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

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

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

$H^0(x) = x$

$H^k(x) = H \circ \cdots \circ H(x)$ for $k$ times

$H^\omega(x) = H^{x+1}(x) = H \circ \cdots \circ H(x)$ for $x + 1$ times

$H^{\omega^2}(x) = H^{\omega \cdot (x+1)} = H^\omega \circ \cdots \circ H^\omega(x)$ for $x + 1$ times

$H^{\omega^3}(x) = H^{\omega^2 \cdot (x+1)} = H^{\omega^2} \circ \cdots \circ H^{\omega^2}(x)$ for $x + 1$ times

\[ H^{\omega \omega}(x) = H^{\omega^{x+1}}(x) \approx \text{ack}(x) \]
**Restating the Result**

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

\[ H^0(x) = x \]

\[ H^k(x) = H \circ \cdots \circ H(x) \]

\[ = x + k \]

\[ H^\omega(x) = H^{x+1}(x) = H \circ \cdots \circ H(x) \]

\[ = 2x + 1 \]

\[ H^{\omega^2}(x) = H^{\omega(x+1)} = H^\omega \circ \cdots \circ H^\omega(x) \]

\[ \approx 2^x \]

\[ H^{\omega^3}(x) = H^{\omega^2(x+1)} = H^{\omega^2} \circ \cdots \circ H^{\omega^2}(x) \]

\[ \approx \text{tower}(x) \]

\[ \vdots \]

\[ H^{\omega^\omega}(x) = H^{\omega(x+1)} \]

\[ \approx \text{ack}(x) \]
**RESTATING THE RESULT**

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

- $H^0(x) = x$

  - $k$ times
  
  $H^k(x) = H \circ \cdots \circ H(x) = x + k$

- $H^\omega(x) = H^{x+1}(x) = H \circ \cdots \circ H(x) = 2x + 1$

  - $x+1$ times

- $H^{\omega^2}(x) = H^{\omega \cdot (x+1)} = H^{\omega \circ \cdots \circ H^\omega(x)} \approx 2^x$

  - $x+1$ times

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

  - $x+1$ times

  : 

- $H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x) \approx \text{ack}(x)$
Restating the Result

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

$H^0(x) = x$

$k$ times

$H^k(x) = H \circ \cdots \circ H(x) = x + k$

$x+1$ times

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

$x+1$ times

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

$x+1$ times

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

\[ \vdots \]

$x+1$ times

$H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x) \approx \text{ack}(x)$
**RESTATING THE RESULT**

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

\[
H^0(x) = x
\]

\[
H^k(x) = H \circ \cdots \circ H(x) = \underbrace{x + 1}_{k \text{ times}}
\]

\[
H^\omega(x) = H^{x+1}(x) = H \circ \cdots \circ H(x) = \underbrace{x + 1}_{\text{x+1 times}}
\]

\[
H^{\omega^2}(x) = H^{\omega \cdot (x+1)} = H^\omega \circ \cdots \circ H^\omega(x) = \underbrace{2^x}_{\text{x+1 times}}
\]

\[
H^{\omega^3}(x) = H^{\omega^2 \cdot (x+1)} = H^{\omega^2} \circ \cdots \circ H^{\omega^2}(x) = \underbrace{\text{tower}(x)}_{\text{x+1 times}}
\]

\[
\vdots
\]

\[
H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x) = \underbrace{\text{ack}(x)}_{\text{x+1 times}}
\]
Restating the Result

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

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

$$H^k(x) = H \circ \cdots \circ H(x)$$

$k$ times

$$= x + k$$

$$H^\omega(x) = H^{x+1}(x) = H \circ \cdots \circ H(x)$$

$x+1$ times

$$= 2x + 1$$

$$H^{\omega^2}(x) = H^{\omega \cdot (x+1)} = H^\omega \circ \cdots \circ H^\omega(x)$$

$x+1$ times

$$\approx 2^x$$

$$H^{\omega^3}(x) = H^{\omega^2 \cdot (x+1)} = H^{\omega^2} \circ \cdots \circ H^{\omega^2}(x)$$

$x+1$ times

$$\approx \text{tower}(x)$$

$$\vdots$$

$$H^{\omega^\omega}(x) = H^{\omega^{x+1}}(x)$$

$$\approx \text{ack}(x)$$
Restating the Result

Define coarse-grained classes:

\[ \mathcal{F}_\alpha \overset{\text{def}}{=} \bigcup_{\beta \prec \omega} \text{FDTIME}(H^\beta(n)) \]

\[ F_\alpha \overset{\text{def}}{=} \bigcup_{f \in \mathcal{F}_\alpha} \text{DTIME}(H^{\omega \alpha}(f(n))) \]

Consequence of (S.’16, Thm. 4.4)

VAS Reachability is in $F_\omega$, and in $F_{d+4}$ in fixed dimension $d$. 
Restating the Result

Define coarse-grained classes:

\[
\mathcal{F}_{<\alpha} \overset{\text{def}}{=} \bigcup_{\beta < \omega^\alpha} \text{FDTIME}(H^\beta(n))
\]

\[
F_\alpha \overset{\text{def}}{=} \bigcup_{f \in \mathcal{F}_{<\alpha}} \text{DTIME}(H^\omega^\alpha(f(n)))
\]

Consequence of (S.’16, Thm. 4.4)

VAS Reachability is in $F_\omega$, and in $F_{d+4}$ in fixed dimension d.
**Complexity Classes Beyond Elementary**

[S.’16]

- **Elementary**
  - $F_3 = \text{Tower}
  - $F_\omega = \text{Ackermann}$
Complexity Classes Beyond Elementary

[S.’16]

\[ F_3 \overset{\text{def}}{=} \bigcup_{e \text{ elementary}} \text{DTIME}(\text{tower}(e(n))) \]
**Complexity Classes Beyond Elementary**

[S.’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.'16]

\[ F_\omega \overset{\text{def}}{=} \bigcup_{p \text{ primitive recursive}} \text{DTime}(\text{ack}(p(n))) \]
**Complexity Classes Beyond Elementary**

[S.’16]

- **Elementary**
- **Primitive Recursive**
- **Multiply Recursive**
- \( F_ω \) = Ackermann

**Examples of Ackermann-Complete Problems:**
- reachability in lossy Minsky machines [Urquhart’98, Schnoebelen’02]
- satisfiability of safety Metric Temporal Logic [Lazić et al.’16]
- satisfiability of Vertical XPath [Figueira and Segoufin’17]
**Complexity Classes Beyond Elementary**

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

[S.’16]
A Related Problem

labelled VAS transitions carry labels from some alphabet

$L(\forall, \text{source}, \text{target})$ the language of labels in runs from source to target

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

Downwards Language Inclusion Problem
input: two labelled VAS $\forall$ and $\forall'$ and configurations source, target, source', target'
question: $\downarrow L(\forall, \text{source}, \text{target}) \subseteq \downarrow L(\forall', \text{source}', \text{target'})$?
A Related Problem

labelled VAS transitions carry labels from some alphabet $L(\mathcal{V}, \text{source, target})$ the language of labels in runs from source to target

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

**Downwards Language Inclusion Problem**

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

question: $\downarrow L(\mathcal{V}, \text{source, target}) \subseteq \downarrow L(\mathcal{V}', \text{source}', \text{target}')$?
A Related Problem

**Downwards Language Inclusion Problem**

**Input:** two labelled VAS \( \mathcal{V} \) and \( \mathcal{V}' \) and configurations source, target, source', target'

**Question:** \( \downarrow L(\mathcal{V}, \text{source, target}) \subseteq \downarrow L(\mathcal{V}', \text{source}', \text{target}') \)?

**Theorem** (Habermehl, Meyer & Wimmel'10)

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

**Corollary**

The Downwards Language Inclusion is in Ackermann.
A Related Problem

Downwards Language Inclusion Problem

**Input:** two labelled VAS $V$ and $V'$ and configurations source, target, source', target'

**Question:** $\downarrow L(V, \text{source, target}) \subseteq \downarrow L(V', \text{source'}, \text{target'})$?

**Theorem (Habermehl, Meyer & Wimmel’10)**

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

**Corollary**

*The Downwards Language Inclusion is in Ackermann.*
A Related Problem

**Downwards Language Inclusion Problem**

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

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

**Corollary**

The Downwards Language Inclusion is in ACKERMANN.

**Theorem (Zetzsche’16)**

The Downwards Language Inclusion is ACKERMANN-hard.
SUMMARY

well-quasi-orders (wqo):
  ▶ proving algorithm termination

a toolbox for wqo-based complexity
  ▶ upper bounds: length function theorems
    (for ordinals, Dickson’s Lemma, Higman’s Lemma, and combinations)
  ▶ lower bounds
  ▶ complexity classes: \((F_\alpha)_\alpha\)

this talk: focus on one problem
  ▶ reachability in vector addition systems in \(F_\omega\)
**Perspectives**

1. Complexity gap for VAS reachability
   - **Tower-hard** [Czerwinski et al.'18]
     - Decomposition algorithm: requires $F_\omega = \text{Ackermann}$ time, because downward language inclusion is $F_\omega$-hard [Zetzsche’16]

2. Reachability in VAS extensions
   - Decidable in VAS with hierarchical zero tests [Reinhardt’08]
   - What about
     - Branching VAS
     - Unordered data Petri nets
     - Pushdown VAS
PERSPECTIVES

1. complexity gap for VAS reachability
   ▶ **TOWER-hard** [Czerwinski et al.’18]
   ▶ decomposition algorithm: requires $F_\omega = \text{ACKERMANN}$ time, because downward language inclusion is $F_\omega$-hard [Zetzsche’16]

2. reachability in VAS extensions
   ▶ decidable in VAS with hierarchical zero tests [Reinhardt’08]
   ▶ what about
     ▶ branching VAS
     ▶ unordered data Petri nets
     ▶ pushdown VAS


**PERSPECTIVES**

1. complexity gap for VAS reachability
   - **Tower-hard** [Czerwinski et al.’18]
   - decomposition algorithm: requires $F_\omega = \text{Ackermann}$ time, because downward language inclusion is $F_\omega$-hard [Zetzsche’16]

2. reachability in VAS extensions
   - decidable in VAS with hierarchical zero tests [Reinhardt’08]
   - what about
     - branching VAS
     - unordered data Petri nets
     - pushdown VAS
Ideal Decomposition Theorem
The Decomposition Algorithm computes the ideal decomposition of the set of runs from source to target.

Upper Bound Theorem
Reachability in vector addition systems is in cubic Ackermann.
Ideals of Well-Quasi-Orders \((X, \lessdot)\)

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

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

**Well-Quasi-Order on Runs**

combination of Dickson’s and Higman’s lemmata

**Syntax**

- $\rightarrow$ 

**Semantics**

$I_0$
DECOMPOSITION THEOREM

WELL-QUASI-ORDER ON RUNS

combination of Dickson’s and Higman’s lemmata

SYNTAX

SEMANTICS

I_0

I_1

I_2
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
**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 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
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

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
  
  Tower-hard
  
  [Lazić & S., ToCL’15]

  Programming Languages

  Database Theory

  Security

  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

\[
\begin{align*}
X & \rightarrow X \text{ ax} & 0 \rightarrow Y, Z \\
X + Y, X \otimes (0 \rightarrow Z) & \otimes R \\
\vdash X \rightarrow Y, X \otimes (0 \rightarrow Z) & \rightarrow R \\
0 & \rightarrow 0 \\
(\lambda \rightarrow Y) & \rightarrow 0 \\
\vdash ((\lambda \rightarrow Y) \rightarrow 0) & \rightarrow (X \otimes (0 \rightarrow Z)) \\
R & \rightarrow R
\end{align*}
\]
Branching VAS Reachability

- important open problem [Bojańczyk’14]

- incorrect decidability proof in [Bimbó’15]

- application domains:
  - Complexity Theory
  - Distributed Computing
  - Proof Theory
  - Computational Biology
  - Complexity Theory
  - Distributed Computing
  - Proof Theory
  - Computational Biology
  - 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, 6, 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:

<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]</td>
<td>[Bertrand et al.’17]</td>
</tr>
</tbody>
</table>

**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 pseudolinear or testing emptiness of a parity tree automaton are problems that are polynomial time equivalent to even 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 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
- a distinguished initial vertex

Here is an example of a parity game:

```plaintext
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, de Groote et al.’04, S., JSL’16]
  - **Computational Biology**
    - population protocols
      - [Bertrand et al.’17]

### Functional Programming Languages

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

fun 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:**

<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]</td>
<td>[Bertrand et al.’17]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programming Languages</th>
<th>Database Theory</th>
<th>Security</th>
<th>Computational Linguistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>observational equivalence</td>
<td>data logics</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Cotton-Barratt et al.’17]</td>
<td>[Bojańczyk et al.’09, Abriola et al.’17]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 Groot et al.’04]</td>
<td>[Bertrand et al.’17]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Programming Languages</th>
<th>Database Theory</th>
<th>Security</th>
<th>Computational Linguistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>fun append (xs, ys) =</td>
<td>data logics</td>
<td>security protocols</td>
<td></td>
</tr>
<tr>
<td>if null xs then ys</td>
<td>[Bojańczyk et al.’09, Abriola et al.’17]</td>
<td>[Verma &amp; Goubault-Larrecq’05]</td>
<td></td>
</tr>
<tr>
<td>else (hd xs):: append (tl xs, ys)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>observational equivalence</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[Cotton-Barratt et al.’17]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 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:

<table>
<thead>
<tr>
<th>Complexity Theory</th>
<th>Distributed Computing</th>
<th>Proof Theory</th>
<th>Computational Biology</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Programming Languages</th>
<th>Database Theory</th>
<th>Security</th>
<th>Computational Linguistics</th>
</tr>
</thead>
</table>

- minimalist syntax [Salvati’10]