Termination Analysis by Learning Terminating Programs

Matthias Heizmann Jochen Hoenicke Andreas Podelski

University of Freiburg

Checking Termination of Programs

- classical approach

  compose termination arguments

- our approach

  decompose program into modules

Checking Termination of Programs

- classical approach

  compose termination arguments

- our approach

  decompose program into modules

Question:

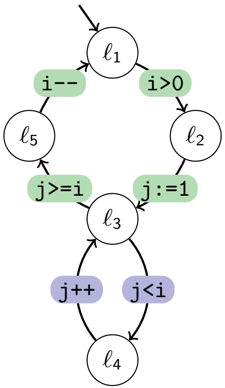
- What kind of module is suitable for termination proof?

Example: Bubble Sort

```
program sort(int i, int a[])
ℓ1 while (i>0)
ℓ2     int j:=1
ℓ3     while(j<i)
           if (a[j]>a[i])
               swap(a,i,j)
ℓ4         j++
ℓ5     i--
```

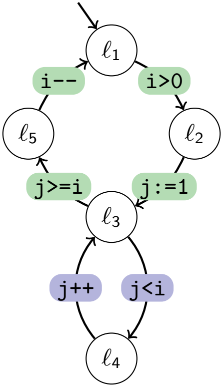
Example: Bubble Sort

```
program sort(int i)
ℓ1  while (i>0)
ℓ2        int j:=1
ℓ3        while(j<i)
ℓ4              j++
ℓ5        i--
```



Example: Bubble Sort

```
program sort(int i)
ℓ1 while (i>0)
ℓ2      int j:=1
ℓ3      while(j<i)
ℓ4           j++
ℓ5      i--
```



quadratic ranking function:

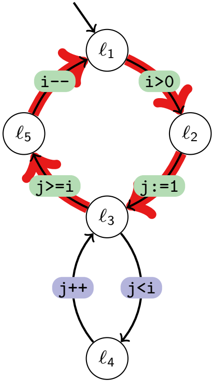
$$f(\mathtt{i},\mathtt{j}) = \mathtt{i}^2 - \mathtt{j}$$

lexicographic ranking function:

$$f(\mathtt{i},\mathtt{j}) = (\mathtt{i}, \mathtt{i} - \mathtt{j})$$

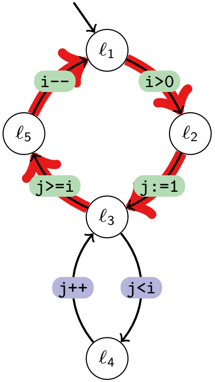
Example: Bubble Sort

single trace $\textsc{Outer}^{\omega}$



Example: Bubble Sort

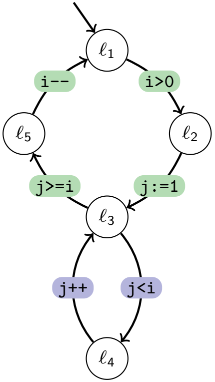
single trace $\textsc{Outer}^{\omega}$
has ranking function $f(\mathtt{i}, \mathtt{j}) = \mathtt{i}$



Example: Bubble Sort

single trace $\textsc{Outer}^\omega$
has ranking function $f(\mathtt{i},\mathtt{j}) = \mathtt{i}$

is also ranking function for
set of traces $(\textsc{Inner}^*.\textsc{Outer})^\omega$

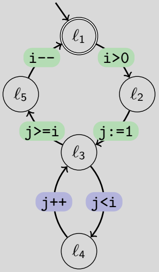


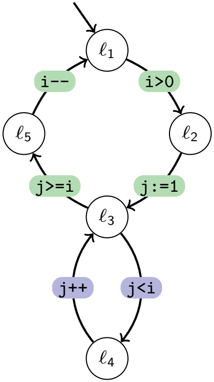
Example: Bubble Sort

single trace $\textsc{Outer}^\omega$
has ranking function $f(\mathtt{i},\mathtt{j}) = \mathtt{i}$

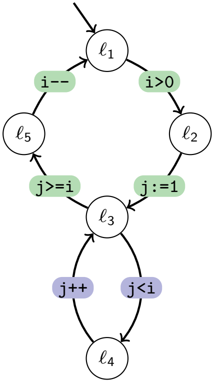
is also ranking function for
set of traces $(\textsc{Inner}^*.\textsc{Outer})^\omega$

module $\mathcal{P}_1$:
program with fairness constraint whose
set of traces is $(\textsc{Inner}^*.\textsc{Outer})^\omega$



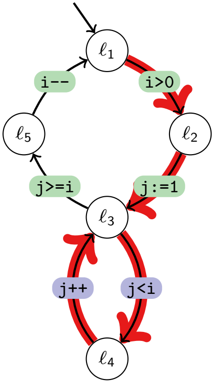


Example: Bubble Sort



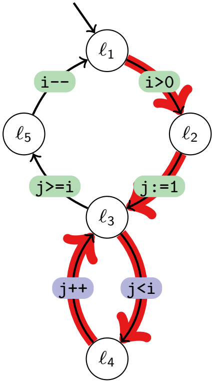
Example: Bubble Sort

new trace $\textsc{Outer}.\textsc{Inner}^{\omega}$



Example: Bubble Sort

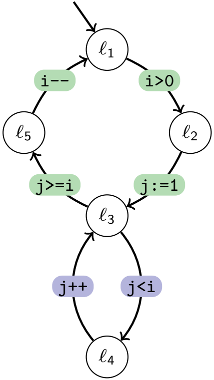
new trace $\textsc{Outer}.\textsc{Inner}^{\omega}$
has ranking function $f(\mathtt{i},\mathtt{j}) = \mathtt{i} - \mathtt{j}$



Example: Bubble Sort

new trace $\textsc{Outer}.\textsc{Inner}^{\omega}$
has ranking function $f(\mathtt{i}, \mathtt{j}) = \mathtt{i} - \mathtt{j}$

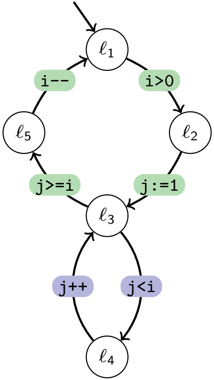
is also ranking function for
set of traces $(\textsc{Inner} + \textsc{Outer})^{*}.\textsc{Inner}^{\omega}$



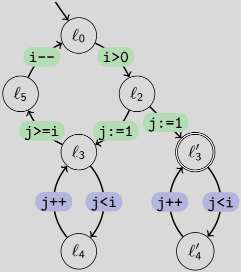
Example: Bubble Sort

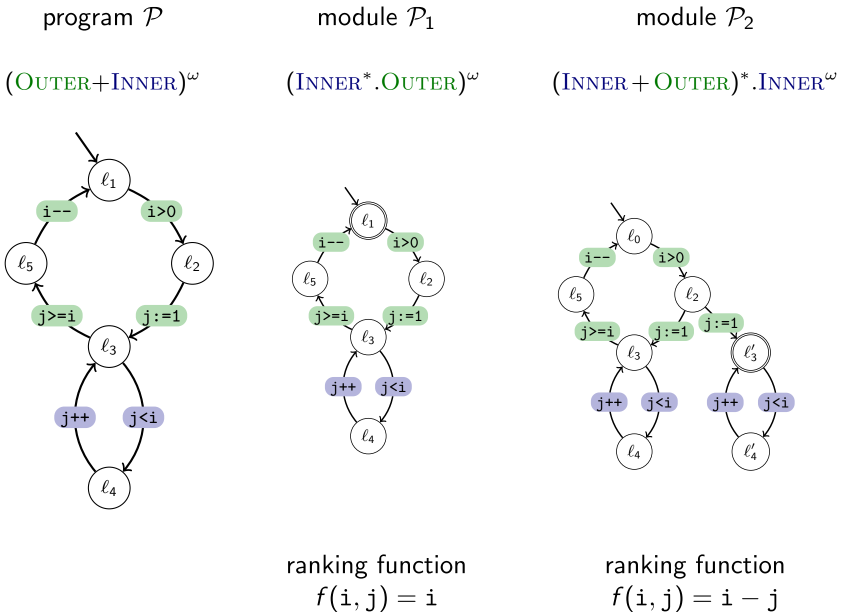
new trace $\textsc{Outer}.\textsc{Inner}^{\omega}$
has ranking function $f(\mathtt{i},\mathtt{j}) = \mathtt{i} - \mathtt{j}$

is also ranking function for
set of traces $(\textsc{Inner} + \textsc{Outer})^{*}.\textsc{Inner}^{\omega}$



module $\mathcal{P}_2$:
program with fairness constraint whose set of traces is $(\textsc{Inner} + \textsc{Outer})^{*}.\textsc{Inner}^{\omega}$

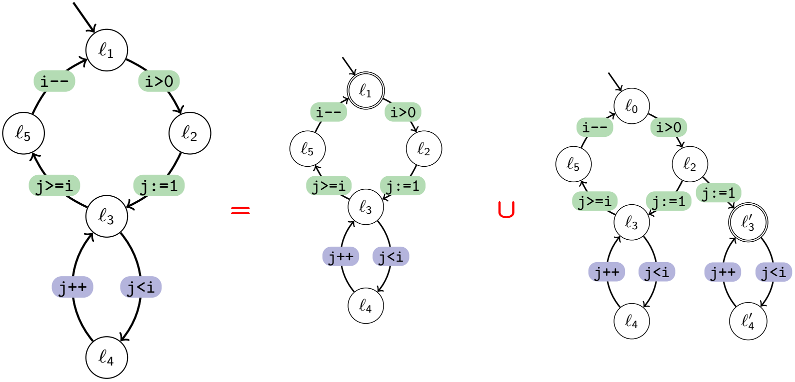




program $\mathcal{P}$
module $\mathcal{P}_1$
module $\mathcal{P}_2$
$(\textsc{Outer}+\textsc{Inner})^\omega$
$(\textsc{Inner}^*.\textsc{Outer})^\omega$
$(\textsc{Inner}+\textsc{Outer})^*.\textsc{Inner}^\omega$
ℓ_0
ℓ_1
ℓ_2
ℓ_3
ℓ_4
ℓ_5
ℓ_3'
ℓ_4'
i--
i>0
j:=1
j>=i
j++
j<i
ranking function
$f(\mathtt{i},\mathtt{j}) = \mathtt{i}$
ranking function
$f(\mathtt{i},\mathtt{j}) = \mathtt{i} - \mathtt{j}$

program $\mathcal{P}$ | module $\mathcal{P}_1$ | module $\mathcal{P}_2$

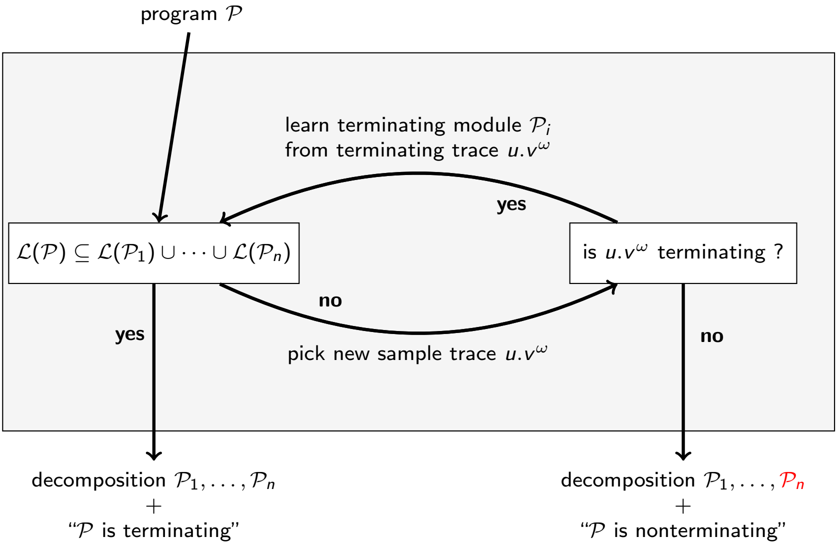
$$(\textsc{Outer}+\textsc{Inner})^\omega = (\textsc{Inner}^*.\textsc{Outer})^\omega + (\textsc{Inner}+\textsc{Outer})^*.\textsc{Inner}^\omega$$



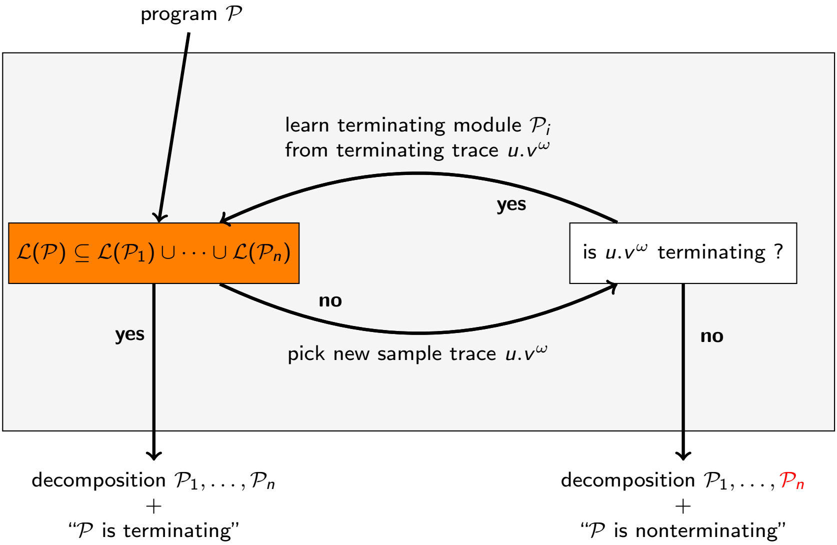
ranking function
$f(\mathtt{i},\mathtt{j}) = \mathtt{i}$

ranking function
$f(\mathtt{i},\mathtt{j}) = \mathtt{i} - \mathtt{j}$

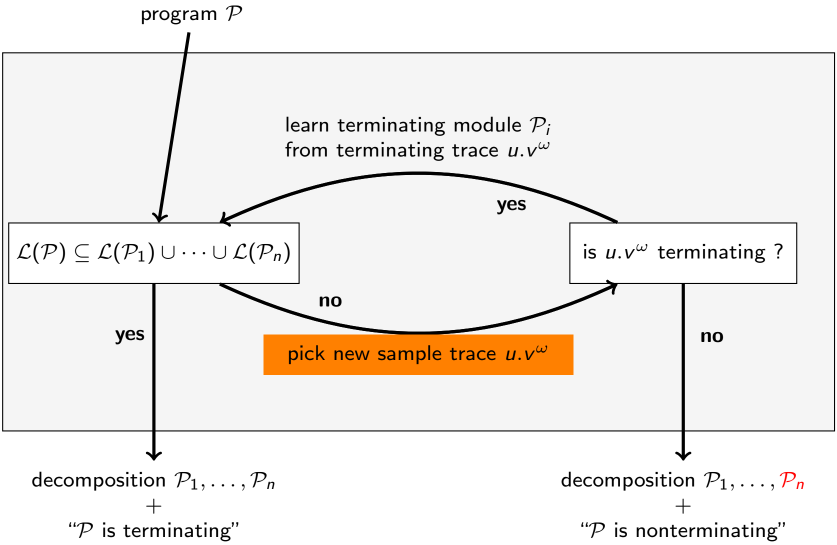
Algorithm for Construction of Decomposition $\mathcal{P}_1, \ldots, \mathcal{P}_n$



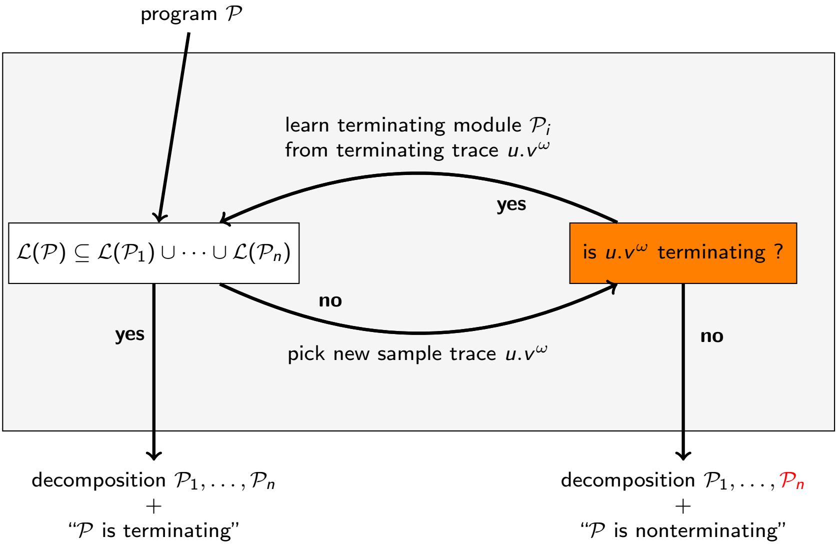
Algorithm for Construction of Decomposition $\mathcal{P}_1, \ldots, \mathcal{P}_n$



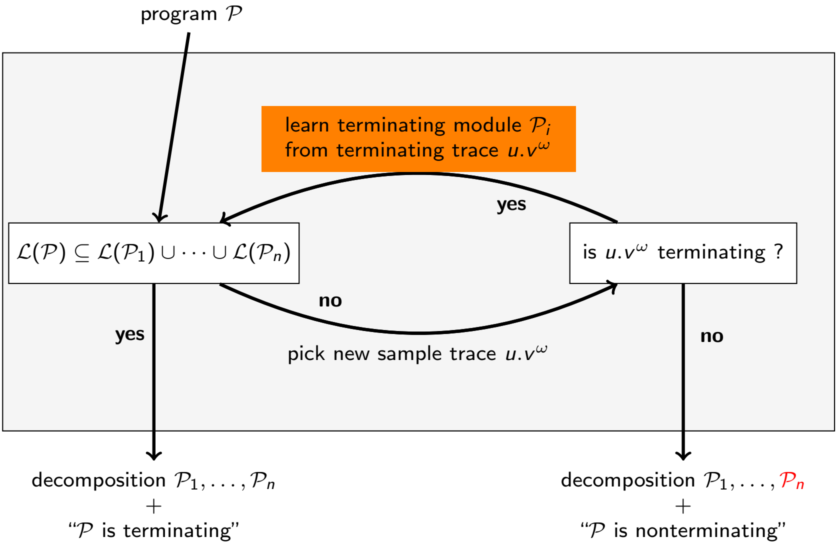
Algorithm for Construction of Decomposition $\mathcal{P}_1, \ldots, \mathcal{P}_n$



Algorithm for Construction of Decomposition $\mathcal{P}_1, \ldots, \mathcal{P}_n$



Algorithm for Construction of Decomposition $\mathcal{P}_1, \ldots, \mathcal{P}_n$



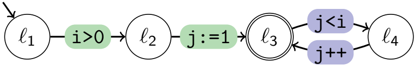
Learn Module from Trace – Example

input: ultimately periodic trace `i>0` `j:=1` (`j<i` `j++`)$^\omega$,

Learn Module from Trace – Example

input: ultimately periodic trace $\boxed{\texttt{i>0}}\,\boxed{\texttt{j:=1}}\,(\boxed{\texttt{j<i}}\,\boxed{\texttt{j++}})^\omega$,

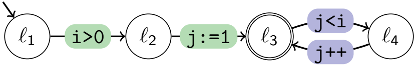
1. construct trivial module



Learn Module from Trace – Example

input: ultimately periodic trace `i>0` `j:=1` (`j<i` `j++`)$^{\omega}$,

1. construct trivial module



2. synthesize ranking function

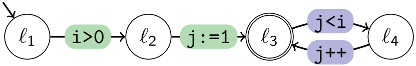
$$f(i,j) = i - j$$

- *Colón, Sipma* **Synthesis of Linear Ranking Functions** (TACAS 2001)
- *Podelski, Rybalchenko* **A complete method for the synthesis of linear ranking functions** (VMCAI 2004)
- *Bradley, Manna, Sipma* **Termination Analysis of Integer Linear Loops** (CONCUR 2005)
- *Bradley, Manna, Sipma* **Linear ranking with reachability** (CAV 2005)
- *Bradley, Manna, Sipma* **The polyranking principle** (ICALP 2005)
- *Ben-Amram, Genaim* **Ranking functions for linear-constraint loops** (POPL 2013)
- *H., Hoenicke, Leike, Podelski* **Linear Ranking for Linear Lasso Programs** (ATVA 2013)
- *Cook, Kroening, Rümmer, Wintersteiger* **Ranking function synthesis for bit-vector relations** (FMSD 2013)
- *Leike, H.* **Ranking Templates for Linear Loops** (TACAS 2014)

Learn Module from Trace – Example

input: ultimately periodic trace `i>0` `j:=1` (`j<i` `j++`)$^{\omega}$,

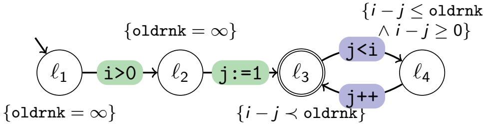
1. construct trivial module



2. synthesize ranking function

$$f(i,j) = i - j$$

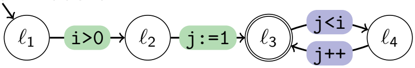
3. compute rank certificate



Learn Module from Trace – Example

input: ultimately periodic trace `i>0` `j:=1` (`j<i` `j++`)$^\omega$,

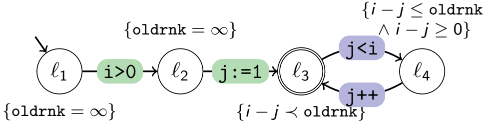
1. construct trivial module



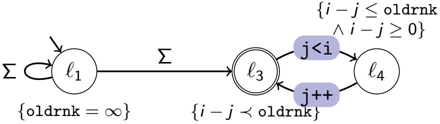
2. synthesize ranking function

$$f(i,j) = i - j$$

3. compute rank certificate



4. add additional transitions



Implementation / Evaluation

Our tool:

Ultimate Büchi Automizer

`http://ultimate.informatik.uni-freiburg.de/BuchiAutomizer/`

Implementation / Evaluation

Our tool:

Ultimate Büchi Automizer

`http://ultimate.informatik.uni-freiburg.de/BuchiAutomizer/`

For synthesis of ranking functions for single traces we use the tool:

Ultimate LassoRanker

`http://ultimate.informatik.uni-freiburg.de/LassoRanker/`

developed together with Jan Leike

Implementation / Evaluation

Our tool:

Ultimate Büchi Automizer

`http://ultimate.informatik.uni-freiburg.de/BuchiAutomizer/`

For synthesis of ranking functions for single traces we use the tool:

Ultimate LassoRanker

`http://ultimate.informatik.uni-freiburg.de/LassoRanker/`

developed together with Jan Leike

Programs with procedures and recursion? Büchi Nested Word Automata!

Implementation / Evaluation

Our tool:

Ultimate Büchi Automizer

`http://ultimate.informatik.uni-freiburg.de/BuchiAutomizer/`

1. in our paper: evalution on benchmark set from

Brockschmidt, Cook, Fuhs **Better termination proving through cooperation** (CAV 2013)

filename	program size	overall runtime	lasso analy time	module co time	Büchi incl time	modules trivial rf	modules non-trivial	module siz (maximum
a.10.c.t2.c	183	9s	2.8s	0.7s	2.1s	2	9	5
bf20.t2.c	156	6s	0.7s	0.9s	1.9s	6	7	9
bubbleSort.t2.c	109	5s	0.7s	0.3s	1.2s	5	5	5
consts1.t2.c	40	2s	0.3s	0.1s	0.2s	2	1	5
edn.t2.c	294	119s	18.8s	7.7s	89.0s	141	15	58
eric.t2.c	53	10s	1.1s	1.7s	5.0s	4	6	14
firewire.t2.c	178	28s	3.6s	1.3s	19.0s	12	7	8
mc91.t2.c	47	12s	1.2s	0.6s	4.3s	4	10	8

2. demonstration category on termination in SV-COMP 2014
3. category *C programs* in Termination Competition 2014

Future Work

- Translate our termination proof (decomposition in terminating modules) into other termination proofs (global ranking function, disjuntive well-founded transition invariant,...) and vice versa.

Future Work

- Translate our termination proof (decomposition in terminating modules) into other termination proofs (global ranking function, disjuntive well-founded transition invariant,...) and vice versa.
- Find the "simplest" termination argument for a trace $u.v^{\omega}$.

Future Work

- Translate our termination proof (decomposition in terminating modules) into other termination proofs (global ranking function, disjuntive well-founded transition invariant,...) and vice versa.

- Find the “simplest” termination argument for a trace $u.v^{\omega}$.

- Evaluation of Büchi complementation algorithms in our setting.

Conclusion

- new termination analysis
- decompose program $\mathcal{P}$ into modules $\mathcal{P}_1, \ldots, \mathcal{P}_n$
- decomposition not guided by the syntax of the program
  decomposition is guided by termination arguments

Conclusion

- new termination analysis
- decompose program $\mathcal{P}$ into modules $\mathcal{P}_1, \ldots, \mathcal{P}_n$
- decomposition not guided by the syntax of the program
  decomposition is guided by termination arguments

Thank You!

Our tool:

Ultimate Büchi Automizer

`http://ultimate.informatik.uni-freiburg.de/BuchiAutomizer/`