Hybrid Compositional Reasoning for Reactive Synthesis from Finite-Horizon Specifications

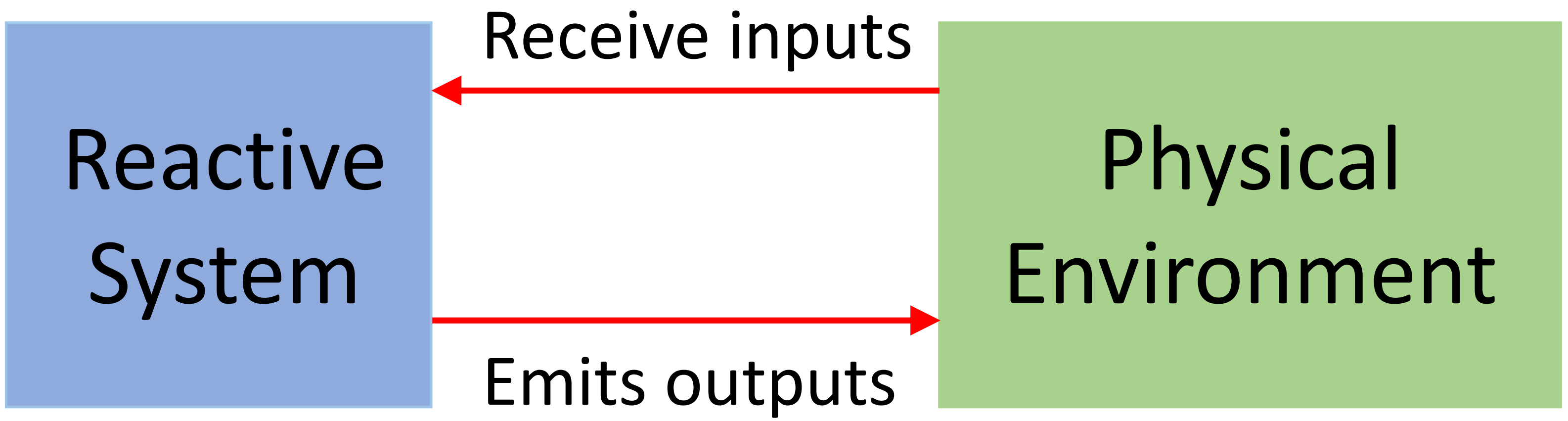
Suguman Bansal
Rice University

Yong Li
Chinese Academy of Science

Lucas M. Tabajara
Rice University

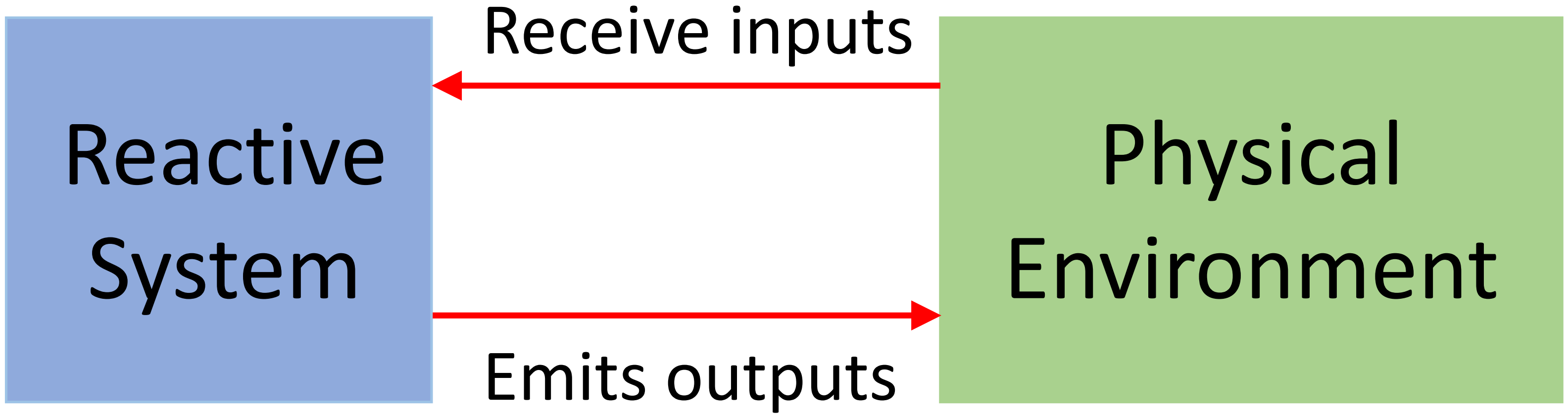
Moshe Y. Vardi
Rice University

Reactive systems



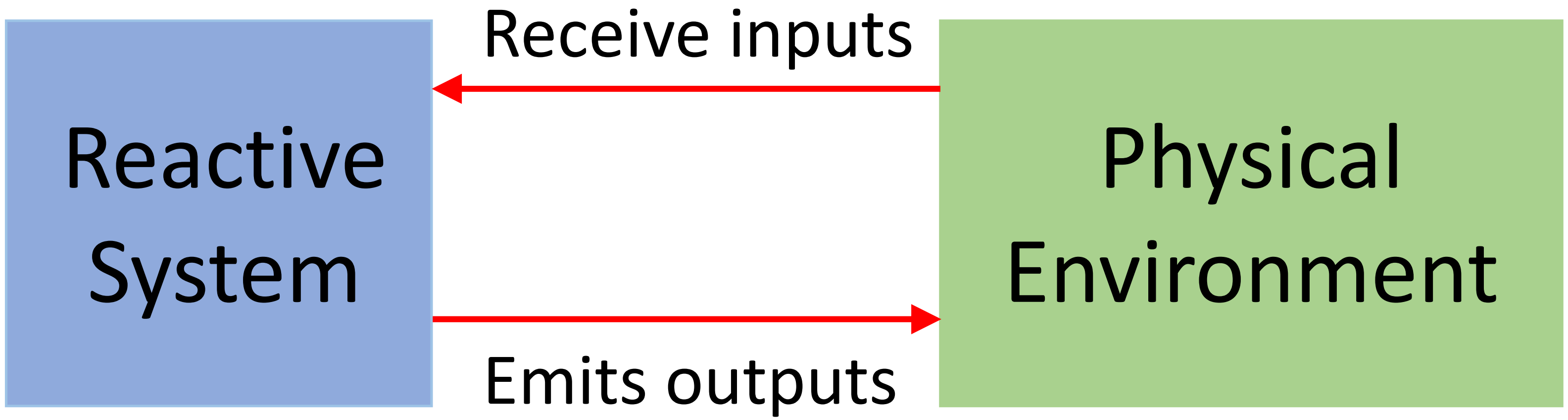
Continuous cycle of interaction

Reactive systems



Continuous cycle of interaction

Reactive systems



Continuous cycle of interaction

“Reactive” systems today

Robot and human interactions

Autonomous vehicles

AARIAN MARSHALL AND ALEX DAVIES TRANSPORTATION 05.24.18 03:38 PM

UBER'S SELF-DRIVING CAR SAW THE WOMAN IT KILLED, REPORT SAYS

…The problem is that it’s hard to find images of every sort of situation that could happen in the wild. Can the system distinguish a tumbleweed from a toddler. …

"Reactive" systems today

Robot and human interactions

Autonomous vehicles

AARIAN MARSHALL AND ALEX DAVIES TRANSPORTATION 05.24.18 03:38 PM

UBER'S SELF-DRIVING CAR SAW THE WOMAN IT KILLED, REPORT SAYS

…The problem is that it's hard to find images of every sort of situation that could happen in the wild. Can the system distinguish a tumbleweed from a toddler. …

Designing correct reactive systems is hard

Specifying intent of a reactive program is easier

Can we automatically generate a reactive program from its specification?

Can we automatically generate a reactive program from its specification?

Reactive synthesis

Can we automatically generate a reactive program from its specification written in LTLf ?

LTLf synthesis

2EXPTIME-complete [De Giacomo and Vardi; IJCAI 2015]

Tools – Use LTL synthesis tools (Acacia+ [2012], BoSy [2016], Strix [2018])

First dedicated LTLf tool – Syft [2017]

Partitioned LTLf synthesis [2019]

FOND planning based [2018]

Contributions

Improving scalability of LTLf synthesis

Address the bottleneck

LTLf to DFA conversion

Algorithmic improvements go a long way

Open source tool Lisa

For LTLf to DFA conversion

For LTLf synthesis

Linear-temporal logic over finite horizon (LTLf)

[Baier and McIlraith; 2006][De Giacomo and Vardi; IJCAI 2013]

- Specification language
  - Temporal logic over discrete time

- Syntax
  - Boolean variables and operators
  - Temporal operators: Always, Eventually, Next, Until …

Example: Always (Request → (Grant ∨ Next Grant))

- "Every request is granted within the two steps"

Linear-temporal logic over finite horizon (LTLf)

[Baier and McIlraith; 2006][De Giacomo and Vardi; IJCAI 2013]

- Specification language
  - Temporal logic over discrete time

- Syntax
  - Boolean variables and operators
  - Temporal operators: Always, Eventually, Next, …

Example: Always (Request → (Grant ∨ Next Grant))

- “Every request is granted within the two steps”

Request	T	F	T	F	T	T	HALT
Grant	T	F	F	T	F	T	

Linear-temporal logic over finite horizon (LTLf)

[Baier and McIlraith; 2006][De Giacomo and Vardi; IJCAI 2013]

- Specification language
  - Temporal logic over discrete time
- Syntax
  - Boolean variables and operators
  - Temporal operators: Always, Eventually, Next, …

Example: Always (Request → (Grant ∨ Next Grant))

- "Every request is granted within the two steps"

Request	T	F	T	F	T	T	HALT
Grant	F	F	F	T	F	T	

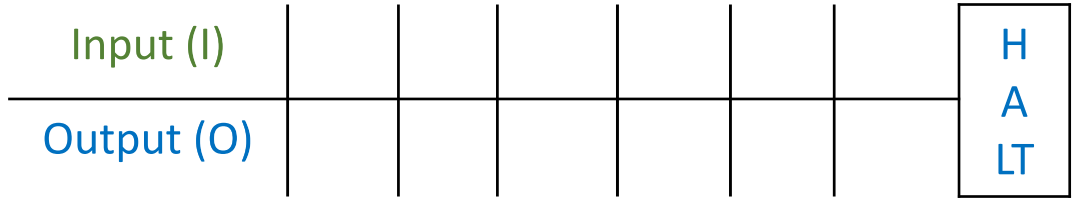
LTLf synthesis

[De Giacomo and Vardi; IJCAI 2015]

LTLf specification over input and output variables

Reactive system

- For each input, generates an output
- Each output depends on all prior inputs
- Input sequence **I**, output sequence **O**



LTLf synthesis: Given LTLf specification **S,**
Generate reactive system s.t. for all **I, (I,O)** satisfies **S.**

LTLf synthesis

[De Giacomo and Vardi; IJCAI 2015]

LTLf specification over input and output variables

Reactive system

- For each input, generates an output
- Each output depends on all prior inputs
- Input sequence **I**, output sequence **O**

Input (I)	T						HALT
Output (O)							

LTLf synthesis: Given LTLf specification **S,**
Generate reactive system s.t. for all **I, (I,O)** satisfies **S.**

LTLf synthesis

[De Giacomo and Vardi; IJCAI 2015]

LTLf specification over input and output variables

Reactive system

- For each input, generates an output
- Each output depends on all prior inputs
- Input sequence **I**, output sequence **O**

Input (I)	T						HALT
Output (O)	F						

LTLf synthesis: Given LTLf specification **S,**
Generate reactive system s.t. for all **I**, **(I,O)** satisfies **S.**

LTLf synthesis

[De Giacomo and Vardi; IJCAI 2015]

LTLf specification over input and output variables

Reactive system

- For each input, generates an output
- Each output depends on all prior inputs
- Input sequence **I**, output sequence **O**

Input (I)	T	T					HALT
Output (O)	F						

LTLf synthesis: Given LTLf specification **S,**
Generate reactive system s.t. for all **I**, **(I,O)** satisfies **S.**

LTLf synthesis

[De Giacomo and Vardi; IJCAI 2015]

LTLf specification over input and output variables

Reactive system

- For each input, generates an output
- Each output depends on all prior inputs
- Input sequence **I**, output sequence **O**

Input (I)	T	T					H A LT
Output (O)	F	T					

LTLf synthesis: Given LTLf specification **S,**
Generate reactive system s.t. for all **I**, **(I,O)** satisfies **S.**

LTLf synthesis

[De Giacomo and Vardi; IJCAI 2015]

LTLf specification over input and output variables

Reactive system

- For each input, generates an output
- Each output depends on all prior inputs
- Input sequence **I**, output sequence **O**

Input (I)	T	T	F				HALT
Output (O)	F	T					

LTLf synthesis: Given LTLf specification **S,**
Generate reactive system s.t. for all **I**, **(I,O)** satisfies **S.**

LTLf synthesis

[De Giacomo and Vardi; IJCAI 2015]

LTLf specification over input and output variables

Reactive system

- For each input, generates an output
- Each output depends on all prior inputs
- Input sequence **I**, output sequence **O**

Input (I)	T	T	F	…	…	T	H A LT
Output (O)	F	T	T	…	…	F	

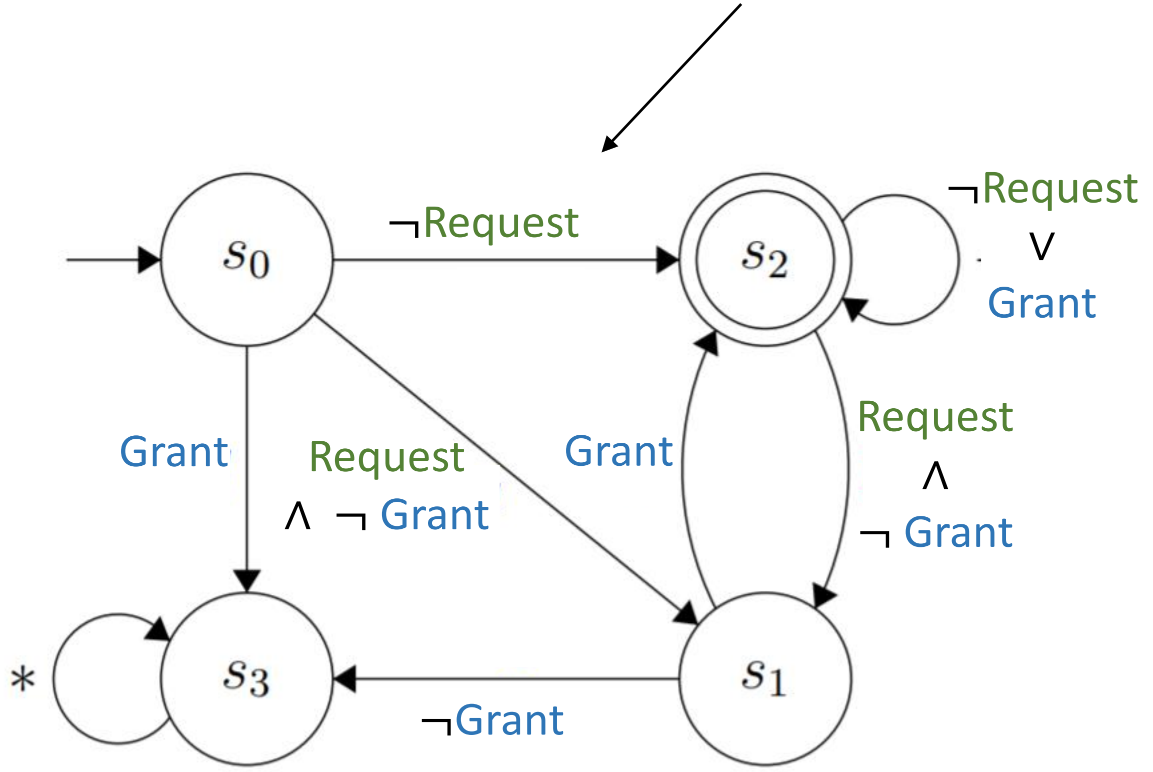
LTLf synthesis: Given LTLf specification **S,**
Generate reactive system s.t. for all **I, (I,O)** satisfies **S.**

LTLf synthesis = LTLf to DFA +

[De Giacomo and Vardi; IJCAI 2015]

¬ Grant ∧ Always (Request →(Grant ∨ Next Grant))

LTLf formula



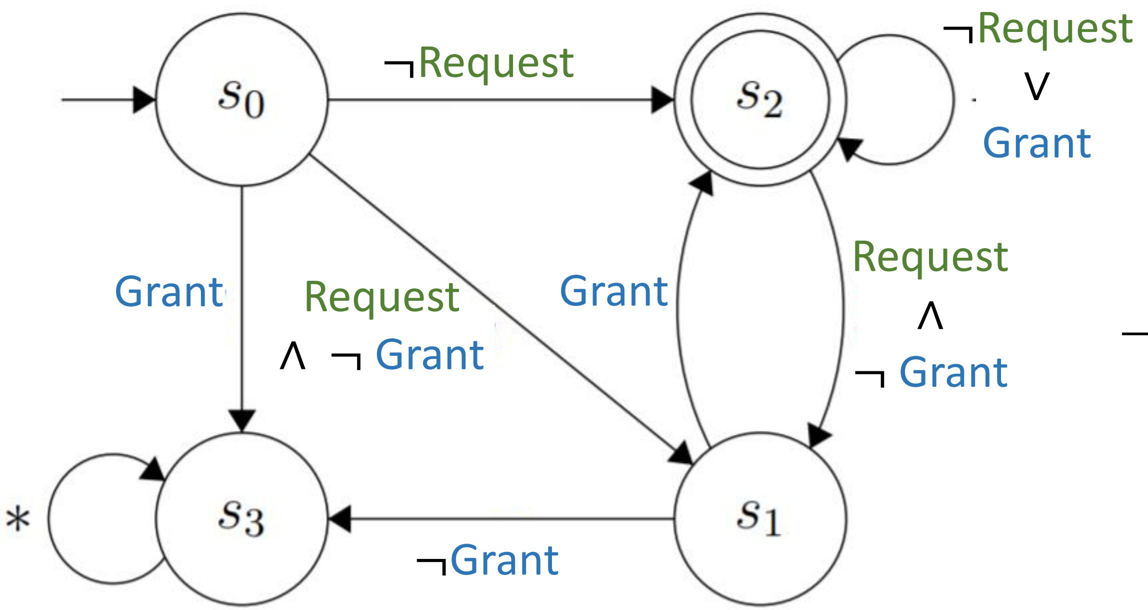
DFA

LTLf synthesis = LTLf to DFA + Reachability game

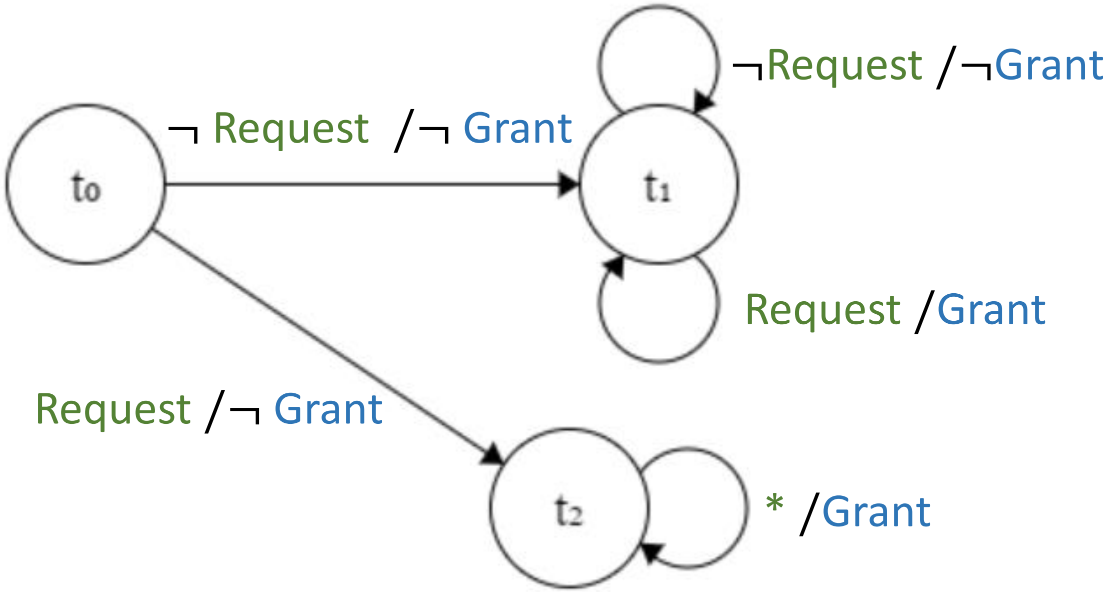
[De Giacomo and Vardi; IJCAI 2015]

$\neg$ Grant $\wedge$ Always (Request $\rightarrow$(Grant $\vee$ Next Grant))

LTLf formula



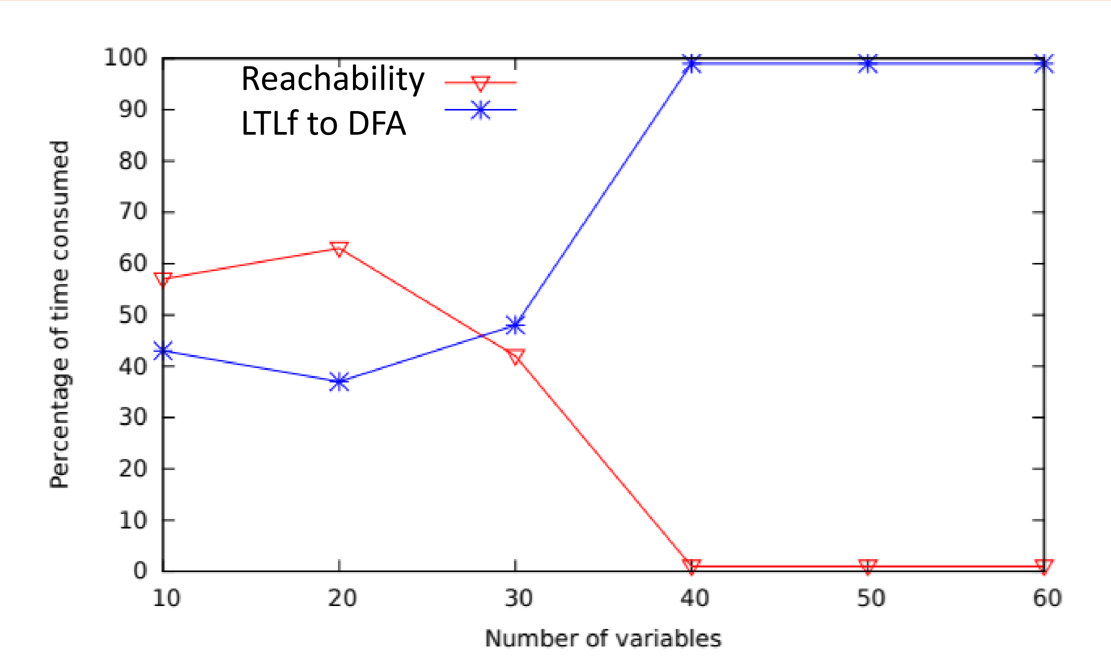
DFA



Reactive system

LTLf synthesis = LTLf to DFA + Reachability game

LTLf to DFA: **Bottleneck**



[Zhu et. al.; IJCAI 2017]

LTLf to DFA conversion

- Worst case, DFA is double exponential in size of formula

- Compositional techniques enhance scalability
  - Decompose formula into sub-formulas
  - Convert each sub-formula into DFA
  - Compose DFAs

- Representation of state-space
  - Explicit state space
  - Symbolic state space: n-states use log(n) variables
  - Impacts technique for composition

I. Explicit-state compositional

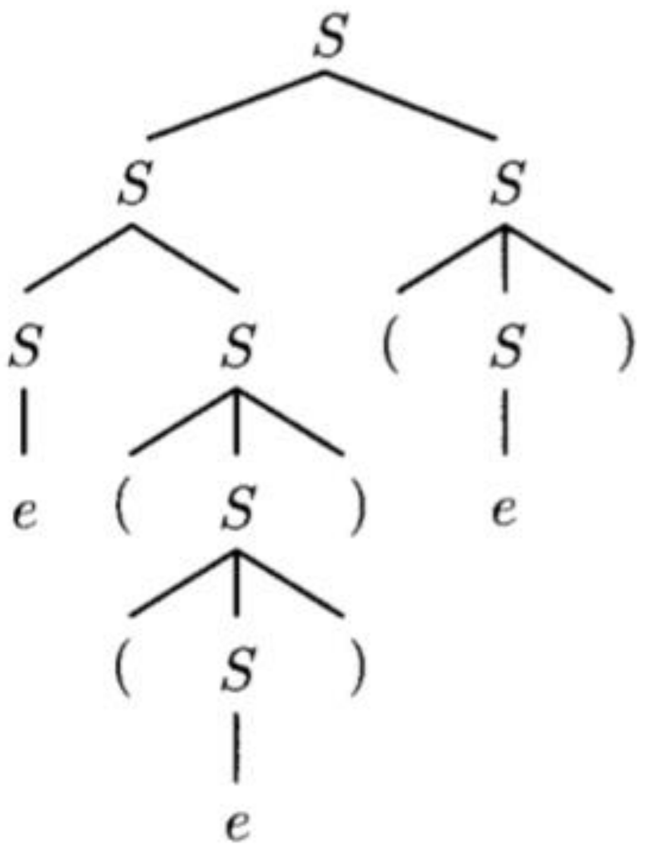
All DFAs are represented explicitly

Composition with DFA minimization

- Intermediate DFA are minimal
- Final DFA is minimal

Minimization becomes expensive.

Does not scale.



Tool – Mona

[Henriksen et al; TACAS 1995]

II. Symbolic-state compositional

All DFAs are represented symbolically

Composition without DFA minimization

- Intermediate DFAs are not minimal
- Final DFA not minimal

Large number of DFA state variables

Minimal DFA = 4100 states
Symbolic minimal DFA = 13 vars

Symbolic-state compositional

# States	# state vars.	# subformulas	Total # state vars
2	1	20	20
3	2	10	20
4	2	1	2
Total	--	**31**	**42**

Explicit state vs. Symbolic state

	Explicit	Symbolic
Runtime	High (Minimization)	Low (No minimization)
DFA state space (Small is good) [Tabajara and Vardi; IJCAI 2019]	Fewest (4100 states, 13 vars)	Very large (42 vars)

Explicit state vs. Symbolic state

	Explicit	Symbolic
Runtime	High (Minimization)	Low (No minimization)
DFA state space (Small is good) [Tabajara and Vardi; IJCAI 2019]	Fewest (4100 states, 13 vars)	Very large (42 vars)

GOAL

Low Runtime + Smaller state space

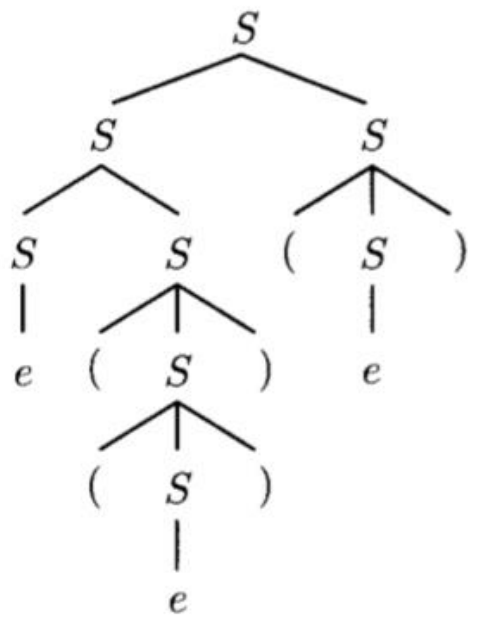
Our approach for LTLf to DFA conversion

1. Greedy heuristic

2. Hybrid heuristic

Heuristic I: Greedy composition

Explicit-state composition

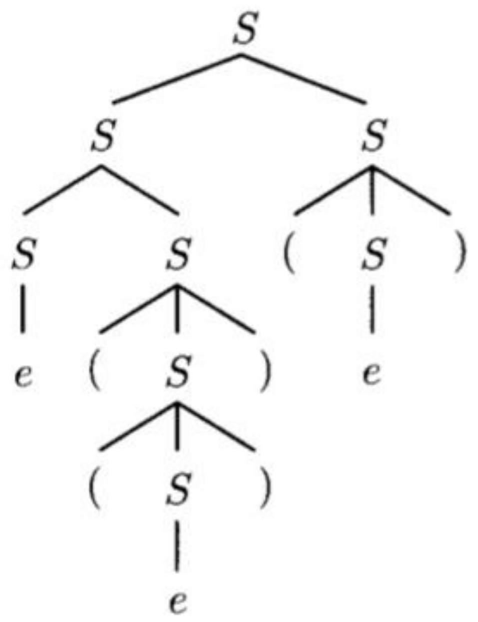


Follows parse tree

Order of composition not optimal

Heuristic I: Greedy composition

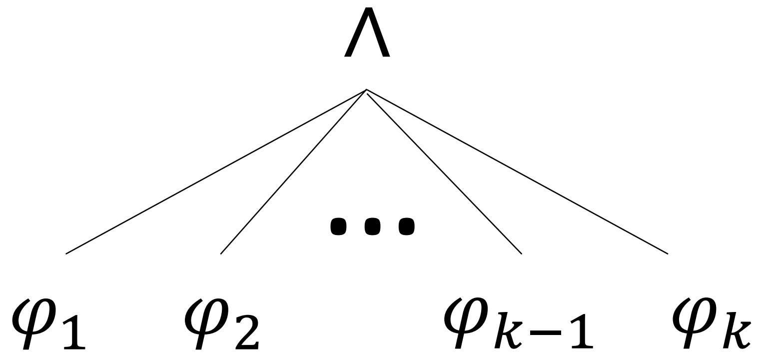
Explicit-state composition



Follows parse tree

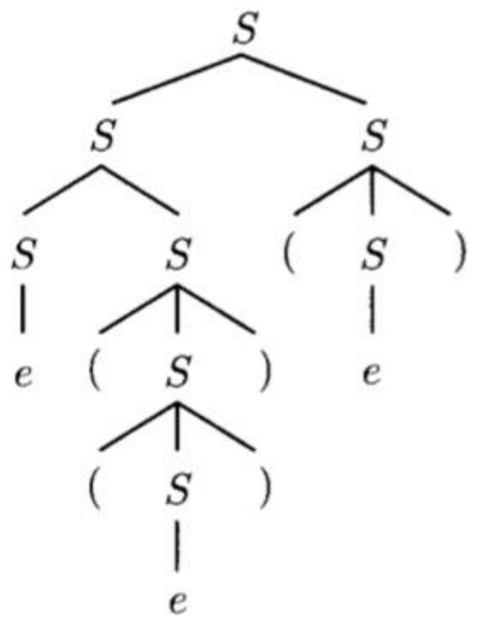
Order of composition not optimal

Greedy composition



Heuristic I: Greedy composition

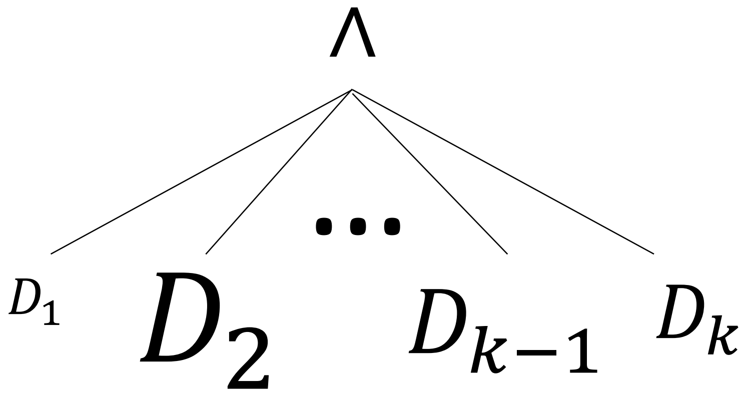
Explicit-state composition



Follows parse tree

Order of composition not optimal

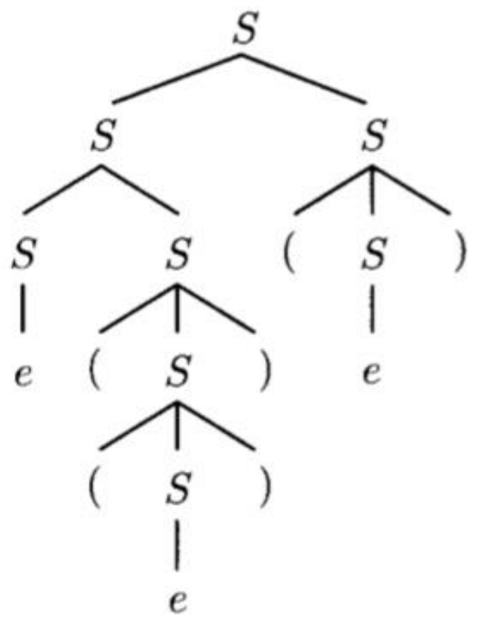
Greedy composition



Make minimal DFA at leaf

Heuristic I: Greedy composition

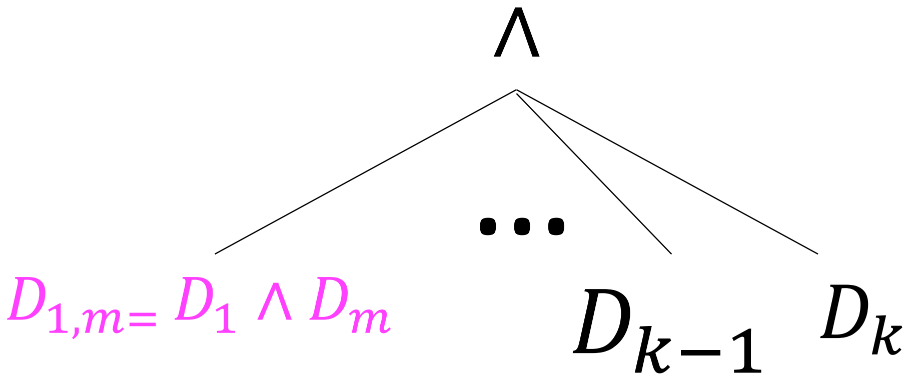
Explicit-state composition



Follows parse tree

Order of composition not optimal

Greedy composition

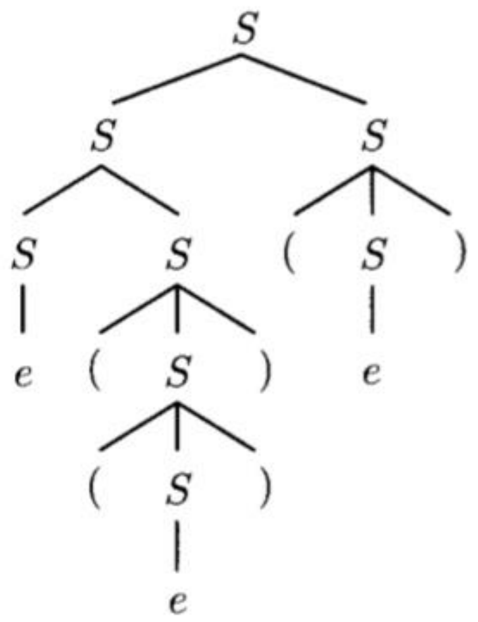


Make minimal DFA at leaf

Compose smallest two

Heuristic I: Greedy composition

Explicit-state composition



Follows parse tree

Order of composition not optimal

Greedy composition

$$D = D_1 \wedge D_2 \wedge \cdots D_k$$

Make minimal DFA at leaf

Compose smallest two

Continue till one DFA remains

Our approach for LTLf to DFA conversion

✓ Greedy heuristic

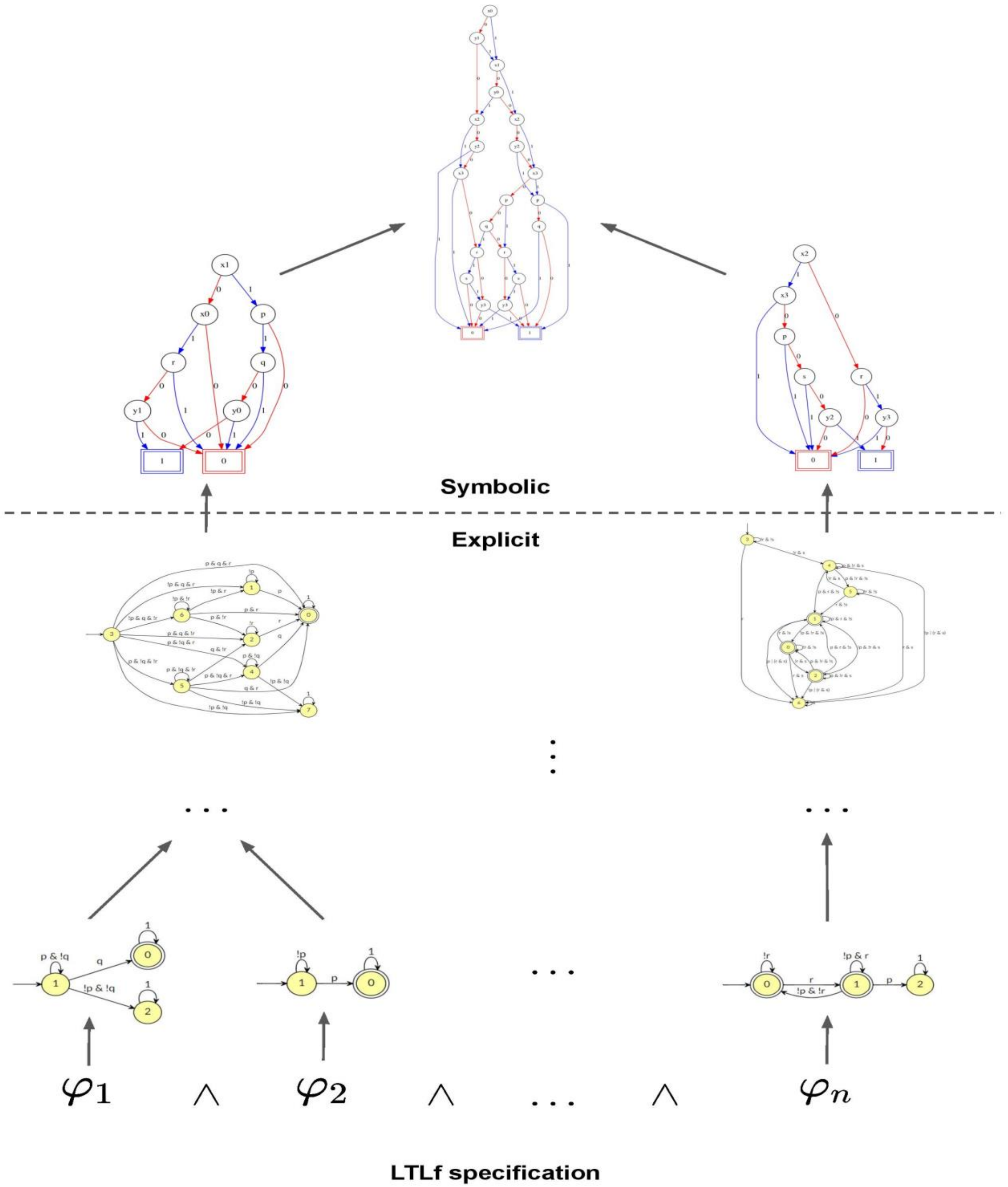
“Smallest first”

2. Hybrid heuristic

Heuristic II: Hybrid composition

Use both state representations

- Initially, greedy composition
- Soon, intermediate DFA become too large
    - Minimization is not effective
- So, switch to symbolic-state
    - Compact DFA representation



Our approach for LTLf to DFA conversion

- ✓ Greedy heuristic

  "Smallest first"

- ✓ Hybrid heuristic

  "Explicit in the beginning, symbolic later on"

Our tool **Lisa**: Implementation details

Lisa: https://github.com/vardigroup/lisa

Functions

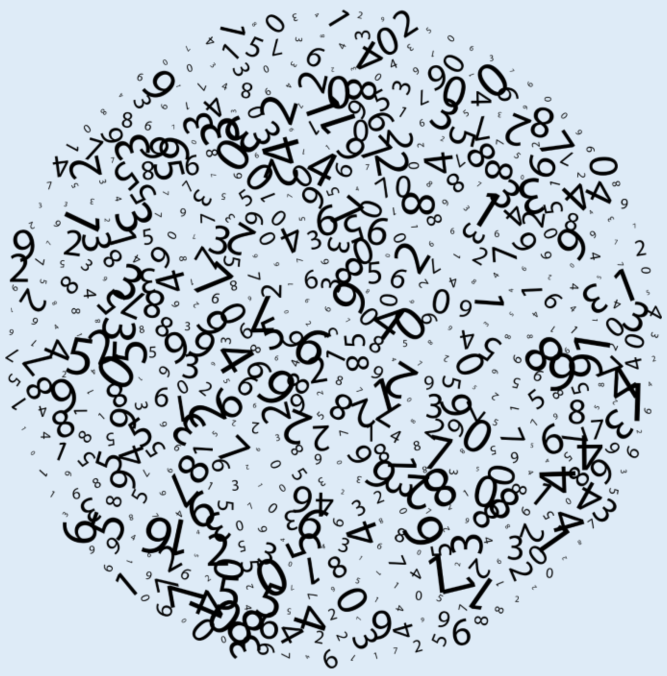
1. LTLf to DFA conversion
2. LTLf synthesis

Modes

- Greedy (Explicit state only)
- Greedy + Hybrid
- …

Empirical evaluation: Benchmark families
~ 450 benchmarks

Randomly generated
n conjunctions of basic formulas
[Zhu et al, IJCAI 2017]



Sequential counters
(n = #bits)
[Tabajara and Vardi, IJCAI 2019]

Single Counter (n = #bits)

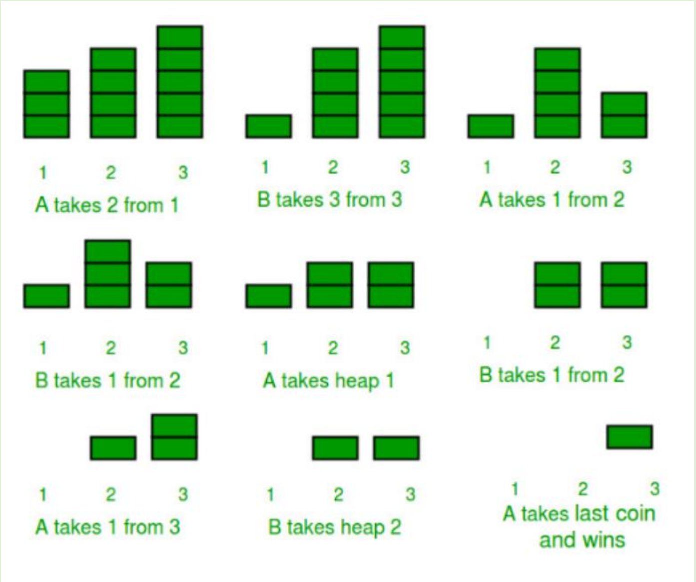
0 1 1 ⇨ 1 0 0 ⇨ …

Double Counter (n = #bits)

0 1 0 ⇨ 0 1 1 ⇨ …

0 0 1 ⇨ 0 1 0 ⇨ …

Nim games
n,m parameters
[Tabajara and Vardi, IJCAI 2019]



Empirical evaluation: I. DFA construction

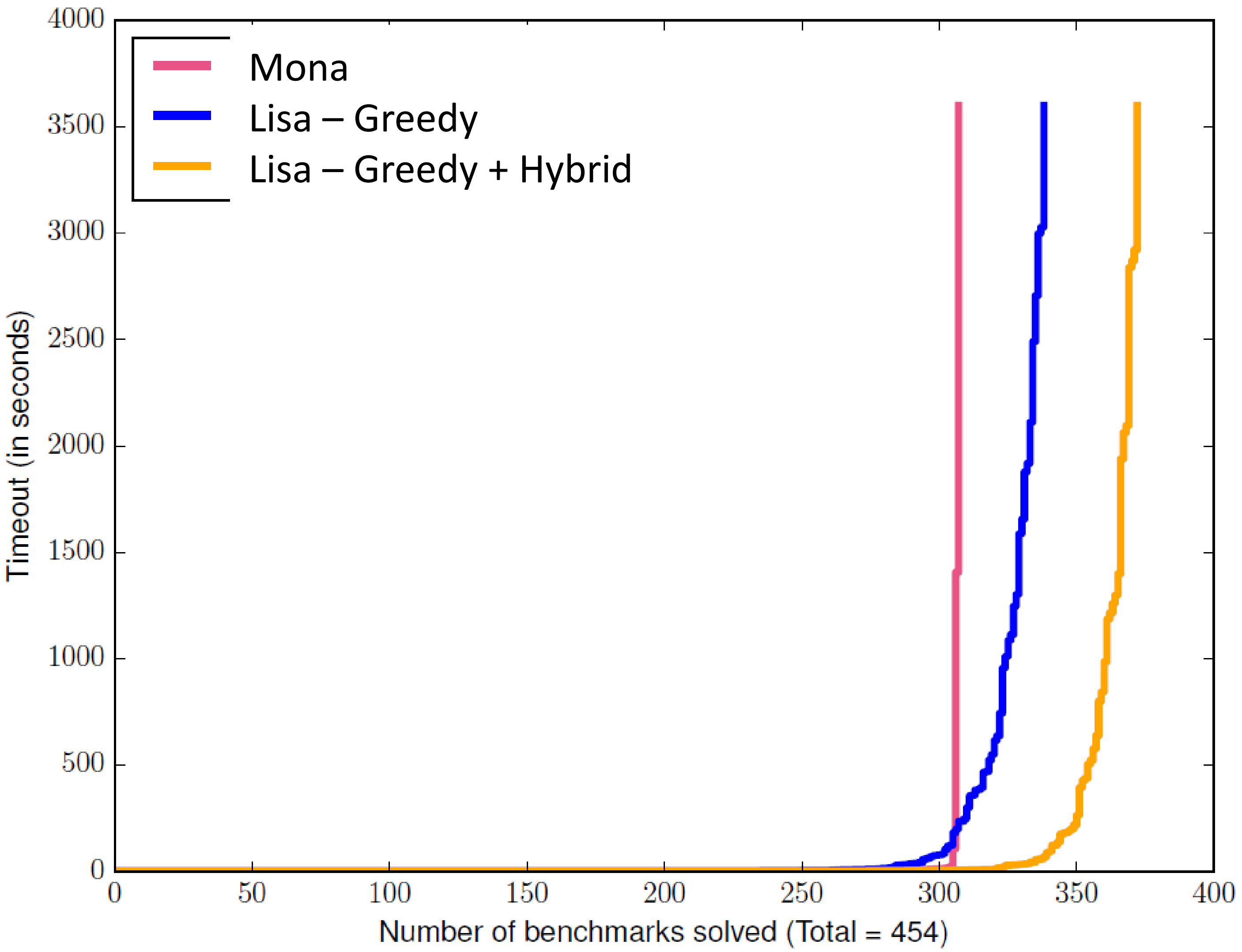
<u>Runtime</u>

Lisa – Greedy + Hybrid

Faster

Solves more benchmarks

Not minimal, but small state space



Empirical evaluation: II. LTLf synthesis

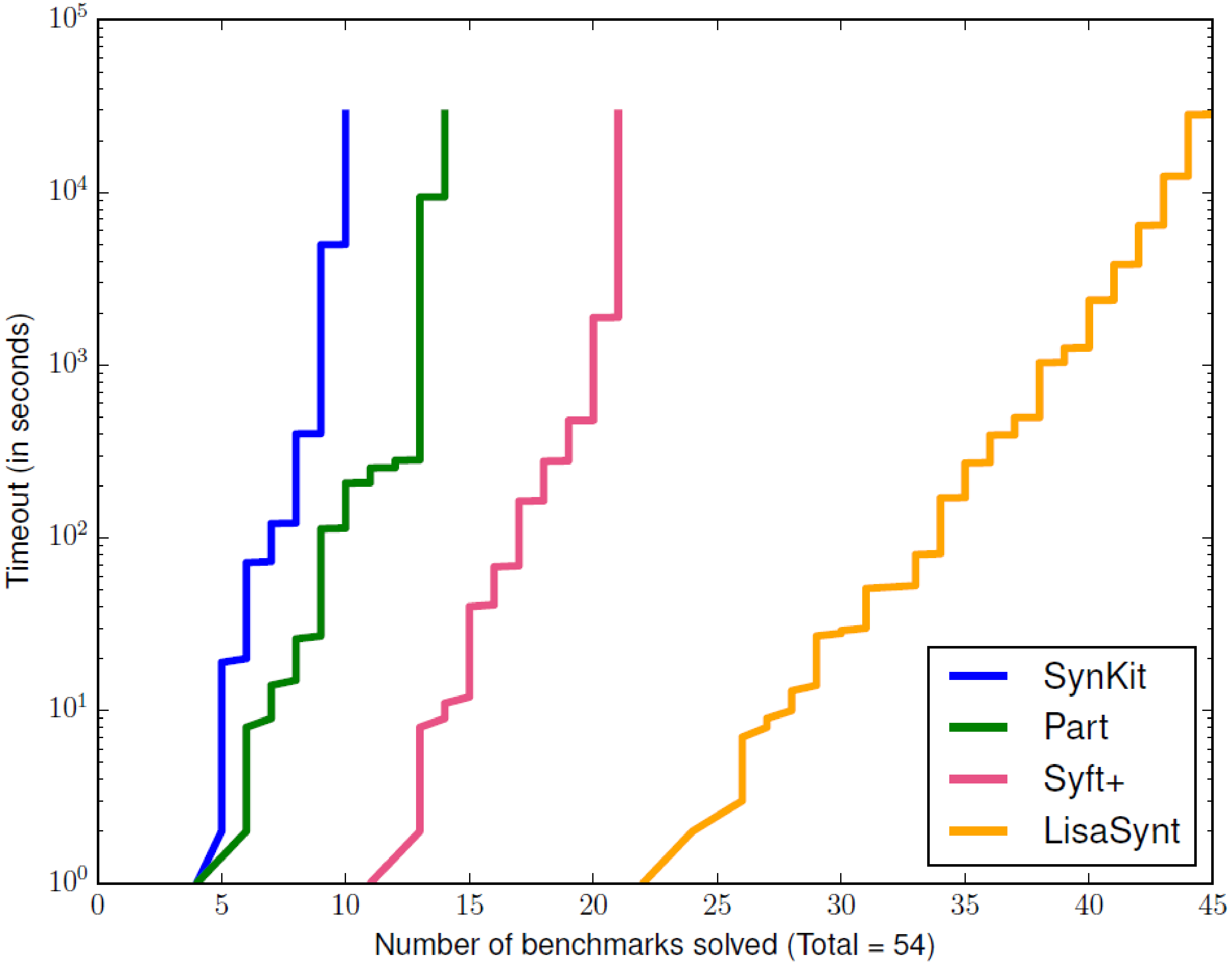
<u>FOND planning</u>

1. SynKit

<u>LTLf to DFA + Game solving</u>

1. Part: Symbolic DFA
2. Syft+: Mona
3. LisaSynt: Lisa –Greedy + Hybrid

LisaSynt solves larger and most benchmarks



In a nutshell

Hybrid Compositional Reasoning for Reactive Synthesis from Finite-Horizon Specifications (Paper Id. 9333)

- LTLf to DFA conversion primary bottleneck in LTLf synthesis
- Algorithmic insights (heuristics) significantly improve DFA conversion

- **Lisa** Open source tool
- LTLf to DFA conversion, LTLf synthesis
- https://github.com/vardigroup/lisa

- Explore further algorithmic improvements/heuristics
  - Symmetry, tunable parameters, …