Automated Black-box Verification of Networking Systems

Alexandra Silva

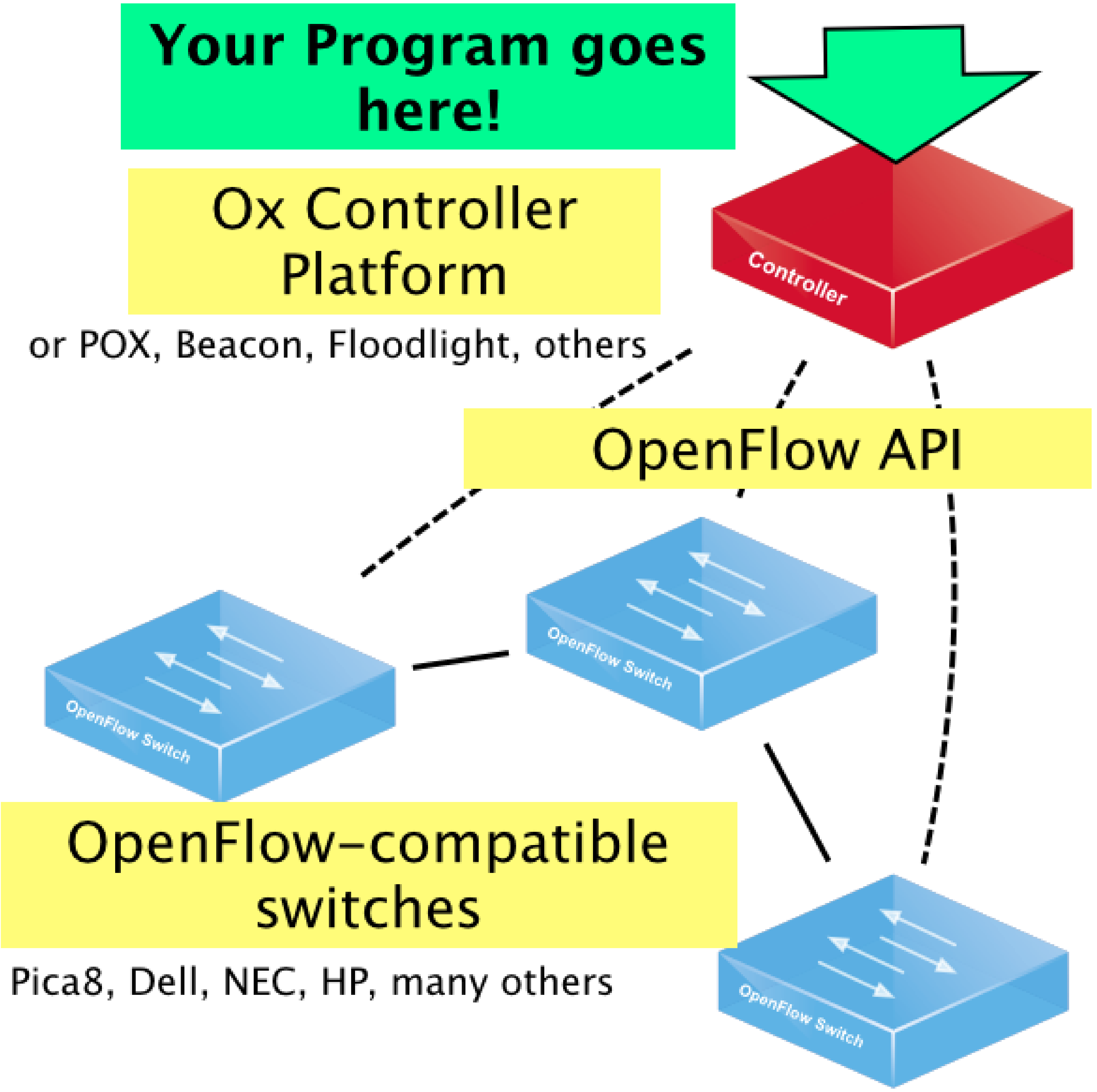
Matteo Sammartino

Stefan Zetzsche

UCL & VeTSS

UCL ENGINEERING
Change the world

Software-Defined Networks



Verification of networks

Trend in PL&Verification after Software-Defined Networks

- Design *high-level languages* that model essential network features
- Develop *semantics* that enables reasoning precisely about behaviour
- Build *tools* to synthesise low-level implementations automatically

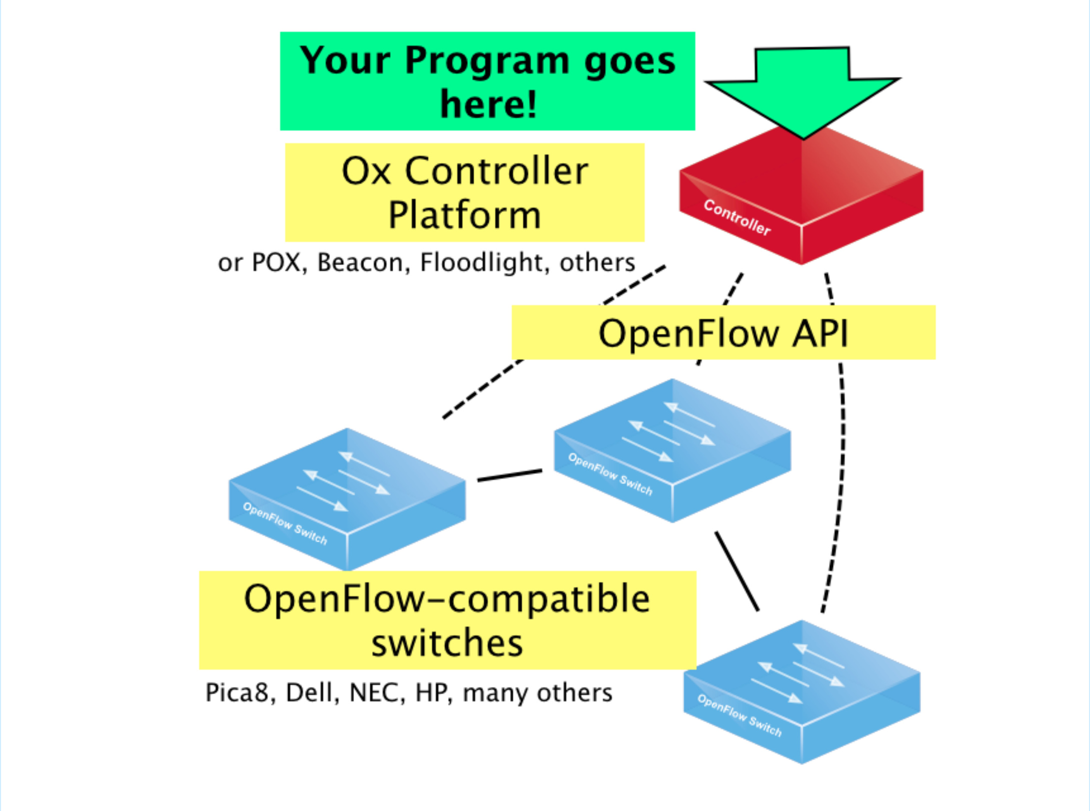
- Frenetic [Foster & al., ICFP 11]
- Pyretic [Monsanto & al., NSDI 13]
- Maple [Voellmy & al., SIGCOMM 13]
- FlowLog [Nelson & al., NSDI 14]
- Header Space Analysis [Kazemian & al., NSDI 12]
- VeriFlow [Khurshid & al., NSDI 13]
- NetKAT [Anderson & al., POPL 14]
- and many others . . .

This is all very nice but..

What if there is no formal model?

Does the low-level implementation really do what it is supposed to do?

What we propose



Build black-box model via interactions with the system

Automated Modelling

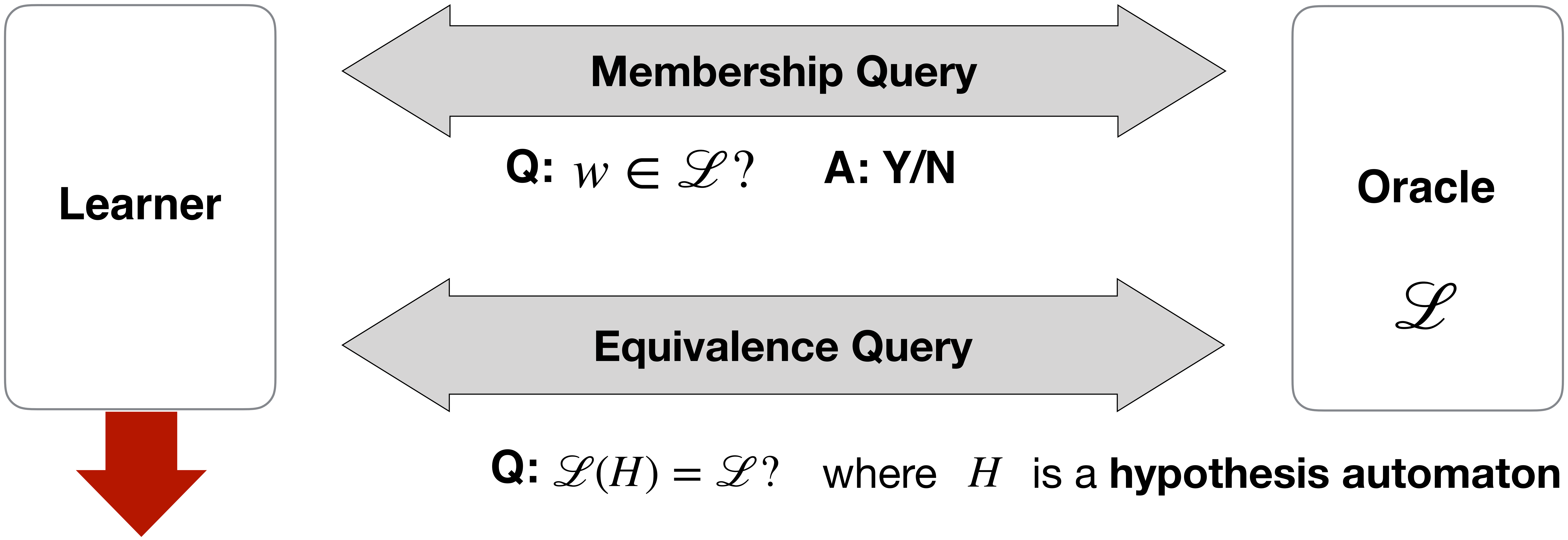
Properties

Automated Verification

Automata learning (Angluin '87)

Finite alphabet of system's actions A

Set of system behaviours is a **regular language** $\mathscr{L} \subseteq A^\star$



A: Y / N + counterexample

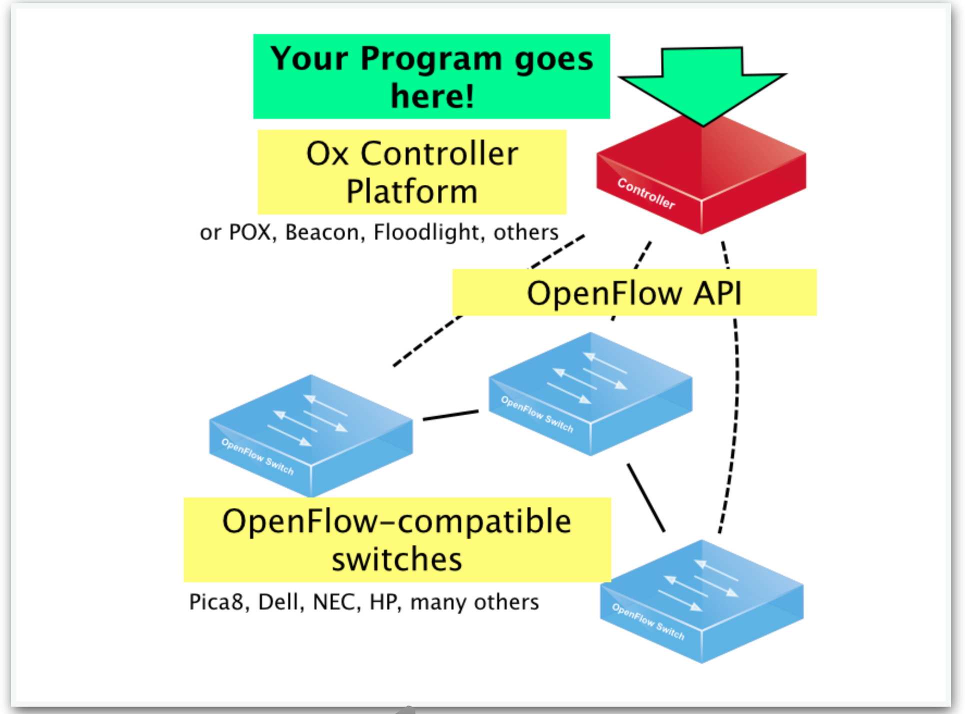
Minimal DFA accepting $\mathscr{L}$

In practice…

Membership Query
= Test Case

Equivalence Query

Oracle



Is spec violation an actual bug?

Test Suite

Model-based testing

Model-checking

In practice…

"Lazy" testing and model-checking

Good for scalability!

Many interesting applications

- Detect TLS implementations flaws [USENIX Sec. Sym. ’15]
- TCP implementations [CAV ’16]
- Analysis of botnet protocols [CCS ’10]
- Bank cards …

To each application domain its model…

Probabilistic

Mealy Machines

Alternating

Weighted

Non-deterministic

Register

Universal

Buchi

Do I need to write my automata learning algorithm from scratch?

NO! Maths can help!

Categorical
Automata
Learning
Framework

calf-project.org

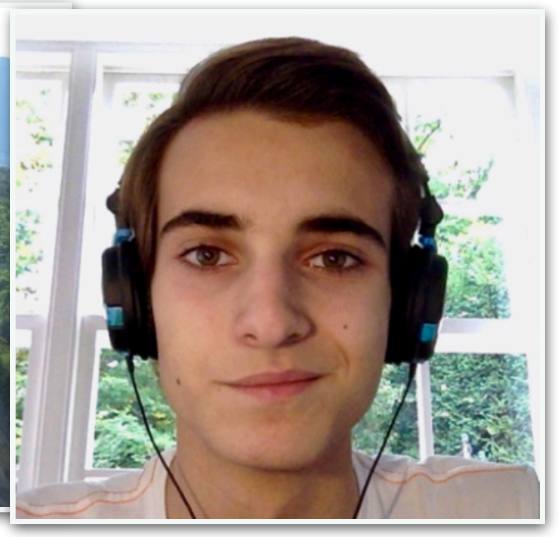
Gerco van Heerdt
UCL

Joshua Moerman
Radboud University

Bartek Klin
Warsaw University

Michal Szynwelski
Warsaw University

Maverick Chardet
ENS Lyon

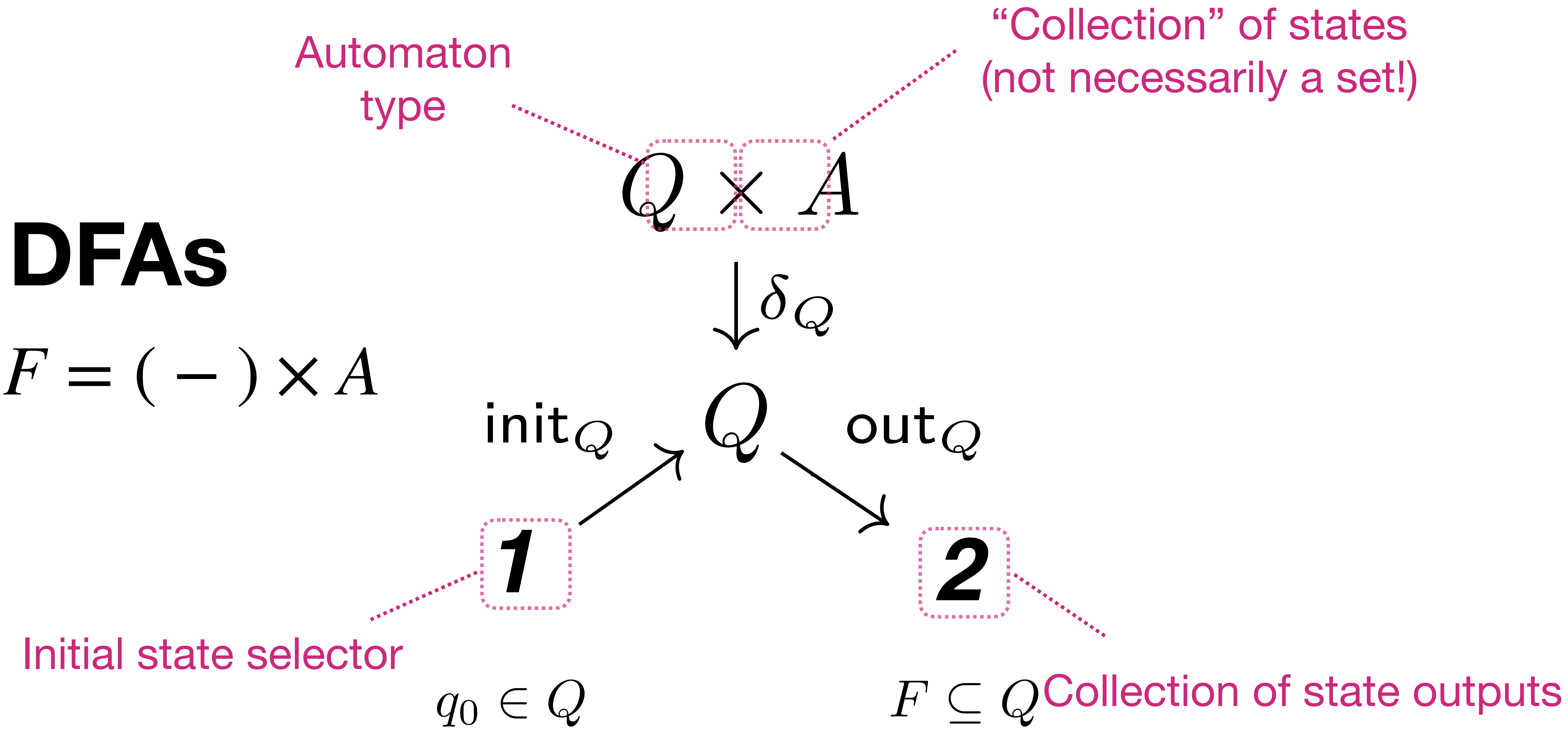


Tiago Ferreira
UCL Intern

Different automata, same structure

DFAs

$$F = (\,-\,) \times A$$



A general framework

Abstract observation data structure

approximates

Target minimal automaton

$$\begin{array}{ccccc} & & FQ & & \\ & & \big\downarrow \delta_Q & & \\ & \mathrm{init}_Q \nearrow & Q & \searrow \mathrm{out}_Q & \\ I & & & & Y \end{array}$$

abstract closedness and consistency

Hypothesis automaton

$$\begin{array}{ccccc} & & FH & & \\ & & \big\downarrow \delta_H & & \\ & \mathrm{init}_H \nearrow & H & \searrow \mathrm{out}_H & \\ I & & & & Y \end{array}$$

General correctness theorem

Guidelines for implementation

New algorithms!

F: Categorical Automata Learning Framework (CSL '17)

Other automata & optimizations

Learning Nominal Automata (POPL '17)

Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, Michal Szynwelski

Nom	**Nominal automata**
Vect	**Weighted automata**

Change main data structure

Observation tables

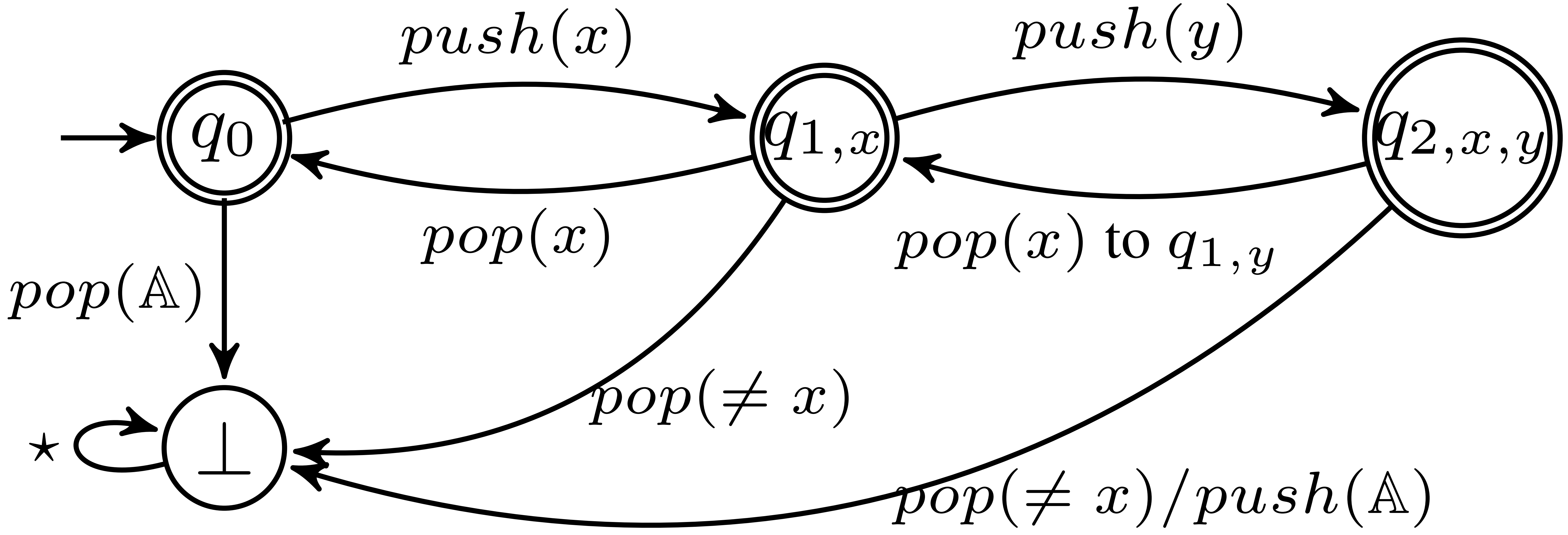
Discrimination trees

Plug monads in

Powerset	**NFAs**
Powerset with intersection	**Universal automata**
Double powerset	**Alternating automata**
Maybe monad	**Partial automata**

Infinite alphabets

infinite-state, but **finitely representable** automata



Other automata & optimizations

Change automaton type

Set	**DFAs**
Nom	**Nominal automata**
Vect	**Weighted automata**

Change main data structure

Observation tables

Discrimination trees

Plug monads in

Powerset	**NFAs**
Powerset with intersection	**Universal automata**
Double powerset	**Alternating automata**
Maybe monad	**Partial automata**

Other automata & optimizations

Learning Nominal Automata (POPL '17)

Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, Michal Szynwelski

Nom	**Nominal automata**
Vect	**Weighted automata**

Change main data structure

Observation tables

Discrimination trees

Plug monads in

Powerset	**NFAs**
Powerset with intersection	**Universal automata**
Double powerset	**Alternating automata**
Maybe monad	**Partial automata**

Other automata & optimizations

Learning Nominal Automata (POPL ’17)

Joshua Moerman, Matteo Sammartino, Alexandra Silva, Bartek Klin, Michal Szynwelski

Nom	**Nominal automata**
Vect	**Weighted automata**

Change main data structure

Observation tables

Discrimination trees

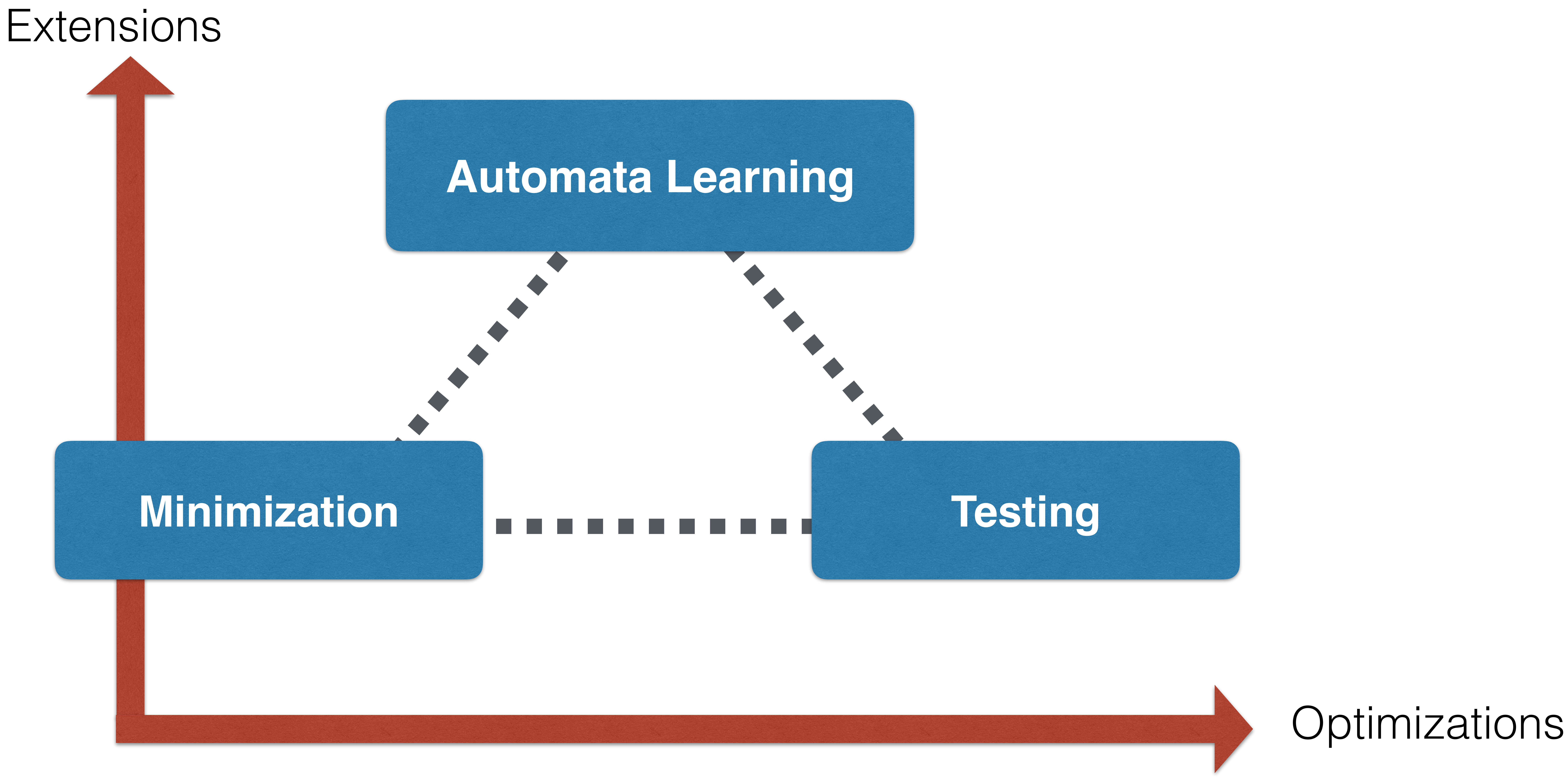
Optimising Automata Learning via Monads

Gerco van Heerdt, Matteo Sammartino, Alexandra Silva **(arXiv:1704.08055)**

Plug monads in

Powerset	**NFAs**
Powerset with intersection	**Universal automata**
Double powerset	**Alternating automata**
Maybe monad	**Partial automata**

Connections with other techniques



Basic ingredients

NetKat

$$sw = 6; pt = 8; dst := 10.0.1.5; pt := 5$$

For all packets located at port 8 of switch 6, set the destination address to 10.0.1.5 and forward it out on port 5.

CKA

[Hoare et al., JLAMP '11] [Kappe et al., CONCUR '17, ESOP '18]

$$a; b \parallel c; d$$

*Thread 1: do **a** and then **b*** *Thread 2: do **c** and then **d***

Verification using NetKAT

Reachability

- Can host A communicate with host B? Can every host communicate with every other host?

Security

- Does all untrusted traffic pass through the intrusion detection system located at C?

Loop detection

- Is it possible for a packet to be forwarded around a cycle in the network?

Verification using NetKAT

Soundness and Completeness [Anderson et al. 14]

- $\vdash p = q$ if and only if $[\![p]\!] = [\![q]\!]$

Decision Procedure [Foster et al. 15]

- NetKAT coalgebra
- efficient bisimulation-based decision procedure
- implementation in OCaml
- deployed in the Frenetic suite of network management tools

This project

Forwarding/
Filtering behaviour

Concurrency

Large data
domains

NetKat **CALF** CKA

in collaboration with amazon

Other research directions

Software Analysis

- Learning the “correct ways” of using undocumented code
- Learning-based automated test generation

Hardware Analysis

- Analysing concurrency in hardware, in collaboration with arm