Learning Nominal Automata

Joshua Moerman (Radboud University)

> Bartek Klin, Michał Szynwelski (Warsaw University)

> > POPL 2017 Paris



Matteo Sammartino, Alexandra Silva (University College London)





Active learning





No formal specification available? Learn it!



Finite alphabet of system's actions A

- set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$



Finite alphabet of system's actions A

Learner

- set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$





Finite alphabet of system's actions A

 $\mathbf{Q}: w \in \mathcal{L}?$ **A: Y/N**

Learner

- set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$





Finite alphabet of system's actions A

Learner

 $\mathbf{Q}: w \in \mathcal{L}?$ **A: Y/N** $\mathbf{Q}: \mathcal{L}(H) = \mathcal{L}?$

- set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$





Finite alphabet of system's actions A



- set of system behaviors is a **regular language** $\mathcal{L} \subseteq A^*$



Observation table



E

$row\colon S\cup S\cdot A\to 2^E$

 $row(s)(e) = 1 \iff se \in \mathcal{L}$



Observation table



E

Hypothesis automaton

$row: S \cup S \cdot A \to 2^{E}$ $row(s)(e) = 1 \iff se \in \mathcal{L}$

- states = $\{row(s) \mid s \in S\}$ final states = $\{row(s) \mid s \in S, row(s)(\epsilon) = 1\}$ initial state = $row(\epsilon)$
- transition function $row(s) \xrightarrow{a} row(sa)$



Observation table



E

Hypothesis automaton

states = $\{row(s) \mid s \in S\}$ final states initial state **Why is this correct?** $(\epsilon) = 1$

transition function $row(s) \xrightarrow{a} row(sa)$

$row: S \cup S \cdot A \to 2^{E}$ $row(s)(e) = 1 \iff se \in \mathcal{L}$



Table properties

Closed

$\forall t \in S \cdot A \quad \exists s \in S$

next state exists

Consistent

next state is unique

$$S \quad row(t) = row(s).$$







next state is unique

Table properties

 $row(s) \xrightarrow{a} row(sa)$

Closed

 $\forall t \in S \cdot A \quad \exists s \in S \quad row(t) = row(s).$

next state exists

Consistent







next state is unique





Applications : Hardware verification, security/network protocols...





Generalizations : Mealy machines, I/O automata, ...



... and shortcomings

operations on data values



- L* learns **control-flow**
- What if program model needs to express data-flow?

comparisons between data values







Automata over infinite alphabets (nominal automata)



Automata over infinite alphabets (nominal automata)

$A = \{a, b, c, d, \dots\}$ infinite alphabet

 $\mathcal{L} = \{aa, bb, cc, dd, \dots\}$



Automata over infinite alphabets (nominal automata)

 $A = \{a, b, c, d, ...\}$

 $\mathcal{L} = \{aa, bb, cc, dd, \dots\}$



infinite automaton

infinite alphabet



Automata over infinite alphabets (nominal automata)

 $A = \{a, b, c, d, ...\}$

 $\mathcal{L} = \{aa, bb, cc, dd, \dots\}$



infinite automaton

infinite alphabet



but with a finite representation





Ad-hoc algorithm? NO!



- **Challenges:**

 $\forall s_1, s_2 \in S \quad row(s_1) = row(s_2) \implies \forall a \in A \quad row(s_1a) = row(s_2a) \mid$

Ad-hoc algorithm? NO!

• table needs to be **infinite** code operates on infinite sets

 $\forall t \in S \cdot A \quad \exists s \in S \quad row(t) = row(s).$



Challenges:



Ad-hoc algorithm? NO!

• table needs to be **infinite** code operates on infinite sets

 $\forall t \in S \cdot A \quad \exists s \in S \quad row(t) = row(s).$ **Everything is "finitely representable"** $\forall s_1, s_2 \in S \quad row(s_1) = row(s_2) \implies \forall a \in A \quad row(s_1a) = row(s_2a)$



functions

(change category from **Set** to **Nom**)

Nominal automata theory

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota: Automata with Group Actions. LICS 2011

(finite) sets (orbit-finite) nominal sets equivariant functions

Nominal Programming languages

Bartek Klin, Michal Szynwelski: **SMT Solving for Functional Programming over Infinite** Structures. MSFP 2016



(change category from **Set** to **Nom**)



Nominal automata theory

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota: Automata with Group Actions. LICS 2011

(finite) sets (orbit-finite) nominal sets functions equivariant functions

Nominal L*

Nominal Programming languages

Bartek Klin, Michal Szynwelski: **SMT Solving for Functional Programming over Infinite** Structures. MSFP 2016



functions

(change category from **Set** to **Nom**)



Nominal automata theory

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota: Automata with Group Actions. LICS 2011

(finite) sets (orbit-finite) nominal sets equivariant functions



First non-trivial application of a new programming paradigm (**NLambda**)

Bartek Kunn, Milenar Ozymweisiki. **SMT Solving for Functional Programming over Infinite** Structures. MSFP 2016



(finite) sets

(change category from Set to Nom)



Nominal automata theory

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota: **Automata with Group Actions**. LICS 2011

(orbit-finite) nominal sets equivariant functions

Works with any (suitable) data domain

First non-trivial application of a new programming paradigm (**NLambda**)



(finite) sets

(change category from Set to Nom)



Nominal automata theory

Mikolaj Bojanczyk, Bartek Klin, Slawomir Lasota: **Automata with Group Actions**. LICS 2011

(orbit-finite) nominal sets equivariant functions

Works with any (suitable) data domain

First non-trivial application of a new programming paradigm (**NLambda**)

Eryk Kopczynski, Szymon Torunczyk: LOIS: syntax and semantics. POPL 2017



Correctness and termination

NLambda guarantees that each line of code terminates

Correctness and termination

NLambda guarantees that each line of code terminates

Algorithm correctness and termination from scratch?

Correctness and termination

Bart Jacobs, Alexandra Silva

Correctness and termination

NLambda guarantees that each line of code terminates

- Algorithm correctness and termination from scratch? Not really
 - Set-based proofs as guidelines
 - L* enjoys a nice category-theoretic generalization
- Automata Learning: A Categorical Perspective, Horizons of the Minds 2014

What we've done

- Nominal L*
- More in the paper: variations, Nominal NL*
- NLambda (Haskell) Implementation
- Experimental results



What's next...

- Improve NLambda
- Other active learning algorithms
- Other optimizations
- Applications: large-scale software, crypto protocols...



https://www.mimuw.edu.pl/~szynwelski/nlambda/



https://github.com/Jaxan/nominal-lstar

Try it yourself

The arcms structure can be	e selected below:	Orciered atoms	Prove Rest allowers	
		Ordered atoms	Freedom stream	
			Equality alotts	
lick the following example	is to see how the language	e worka:		
Atoms and formulas 8	Sola Conditional Sup	port and orbits		
Normal Vicence				
Encodered in the la	to a state of the second sectors of the	a strike station	an in defending many constructor, at	
Atoms can be compared as fold	WS OR A D. BOO A D. LT	a D. Le a D. De a b. Ct	a. For equality atoms only the first two	comparisons are defined.
Ferreulas				
The laukalogy and some distinguistics (impairs), web. Keen (impairs)	are created by functions iteras atoms) and Keep (equivalence)	and iffalling. More complex form	ulas are formed by connectives lunctions	A (conjunction), W (Solunction), set
not (eq a b / a eq a c)	U eç b o, (eç a b /l. e	s d pe ć== (o z p		
Solving				
To directly check whether the fo	mula is equal to taxa or data	as the have the following function	o: istrue, istalse, moive, We co	in solve the above fermula as follows:
moive é not (eg a h 🔿	neq s c) \/ eq b c, solv	e ő (og a b /∖ og a c) ==	or eq is a	
We can also use the function a	ingligy which returns the for	sprivice refit aller		
simplify 8 mon (eq.a b.	A neg a ci Vieg b c, s	implify 5 (eq.n.b. // eq.a	e) and og b e	

