Computation Theory in Sets with Atoms

Bartek Klin
University of Oxford

IFIP WG 2.2

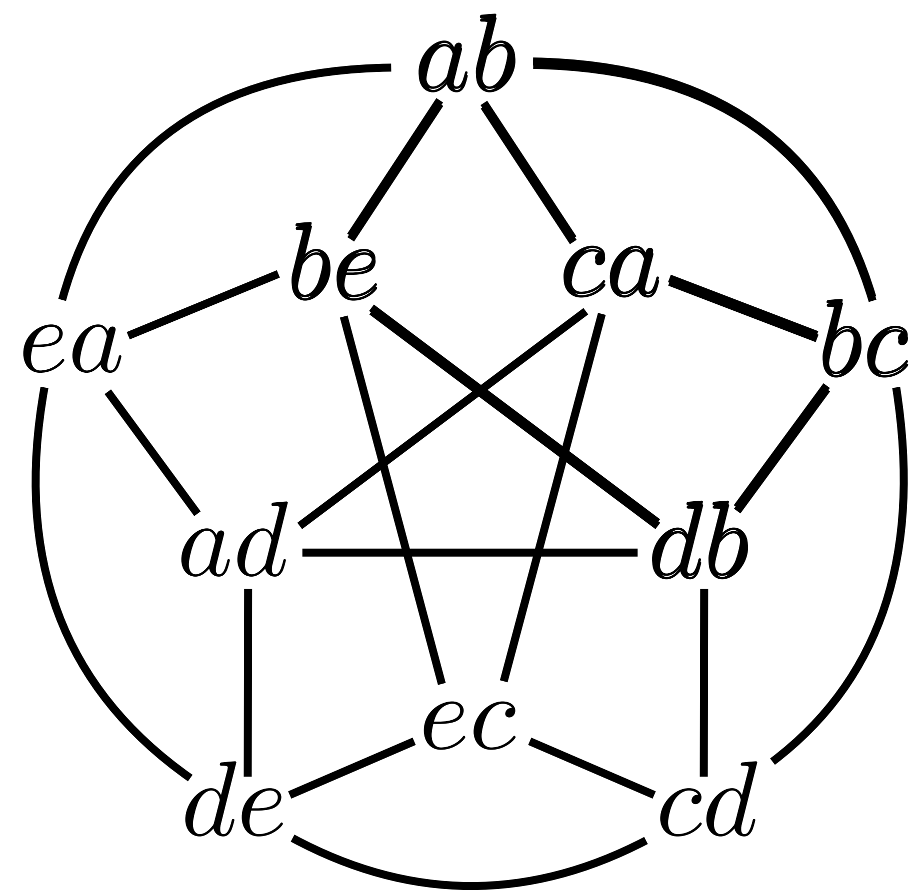
Bologna, 5-8/09/23

A puzzle

An infinite graph:

- nodes: ab $\qquad a \neq b \in \mathbb{N}$
- edges: $ab — bc$ $\qquad a \neq c$

Is it 3-colorable?



No.

Is it *k*-colorable?

Is 3-colorability decidable?

General theme

Replace finite structures

with infinite, but highly symmetric ones

in:

- automata theory
- computability theory
- modelling / verification
- algorithms

...

- all the way down to set theory

Plan

1. Register automata

2. Sets with atoms

-- Constraint satisfaction problems with atoms

-- Turing machines with atoms

-- Temporal logics with atoms

-- Linear algebra with atoms

3. Programming with atoms

-- ...

1
Register automata

Finite automata

A finite automaton is:

finite

- a set Q of states
- an alphabet Σ
- initial state $q_0 \in Q$, accepting states $F \subseteq Q$
- transition function $\delta : Q \times \Sigma \to Q$

  (or relation $\delta \subseteq Q \times \Sigma \times Q$)

Example language: $\displaystyle\bigcup_{a \in \Sigma} a(\Sigma \setminus a)^*$

What about infinite alphabets?

A register automaton is:

- a set Q of states
- a set R of registers
- an alphabet $\mathbb{A}$ (or $\Sigma \times \mathbb{A}$)
- initial state $q_0 \in Q$, accepting states $F \subseteq Q$
- configurations: $\Gamma = Q \times (\mathbb{A} \cup \{\perp\})^R$
- transition function $\delta : \Gamma \times \mathbb{A} \to \Gamma$

  (or relation $\delta \subseteq \Gamma \times \mathbb{A} \times \Gamma$)

finite

infinite

that $\boxed{\text{only checks } \mathbb{A} \text{ for equality.}}$?

“Only checking for equality”, syntactically

Every transition:

$$q \xrightarrow{a} q'$$

is guarded by a Boolean combination of conditions:

$$a = r_i \qquad a = r'_j \qquad r_i = r_j \qquad r_i = r'_j$$

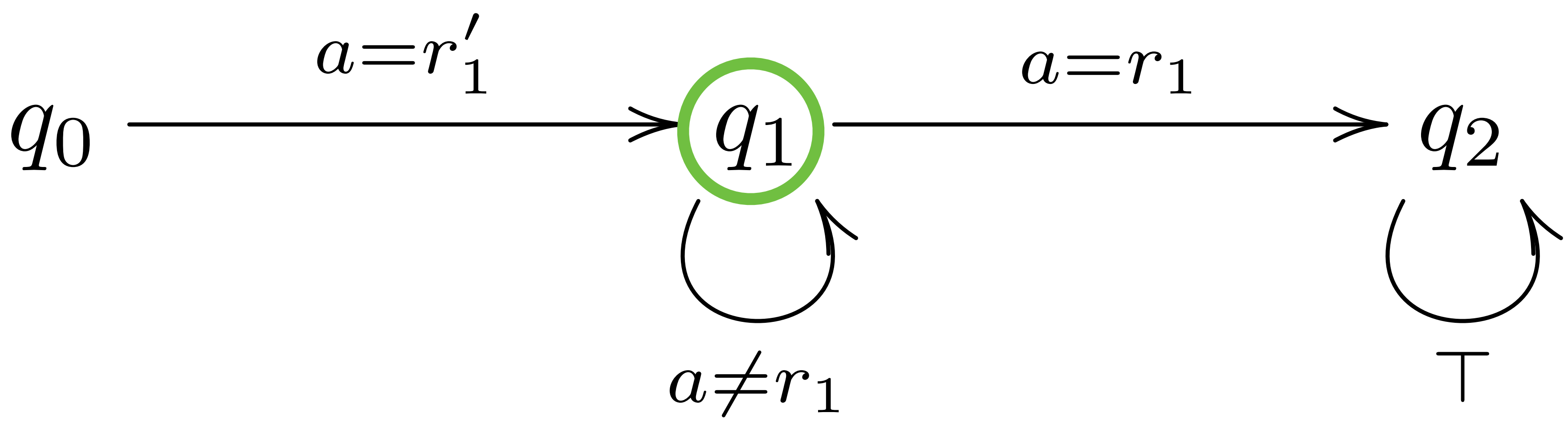
(so a is a “letter variable”, not an actual letter)

r_i - old i-th register

r'_i - new i-th register

Example

$$\bigcup_{a \in \mathbb{A}} a(\mathbb{A} \setminus a)^*$$



This is a deterministic register automaton.

“Only checking for equality”, semantically

Every bijection $\pi : \mathbb{A} \to \mathbb{A}$ acts on configurations:

$$(q, a_1, \ldots, a_k) \cdot \pi = (q, \pi(a_1), \ldots, \pi(a_k))$$

This defines a group action of $\mathrm{Aut}(\mathbb{A})$ on Γ.

We require δ to be equivariant:

if $(\gamma, a, \gamma') \in \delta$ then $(\gamma \cdot \pi, \pi(a), \gamma' \cdot \pi) \in \delta$
for all π.

Fact: The syntactic and the semantic conditions are equivalent.

11
Sets with Atoms

Slogans

X = set, function, relation, automaton,
Turing machine, grammar, graph,
system of equations...

X with atoms

Infinite but with lots of symmetries

orbit-finite

Infinite but symbolically finitely presentable

We can compute on them

Sets with atoms

$\mathbb{A}$ - a countable set of atoms

A hierarchy of universes:

$$\mathcal{U}_0 = \emptyset$$
$$\mathcal{U}_{\alpha+1} = \mathcal{P}\mathcal{U}_\alpha + \mathbb{A}$$
$$\mathcal{U}_\beta = \bigcup_{\alpha<\beta} \mathcal{U}_\alpha$$

Elements of sets with atoms are atoms
or other sets with atoms, in a well founded way

A canonical group action:

$$\_ \cdot \_ : \mathcal{U} \times \mathrm{Aut}(\mathbb{A}) \to \mathcal{U}$$

Finite support

$S \subseteq \mathbb{A}$ supports X if

$$\forall a \in S. \pi(a) = a \quad \text{implies} \quad x \cdot \pi = x$$

A legal set with atoms:

- has a finite support,
- every element has a finite support,
- and so on.

A set is equivariant if it has empty support.

Examples

$a \in \mathbb{A}$	is supported by	$\{a\}$
$\mathbb{A}$	is equivariant	
$S \subseteq \mathbb{A}$	is supported by	S
$\mathbb{A} \setminus S$	is supported by	S

Fact: $S \subseteq \mathbb{A}$ is fin. supp. iff it is finite or co-finite

$\mathbb{A}^{(2)} = \{(d, e) \mid d, e \in \mathbb{A}, d \neq e\}$ is equivariant

$\binom{\mathbb{A}}{2} = \{\{d, e\} \mid d, e \in \mathbb{A}, d \neq e\}$ is equivariant

Closure properties

Legal sets with atoms are closed under:

- unions, intersections, set differences
- Cartesian products
- taking finitely supported subsets
- quotienting by finitely supported equivalence relations

BUT not under powersets!

$\mathcal{P}(\mathbb{A})$ is equivariant but not legal.

They are closed under finite powersets $\mathcal{P}_{\mathrm{fin}}(\mathbb{A})$ and finitely supported powersets $\mathcal{P}_{\mathrm{fs}}(\mathbb{A})$

Relations and functions are sets too, so:

$R \subseteq X \times Y$ is equivariant iff

xRy implies $(x \cdot \pi) R (y \cdot \pi)$ for all π

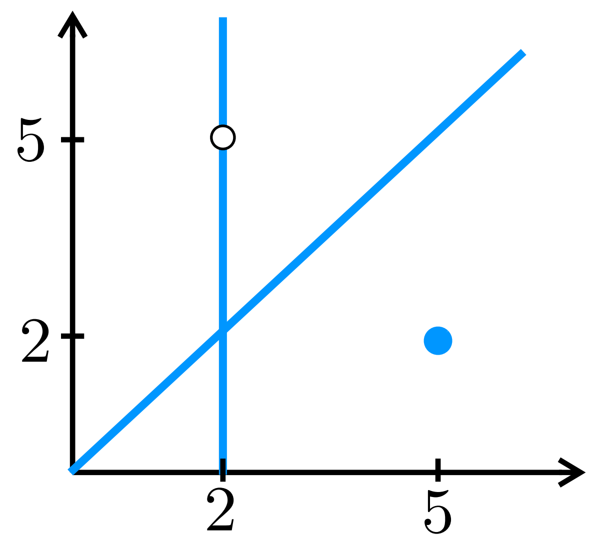
$f : X \to Y$ is equivariant iff

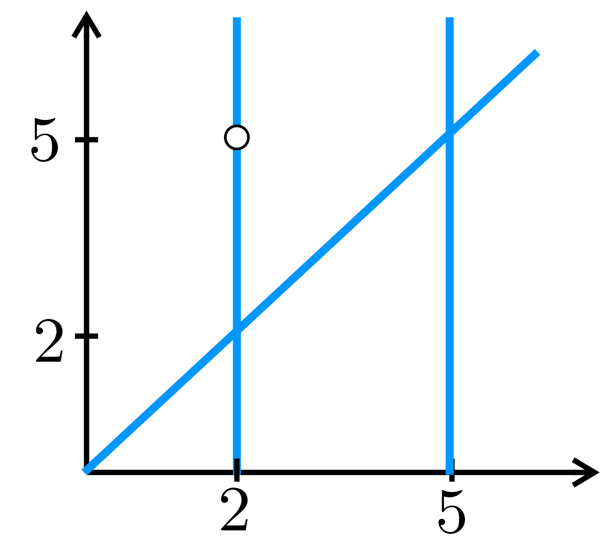
$f(x \cdot \pi) = f(x) \cdot \pi$ for all π

For fixed $2, 5 \in \mathbb{A}$:

$$R = \{(5,2)\} \cup \{(2,d) \mid d \neq 5\} \cup \{(d,d)\}$$

R





R^*

R , R^* are supported by $\{2, 5\}$

Examples ctd.

Equivariant binary relations on $\mathbb{A}$:

- empty
- equality
- total
- inequality

No equivariant function from $\binom{\mathbb{A}}{2}$ to $\mathbb{A}$, but

$$\{(\{a,b\},a) \mid a,b \in \mathbb{A}\}$$

is an equivariant relation.

Only equiv. functions from $\mathbb{A}^2$ to $\mathbb{A}$ are projections

Only equiv. function from $\mathbb{A}$ to $\mathbb{A}^2$ is the diagonal

Orbits

The orbit of x is the set $\{x \cdot \pi \mid \pi \in \mathrm{Aut}(\mathbb{A})\}$

Every equivariant set is a disjoint union of orbits.

Orbit-finite set if the union is finite.

More generally: the S-orbit of x is

$$\{x \cdot \pi \mid \pi \in \mathrm{Aut}_S(\mathbb{A})\}$$

Fact: An orbit-finite set is S-orbit-finite for every finite S.

Examples

Orbit-finite sets:

$$\mathbb{A} \qquad \mathbb{A}^n \qquad \binom{\mathbb{A}}{n}$$

$$\mathbb{A}^{\triangleleft} = \{\{(a,b,c),(b,c,a),(c,a,b)\} \mid a,b,c \in \mathbb{A}\}$$

- closed under finite union, intersection difference, finite Cartesian product
- but not under (even finite) powerset!

Not orbit-finite:

$$\mathbb{A}^* \qquad \mathcal{P}_{\text{fin}}(\mathbb{A})$$

Finite presentation

A set-builder expression:

$$\{e \mid a_1, \ldots, a_n \in \mathbb{A},\ \phi[a_1, \ldots, a_n, b_1, \ldots, b_m]\}$$

expression

bound variables

free variables

FO($=$)-formula

Add also $\emptyset$ and $\cup$.

Fact: s.-b. e. + interpretation of free vars. as atoms
= a hereditarily orbit-finite set with atoms

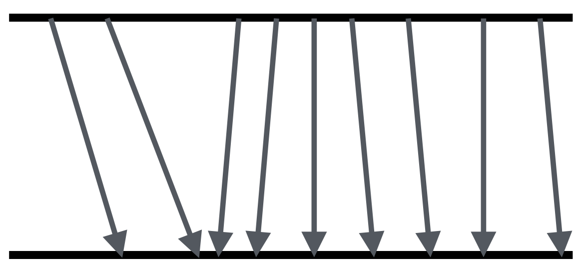
Fact: Every h. o.-f. set is of this form.

Ordered atoms

$$\mathbb{A} = \{a, b, c, d, \ldots\} \qquad \mathrm{Aut}(\mathbb{A})$$

can be replaced by

$$\mathbb{Q} \qquad \mathrm{Aut}(\mathbb{Q}, <)$$



New legal sets:

$$\{(a, b) : a, b \in \mathbb{A} : a < b\}$$

- orbit-finite sets remain orbit-finite
- equivariant functions are monotone-equivariant

A automaton with atoms is:

- a set Q of states
- an alphabet Σ
- initial state $q_0 \in Q$, accepting states $F \subseteq Q$
- transition function $\delta : Q \times \Sigma \to Q$

  (or relation $\delta \subseteq Q \times \Sigma \times Q$)

orbit-finite

equivariant

Fact: these are expressively equivalent to reg. aut.

Set theory with atoms

Sets with atoms are a topos

A lot of mathematics can be done with atoms

set → set with atoms

finite → orbit-finite

function → equivariant function

EXCEPT:

- axiom of choice fails, even orbit-finite choice
- powerset does not preserve orbit-finiteness

III
Programming with Atoms

Programming with finite sets

Haskell syntax used

```
type Set a = [a]

empty :: Set a

insert :: a -> Set a -> Set a

map :: (a -> b) -> Set a -> Set b

filter :: (a -> Bool) -> Set a -> Set a

sum :: Set (Set a) -> Set a

...
```

Example: transitive closure

```
comp :: Set (a,b) -> Set (b,c) -> Set (a,c)
comp s r = ...

transCl :: Set (a,a) -> Set (a,a)
transCl r =
    let r1 = comp r r in
      if isSubsetOf r1 r
      then r
      else transCl (union r1 r)
```

```
> transCl [(1,2),(2,3)]
[(1,2),(2,3),(1,3)]
```

Other examples

1. Graph 2-colorability

```
twoColorable :: Set (a,a) -> Bool
```

- look for cycles of odd length

2. Graph 3-colorability

```
threeColorable :: Set (a,a) -> Bool
```

- generate all 3-partitions of vertices
- for each of them, check legality

NLambda: a Haskell library

```haskell
type Atom
type Set a = ...

empty :: Set a
atoms :: Set Atom
insert :: a -> Set a -> Set a
map :: (a -> b) -> Set a -> Set b
sum :: Set (Set a) -> Set a
isEmpty :: Set a -> Formula

...
```

Example

```
> atoms
{a : for a in A}
> map (\a -> map (\b -> (a,b)) atoms) atoms
{{(a,b) : for b in A} : for a in A}
> sum it
{(a,b) : for a,b in A}
> filter (\(a,b) -> eq a b) it
{(a,a) : for a in A}
> forAll (\a -> member a atoms) atoms
True
```

Semantics

- Orbit-finite sets internally represented by FO formulas and set-builder expressions
- Condition evaluation delayed when possible:

```
> if (eq a b) (singleton c) atoms
{c : a=b, d : a!=b for d in A}
```

- Formulas evaluated by calling an SMT solver

```
> isEmpty atoms
False
```

Example: transitive closure

```
comp :: Set (a,b) -> Set (b,c) -> Set (a,c)
comp s r = ...

transCl :: Set (a,a) -> Set (a,a)
transCl r =
    let r1 = comp r r in
      if (isSubsetOf r1 r)
          r
          (transCl (union r1 r))
```

The same code!*

*essentially

Other examples

- Graph 2-colorability

```
twoColorable :: Set (a,a) -> Bool
```

- Angluin algorithm for automata learning
  - interact with a teacher to learn an automaton
  - Moerman, Sammartino, Silva, K., Szynwelski: *Learning nominal automata*, POPL'17

Also the same code*

*essentially

3-colorability

```
threeColorable :: Set (a,a) -> Bool
```

- generate all 3-partitions of vertices ...

Cannot be done!

Different code:

- if coloring exists then an equivariant one exists
- generate 3-partitions of orbits ...

```
supports :: NType a => [Atom] -> a -> Bool
orbits   :: NType a => Set a -> Set (Set a)
```

...

Ordered atoms needed

The easy, the hard & the impossible

Easy: code copied verbatim*

*essentially

```
transCl :: Set (a,a) -> Set (a,a)
twoColorable :: Set (a,a) -> Bool
learnAngluin :: ...
```

Hard: supports, orbits etc. required

```
threeColorable :: Set (a,a) -> Bool
```

Impossible: atom enumeration

```
toList :: Set a -> [a]
foldl :: (b -> a -> b) -> b -> Set a -> b
```

λX.(X with atoms)

A recipe for adding atoms to everything:

1. Take your favourite definition.
2. Replace all sets (relations, functions etc.) with sets with atoms (equivariant if you wish).
3. Replace every "finite" with "orbit-finite".
4. Check if your favourite theorems still hold.

(take with a pinch of salt)