Automata over Infinite Alphabets

Nikos Tzevelekos
Queen Mary University of London

MOVEP 2018

Joint work with:
– Andrzej Murawski (Oxford)
– Steven Ramsay (Bristol)
– Radu Grigore (Kent)
– Dino Distefano (QMUL, Facebook) and Rasmus Petersen

Based on joint ESSLLI'15 notes with Andrzej Murawski

Finite alphabets

- ☐ formal language theory
- ☐ theory of computation
- ☐ software verification

Chomsky hierarchy

regular, context-free, context-sensitive, recursively enumerable

Automata models

finite automata, pushdown automata, linear bounded automata, Turing machines

Other models

counter machines, queue machines, vector addition systems, Petri nets, higher-order pushdown automata

Infinite alphabets

What is a "reasonable" definition of automata over infinite alphabets?

Criteria

- ☐ finitariness (analogous to "finite-state")
- ☐ expressivity
- ☐ good closure properties
- ☐ decision procedures (emptiness, equivalence, etc.)
- ☐ minimisation
- ☐ connections to logic

Why infinite alphabets?

To model computational/mathematical/linguistic situations involving finite alphabets that:

- while bounded, they get so large/abstract that are best viewed as infinite,
- or alphabets are not known to be bounded,
- behaviour is parametric to the alphabet values – only their finite set of types/classes/hierarchy matters.

Computational significance:

- in programming languages semantics and verification: identifiers, references (pointers), objects, channels, etc.
- in XML document validation: attributes.

To preserve decidability, we assume the elements of the infinite alphabet are **names**: they can only be compared for equality.

OCaML

```
        Objective Caml version 4.05.0

# let v=ref(0);;
val v : int ref = {contents = 0}

# let vv=ref(v);;
val vv : int ref ref = {contents = {contents = 0}}

# !vv==v;;
- : bool = true

# !vv==ref(0);;
- : bool = false
```

Java

```
Object obj1 = new Object();
Object obj2 = new Object();

result = (obj1 == obj2);
```

```
public void foo() {
  // Create new list
  List x = new ArrayList();

  x.add(1); x.add(2);
  Iterator i = x.iterator();
  Iterator j = x.iterator();
  i.next(); i.remove(); j.next();
}
```

Plan

1. Register automata
   - ☐ Motivation for automata over infinite alphabets
   - ☐ Definitions and examples
   - ☐ Closure properties, emptiness and equivalence
   - ☐ Application to Java runtime verification
2. Fresh-register automata
3. Applications, other formalisms and extensions

Register automata (informally)

- A basic model of computation over an **infinite alphabet** $\mathcal{D}$.
- Elements of $\mathcal{D}$ are referred to as **names** or **data values**.

Features: finitely many states & finitely many $\mathcal{D}$-valued **registers**.

Let $r \in \mathbb{N}$ be the number of registers. Let us write $[r]$ for $\{1, \cdots, r\}$. Fix $\# \notin \mathcal{D}$ to be a *blank* symbol.

- By an r-**register assignment** we mean a map $\rho : [r] \to \mathcal{D} \uplus \{\#\}$ that is *injective* on $\mathcal{D}$:

$$\forall i, j \in [r].\ \rho(i) = \rho(j) \wedge i \neq j \implies \rho(i) = \#.$$

1	2	3	4
$\#$	d	d'	$\#$

$$\longmapsto \qquad [\#, d, d', \#]$$

Write Reg_r for the set of all such assignments.

Register automata

An r-**register automaton** (r-RA) is a tuple $\mathcal{A} = \langle Q, q_0, \rho_0, F, \delta \rangle$, where:

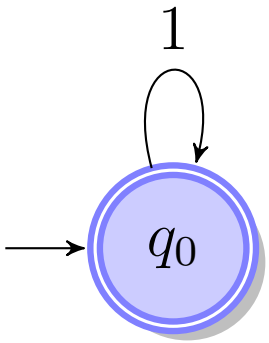
- Q is a *finite* set of states,
- $q_0 \in Q$ is the *initial state*,
- $\rho_0 \in Reg_r$ is the *initial register assignment*,
- $F \subseteq Q$ is the set of *final states*,
- $\delta \subseteq Q \times Op_r \times Q$ is the *transition relation*, where

$$Op_r = \{1, \cdots, r\} \cup \{1^\bullet, \cdots, r^\bullet\}.$$

Intuition

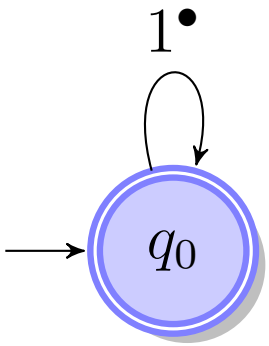
- $i \in Op_r$ represents reading the letter in the ith register
- $i^\bullet \in Op_r$ represents reading a letter that is not currently in any of the registers. It will overwrite the current value in register i.

Example 1-RAs ($\mathcal{A} = \langle Q, q_0, \rho_0, F, \delta \rangle$)



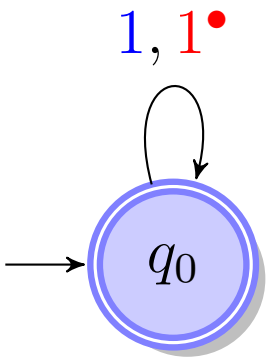
if $\rho_0 = [\#]$: $(q_0, [\#]) \to ? \longmapsto \{\epsilon\}$

if $\rho_0 = [d]$: $(q_0, [d]) \xrightarrow{d} (q_0, [d]) \xrightarrow{d} \cdots \longmapsto \{d\}^*$



$(q_0, [\#]) \xrightarrow{d_1} (q_0, d_1) \xrightarrow{d_2\ (\neq d_1)} (q_0, d_2) \xrightarrow{d_3\ (\neq d_2)} (q_0, d_3) \cdots$

$\longmapsto \{d_1 \cdots d_n \mid n \in \mathbb{N} \wedge \forall 1 \leq i < n.\, d_i \neq d_{i+1}\}$



$(q_0, [\#]) \xrightarrow{d_1} (q_0, d_1) \xrightarrow{d_2} (q_0, d_2) \xrightarrow{d_3} (q_0, d_3) \xrightarrow{d_4} \cdots$

$(q_0, d_1) \xrightarrow{d_1} (q_0, d_1) \xrightarrow{d_1} (q_0, d_1) \xrightarrow{d_2} \cdots$

$\longmapsto \mathcal{D}^*$

Configurations of $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$

- **Set of configurations** $Conf_r = Q \times Reg_r$
- **Evolution of configurations (successor configurations)**:
  - if $(q_1, i, q_2) \in \delta$ and $\rho_1(i) = d$ then $(q_1, \rho_1) \xrightarrow{d} (q_2, \rho_1)$
  - if $(q_1, i^\bullet, q_2) \in \delta$ and $d \notin \rho_1([r])$ (i.e. d is **fresh** for ρ_1) then $(q_1, \rho_1) \xrightarrow{d} (q_2, \rho_1[i \mapsto d])$.

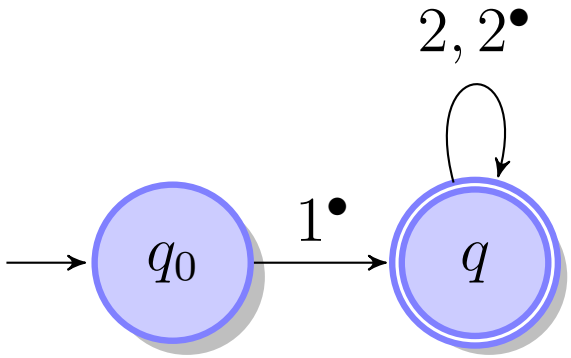
A **run** of $\mathcal{A}$ is a sequence $\kappa_0 \xrightarrow{d_1} \kappa_1 \cdots \xrightarrow{d_n} \kappa_n$ evolutions such that $\kappa_0 = (q_0, \rho_0)$.

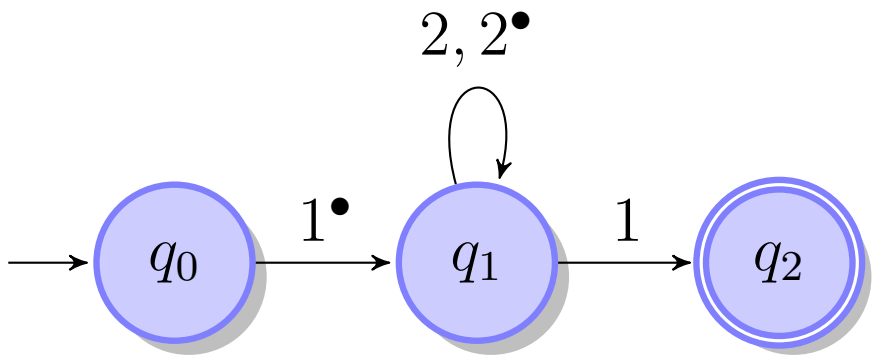
It is **accepting** if $\kappa_n = (q_n, \rho_n)$ for some $q_n \in F$. In this case we say that $\mathcal{A}$ **accepts** $d_1 \cdots d_n \in \mathcal{D}^*$.

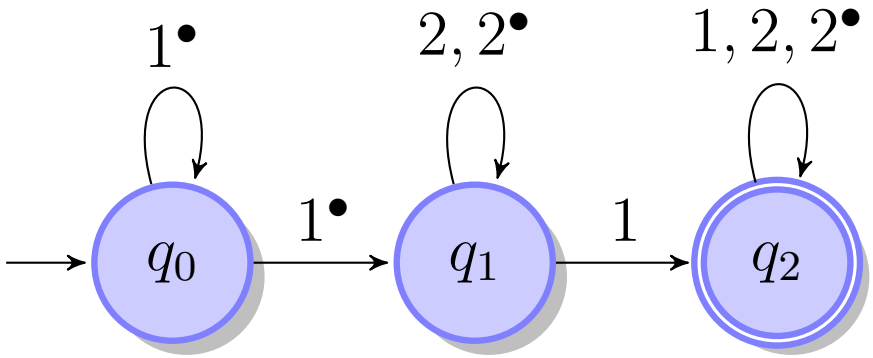
$\mathcal{L}(\mathcal{A}) = \{d_1 \cdots d_n \in \mathcal{D}^* \mid d_1 \cdots d_n \text{ accepted by } \mathcal{A}\}$.

More RA's

Consider the following 2-register automata ($\rho_0 = [\#, \#]$).





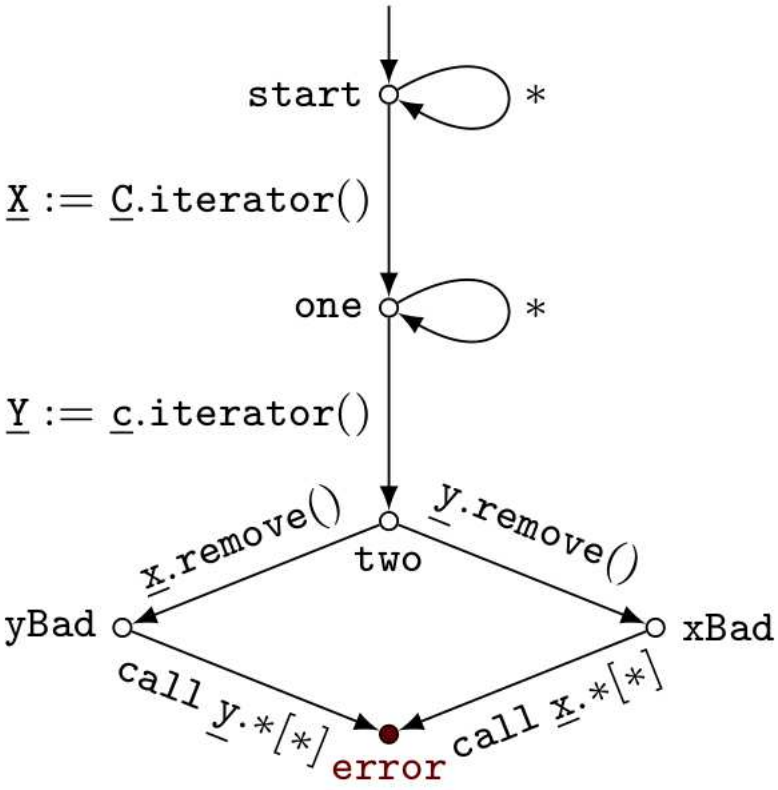


RA languages

n ranges over the set of natural numbers (including 0).

- $\{\epsilon\}$
- $\{d_1 \cdots d_n \in \mathcal{D}^* \mid \forall_{1 \leq i < n}.\, d_i \neq d_{i+1}\}$
- $\mathcal{D}^*$
- $\{d d_1 \cdots d_n \mid \forall_{1 \leq i \leq n}.\, d_i \neq d\}$
- $\{d d_1 \cdots d_n d \mid \forall_{1 \leq i \leq n}.\, d_i \neq d\}$
- $\{d_1 \cdots d_n \mid \exists i \neq j.\, d_i = d_j\}$
- $\{d_1 \cdots d_n \mid \forall_{i \neq j}.\, d_i \neq d_j\}$

Application: Runtime Verification based on RAs (TACAS'13)



rgrig.github.io/topl/

TOPL

monitor Java programs, easy

Overview

APIs often come with temporal constraints, such as 'method `x.foo(y)` must be called only after `y.bar()`'. Sometimes these are enforced at runtime: The library throws an exception if a constraint is broken. Sometimes the checks don't catch all bad usages, because that would be too slow.

Here is where TOPL comes in.

TOPL is a language that can express many temporal constraints. It is understood by a compiler that automatically inserts the appropriate checks in the bytecode of the library. The main advantages of this approach are the following:

- The description of the illegal usage patterns is almost as concise as the English prose of comments, but has a precise meaning.
- The API designer needs not write code to detect illegal sequences of calls. This code is generated automatically by the TOPL compiler.
- The user of the library can choose to use the non-instrumented and faster version of the library, after testing is completed.

RA properties: Name invariance

The one-step successor relation is invariant with respect to permutations on names.

Let π be a permutation of $\mathcal{D}$. Abusing notation, we can apply π to objects containing elements of $\mathcal{D}$, e.g. $\pi([d_1, d_2, \#]) = [\pi(d_1), \pi(d_2), \#]$.

Suppose $\kappa \xrightarrow{d} \kappa'$. Then

$$\pi(\kappa) \xrightarrow{\pi(d)} \pi(\kappa').$$

Consider a run $\kappa_0 \xrightarrow{d_1} \kappa_1 \xrightarrow{d_2} \cdots \xrightarrow{d_n} \kappa_n$ and a permutation π of $\mathcal{D}$. If $\pi(\kappa_0) = \kappa_0$ then the following is also a run of $\mathcal{A}$.

$$\kappa_0 = \pi(\kappa_0) \xrightarrow{\pi(d_1)} \pi(\kappa_1) \xrightarrow{\pi(d_2)} \cdots \xrightarrow{\pi(d_n)} \pi(\kappa_n)$$

Invariance theorems

Let $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$ be an r-RA.

- Let $\rho_0 = [\#, \cdots, \#]$ and π be a permutation of $\mathcal{D}$:

  If $d_1 \cdots d_n \in \mathcal{L}(\mathcal{A})$ then $\pi(d_1) \cdots \pi(d_n) \in \mathcal{L}(\mathcal{A})$.

- More generally, for arbitrary ρ_0 and π such that $\pi(\rho_0) = \rho_0$:

  If $d_1 \cdots d_n \in \mathcal{L}(\mathcal{A})$ then $\pi(d_1) \cdots \pi(d_n) \in \mathcal{L}(\mathcal{A})$.

Register automata do not distinguish specific names other than those provided in the initial assignment.

RA properties: Bounded alphabet

New names can be compared only with what the automaton has in memory.

Consider an accepting run

$$\kappa_0 \xrightarrow{d_1} \kappa_1 \xrightarrow{d_2} \cdots \xrightarrow{d_n} \kappa_n$$

and let $\mathcal{D}_0 = \{d_1, \ldots, d_h\}$ be the set of all names of ρ_0.

Pick fresh distinct names $d_{h+1}, \cdots, d_{r+1}$ and let $\mathcal{D}_b = \{d_1, \cdots, d_{r+1}\}$.

Then, the above accepting run can be transformed into another accepting run that features only elements of the *finite* set $\mathcal{D}_b$.

Crucial property

An r-RA can store at most r values from $\mathcal{D}$. Consequently, we can always implement the $i^\bullet$ transitions by drawing a letter from $\mathcal{D}_b$.

Boundedness

Theorem. Let $\mathcal{A}$ be an r-RA. There exists a subset $\mathcal{D}_b$ of $\mathcal{D}$ with $r+1$ elements such that if $w \in \mathcal{L}(\mathcal{A})$ then there exists $w' \in (\mathcal{D}_b)^*$ such that $w' \in \mathcal{L}(\mathcal{A})$ and $|w| = |w'|$.

I.e. if an r-RA $\mathcal{A}$ accepts a word then it accepts a word consisting of at most $r+1$ names.

Corollary. No register automaton can accept the language

$$\{d_1 \cdots d_n \in \mathcal{D}^* \,|\, n \in \mathbb{N}, \forall_{i \neq j}.\, d_i \neq d_j\}.$$

No r-register automaton accepts $\{d_1 \cdots d_{r+2} \in \mathcal{D}^* \,|\, \forall_{i \neq j}.\, d_i \neq d_j\}$, but there exists an r-register automaton that accepts

$$\{d_1 \cdots d_{r+1} \in \mathcal{D}^* \,|\, \forall_{i \neq j}.\, d_i \neq d_j\}.$$

Complementation

No register automaton will accept

$$\mathcal{L}_1 = \{d_1 \cdots d_n \in \mathcal{D}^* \mid n \in \mathbb{N}, \quad \forall_{i \neq j}.\, d_i \neq d_j\}.$$

But there exists a register automaton that accepts

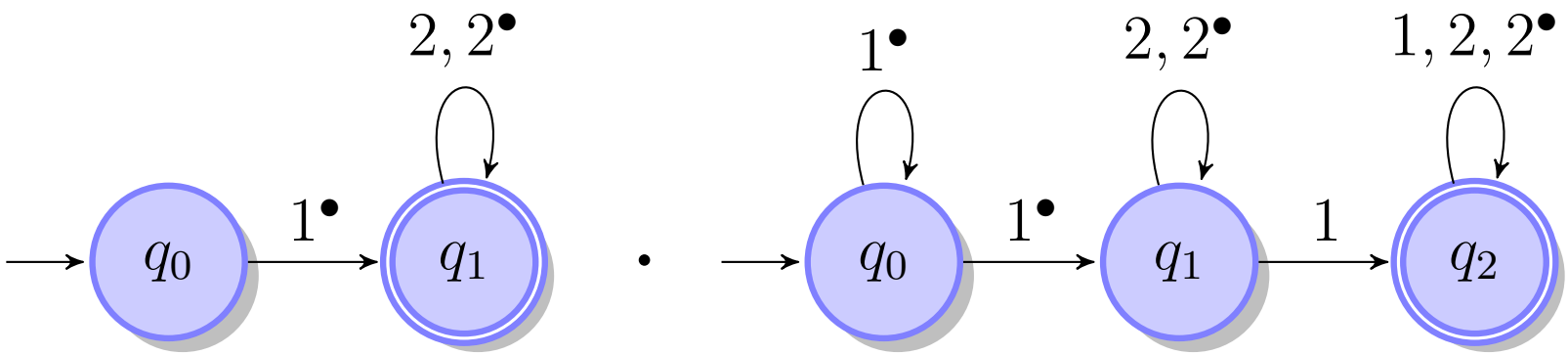
$$\mathcal{L}_2 = \{d_1 \cdots d_n \in \mathcal{D}^* \mid n \in \mathbb{N}, \quad \exists i \neq j.\, d_i = d_j\}.$$

and $\mathcal{L}_1 = \mathcal{D}^* \setminus \mathcal{L}_2$.

Conclusion. Languages accepted by register automata are not closed under complementation.

Closure properties

RA-languages turn out to be closed under union, intersection and concatenation, though.



RAs cannot be immediately composed:

- union of $\mathcal{A}_1$ and $\mathcal{A}_2$: what if different initial register assignments?
- concatenation of $\mathcal{A}_1$ and $\mathcal{A}_2$: final register assignments of $\mathcal{A}_1$ may not be the same as initial ones of $\mathcal{A}_2$
- same with Kleene-star...
- product of $\mathcal{A}_1$ and $\mathcal{A}_2$: registers need synchronising!

M-automata

Let us relax the injectivity requirement for register assignments and set

$$Reg_r^{\neg inj} = [r] \to \mathcal{D} + \{\#\}.$$

Accordingly, configurations will be defined by

$$Conf_r^{\neg inj} = Q \times Reg_r^{\neg inj}.$$

An r-**register** M-**automaton** (r-RA(M)) is a tuple $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$, where everything as before except:

- $\delta \subseteq Q \times \mathcal{P}([r]) \times \mathcal{P}([r]) \times Q$ is the *transition relation*.

Intuition: transition $q_1 \xrightarrow{X,Y} q_2$ represents

- reading a letter currently stored exactly in registers listed in X,
- writing it to registers listed in Y.

Runs of M-automata

$$(q_1, \rho_1) \xrightarrow{d} (q_2, \rho_2)$$

- $(q_1, X, Y, q_2) \in \delta$
- $X = \{i \mid \rho_1(i) = d\}$
- $\forall_{i \in Y}.\, \rho_2(i) = d$ and $\forall_{i \notin Y}.\, \rho_2(i) = \rho_1(i)$

Previous model

$$q_1 \xrightarrow{\{i\},\emptyset} q_2 \qquad\qquad q_1 \xrightarrow{\emptyset,\{i\}} q_2$$

Definition of runs and acceptance analogous to RAs.

Equiexpressivity

Theorem. For any r-register M-automaton $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$, there exists an $(r+1)$-register automaton $\mathcal{A}' = \langle Q', q_0', \rho_0', \delta', F' \rangle$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.

Ideas

- We cannot store multiple names. Instead we shall maintain a map that tells us in which (possibly multiple) registers each name would occur in the corresponding M-automaton, e.g. $\{1, 5, 7\}$.

- M-automata can overwrite multiple registers in a single step. In register automata we can achieve the same effect by writing to a single register and updating the above-mentioned map.

- M-automata can read a letter without recording it. Register automata can't. We shall use an extra register to handle such cases (we shall write to that register but it will be mapped to $\emptyset$).

Proof sketch: from r-RA(M) $\mathcal{A}$ to $(r+1)$-RA $\mathcal{A}'$

In order to keep track of the original locations of names we introduce *partitions*. We write $Part_r$ for the set of all partitions.

A **partition** of $[r+1]$ is a map

$$p : [r+1] \to \mathcal{P}([r])$$

such that $\forall_{i \neq j}.\, p(i) \cap p(j) = \emptyset$.

Construct $\mathcal{A}' = \langle Q', (q_0, p_0), \rho_0', \delta', F' \rangle$ by setting $Q' = Q \times Part_r$ and:

- Let $X_1, \cdots, X_n$ be the equivalence classes on $\{\, i \in [r] \mid \rho_0(i) \in \mathcal{D} \,\}$ determined by

$$i_1 \sim i_2 \iff \rho_0(i_1) = \rho_0(i_2)$$

and let $x_1, \cdots, x_k$ be the respective representatives. Define:

$$p_0(x) = \begin{cases} X_i & x = x_i \\ \emptyset & \text{otherwise} \end{cases} \quad \text{and} \quad \rho_0'(x) = \begin{cases} \rho_0(x) & p_0(x) \neq \emptyset \\ \# & \text{otherwise.} \end{cases}.$$

Proof sketch (transitions)

Given $q_1 \xrightarrow{X,Y} q_2$:

- for any $i \in [r+1]$ and $p_1 \in Part_r$ such that $p_1(i) = X$, add

$$(q_1, p_1) \xrightarrow{i} (q_2, p_2)$$

where $p_2(j) = \begin{cases} p_1(j) \setminus Y & j \neq i \\ p_1(i) \cup Y & j = i \end{cases}$.

- in addition, if $X = \emptyset$ then for any $p_1 \in Part_r$, take some $i \in [r+1]$ such that $p_1(i) = \emptyset$ (i must exist because values of p_1 are disjoint), and add

$$(q_1, p_1) \xrightarrow{i^\bullet} (q_2, p_2)$$

where $p_2(j) = \begin{cases} p_1(j) \setminus Y & j \neq i \\ Y & j = i \end{cases}$.

Finally, take $F' = F \times Part_r$.

Union

Theorem. Given an r_1-RA(M) $\mathcal{A}^1 = \langle Q^1, q_0^1, \rho_0^1, \delta^1, F^1 \rangle$ and an r_2-RA(M) $\mathcal{A}^2 = \langle Q^2, q_0^2, \rho_0^2, \delta^2, F^2 \rangle$ there exists an $(r_1 + r_2)$-RA(M) $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}^1) \cup \mathcal{L}(\mathcal{A}^2).$$

Idea. Create a new initial state q_0, merge initial assignments and make it possible for $\mathcal{A}$ to explore both $\mathcal{A}^1$ and $\mathcal{A}^2$.

$$\begin{array}{rcl} Q & = & \{q_0\} \uplus Q^1 \uplus Q^2 \\ \rho_0 & = & [\rho_0^1, \rho_0^2] \\ F & = & \{q_0 \mid q_0^1 \in F^1 \text{ or } q_0^2 \in F^2\} \uplus F^1 \uplus F^2 \end{array}$$

Proof sketch (embed each $\mathcal{A}^i$ into $\mathcal{A}$)

- For each $q \xrightarrow{X,Y} q'$ in δ^1 and $Z_2 \subseteq \{r_1+1, \cdots, r_1+r_2\}$, add to δ:

$$q \xrightarrow{X \cup Z_2, Y} q'$$

- for each $q_0^1 \xrightarrow{X,Y} q'$ in δ^1 and Z_2 [...], add to δ: $q_0 \xrightarrow{X \cup Z_2, Y} q'$.

- **Notation.** We write $r + X$ for $\{r + x \mid x \in X\}$.

  For each $q \xrightarrow{X,Y} q'$ in δ^2 and $Z_1 \subseteq \{1, \cdots, r_1\}$, add to δ:

$$q \xrightarrow{Z_1 \cup (r_1+X), r_1+Y} q'$$

- for each $q_0^2 \xrightarrow{X,Y} q'$ in δ^2 and Z_1 [...], add to δ: $q_0 \xrightarrow{Z_1 \cup (r_1+X), r_1+Y} q'$.

Intersection

Theorem. Given an r_1-RA(M) $\mathcal{A}^1 = \langle Q^1, q_0^1, \rho_0^1, \delta^1, F^1 \rangle$ and an r_2-RA(M) $\mathcal{A}^2 = \langle Q^2, q_0^2, \rho_0^2, \delta^2, F^2 \rangle$ there exists an $(r_1 + r_2)$-RA(M) $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2).$$

Idea. Run $\mathcal{A}_1$ and $\mathcal{A}_2$ in parallel.

- $Q = Q^1 \times Q^2$, $q_0 = (q_0^1, q_0^2)$, $F = F^1 \times F^2$ and $\rho_0 = [\rho_0^1, \rho_0^2]$.
- Given

$$q_1^1 \xrightarrow{X_1, Y_1} q_2^1 \in \delta^1 \quad \text{and} \quad q_1^2 \xrightarrow{X_2, Y_2} q_2^2 \in \delta^2,$$

add to δ:

$$(q_1^1, q_1^2) \xrightarrow{X_1 \cup (r_1 + X_2),\ Y_1 \cup (r_1 + Y_2)} (q_2^1, q_2^2).$$

Concatenation and Kleene star

Theorem. Given an r_1-RA(M) $\mathcal{A}^1 = \langle Q^1, q_0^1, \rho_0^1, \delta^1, F^1 \rangle$ and an r_2-RA(M) $\mathcal{A}^2 = \langle Q^2, q_0^2, \rho_0^2, \delta^2, F^2 \rangle$ there exists an $(r_1 + r_2)$-RA(M) $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$ such that

$$\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}^1) \cdot \mathcal{L}(\mathcal{A}^2).$$

Theorem. For any r-RA(M) $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$ there exists a $2r$-RA(M) $\mathcal{A}' = \langle Q', q_0', \rho_0', \delta', F' \rangle$ such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})^*$.

Taking stock: RAs are closed under union, intersection, concatenation and Kleene star. They are not closed under complementation.

Classic decision problems

- emptiness
$$\mathcal{L}(\mathcal{A}) = \emptyset$$
- universality
$$\mathcal{L}(\mathcal{A}) = \mathcal{D}^*$$
- equivalence
$$\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$$
- inclusion
$$\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$$

Emptiness

Let $\mathcal{A} = \langle Q, q_0, \rho_0, \delta, F \rangle$ be an r-RA.

- Recall the boundedness property: if $w \in \mathcal{L}(\mathcal{A})$ then there exists a finite set $\mathcal{D}_b \subseteq \mathcal{D}$ and $w' \in \mathcal{D}_b^*$ such that $w' \in \mathcal{L}(\mathcal{A})$.
- Concretely, we can take $\mathcal{D}_b$ to be an arbitrary superset of $\rho_0([r]) \cap \mathcal{D}$ of size $r + 1$.

Observation

Emptiness of $\mathcal{A}$ can be verified by searching through words from $\mathcal{D}_b^*$.

This is a finite alphabet, so the problem is reduced to one for a finite automaton with states:

$$(q, \rho) \in Q \times \{\rho \in Reg_r \mid \rho([r]) \cap \mathcal{D} \subseteq \mathcal{D}_b\}$$

The resulting finite automaton is *very* large! (solution in PSPACE)

Emptiness in NP (in fact, NP-complete)

Better observation

- For (non)emptiness, the content of the registers is irrelevant – what matters is whether they are empty or not. If there is an accepting run:

$$(q_0, \rho_0) \xrightarrow{d_1} \cdots \boxed{(q, \rho) \xrightarrow{d} \cdots} (q, \rho') \xrightarrow{d'} \cdots (q_F, \rho_F)$$

  with $\mathrm{dom}(\rho) = \mathrm{dom}(\rho')$ then there is one without the boxed bit.

- Moreover, register content cannot be deleted.

Thus, in a **minimal** accepting run we never see the same state with two register assignments of the same domain size.
Assuming $|Q| = n + 1$, the longest such has length $(r + 1)n + r$:

$$(q_0, [\#, \ldots, \#]) \xrightarrow{d_1} \cdots \xrightarrow{d_n} (q_n, [\#, \ldots, \#]) \xrightarrow{d^1}$$
$$(q_0, [d^1, \ldots, \#]) \xrightarrow{d'_1} \cdots \xrightarrow{d'_n} (q_n, [d^1, \ldots, \#]) \xrightarrow{d^2}$$
$$\cdots \xrightarrow{d'^r_1} \cdots \xrightarrow{d'^r_n} (q_n, [d^1, \ldots, d^r])$$

Universality

$$\mathcal{L}(\mathcal{A}) = \mathcal{D}^*$$

- In presence of closure under complementation, universality can be reduced to emptiness.
- But RAs are not closed under complementation.
- In fact, universality for RAs turns out undecidable.
- It follows that equivalence and inclusion are not decidable either.

Proof idea: Reduction from termination of 2-Counter Machines:

Given a 2-CM $\mathcal{M} = (Q, \delta, q_0, q_F)$, is there an accepting run for $\mathcal{M}$?

Transition labels: $\Sigma = \{\mathsf{inc}_i, \mathsf{dec}_i, \mathsf{ifz}_i \mid i = 1, 2\}$

Determinism: for each q, either $q \xrightarrow{\mathsf{inc}_i} q'$ or $q_1 \xleftarrow{\mathsf{dec}_i} q \xrightarrow{\mathsf{ifz}_i} q_2$ (or none at all)

Machine configurations: (q, c_1, c_2) with $q \in Q$ and $c_1, c_2 \in \mathbb{N}$

Representation scheme

An *accepting run* of $\mathcal{M}$ is a sequence of transitions of the form:

$$(q_0, c_{1,0}, c_{2,0}) \xrightarrow{\ell_1} (q_1, c_{1,1}, c_{2,1}) \xrightarrow{\ell_2} \cdots \xrightarrow{\ell_n} (q_n, c_{1,n}, c_{2,n})$$

such that $c_{1,0} = c_{2,0} = 0$, $q_n = q_F$ and, for each i, $q_i \xrightarrow{\ell_{i+1}} q_{i+1} \in \delta$ and:

- if $\ell_{i+1} = \mathsf{inc}_1$ then $c_{1,i+1} = c_{1,i} + 1$ and $c_{2,i+1} = c_{2,i}$
- if $\ell_{i+1} = \mathsf{dec}_1$ then $c_{1,i+1} = c_{1,i} - 1$ and $c_{2,i+1} = c_{2,i}$
- if $\ell_{i+1} = \mathsf{ifz}_1$ then $c_{1,i+1} = c_{1,i} = 0$ and $c_{2,i+1} = c_{2,i}$

and dually if ℓ_{i+1} refers to c_2.

Fix an encoding of labels: $\langle \mathsf{inc}_i \rangle = 1 + i$, $\langle \mathsf{dec}_i \rangle = 3 + i$, $\langle \mathsf{ifz}_i \rangle = 5 + i$.

Fix some $\smiley \in \mathcal{D}$. We will represent a run as above by a string in $\mathcal{D}^*$:

$$\smiley\, d_1^{\langle \ell_1 \rangle}\, \smiley\, d_2^{\langle \ell_2 \rangle}\, \smiley \cdots \smiley\, d_n^{\langle \ell_n \rangle}\, \smiley$$

with $d_1, \ldots, d_n \in \mathcal{D}$.

Correctness conditions

$$\☺\ d_1^{\langle \ell_1 \rangle}\ \☺\ d_2^{\langle \ell_2 \rangle}\ \☺ \cdots \☺\ d_n^{\langle \ell_n \rangle}\ \☺$$

- each inc_i has a fresh name: $\ell_k = \mathsf{inc}_i \implies \forall k' < k.\, d_{k'} \neq d_k$
- ditto for each ifz_i
- the j-th dec_i has the same name as the j-th inc_i, which must precede it: if ℓ_k is the j-th dec_i from the start of the word, then $\exists k' < k$:
  - $d_{k'} = d_k$ and $\ell_{k'}$ is the j-th inc_i from the start

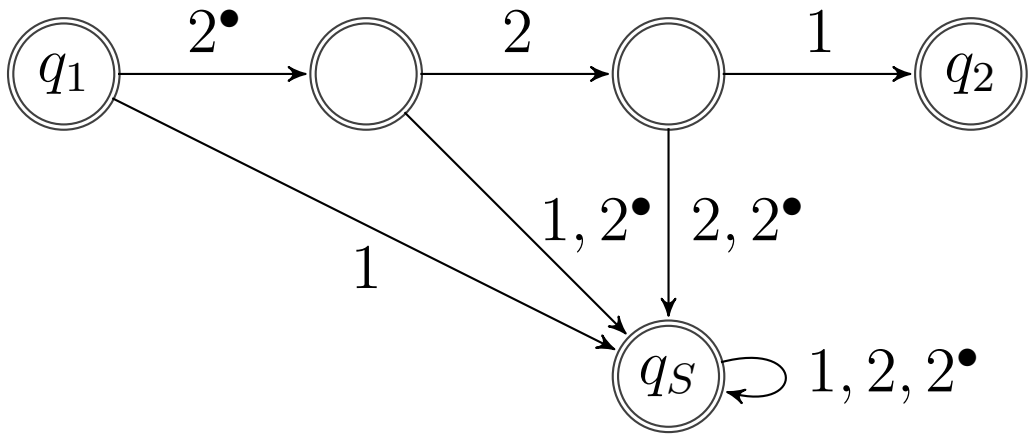
  In other words, each counter increment is assigned a fresh name, and the same name is assigned to each corresponding decrement.
- if $\ell_k = \mathsf{ifz}_i$ and there are j inc_i's and j' dec_i's before ℓ_k then $j = j'$.

Key idea: For each condition above, we can construct a register automaton gadget that will detect a violation (use nondeterminism).
We put our gadgets together into a 3-RA $\mathcal{A}_{\mathcal{M}}$:

$$\mathcal{M} \text{ terminates} \iff \mathcal{A}_{\mathcal{M}} \text{ does } \textit{not} \text{ accept any word}$$

Transition gadgets (work on $d_1^{\langle \ell_1 \rangle}$ ☺ $d_2^{\langle \ell_2 \rangle}$ ☺ $\cdots$ ☺ $d_n^{\langle \ell_n \rangle}$ ☺)

Let us assume that register 1 contains ☺. For each transition $q_1 \xrightarrow{\mathsf{inc}_1} q_2$ of $\mathcal{M}$ we construct a gadget (note $\langle \mathsf{inc}_1 \rangle = 2$):



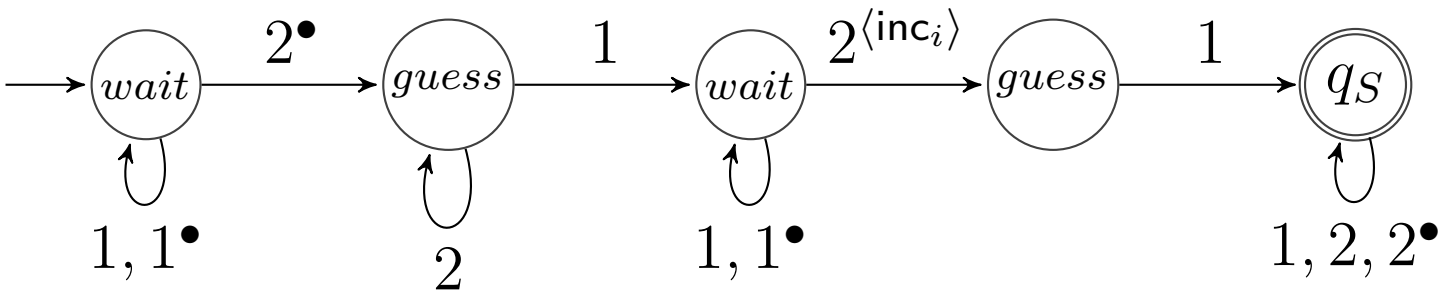
and similarly if $q_1 \xrightarrow{\mathsf{inc}_2} q_2$. We let q_S be some sink state.

The gadgets for $q_2 \xleftarrow{\mathsf{dec}_i} q_1 \xrightarrow{\mathsf{ifz}_i} q_2'$ are similar, though more elaborate...

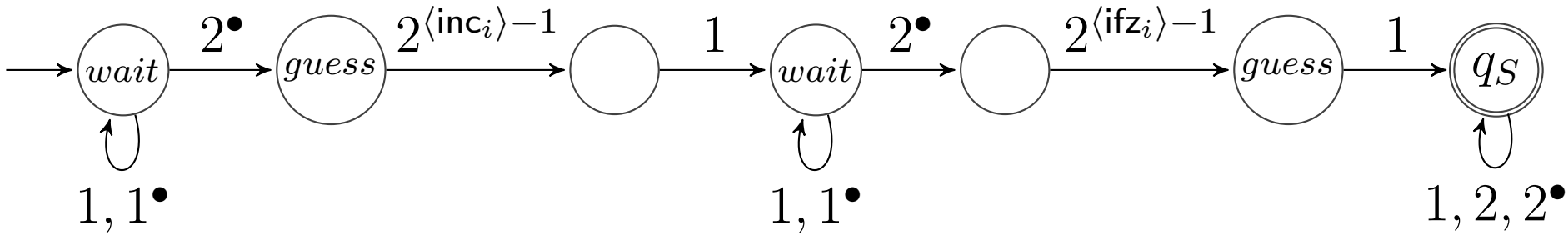
We put all transition gadgets together into a 2-RA $\mathcal{A}_1$.
We make q_F *not final* in $\mathcal{A}_1$ – all of its other states are final.

Correctness gadgets (cf. $d_1^{\langle \ell_1 \rangle} \, ☺ \, d_2^{\langle \ell_2 \rangle} \, ☺ \cdots ☺ \, d_n^{\langle \ell_n \rangle} \, ☺$)

- Gadget $\mathcal{A}_{\mathsf{inc}_i}$ spots any inc_i that has an *unfresh* name (same for $\mathcal{A}_{\mathsf{ifz}_i}$):

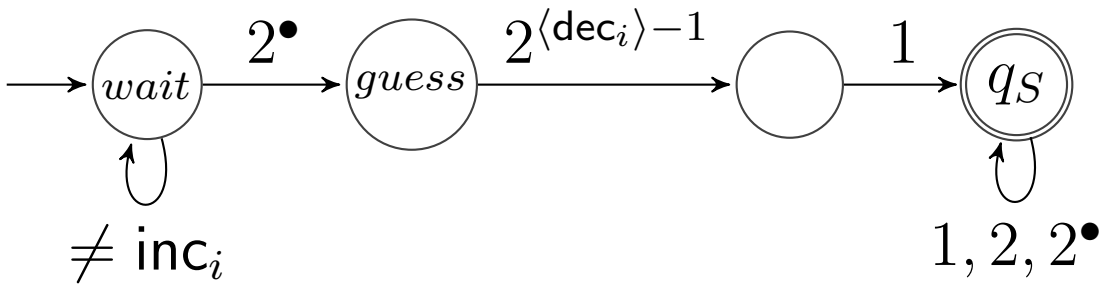


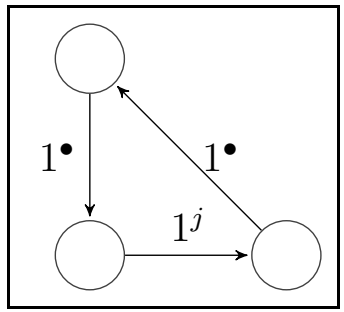
- Gadget $\mathcal{A}'_{\mathsf{ifz}_i}$ spots any ifz_i with an unmatched inc_i before it:



Correctness gadgets 2 (cf. $d_1^{\langle \ell_1 \rangle}$ ☺ $d_2^{\langle \ell_2 \rangle}$ ☺ $\cdots$ ☺ $d_n^{\langle \ell_n \rangle}$ ☺)

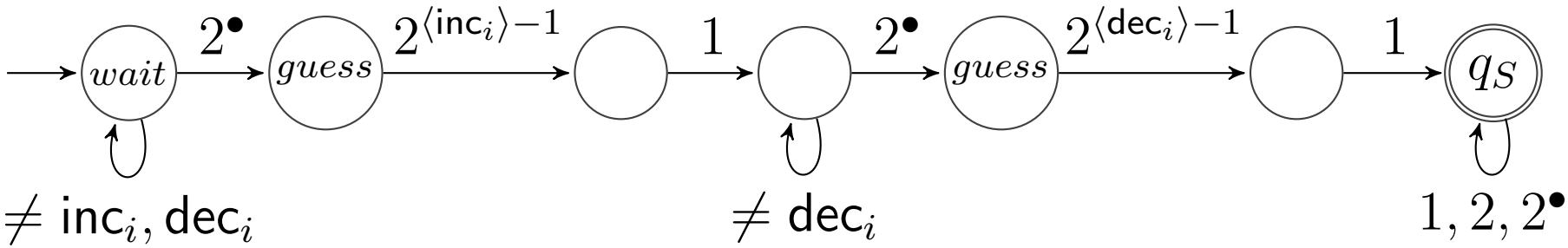
- Gadget $\mathcal{A}_{\mathsf{dec}_i}$ spots any dec_i that has no preceding inc_i:





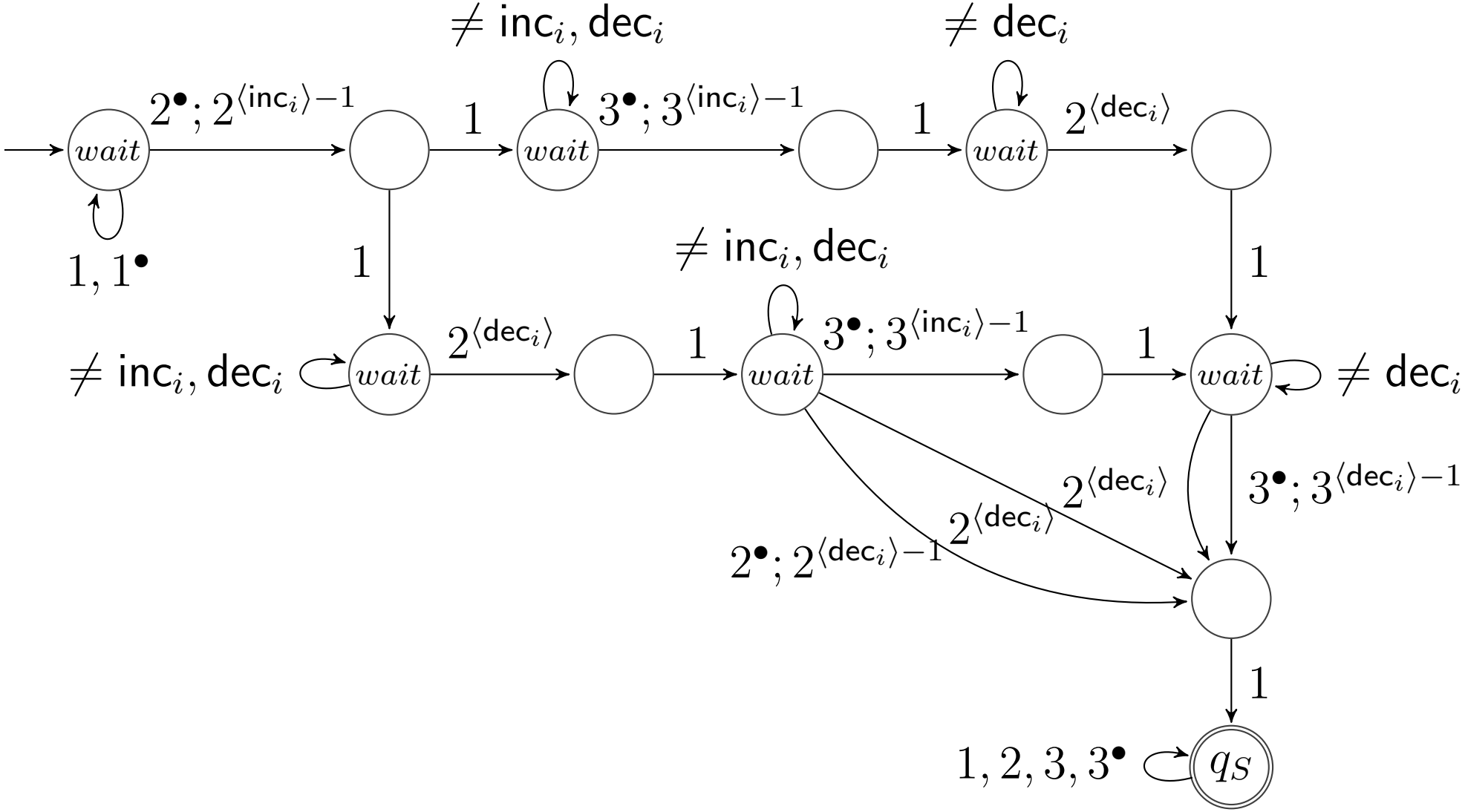
$\neq \mathsf{inc}_i$ are the components in the box, for $j \in \{1, ..., 6\} \setminus \{\langle \mathsf{inc}_i \rangle - 1\}$.

- Gadget $\mathcal{A}'_{\mathsf{dec}_i}$ spots if the first dec_i does not match the first inc_i:



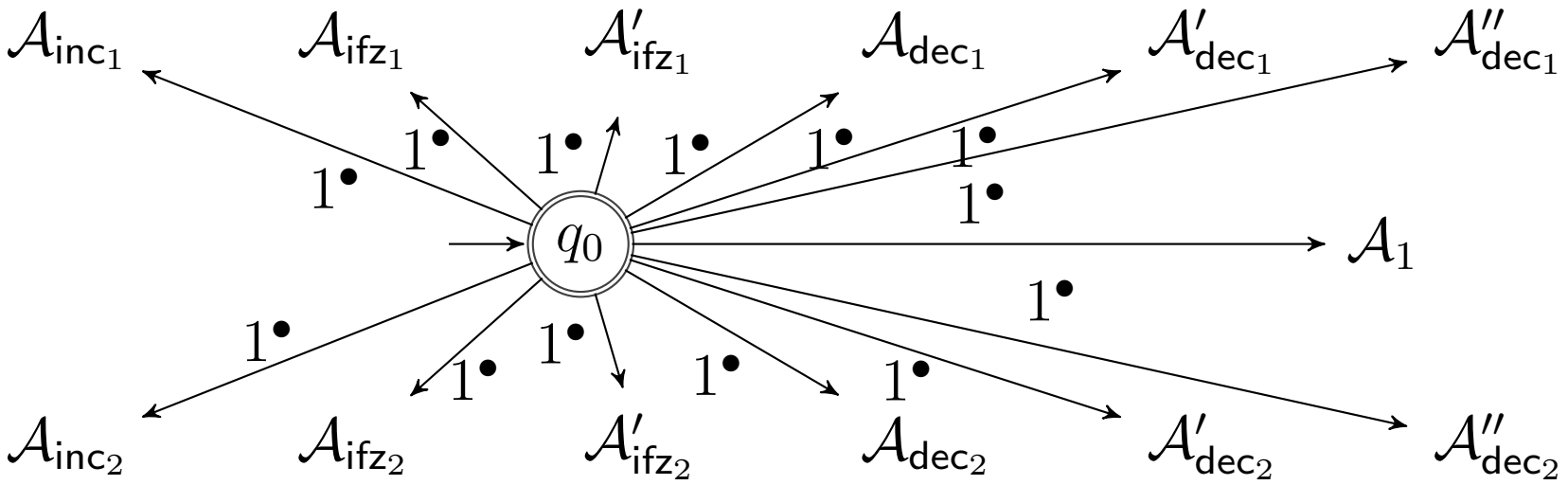
Correctness gadgets 3 (cf. $d_1^{\langle \ell_1 \rangle} \ \☺ \ d_2^{\langle \ell_2 \rangle} \ \☺ \cdots \☺ \ d_n^{\langle \ell_n \rangle} \ \☺$)

- $\mathcal{A}''_{\mathsf{dec}_i}$ spots if any subsequent dec_i does not match its corresponding inc_i:



Putting all together

The resulting 3-RA $\mathcal{A}_{\mathcal{M}}$ is now the following (with $\rho_0 = [\#, \#, \#]$):



Theorem. Given a 2-CM $\mathcal{M}$ as above we can effectively construct a 3-RA $\mathcal{A}_{\mathcal{M}}$ such that $\mathcal{M}$ terminates iff $\mathcal{L}(\mathcal{A}_{\mathcal{M}}) \neq \mathcal{D}^*$.

Hence, given any 3-RA $\mathcal{A}$, deciding whether $\mathcal{L}(\mathcal{A}) = \mathcal{D}^*$ is undecidable.

Summing up

- emptiness (NP-complete)

$$\mathcal{L}(\mathcal{A}) = \emptyset$$

- universality (undecidable)

$$\mathcal{L}(\mathcal{A}) = \mathcal{D}^*$$

- equivalence (undecidable)

$$\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$$

- inclusion (undecidable)

$$\mathcal{L}(\mathcal{A}_1) \subseteq \mathcal{L}(\mathcal{A}_2)$$

Plan

1. Register automata
   - ☐ Motivation for automata over infinite alphabets
   - ☐ Definitions and examples
   - ☐ Closure properties, emptiness and equivalence
   - ☐ Application to Java runtime verification
2. Fresh-register automata
   - ☐ Adding freshness
   - ☐ Basic properties
   - ☐ Bisimulation equivalence
3. Applications, other formalisms and extensions

Freshness

We saw automata for recognising languages like:

$$\mathcal{L} = \{d_1 d_2 \cdots d_n \in \mathcal{D}^* \mid n \geq 0 \wedge \forall i.\, d_i \neq d_{i+1}\}$$

$$\mathcal{L} = \{d_0 d_1 d_2 \cdots d_n \in \mathcal{D}^* \mid n \geq 0 \wedge \forall i > 0.\, d_i \neq d_0\}$$

Such languages are based on *local freshness*: being able to distinguish a name from a *bounded* number of names in memory.

Consider this language that describes e.g. a memory allocator in Java or ML:

$$\mathcal{L}_{\text{fresh}} = \{d_1 d_2 \cdots d_n \in \mathcal{D}^* \mid n \in \mathbb{N} \wedge \forall_{i \neq j}.\, d_i \neq d_j)\}$$

Such examples require *global freshness*, which we examine next.

Fresh-Register Automata

An **r-Fresh-Register Automaton (r-FRA)** is a tuple $\mathcal{A} = (Q, q_0, \rho_0, \delta, F)$, where:

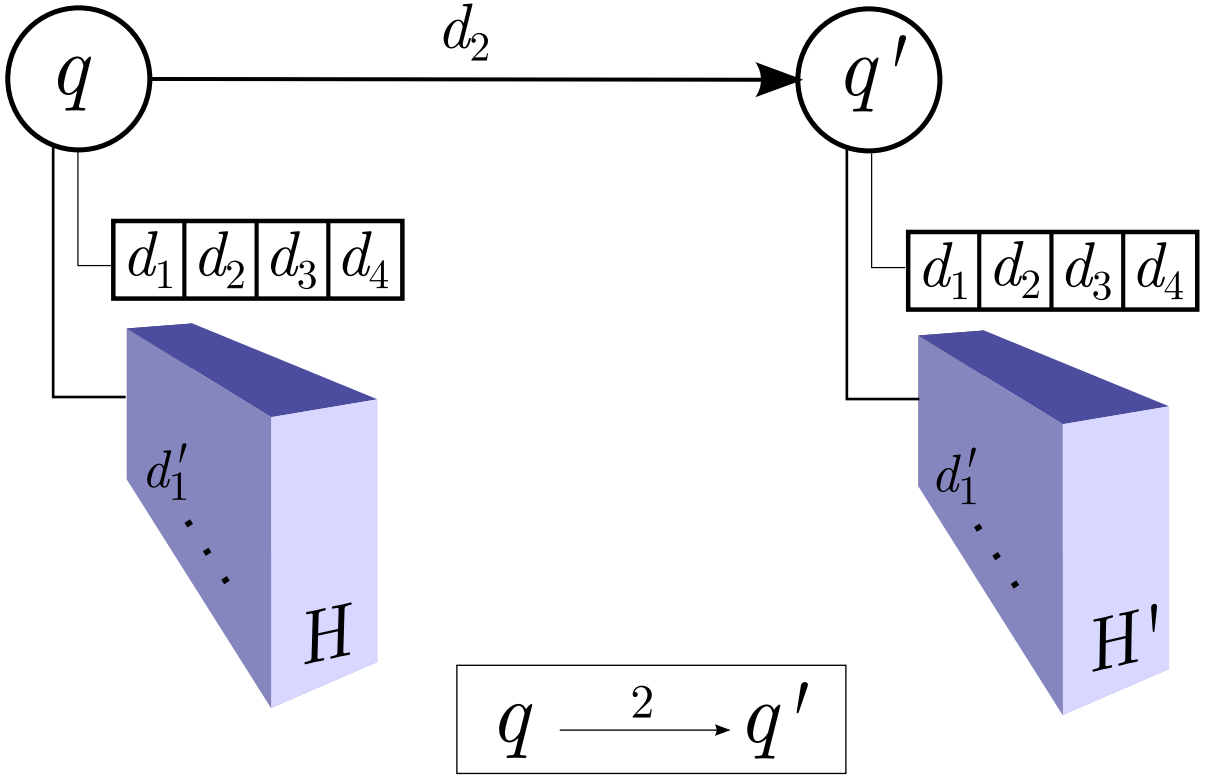
- Q is a finite set of states,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final states,
- $\rho_0 \in Reg_r$ is the initial r-register assignment,
- and $\delta \subseteq Q \times Op_r^f \times Q$ is the transition relation,

where $Op_r^f = \{i, i^\bullet, i^\circledast \mid 1 \leq i \leq r\}$.

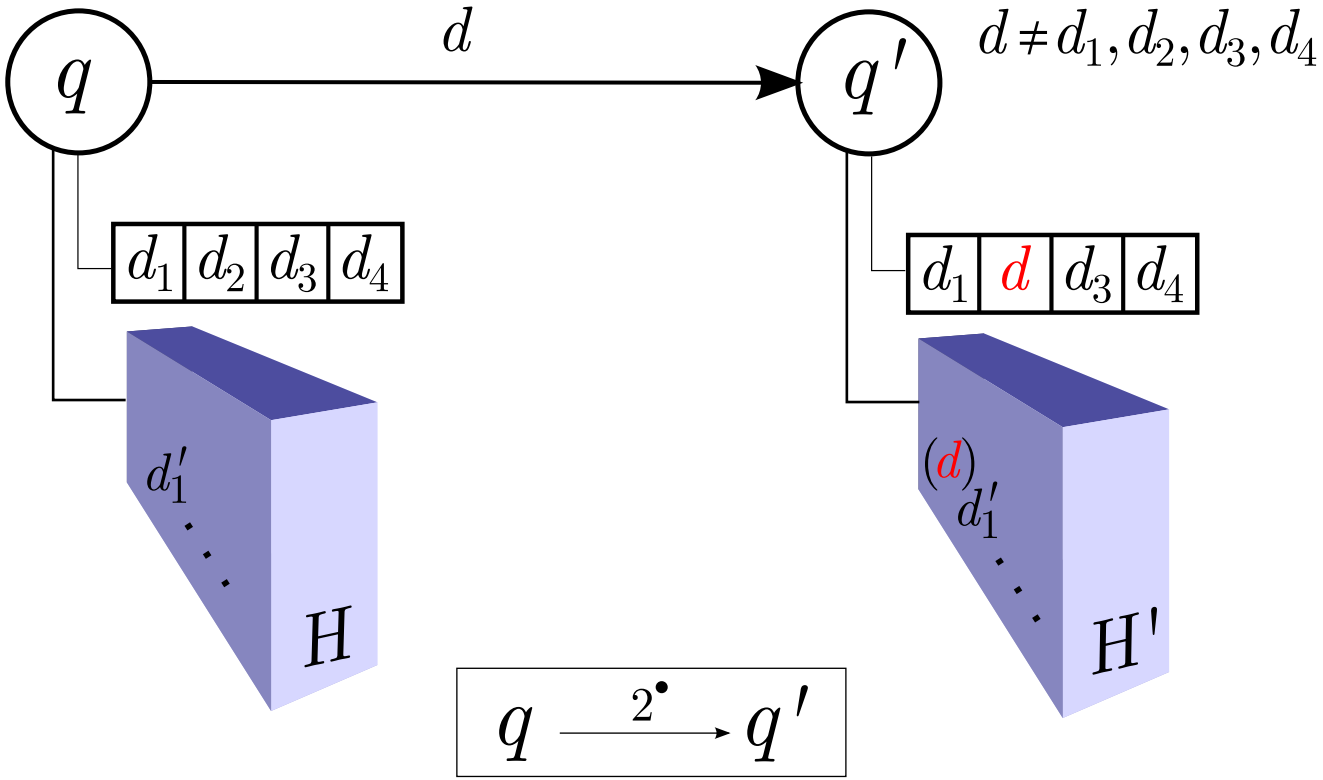
Thus, the new operation is: $q \xrightarrow{i^\circledast} q'$

It means: *accept a globally fresh name and store it in register i*

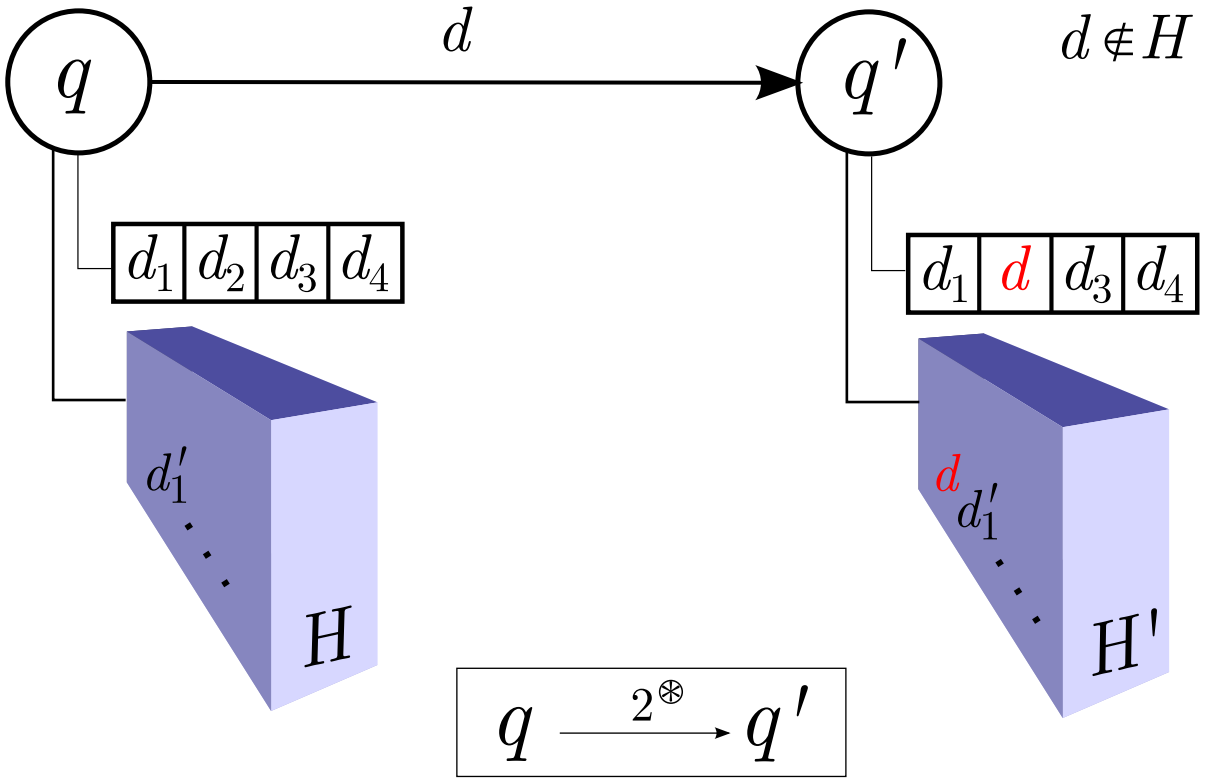
Semantics of FRAs: in pictures



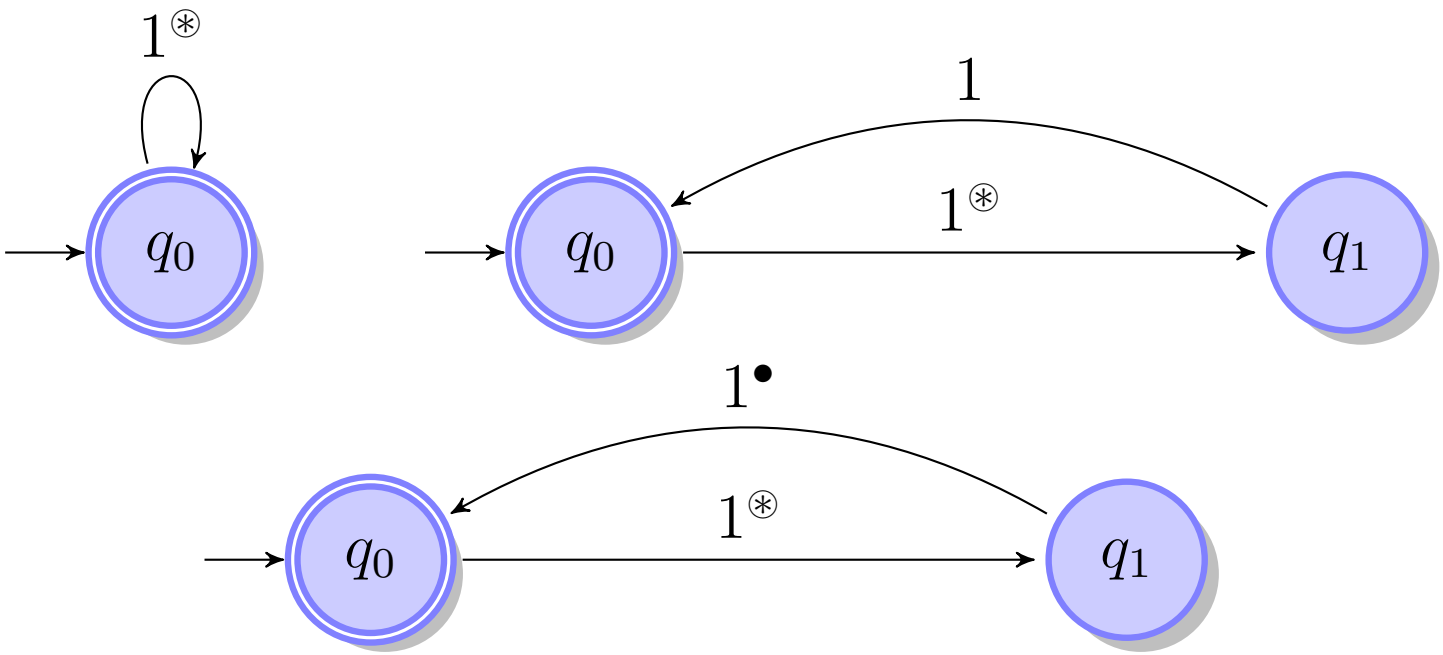
Semantics of FRAs: in pictures



Semantics of FRAs: in pictures



Examples



$$\mathcal{L}_{\text{fresh}} = \{d_1 d_2 \cdots d_n \in \mathcal{D}^* \mid n \in \mathbb{N} \wedge \forall_{i \neq j}. (d_i \neq d_j)\}$$

$$\mathcal{L} = \{d_1 d_1 d_2 d_2 \cdots d_n d_n \in \mathcal{D}^* \mid n \in \mathbb{N} \wedge \forall_{i \neq j}. (d_i \neq d_j)\}$$

$$\mathcal{L}' = \{d_1 d_1' d_2 d_2' \cdots d_n d_n' \in \mathcal{D}^* \mid n \in \mathbb{N} \wedge \forall_{i < j}. (d_j \neq d_i, d_i', d_j')\}$$

Formal semantics of FRAs

Let $\mathcal{A} = (Q, q_0, \rho_0, \delta, F)$ be an r-FRA. We extend configurations by:

$$Conf^f_{\mathcal{A}} = \{(q, \rho, H) \in Q \times Reg_r \times \mathcal{P}_{\text{fin}}(\mathcal{D}) \mid \nu(\rho) \subseteq H\}$$

Notation: For any X, we write $\nu(X)$ for the set of all names in X.
H is a *history*: the set of all names seen so far by the automaton.

An **evolution** $(q_1, \rho_1, H_1) \xrightarrow{d} (q_2, \rho_2, H_2)$ between configurations needs to satisfy one of the following conditions, for some $i \in [r]$:

- $(q_1 \xrightarrow{i} q_2) \in \delta$, and $\rho_1(i) = d$, $\rho_2 = \rho_1$ and $H_2 = H_1$;
- $(q_1 \xrightarrow{i^\bullet} q_2) \in \delta$, and $d \notin \nu(\rho_1)$, $\rho_2 = \rho_1[i \mapsto d]$ and $H_2 = H_1 \cup \{d\}$;
- $(q_1 \xrightarrow{i^\circledast} q_2) \in \delta$, and $d \notin H_1$, $\rho_2 = \rho_1[i \mapsto d]$ and $H_2 = H_1 \cup \{d\}$.

The **initial configuration** is: $\kappa_0 = (q_0, \rho_0, \nu(\rho_0))$.

The *configuration graph* of $\mathcal{A}$ contains all its evolutions.

Closure properties

Following a similar route as for RAs, we can show:

- for any pair of FRAs $\mathcal{A}_1, \mathcal{A}_2$ there is FRA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$
- for any pair of FRAs $\mathcal{A}_1, \mathcal{A}_2$ there is FRA $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$

However, we lose the following closures:

concatenation e.g. $\mathcal{L}^2_{\text{fresh}}$ $(= \mathcal{L}_{\text{fresh}}\mathcal{L}_{\text{fresh}})$ is not FRA-recognisable

Kleene star we can find a similar example as above [exercise]

complement e.g. $\overline{\mathcal{L}^2_{\text{fresh}}}$ can be recognised by an (F)RA [exercise]

Non-Examples

$$\mathcal{L}_{\text{palindrome}} = \{d_1 d_2 \cdots d_n d_n \cdots d_2 d_1 \in \mathcal{D}^* \mid n \geq 0\}$$
$$\mathcal{L}^2_{\text{fresh}} = \{ww' \in \mathcal{D}^* \mid w, w' \in \mathcal{L}_{\text{fresh}}\}$$

These follow from the next boundedness result.

Theorem. *Let $\mathcal{L}$ be an FRA-recognisable language. There is an $r \in \mathbb{N}$ such that, for any word $w_1 w_2 \in \mathcal{L}$ with $\nu(w_2) \subseteq \nu(w_1)$, there is some $w_1 w_2' \in \mathcal{L}$ with $|w_2'| = |w_2|$ and $|\nu(w_2')| \leq r + 1$.*

Sketch: Suppose $\mathcal{L}$ accepted by an r-FRA, and $w_1 w_2 \in \mathcal{L}$ has run:

$$\underbrace{\kappa_0 \xrightarrow{d_1} \cdots \xrightarrow{d_n} \kappa_n}_{w_1} \underbrace{\xrightarrow{d'_1} \kappa_{n+1} \cdots \xrightarrow{d'_m} \kappa_{n+m}}_{w_2}$$

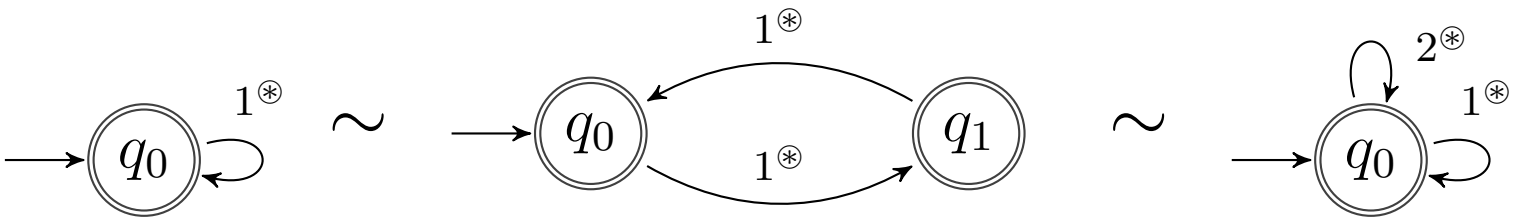
Then, $\nu(w_2) \subseteq \nu(w_1)$ implies that there are no $\circledast$ transitions after κ_n, so use boundedness result for RAs.

Another notion of equivalence: Bisimulation

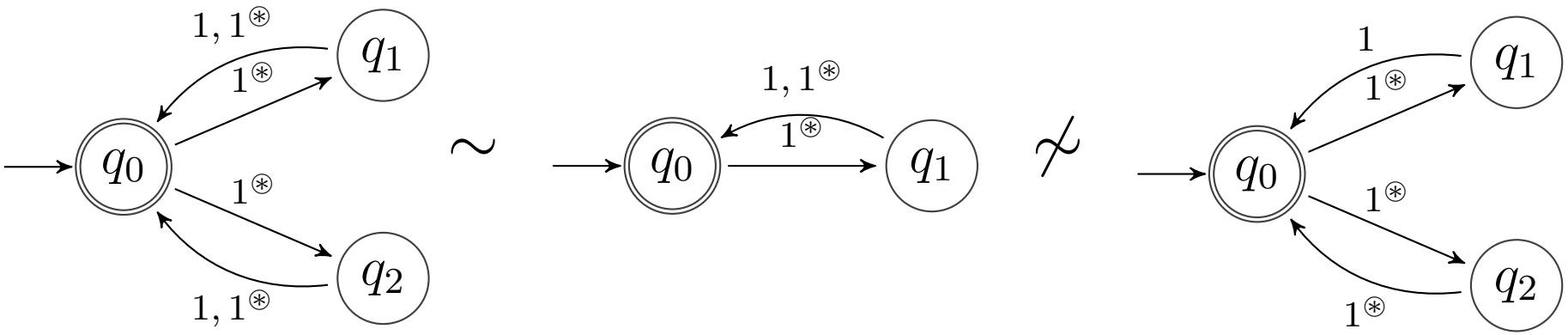
A more *behavioural* notion of equivalence says:

> Two automata are equivalent if they can simulate the operation of one another in a name-by-name manner

For instance (assuming empty initial registers):



but also:



Bisimulation formally

Let $\mathcal{A}$ be an r-FRA. A relation $R \subseteq Conf^f_{\mathcal{A}} \times Conf^f_{\mathcal{A}}$ is called a **bisimulation** if, whenever $(\kappa_1, \kappa_2) \in R$ (written $\kappa_1 \, R \, \kappa_2$):

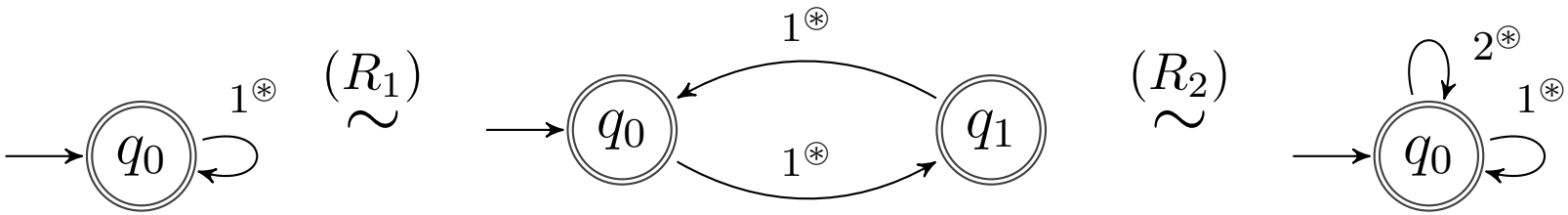
- for all $\kappa_1 \xrightarrow{d} \kappa_1'$ there is some $\kappa_2 \xrightarrow{d} \kappa_2'$ such that $\kappa_1' \, R \, \kappa_2'$,
- for all $\kappa_2 \xrightarrow{d} \kappa_2'$ there is some $\kappa_1 \xrightarrow{d} \kappa_1'$ such that $\kappa_1' \, R \, \kappa_2'$,
- if $\kappa_i = (q_i, \rho_i, H_i)$, for $i = 1, 2$, then $q_1 \in F \iff q_2 \in F$.

Moreover:

- If $R_1 \cup R_2$ are bisimulations then so is $R_1 \cup R_2$. We take $\sim$ to be the union of all bisimulations, called **bisimilarity**.
  I.e. $\kappa_1 \sim \kappa_2$ if $\kappa_1 \, R \, \kappa_2$ for some bisimulation R.
- Bisimilarity can be defined directly on pairs of states of a configuration graph, assuming designated final configurations.

  For FRAs $\mathcal{A}_1$ and $\mathcal{A}_2$ we let $\mathcal{A}_1 \sim \mathcal{A}_2$ if their initial configurations are bisimilar in the union configuration graph of $\mathcal{A}_1$ and $\mathcal{A}_2$.

Examples revisited



- $R_1 = \{((q_0, \rho, H), (q_0, \rho, H)) \mid \nu(\rho) \subseteq H\}$
  $\cup \{((q_0, \rho, H), (q_1, \rho, H)) \mid \nu(\rho) \subseteq H\}$
- $R_2 = \{((q_0, \rho, H), (q_0, \rho', H)) \mid \nu(\rho) \cup \nu(\rho') \subseteq H\}$
  $\cup \{((q_1, \rho, H), (q_0, \rho', H)) \mid \nu(\rho) \cup \nu(\rho') \subseteq H\}$
- $R_1; R_2$ witnesses bisimilarity of first and last automaton

Bisimilarity vs language equivalence

Theorem. *If* $\mathcal{A}_1 \sim \mathcal{A}_2$ *then* $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$.

Proof idea. Given an FRA-configuration graph $\mathcal{G}$ and a configuration κ, let $\mathcal{L}(\kappa)$ be the language of all paths from κ to some final configuration:

$$\mathcal{L}(\kappa) = \{w \in \mathcal{D}^* \mid \text{there is a } w\text{-labelled path in } \mathcal{G} \text{ from } \kappa \text{ to a final } \kappa_F\}$$

It suffices then to show that $\kappa \sim \kappa'$ implies $\mathcal{L}(\kappa) = \mathcal{L}(\kappa')$. □

The converse does not hold in general, unless the FRAs are *deterministic*.

Application: equivalence with M-automata

We can define M-type automata similarly to RA(M)'s by simply extending transition labels to:

$$Op_r = (\mathcal{P}([r]) \cup \{\circledast\}) \times \mathcal{P}([r])$$

e.g. in $q \xrightarrow{\circledast, Y} q'$ we *read a globally fresh name and store it in registers* Y.

Theorem. *For any* r*-FRA(M)* $\mathcal{A}$ *there is an* $(r+1)$*-FRA* $\mathcal{A}'$ *such that* $\mathcal{A} \sim \mathcal{A}'$.

Proof. [Exercise] □

Main result: bisimilarity is decidable

As FRAs extend RAs, language equivalence for FRAs in undecidable. However, we are going to show the following.

Theorem. *The following problem, called* BISIMILARITY, *is decidable.*

INPUT: *An r-FRA $\mathcal{A}$ and configurations κ_1, κ_2 with common history.*
QUESTION: *Is it the case that $\kappa_1 \sim \kappa_2$?*

Some notes:

- Important for applicability of FRAs as a modelling paradigm.
- The restriction to common history is not essential (but easier).
- Strictly speaking, the input $(\mathcal{A}, \kappa_1, \kappa_2)$ is not finite as κ_1, κ_2 may contain names, which come from an infinite domain.
  However, in this case it suffices to think of the domain as being just the common history H, which is finite.

Proof idea: symbolic reasoning

Consider a pair (κ_1, κ_2) of configurations with $\kappa_i = (q_i, \rho_i, H)$.

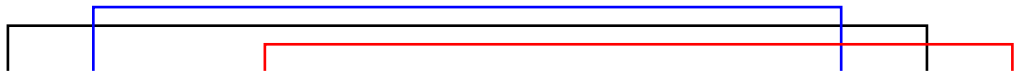
To check whether $\kappa_1 \sim \kappa_2$ are bisimilar, we do not need to know ρ_1, ρ_2, H in full detail. Rather, it suffices to know:

- what are the common names of ρ_1, ρ_2 (and in which positions),
- what are the private names of ρ_1, ρ_2 (and in which positions),
- what is the size of H with respect to the names in $\nu(\rho_1) \cup \nu(\rho_2)$.

Given the above, we can then reason *symbolically*, by looking directly at $\mathcal{A}$ (which is finite) rather than its configuration graph (that is infinite).

Symbolic reasoning

Consider the following situation:



$$\kappa_1 = (q_1, [d_1, d_2, d_3, d_4, d_5], H) \qquad \kappa_2 = (q_2, [d_3, d_2, d_5, \#, \#], H)$$

where $|H| = 8$.

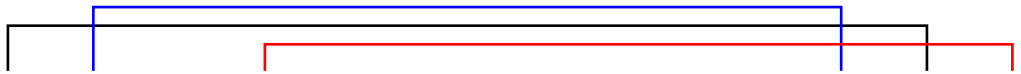
In order for $\kappa_1 \sim \kappa_2$ to hold:

- ☐ if $q_1 \xrightarrow{3} q_1'$ then there must be $q_2 \xrightarrow{1} q_2'$,
- ☐ if $q_1 \xrightarrow{1} q_1'$ then there must be $q_2 \xrightarrow{j^\bullet} q_2'$,
- ☐ if $q_2 \xrightarrow{1^\bullet} q_2'$ then there must be $q_1 \xrightarrow{i^\bullet} q_1'$,
- ☐ if $q_2 \xrightarrow{1^\bullet} q_2'$ then there must be $q_1 \xrightarrow{1} q_1'$ and $q_1 \xrightarrow{4} q_1''$,
- ☐ if $q_2 \xrightarrow{1^\circledast} q_2'$ then there must be $q_1 \xrightarrow{i^\circledast} q_1'$ or $q_1 \xrightarrow{i^\bullet} q_1'$.

In the last case we use the fact that local freshness is more general than global freshness (i.e. it can accept more names).

Global freshness could be as general as local

Consider a the following situation:



$$\kappa_1 = (q_1, [d_1, d_2, d_3, d_4, d_5], H) \qquad \kappa_2 = (q_2, [d_3, d_2, d_5, \#, \#], H)$$

where $|H| = 5$.

In order for $\kappa_1 \sim \kappa_2$ to hold:

- ☐ ...
- ☐ if $q_2 \xrightarrow{1^\bullet} q_2'$ then there must be $q_1 \xrightarrow{1} q_1'$ and $q_1 \xrightarrow{4} q_1''$,
- ☐ if $q_2 \xrightarrow{1^\bullet} q_2'$ then there must be $q_1 \xrightarrow{i^\bullet} q_1'$ or $q_1 \xrightarrow{i^\circledast} q_1'$

  This is because $q_2 \xrightarrow{1^\bullet} q_2'$ can accept:

  - d_1 and d_4 (taken care of by previous case)
  - any name in $H \setminus \{d_1, \cdots, d_5\}$ (empty!)
  - any name not in H (taken care by either of $q_1 \xrightarrow{i^\bullet / i^\circledast} q_1'$)

Symbolic configurations

We shall represent each pair:

$$(q_1, [d_1, d_2, d_3, d_4, d_5], H) \qquad (q_2, [d_3, d_2, d_5, \#, \#], H) \qquad |H| = 5$$

with: $(q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\}, 5).$

Thus, we define *symbolic configuration pairs* as:

$$\mathcal{U} = \{\, (q_1, S_1, \sigma, q_2, S_2, h) \in Q \times \mathcal{P}([r]) \times ([r] \overset{\cong}{\rightharpoonup} [r]) \times Q \times \mathcal{P}([r]) \times \mathbb{N}$$

$$\mid \mathsf{dom}(\sigma) \subseteq S_1 \wedge \mathsf{cod}(\sigma) \subseteq S_2 \wedge 0 \leq h \leq 2r+1 \,\}$$

Notes:

- Symbolic configuration pairs describe pairs of concrete configurations.
- The components (S_1, σ, S_2) describe how the two register assignments are related.
- We only need to count the size of H up to $2r+1$.

Symbolic representation of configuration pairs

We shall represent each pair:

$$(q_1, [d_1, d_2, d_3, d_4, d_5], H) \qquad (q_2, [d_3, d_2, d_5, \#, \#], H) \qquad |H| = 5$$

with: $(q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\}, 5).$

Thus, we define *symbolic configuration pairs* as:

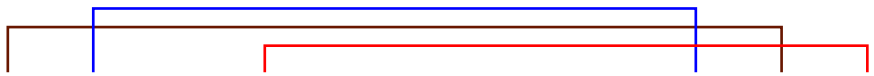
$$\mathcal{U} = \{\, (q_1, S_1, \sigma, q_2, S_2, h) \in Q \times \mathcal{P}([r]) \times ([r] \xrightarrow{\cong} [r]) \times Q \times \mathcal{P}([r]) \times \mathbb{N} \\ \mid \mathsf{dom}(\sigma) \subseteq S_1 \wedge \mathsf{cod}(\sigma) \subseteq S_2 \wedge 0 \leq h \leq 2r+1 \,\}$$

Given a pair of configurations $\kappa_i = (q_i, \rho_i, H)$, $i = 1, 2$, we define its *symbolic representation*, as:

$$\mathsf{symb}(\kappa_1, \kappa_2) = (q_1, \mathsf{dom}(\rho_1), \rho_1; \rho_2^{-1}, q_2, \mathsf{dom}(\rho_2), |H|_{2r+1}).$$

Here, $\rho_1; \rho_2^{-1} = \{(i,j) \mid \rho_1(i) = \rho_2(j)\}$ and $|H|_{2r+1} = \begin{cases} |H| & |H| < 2r+1 \\ 2r+1 & \text{otherwise} \end{cases}$

Definition of symbolic bisimulation



$$(q_1, [d_1, d_2, d_3, d_4, d_5], H) \qquad (q_2, [d_3, d_2, d_5, \#, \#], H) \qquad |H| = 5$$

$$\mapsto (q_1, S_1, \sigma, q_2, S_2, h) = (q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\}, 5)$$

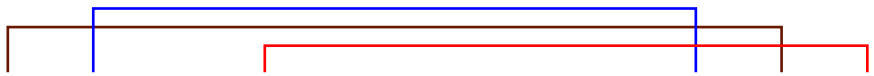
Given r-FRA $\mathcal{A}$, a relation $R \subseteq \mathcal{U}$ is called a **symbolic bisimulation** if, whenever $(q_1, S_1, \sigma, q_2, S_2, h) \in R$, $q_1 \in F \iff q_2 \in F$ and:

- $\forall\ q_1 \xrightarrow{i} q_1'$ with $i \in \mathsf{dom}(\sigma)$, $\exists\ q_2 \xrightarrow{\sigma(i)} q_2'$ with $(q_1', S_1, \sigma, q_2', S_2, h) \in R$,
- $\forall\ q_1 \xrightarrow{i} q_1'$ with $i \in S_1 \setminus \mathsf{dom}(\sigma)$, $\exists\ q_2 \xrightarrow{j^\bullet} q_2'$ with $(q_1', S_1, \sigma[i \mapsto j], q_2', S_2 \cup \{j\}, h) \in R$,
- $\forall\ q_1 \xrightarrow{i^\circledast} q_1'$, $\exists\ q_2 \xrightarrow{j^\bullet / j^\circledast} q_2'$ with $(q_1', S_1 \cup \{i\}, \sigma[i \mapsto j], q_2', S_2 \cup \{j\}, h \oplus 1) \in R$,

where: $\sigma[i \mapsto j] = \{(k, \sigma(k)) \mid k \neq i \wedge \sigma(k) \neq j\} \cup \{(i,j)\}$

$h \oplus 1 = h + 1$ if $h \leq 2r$, and $(2r+1) \oplus 1 = 2r + 1$

Definition of symbolic bisimulation (ctd)



$$(q_1, [d_1, d_2, d_3, d_4, d_5], H) \qquad (q_2, [d_3, d_2, d_5, \#, \#], H) \qquad |H| = 5$$

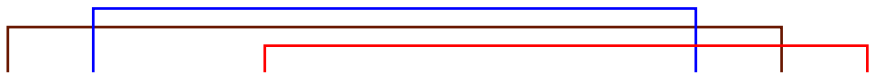
$$\mapsto (q_1, S_1, \sigma, q_2, S_2, h) = (q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\}, 5)$$

Given r-FRA $\mathcal{A}$, a relation $R \subseteq \mathcal{U}$ is called a **symbolic bisimulation** if, whenever $(q_1, S_1, \sigma, q_2, S_2, h) \in R$, $q_1 \in F \iff q_2 \in F$ and:

- for all $q_1 \xrightarrow{i^\bullet} q_1'$:
  - $\forall\ j \in S_2 \setminus \mathsf{cod}(\sigma)$, $\exists\ q_2 \xrightarrow{j} q_2'$ with $(q_1', S_1 \cup \{i\}, \sigma[i \mapsto j], q_2', S_2, h) \in R$,
  - if $\hat{h} < h$, $\exists\ q_2 \xrightarrow{j^\bullet} q_2'$ with $(q_1', S_1 \cup \{i\}, \sigma[i \mapsto j], q_2', S_2 \cup \{j\}, h) \in R$,
  - $\exists\ q_2 \xrightarrow{j^\bullet/j^\circledast} q_2'$ with $(q_1', S_1 \cup \{i\}, \sigma[i \mapsto j], q_2', S_2 \cup \{j\}, h \oplus 1) \in R$; (because $i^\bullet$ can capture some globally fresh name)

where $\hat{h} = |S_1| + |S_2| - |\mathsf{dom}(\sigma)|$ is the number of all names in the two simulated assignments (removing repetitions)

Definition of symbolic bisimulation (ctd ctd)



$$(q_1, [d_1, d_2, d_3, d_4, d_5], H) \qquad (q_2, [d_3, d_2, d_5, \#, \#], H) \qquad |H| = 5$$

$$\mapsto (q_1, S_1, \sigma, q_2, S_2, h) = (q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\}, 5)$$

Given r-FRA $\mathcal{A}$, a relation $R \subseteq \mathcal{U}$ is called a **symbolic bisimulation** if, whenever $(q_1, S_1, \sigma, q_2, S_2, h) \in R$, $q_1 \in F \iff q_2 \in F$ and:

- the symmetric conditions apply for all $q_2 \xrightarrow{\ell} q_2'$.

Define **symbolic bisimilarity**, $\sim_s \subseteq \mathcal{U}$, as the union of all symbolic bisimulations.

Given κ_1, κ_2 with $\kappa_i = (q_i, \rho_i, H)$, we write $\kappa_1 \sim_s \kappa_2$ if $\mathsf{symb}(\kappa_1, \kappa_2) \in \sim_s$, i.e. if there is symbolic bisimulation R such that $\mathsf{symb}(\kappa_1, \kappa_2) \in R$.

Recall: $\mathsf{symb}(\kappa_1, \kappa_2) = (q_1, \mathsf{dom}(\rho_1), \rho_1; \rho_2^{-1}, q_2, \mathsf{dom}(\rho_2), |H|_{2r+1})$.

Bisimilarity is decidable

We can show the following.

Theorem. *For any pair of configurations κ_1, κ_2 with common history, $\kappa_1 \sim \kappa_2$ iff $\kappa_1 \sim_{\mathsf{s}} \kappa_2$.*

and therefore:

Corollary. Bisimilarity *is decidable.*

Proof. Given $(\mathcal{A}, \kappa_1, \kappa_2)$, it suffices to check whether $\kappa_1 \sim_{\mathsf{s}} \kappa_2$, i.e. whether there is symbolic bisimulation $R \subseteq \mathcal{U}$ such that $\mathsf{symb}(\kappa_1, \kappa_2) \in R$.

But $\mathcal{P}(\mathcal{U})$ is bounded, so we can exhaustively search in it for a relation R satisfying the required conditions. □

Correspondence

The proof of the Theorem relies on two correspondences.

Lemma. *Suppose R is a bisimulation. Then, the relation*

$$R' = \{\mathsf{symb}((q_1, \rho_1, H), (q_2, \rho_2, H)) \mid ((q_1, \rho_1, H), (q_2, \rho_2, H)) \in R\}$$

is a symbolic bisimulation.

Lemma. *Suppose R is a symbolic bisimulation. Then, the relation*

$$R' = \{((q_1, \rho_1, H), (q_2, \rho_2, H)) \mid \mathsf{symb}((q_1, \rho_1, H), (q_2, \rho_2, H)) \in R\}$$

is a bisimulation.

Complexity

Given r-FRA $\mathcal{A} = (Q, q_0, \rho_0, \delta, F)$, symbolic bisimulations for $\mathcal{A}$ live in:

$$2^{Q \times \mathcal{P}([r]) \times ([r] \overset{\cong}{\backsimeq} [r]) \times Q \times \mathcal{P}([r]) \times \{0,\ldots,2r+1\}}$$

a space exponential in size in $|Q|$ and doubly exponential in r, so our (naive) solution is in 2-EXPTIME.

This is the same even if $\mathcal{A}$ is just an RA. In such a case, we can drop the symbolic history component and let symbolic bisimulations be sets with elements from the set:

$$\mathcal{U} \;=\; Q \times \mathcal{P}([r]) \times ([r] \overset{\cong}{\backsimeq} [r]) \times Q \times \mathcal{P}([r])$$

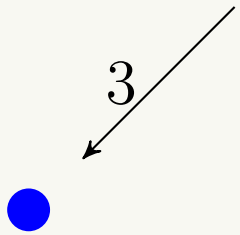
such that $(q_1, S_1, \sigma, q_2, S_2) \in \mathcal{U}$ implies $\sigma \subseteq S_1 \times S_2$.

Symbolic bisimulation game

Things can get a bit better, and then a lot better.

The symbolic bisimulation conditions can be interpreted as a game between Attacker and Defender:

$$(q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\})$$

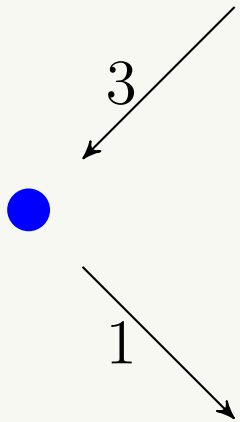


Symbolic bisimulation game

Things can get a bit better, and then a lot better.

The symbolic bisimulation conditions can be interpreted as a game between Attacker and Defender:

$$(q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\})$$



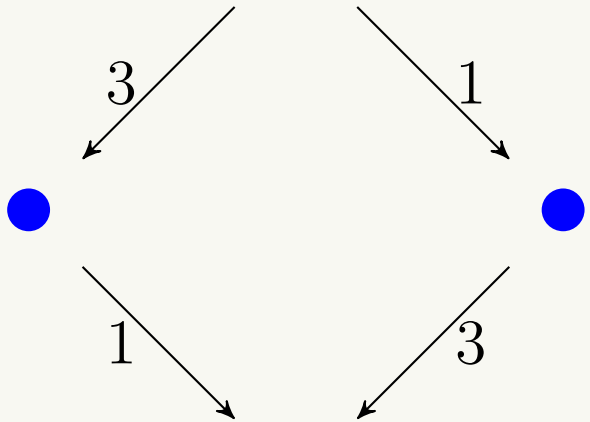
$$(q_1', \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2', \{1,2,3\})$$

Symbolic bisimulation game

Things can get a bit better, and then a lot better.

The symbolic bisimulation conditions can be interpreted as a game between Attacker and Defender:

$$(q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\})$$



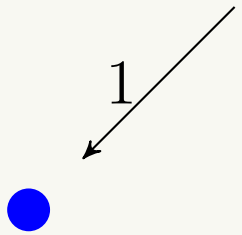
$$(q_1', \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2', \{1,2,3\})$$

Symbolic bisimulation game

Things can get a bit better, and then a lot better.

The symbolic bisimulation conditions can be interpreted as a game between Attacker and Defender:

$$(q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\})$$

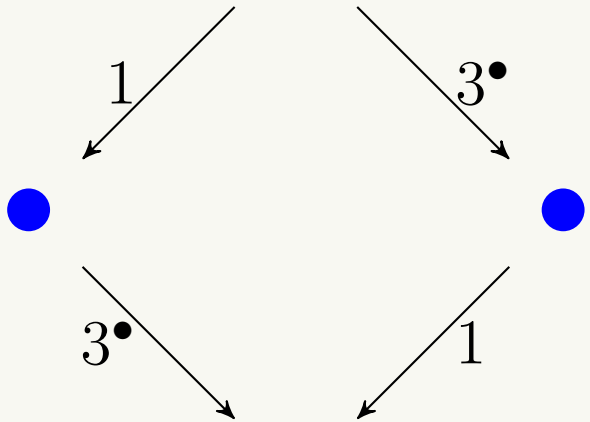


Symbolic bisimulation game

Things can get a bit better, and then a lot better.

The symbolic bisimulation conditions can be interpreted as a game between Attacker and Defender:

$$(q_1, \{1,2,3,4,5\}, \{(2,2),(3,1),(5,3)\}, q_2, \{1,2,3\})$$

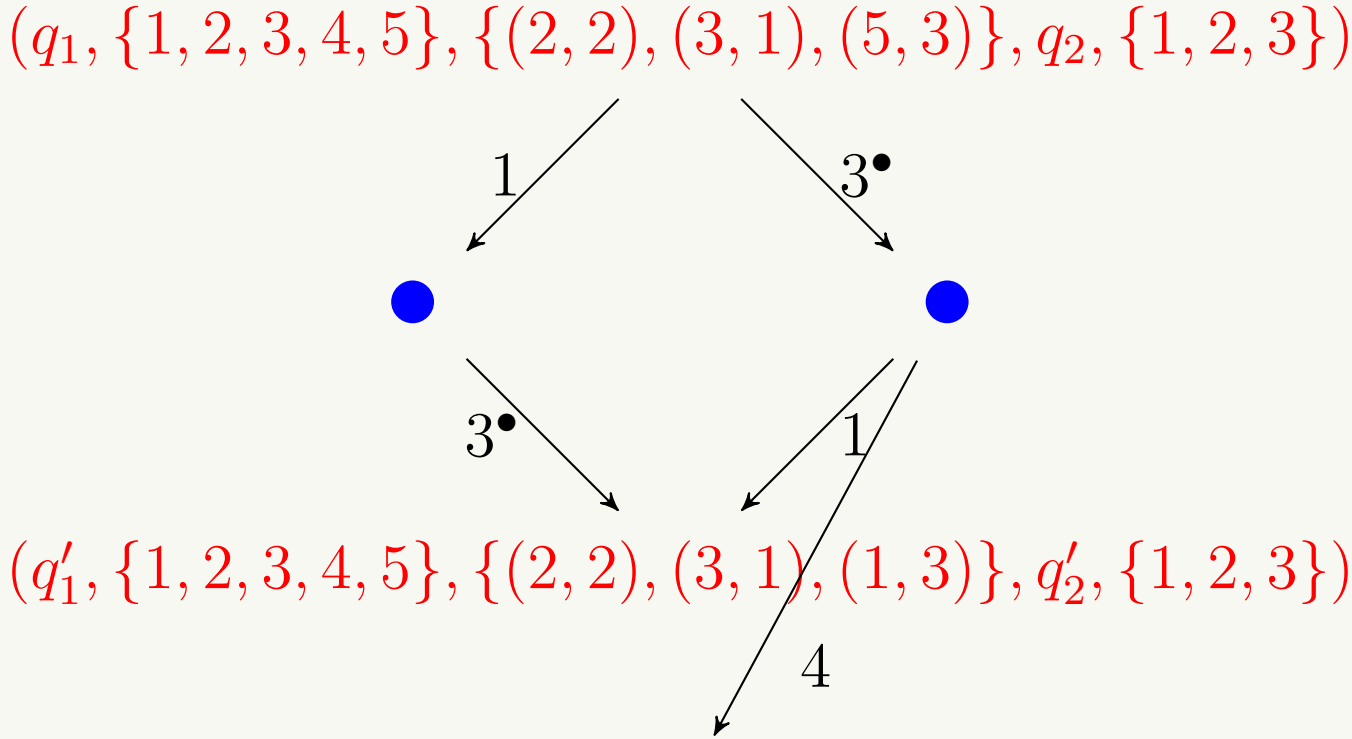


$$(q_1', \{1,2,3,4,5\}, \{(2,2),(3,1),(1,3)\}, q_2', \{1,2,3\})$$

Symbolic bisimulation game

Things can get a bit better, and then a lot better.

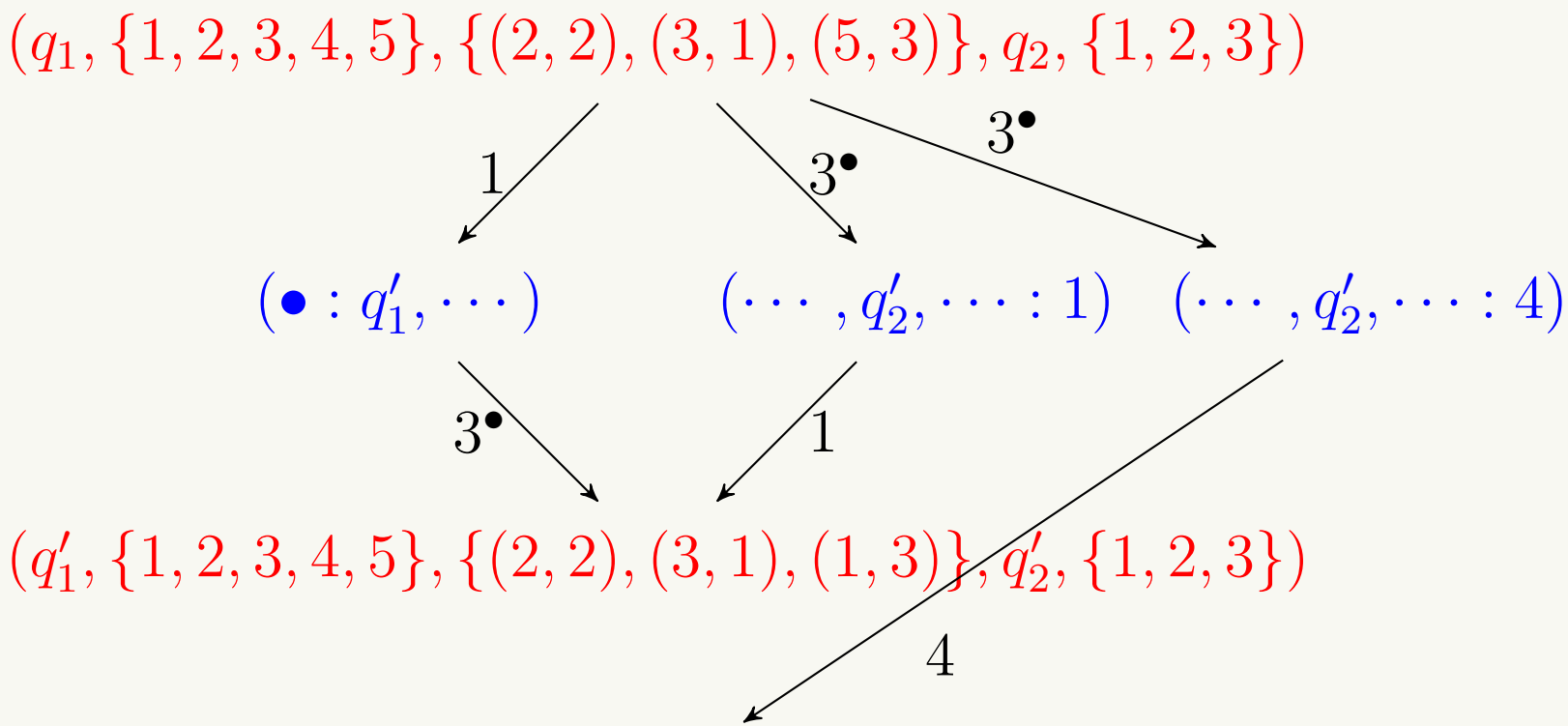
The symbolic bisimulation conditions can be interpreted as a game between Attacker and Defender:



Symbolic bisimulation game

Things can get a bit better, and then a lot better.

The symbolic bisimulation conditions can be interpreted as a game between Attacker and Defender:



Symbolic bisimulation game

Things can get a bit better, and then a lot better.

The symbolic bisimulation conditions can be interpreted as a game between Attacker and Defender. Starting from u_0 (is $u_0 \in \sim_{\mathsf{s}}$?):

- Attacker plays red nodes $u \in \mathcal{U}$
- Defender plays blue nodes $(\ell : u), (u : \ell)$ where $u \in \mathcal{U}$ and $\ell \in Op_r$.
- Attacker wins if Defender gets stuck, or if game reaches some u that mixes final/non-final states.

Observation If Attacker has a winning strategy then they have one in which red nodes are not repeated in any path.

So, if Attacker wins then they can win in exponentially many steps, and therefore the game can be decided in alternating polynomial space.

This gives $\mathrm{EXPTIME}$. But we can do even better.

Indexed bisimilarity

We define approximations of $\sim_{\mathsf{s}}$:

- $\sim_0 = \{(q_1, S_1, \sigma, q_2, S_2) \in \mathcal{U} \mid q_1 \in F \iff q_2 \in F\}$,
- $u \in \sim_{i+1}$ if $u \in \sim_0$ and Defender brings the game from u into $\sim_i$ in one round.

Lemma. *For each* $i \in \mathbb{N}$, $\sim_{i+1} \subseteq \sim_i$. *Moreover,* $\sim_{\mathsf{s}} = \bigcap_{i \in \mathbb{N}} \sim_i$.

Observations

- There is a bound $B \in \mathbb{N}$:

$$\sim_0 \supseteq \sim_1 \supseteq \cdots \supseteq \sim_B = \sim_{\mathsf{s}}$$

- If Attacker has a winning strategy then they have one winning in at most B rounds.
- We can show B is of polynomial size (wrt the size of the RA $\mathcal{A}$).

Idea: group-based representations $\sim_i$'s

Each $\sim_i$ is closed under the following rules for $R \subseteq \mathcal{U}$:

$$\frac{}{(q, S, \mathsf{id}_S, q, S) \in R}\ (\textsc{Id}) \qquad \frac{(q_1, S_1, \sigma, q_2, S_2) \in R}{(q_2, S_2, \sigma^{-1}, q_1, S_1) \in R}\ (\textsc{Sym})$$

$$\frac{(q_1, S_1, \sigma_1, q_2, S_2) \in R \qquad (q_2, S_2, \sigma_2, q_3, S_3) \in R'}{(q_1, S_1, \sigma_1; \sigma_2, q_3, S_3) \in R}\ (\textsc{Tr})$$

$$\frac{(q_1, S_1, \sigma, q_2, S_2) \in R \qquad \sigma \leq_{S_1, S_2} \sigma'}{(q_1, S_1, \sigma', q_2, S_2) \in R}\ (\textsc{Ext})$$

- For each state q and set $S \subseteq [r]$, there is a *characteristic set* $X_{(q,S)}$:
  - $(q, S, \mathsf{id}_{S'}, q, S) \in \sim_i \iff X_{(q,S)} \subseteq S' \subseteq S$
  - $G_{(q,S)} = \{\sigma \subseteq (X_{(q,S)} \times X_{(q,S)}) \mid (q, S, \sigma, q, S) \in \sim_i\}$ is a group.

  So, $\sim_i$ restricted to $((q, S), (q, S))$ is represented by $(X_{(q,S)}, G_{(q,S)})$.
- $\sim_i$ restricted to $((q_1, S_1), (q_2, S_2))$ is repr. by an $(q_1, S_1, \sigma, q_2, S_2) \in \sim_i$.

Polynomial game bound

Coming back to the bound $B \in \mathbb{N}$:

$$\sim_0 \supseteq \sim_1 \supseteq \cdots \supseteq \sim_B = \sim_s$$

Each inclusion is strict, so one of the following has to give in each case:

- some characteristic set becomes larger,
- some characteristic group becomes smaller,
- some connection $((q_1, S_1), (q_2, S_2))$ is lost.

We can show that $B \in O(r|Q|^2 + r^3|Q|)$, and therefore that the bisimulation game can be won/lost in polynomial time.

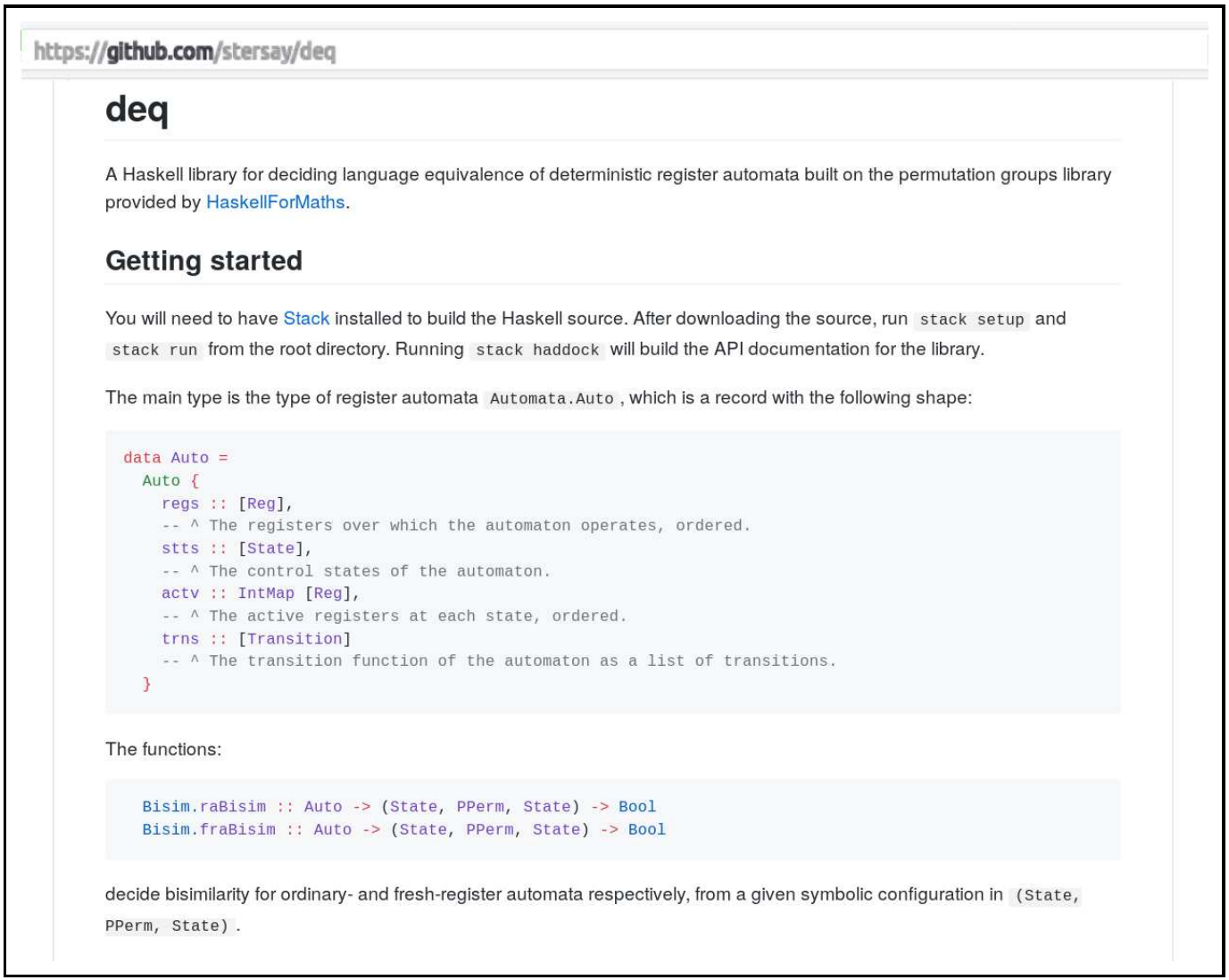
The whole argument can be adapted to FRAs. In fact:

Theorem. BISIMILARITY *is* PSPACE*-complete for RAs and FRAs.*

The deterministic case (MFCS'18)

For *deterministic* FRAs the upper bound can be brought down to NP.

If, in addition, the set of full registers of the FRA is *fixed per state*, then $\textsc{Bisimilarity}$ is in P.



https://github.com/stersay/deq

deq

A Haskell library for deciding language equivalence of deterministic register automata built on the permutation groups library provided by HaskellForMaths.

Getting started

You will need to have Stack installed to build the Haskell source. After downloading the source, run `stack setup` and `stack run` from the root directory. Running `stack haddock` will build the API documentation for the library.

The main type is the type of register automata `Automata.Auto`, which is a record with the following shape:

```
data Auto =
  Auto {
    regs :: [Reg],
    -- ^ The registers over which the automaton operates, ordered.
    stts :: [State],
    -- ^ The control states of the automaton.
    actv :: IntMap [Reg],
    -- ^ The active registers at each state, ordered.
    trns :: [Transition]
    -- ^ The transition function of the automaton as a list of transitions.
  }
```

The functions:

```
  Bisim.raBisim :: Auto -> (State, PPerm, State) -> Bool
  Bisim.fraBisim :: Auto -> (State, PPerm, State) -> Bool
```

decide bisimilarity for ordinary- and fresh-register automata respectively, from a given symbolic configuration in `(State, PPerm, State)`.

Plan

1. Register automata
   - ☐ Motivation for automata over infinite alphabets
   - ☐ Definitions and examples
   - ☐ Closure properties, emptiness and equivalence
   - ☐ Application to Java runtime verification
2. Fresh-register automata
   - ☐ Adding freshness
   - ☐ Basic properties
   - ☐ Bisimulation equivalence
3. Applications, other formalisms and extensions
   - ☐ Pushdown FRAs and game-semantics model-checking
   - ☐ Session automata
   - ☐ Nominal automata
   - ☐ Class memory automata

Pushdown RAs (PDRAs)

The extension of r-RAs with a pushdown stack adds transition labels:

- push(i), for $i \in [r]$,
- pop(i), for $i \in [r]$, and pop$(\bullet)$;

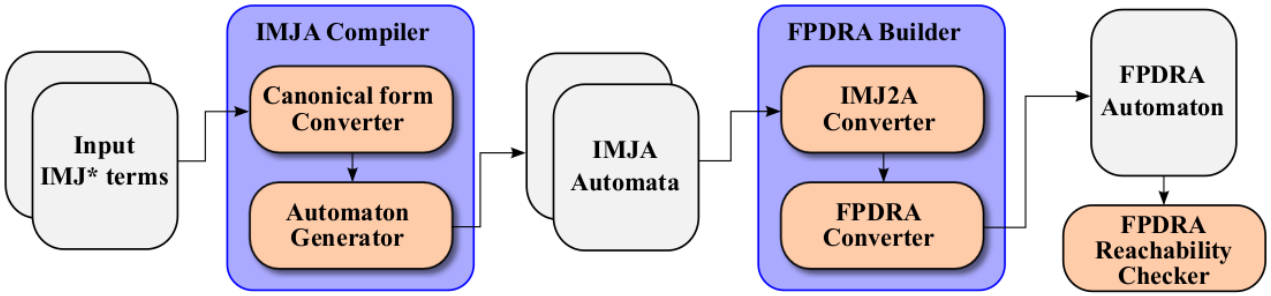
with the expected semantics: push/pop the name in the ith register, or pop some locally fresh symbol.
Results:

Theorem. *Let $\mathcal{A}$ be an r-PDRA. There exists a subset $\mathcal{D}_b$ of $\mathcal{D}$ with $3r$ elements such that if $w \in \mathcal{L}(\mathcal{A})$ then there exists $w' \in (\mathcal{D}_b)^*$ such that $w' \in \mathcal{L}(\mathcal{A})$ and $|w| = |w'|$.*

Theorem. *Non-emptiness for PDRAs is* $\mathrm{EXPTIME}$*-complete.*

Moreover, the result extends to fresh-PDRAs.

Program equivalence checker for IMJ (ATVA'15)



https://bitbucket.org/sjr/coneqct/wiki/Home

Search

Steven Ramsay / coneqct

Wiki

coneqct / Home

Coneqct: a contextual equivalence checking tool for Interface Middleweight Java

Home | Downloads | Syntax | Examples

Requirements

The checker runs on the .NET platform (>=4.5), and hence requires a recent implementation of the .NET Common Language Infrastructure (CLI) to be installed on your system.

- On Windows we recommend Microsoft's ".NET Framework", the latest stable version is 4.5.2: https://www.microsoft.com/en-us/download/details.aspx?id=42643.
- On Linux or Mac we recommend Xamarin's "Mono": http://www.mono-project.com/download/

Installation

1. Download the latest assemblies from downloads. All the required assemblies are packaged together in a zip file named "coneqct-XXX.zip" where "XXX" denotes the revis
2. Unzip to any convenient location, this creates a new directory "coneqct-XXX" in which resides the executable "coneqct.exe".
3. To verify that all is well, on the command line navigate to the directory "coneqct-XXX" and run the command:
   - ".\coneqct.exe" on Windows or,
   - "mono ./coneqct.exe" on Linux or Mac. If the installation is working correctly, the usage message will be printed out to the terminal.

Many more formalisms

What we presented is a very small selection of automata over infinite alphabets. More exciting work (see references at the end):

- **Session automata** Undecidability of RA-equivalence can be avoided by removing local freshness and keeping only the global one. This gives rise to a **robust** class of automata.
- **Nominal automata** We can abstract away from registers and assume that states and transitions are **nominal**: they each contain finitely names and can be acted upon by name permutations (formally, they come from a *nominal set*).
  In this setting, we recover RA-languages via automata whose sets of states have **finitely many orbits**.
- **Class memory automata** Instead of sets storing a single name, we can assume they store a finite set of names (a *class*). All names stored in the same classes are indistinguishable. This subsumes FRAs (history $\sim$ an all-storing class) and has VASS-like reachability behaviour.

References

On register automata, closure properties, reachability, equivalence, applications:

- M. Kaminski, N. Francez. Finite-memory automata, Theor.Comput.Sci. 134 (2) (1994).
- H. Sakamoto, D. Ikeda. Intractability of decision problems for finite-memory automata, Theor.Comput.Sci. 231(2) (2000).
- F. Neven, T. Schwentick, V. Vianu. Finite state machines for strings over infinite alphabets, ACM Trans.Comput.Log. 5(3) (2004).
- S. Demri, R. Lazić. LTL with the freeze quantifier and register automata, ACM Trans.Comput.Log. 10(3) (2009).
- T. Schwentick. Automata for XML - A survey. J.Comput.Syst.Sci., 73(3), 2007.
- R. Grigore, D. Distefano, R. L. Petersen, and N. Tzevelekos. Runtime verification based on register automata. In TACAS 2013.
- F. Aarts, P. Fiterau-Brostean, H. Kuppens, and F. W. Vaandrager. Learning register automata with fresh value generation. In ICTAC 2015.
- S. Cassel, F. Howar, B. Jonsson, and B. Steffen. Active learning for extended finite state machines. Formal Asp.Comput., 28(2), 2016.

Fresh-register automata, symbolic methods and bisimilarity:

- N. Tzevelekos. Fresh-register automata. In POPL 2011.
- A. S. Murawski, S. J. Ramsay, N. Tzevelekos. Bisimilarity in fresh-register automata. In LICS 2015.
- A. S. Murawski, S. J. Ramsay, N. Tzevelekos. Polynomial-time equivalence testing for deterministic fresh-register automata. In MFCS 2018.

Pushdown (fresh-)register automata, reachability and application to program equivalence checking:

- E.Y.C. Cheng, M. Kaminski. Context-free languages over infinite alphabets, Acta Inform. 35(3) (1998).
- L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In CSL 2006.
- A.S. Murawski, S.J. Ramsay, N. Tzevelekos. Reachability in Pushdown Register Automata. J.Comput.Syst.Sci., 87 (2017).
- A.S. Murawski, N. Tzevelekos. Algorithmic games for full ground references. In ICALP 2012.
- A.S. Murawski, S.J. Ramsay, N. Tzevelekos. A contextual equivalence checker for IMJ*. In ATVA 2015.

Session automata, nominal automata, class memory automata:

- B. Bollig, P. Habermehl, M. Leucker, B. Monmege. A Robust Class of Data Languages and an Application to Learning. LMCS 10(4) (2014).
- M. Bojańczyk, B. Klin, S. Lasota: Automata theory in nominal sets. Logical Methods in Computer Science 10(3) (2014)
- J. Moerman, M. Sammartino, A. Silva, B. Klin, and M. Szynwelski. Learning nominal automata. In POPL 2017.
- H. Björklund, T. Schwentick. On notions of regularity for data languages. Theor.Comput.Sci. 411(4-5) (2010).
- M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, L. Segoufin. Two-variable logic on data words. ACM Trans.Comput.Log. 12(4) (2011).
- R. Grigore, N. Tzevelekos. History-Register Automata. LMCS 12(1) (2016).

Exercises

1. Let an r-delRA be defined exactly as an r-RA but with additional transitions of the form $q \xrightarrow{\mathsf{del}(i)} q'$ $(i \in [r])$. The semantics of each such transition is given by evolutions of the form $(q, \rho) \xrightarrow{\epsilon} (q', \rho[i \mapsto \#])$. Show that for each r-delRA $\mathcal{A}$ there is an r-RA $\mathcal{A}'$ such that $\mathcal{A} \sim \mathcal{A}'$.
2. Let an r-permRA be defined exactly as an r-RA but with additional transitions $q \xrightarrow{\pi} q'$, for $\pi : [r] \xrightarrow{\cong} [r]$, with semantics: $(q, \rho) \xrightarrow{\epsilon} (q', \rho \circ \pi)$. Show that for each r-permRA $\mathcal{A}$ there is an r-RA $\mathcal{A}'$ such that $\mathcal{A} \sim \mathcal{A}'$.
3. Design an RA recognising the complement of $\mathcal{L}^2_{\text{fresh}}$.
4. Taking $\mathcal{L}_d = \{dw \in \mathcal{D}^* \mid w \in \mathcal{L}_{\text{fresh}} \wedge d \notin \nu(w)\}$ for some fixed $d \in \mathcal{D}$, show that $\mathcal{L}^*_d$ is not FRA-recognisable.
5. Show that bisimilarity is an equivalence relation.
6. Show that for any r-FRA(M) $\mathcal{A}$ there is an $(r{+}1)$-FRA $\mathcal{A}'$ with $\mathcal{A} \sim \mathcal{A}'$.
7. Show that symbolic bisimilarity is an equivalence relation.
8. Given an r-FRA $\mathcal{A}$ with initial assignment ρ_0 and a sequence of names $s = d_1 \cdots d_n$ disjoint from $\nu(\rho_0)$, construct an $(r{+}n)$-FRA $\mathcal{A}'$ with initial assignment $\rho_0 + s$ and such that $\mathcal{A} \sim \mathcal{A}'$.
9. Using the above construction, show that bisimilarity is decidable even in the case of configurations with different histories.