# Automata over Infinite Alphabets

## Register Automata

Andrzej Murawski
University of Warwick

Nikos Tzevelekos
Queen Mary University of London

http://warwick.ac.uk/amurawski/esslli15

ESSLLI 2015

# 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 ("finite-state")
- good closure properties
- decision procedures (emptiness, equivalence, etc)
- minimisation
- connections to logic

## DFA

$$\delta : Q \times \Sigma \to Q$$

# Why infinite alphabets?

- To model computational/mathematical/linguistic situations involving finite values that get so large/abstract that they are best viewed as infinite.
- To model computational/mathematical/linguistic situations involving infinite values.
- To preserve decidability, one needs to restrict the range of allowable operations. This will also allow us to prevent encodings. Therefore, in a typical scenario, only equality will be available.
- Computational significance: identifiers, references (pointers), objects, XML.

# ML

```
        Objective Caml version 3.12.0

# 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
```

# Pointers and references

C

```
int arr[ 10 ] ;

...   = arr[6];

... = *(arr+6);
```

Java

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

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

# XML

```
<menu>
  <menu-item>
    <portion unit="mL">250</portion>
    <name>Small soft drink</name>
  </menu-item>
  <menu-item>
    <portion unit="g">500</portion>
    <name>Sirloin steak</name>
  </menu-item>
</menu>
```

- XML documents support attributes: they are intended to be peripheral to the main communication and meant to help applications process the main communication (application-specific).
- XML queries can perform tests on attribute values. For formal analysis, it makes sense to regard them as elements of an infinite set.

# Abstract syntax with binding

$$\forall x.(P(x) \Rightarrow \forall y.Q(y) \wedge \exists z.S(z))$$

$$\lambda x.xy(\lambda z.z(\lambda v.vz))(\lambda v.v)$$

```
fun fact 0 = 1
  | fact n = n * fact (n - 1)
```

```
void copy(int i) { return (i+1); }
```

# 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
- bounded number of $\mathcal{D}$-valued registers

Let $r$ be the number of registers. Let us write $[r]$ for $\{1, \cdots, r\}$.

- By an $r$-register assignment we mean an *injective* map from $[r]$ to $\mathcal{D} + \{\#\}$.
- We write $Reg_r^{\mathrm{i}}$ for the set of all such assignments.

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

# Register automata

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

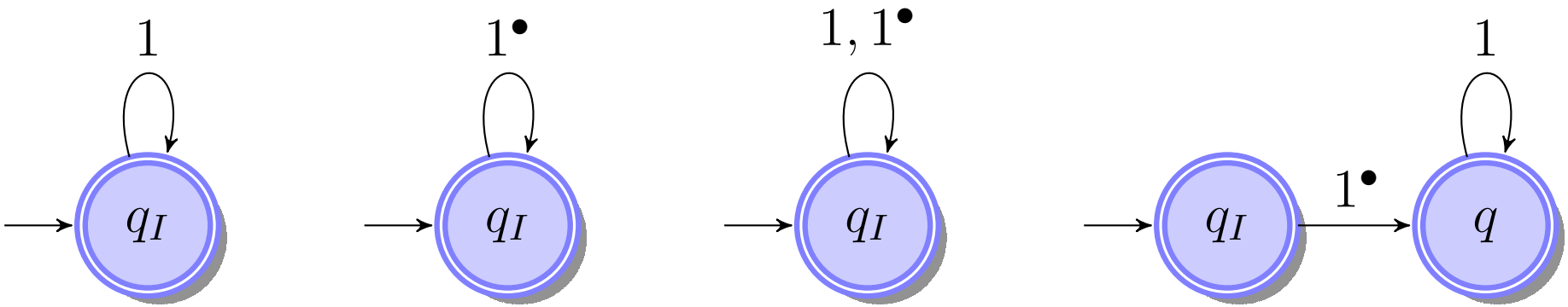
- $Q$ is a *finite* set of states,
- $q_I \in Q$ is the *initial state*,
- $\tau_I \in Reg_r^{\mathrm{i}}$ is the *initial register assignment*,
- $F \subseteq Q$ is the set of *final states*,
- $\delta \subseteq Q \times Op_r \times Q$ is the *transition function*, where $Op_r = \{1, \cdots, r\} \cup \{1^\bullet, \cdots, r^\bullet\}$.

## Intuition

- $i \in Op_r$ represents reading the letter in the $i$th 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 $i$.

# Examples

Consider the following 1-register automata $\langle Q_i, q_I, [\#], F_i, \delta_i \} \rangle$ $(i = 1, 2, 3, 4)$.




- $Q_1 = \{q_I\}$, $\delta_1 = \{(q_I, 1, q_I)\}$, $F = \{q_I\}$
- $Q_2 = \{q_I\}$, $\delta_2 = \{(q_I, 1^\bullet, q_I)\}$, $F = \{q_I\}$
- $Q_3 = \{q_I\}$, $\delta_3 = \{(q_I, 1, q_I), (q_I, 1^\bullet, q_I)\}$, $F = \{q_I\}$
- $Q_4 = \{q_I, q\}$, $\delta_4 = \{(q_I, 1^\bullet, q), (q, 1, q)\}$, $F = \{q_I, q\}$

# Configurations of $\mathcal{A} = \langle\, Q, q_I, \tau_I, \delta, F \,\rangle$

**Set of configurations**

$$Conf_r^{\mathrm{i}} = Q \times Reg_r^{\mathrm{i}}$$

**Initial configuration**

$$\kappa_I = (q_I, \tau_I) \in Conf_r^{\mathrm{i}}$$

**Evolution of configurations (successor configurations)**

$$(q_1, \tau_1) \xrightarrow{d} (q_2, \tau_2)$$

- $(q_1, i, q_2) \in \delta,\ \tau_1 = \tau_2,\ d = \tau_1(i)$
- $(q_1, i^{\bullet}, q_2) \in \delta,\ d \notin \tau_1([r]),\ d = \tau_2(i),\ \forall_{j \neq i}\ \tau_1(j) = \tau_2(j)$

# Runs and languages

A **run** of $\mathcal{A}$ is a sequence $\kappa_0, \cdots, \kappa_k$ of configurations such that

- $\kappa_0 = \kappa_I$,
- for all $0 \leq i < k$, $\kappa_i \xrightarrow{d_i} \kappa_{i+1}$ for some $d_i \in \mathcal{D}$.

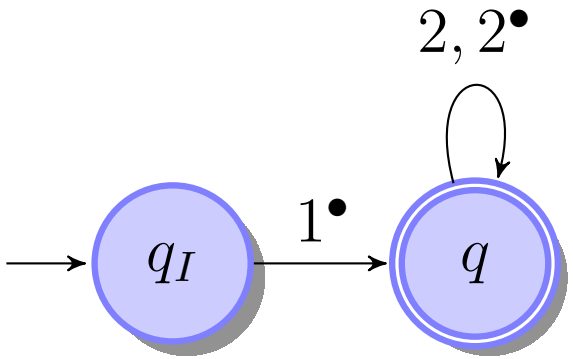
A run is **accepting** if $\kappa_k = (q_k, \tau_k)$ for some $q_k \in F$. In this case we say that $\mathcal{A}$ **accepts** $d_0 \cdots d_k \in \mathcal{D}^*$.

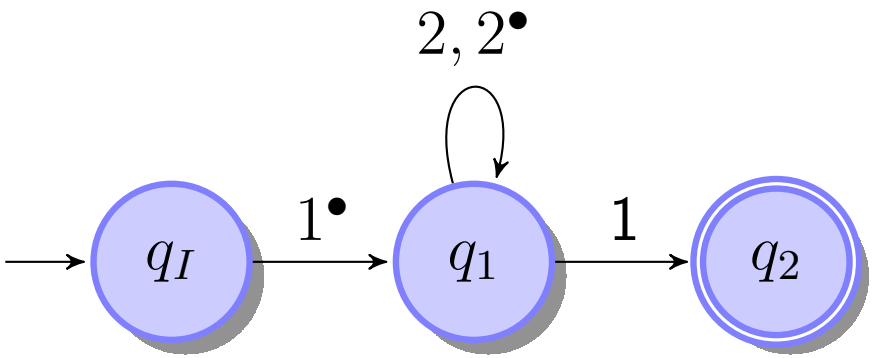
The set of all sequences $w \in \mathcal{D}^*$ accepted by $\mathcal{A}$ is called the language of $\mathcal{A}$ and denoted by $\mathcal{L}(\mathcal{A})$.

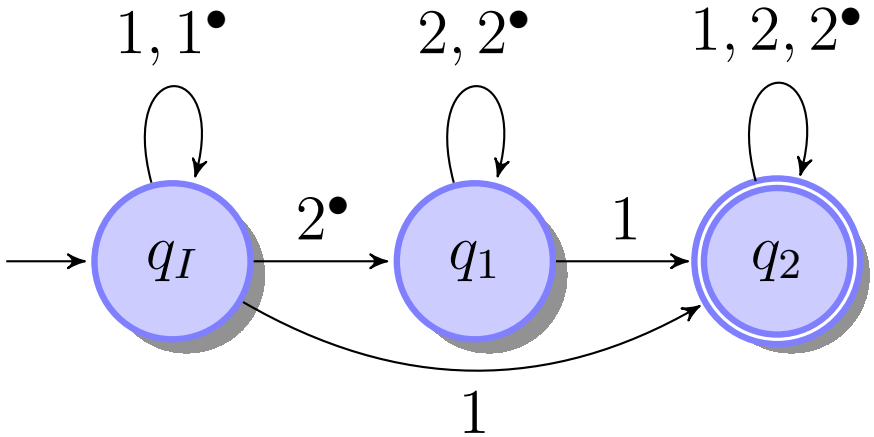
A language $L \subseteq \mathcal{D}^*$ is called an **RA-language** (or a **quasi-regular** language) if there exists an RA that accepts it.

# More RA's

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



# RA languages

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

- $\{\epsilon\}$
- $\{d_1 \cdots d_k \in \mathcal{D}^* \quad | \quad \forall_{1 \leq i < k}\, d_i \neq d_{i+1}\}$
- $\mathcal{D}^*$
- $\{d_1 \cdots d_k \in \mathcal{D}^* \quad | \quad \forall_{1 \leq i < k}\, d_i = d_{i+1}\}$
- $\{dd_1 \cdots d_k \quad | \quad \forall_{1 \leq i \leq k}\, d_i \neq d\}$
- $\{dd_1 \cdots d_k d \quad | \quad \forall_{1 \leq i \leq k}\, d_i \neq d\}$
- $\{d_1 \cdots d_k \quad | \quad \exists_{i \neq j}\, d_i = d_j\}$

# Name invariance

Note that the one-step successor relation is invariant with respect to permutations on names. Let $\sigma$ be a permutation of $\mathcal{D}$.

Abusing notation, we shall also apply $\sigma$ to various other objects containing elements of $\mathcal{D}$, e.g. $\sigma([d_1, d_2]) = [\sigma(d_1), \sigma(d_2)]$.

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

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

Consider a run $\kappa_I \xrightarrow{d_0} \kappa_1 \xrightarrow{d_1} \cdots \xrightarrow{d_k} \kappa_{k+1}$ and a permutation $\sigma$ of $\mathcal{D}$. If $\sigma(\kappa_I) = \kappa_I$ then the following is also a run of $\mathcal{A}$.

$$\kappa_I = \sigma(\kappa_I) \xrightarrow{\sigma(d_0)} \sigma(\kappa_1) \xrightarrow{\sigma(d_1)} \cdots \xrightarrow{\sigma(d_k)} \sigma(\kappa_{k+1})$$

# Invariance theorems

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

- Let $\tau_I = [\#, \cdots, \#]$ and $\sigma$ be a permutation of $\mathcal{D}$.

  If $d_1 \cdots d_k \in \mathcal{L}(\mathcal{A})$ then $\sigma(d_1) \cdots \sigma(d_k) \in \mathcal{L}(\mathcal{A})$.

- Let $\tau_I$ be arbitrary and $\sigma$ be a permutation of $\mathcal{D}$ such that $\sigma(\tau_I) = \tau_I$, i.e. for all $1 \leq i \leq r$, if $\tau_I(i) \in \mathcal{D}$ then $\sigma(\tau_I(i)) = \tau_I(i)$.

  If $d_1 \cdots d_k \in \mathcal{L}(\mathcal{A})$ then $\sigma(d_1) \cdots \sigma(d_k) \in \mathcal{L}(\mathcal{A})$.

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

# Motto

> An important facet of our theory is a certain indistinguishability view of the finite alphabet embedded in the modus operandi of the automaton. Languages are only unique up to automorphisms of the alphabet. Thus the actual letters occurring in the input are of no real significance. Only the initial and repetition patterns matter.
>
> Francez & Kaminski

New letters can be compared only with what the automaton has seen so far.

# Bounded alphabet property

Consider an accepting run

$$\kappa_I \xrightarrow{d_0} \kappa_1 \xrightarrow{d_1} \cdots \xrightarrow{d_k} \kappa_{k+1}.$$

Suppose $\mathcal{D}_I = \tau_I([r]) \cap \mathcal{D} = \{d_1, \cdots, d_h\}$. Then $h \leq r$. Pick $r + 1 - h$ different elements $d_{h+1}, \cdots, d_{r+1}$ of $\mathcal{D}$ disjoint from $\mathcal{D}_I$. Let $\mathcal{D}_b = \{d_1, \cdots, d_{r+1}\}$.

Observe that 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 finite subset $\mathcal{D}_b$ of $\mathcal{D}$ 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 $|w| = |w'|$.

If an $r$-RA $\mathcal{A}$ accepts a word then it accepts a word consisting of at most $r + 1$ letters.

**Corollary.** No register automaton will accept the language

$$\{d_1 \cdots d_k \in \mathcal{D}^* \mid k \in \mathbb{N}, \forall_{i \neq j}. d_i \neq d_j\}.$$

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

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

# Complementation

No register automaton will accept

$$L_1 = \{d_1 \cdots d_k \in \mathcal{D}^* \mid k \in \mathbb{N}, \qquad \forall_{i \neq j}.d_i \neq d_j\}.$$

But there exists a register automaton that accepts

$$L_2 = \{d_1 \cdots d_k \in \mathcal{D}^* \mid k \in \mathbb{N}, \qquad \exists_{i \neq j}.d_i = d_j\}.$$

Note that the languages complement each other:

$$L_1 = \mathcal{D}^* \setminus L_2 \qquad \text{and} \qquad L_2 = \mathcal{D}^* \setminus L_1.$$

**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.

To show this, we are going to rely on a more flexible notion of automata.

For a start, let us relax the injectivity requirement for register assignments and set

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

Accordingly, configurations will be defined by

$$Conf_r = Q \times Reg_r.$$

# $M$-automata

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

- $Q$ is a *finite* set of states,
- $q_I \in Q$ is the *initial state*,
- $\tau_I \in Reg_r$ is the *initial register assignment*,
- $\delta \subseteq Q \times \mathcal{P}([r]) \times \mathcal{P}([r]) \times Q$ is the *transition function*,
- $F \subseteq Q$ is the set of *final states*.

## Intuition

$(q_1, X, Y, q_2) \in \delta$ represents

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

# Runs of $M$-automata

**Evolution of configurations (successor configurations)**

$$(q_1, \tau_1) \xrightarrow{d} (q_2, \tau_2)$$

- $(q_1, X, Y, q_2) \in \delta$
- $X = \{i \mid \tau_1(i) = d\}$
- $\forall_{i \in Y}\ \tau_2(i) = d$
- $\forall_{i \notin Y}\ \tau_2(i) = \tau_1(i)$

**Previous model**

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

Definition of runs and accceptance analogous to RA.

# Equiexpressivity

**Theorem.** For any $r$-register $M$-automaton $\mathcal{A} = \langle Q, q_I, \tau_I, \delta, F \rangle$, there exists an $(r+1)$-register automaton $\mathcal{A}' = \langle Q', q'_I, \tau'_I, \delta', F' \rangle$ 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

In order to keep track of the original locations of names we introduce partitions.

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

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

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

We shall write $Part_r$ for the set of all partitions and take

$$Q' = Q \times Part_r$$

The initial state $q'_I$ will be $(q_I, \pi_I)$, where $\pi_I$ is defined next.

# Proof sketch (initial state, assignment)

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

$$i_1 \sim i_2 \iff \tau_I(i_1) = \tau_I(i_2).$$

- Let $x_1, \cdots, x_k$ be the respective representatives. We define $\pi_I \in Part_r$ by

$$\pi_I(x) = \begin{cases} X_i & x = x_i \\ \emptyset & \text{otherwise} \end{cases}$$

and let the initial assignment $\tau'_I$ be

$$\tau'_I(x) = \begin{cases} \tau_I(x) & x = x_i \\ \# & \text{otherwise.} \end{cases}$$

# Proof sketch (transitions)

Given

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

for any $i \in [r+1]$ and $\pi_1 \in Part_r$ such that $\pi_1(i) = X$,

add

$$(q_1, \pi_1) \xrightarrow{i} (q_2, \pi_2)$$

where

$$\pi_2(j) = \begin{cases} \pi_1(j) \setminus Y & j \neq i \\ \pi_1(i) \cup Y & j = i \end{cases}$$

# Proof sketch (transitions)

In addition, given

$$q_1 \xrightarrow{\emptyset, Y} q_2,$$

for any $\pi_1 \in Part_r$, take some $i \in [r+1]$ such that $\pi_1(i) = \emptyset$ ($i$ must exist because values of $\pi_1$ are disjoint), and

add

$$(q_1, \pi_1) \xrightarrow{i^\bullet} (q_2, \pi_2)$$

where

$$\pi_2(j) = \begin{cases} \pi_1(j) \setminus Y & j \neq i \\ \pi_1(i) \cup Y & i = j \end{cases}$$

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

# Union

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

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

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

$$\begin{aligned} Q &= \{q_I\} + Q^1 + Q^2 \\ \tau_I &= [\tau_I^1, \tau_I^2] \\ F &= \{q_I \mid q_I^1 \in F^1 \text{ or } q_I^2 \in F^2\} + F^1 + F^2 \end{aligned}$$

# Proof sketch (embed $\mathcal{A}^1$ into $\mathcal{A}$)

- Given $q \xrightarrow{X,Y} q'$ from $\delta^1$, add

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

to $\delta$ for any $Z_2 \subseteq \{r_1 + 1, \cdots, r_1 + r_2\}$.

- Given $q_I^1 \xrightarrow{X,Y} q'$ from $\delta^1$, add

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

to $\delta$ for any $Z_2 \subseteq \{r_1 + 1, \cdots, r_1 + r_2\}$.

# Proof sketch (embed $\mathcal{A}^2$ into $\mathcal{A}$)

**Notation.** We write $r + X$ for $\{r + x \,|\, x \in X\}$.

- Given $q \xrightarrow{X,Y} q'$ from $\delta^2$, add

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

to $\delta$ for any $Z_1 \subseteq \{1, \cdots, r_1\}$.

- Given $q_I^2 \xrightarrow{X,Y} q'$ from $\delta^2$, add

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

to $\delta$ for any $Z_1 \subseteq \{1, \cdots, r_1\}$.

# Intersection

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

$$\begin{aligned} Q &= Q^1 \times Q^2 \\ q_I &= (q_I^1, q_I^2) \\ \tau_I &= [\tau_I^1, \tau_I^2] \\ F &= F^1 \times F^2 \end{aligned}$$

# Proof sketch (synchronisation)

Given

$$q_1^1 \xrightarrow{X_1, Y_1} q_2^1 \in \delta^1$$

and

$$q_1^2 \xrightarrow{X_2, Y_2} q_2^2 \in \delta^2,$$

add

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

to $\delta$.

# Concatenation

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

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

**Idea.** Run $\mathcal{A}^1$ until (local) acceptance, then move on to $\mathcal{A}^2$ and again continue until acceptance.

$$\begin{aligned} Q &= Q^1 + Q^2 \\ q_I &= q_I^1 \\ \tau_I &= [\tau_I^1, \tau_I^2] \\ F &= F^2 \cup \{ f \in F^1 \mid q_I^2 \in F^2 \} \end{aligned}$$

# Proof sketch

- Given $q_1^1 \xrightarrow{X,Y} q_2^1 \in \delta^1$ add

$$q_1^1 \xrightarrow{X \cup Z_2, Y} q_2^1$$

to $\delta$ for any $Z_2 \subseteq \{r_1 + 1, \cdots, r_1 + r_2\}$.

- Given $q_1^2 \xrightarrow{X,Y} q_2^2 \in \delta^2$ add

$$q_1^2 \xrightarrow{Z_1 \cup (r_1+X), r_1+Y} q_2^2$$

to $\delta$ for any $Z_1 \subseteq \{1, \cdots, r_1\}$.

- Given $q_I^2 \xrightarrow{X,Y} q^2 \in \delta^2$, for any $f \in F^1$, add

$$f \xrightarrow{Z_1 \cup (r_1+X), r_1+Y} q^2$$

to $\delta$ for any $Z_1 \subseteq \{1, \cdots, r_1\}$.

# Kleene star

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

**Idea.** Use the extra $r$ registers to store $\tau_I$, which is needed to restart $\mathcal{A}$. After the restart, use the first $r$ registers to run the new iteration, keeping track of which registers have outdated content (filled with names from the previous iteration). Redirect accesses to outdated registers to the initial assignment (second set of $r$ registers).

$$\begin{aligned} Q' &= \{q'_I\} + (Q \times \mathcal{P}([r])) \\ q'_I &= q'_I \\ \tau'_I &= [\tau_I, \tau_I] \\ F &= \{q'_I\} + (F \times \mathcal{P}([r])) \end{aligned}$$

# Proof sketch

$Z \subseteq [r]$ corresponds to the indices of the first $r$ registers that are consistent with the current iteration of the automaton.

- Given $q_1 \xrightarrow{X,Y} q_2 \in \delta$, for any $Z' \subseteq [r] \setminus Z$ and any $Z'' \supseteq X \setminus Z$, add

$$(q_1, Z) \xrightarrow{(X \cap Z) \cup Z' \cup (r+Z''),Y} (q_2, Z \cup Y)$$

  to $\delta$. $Z'$ corresponds to outdated content, which can be ignored.

- Transitions from the initial state need extra treatment: we need to accept $\epsilon$ (hence the need for the new initial state) as well as creating loops supporting iteration. Given $q_I \xrightarrow{X,Y} q \in \delta$,
  - add $q'_I \xrightarrow{X \cup (r+X),Y} (q, [r])$ to $\delta$,
  - for any $f \in F$ and $Z, Z' \subseteq [r]$, add $(f, Z) \xrightarrow{Z' \cup (r+X),Y} (q, Y)$ to $\delta$.

# 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 $\mathcal{A} = \langle Q, q_I, \tau_I, \delta, F \rangle$ 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 $w' \in \mathcal{L}(\mathcal{A})$.
- More concretely, one take $\mathcal{D}_b$ to be an arbitrary superset of $\tau_I([r]) \cap \mathcal{D}$ of size $r + 1$.

**Observation**

Emptiness of an RA 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.

# Reduction to FA

It suffices to consider $\mathcal{A}' = \langle\, Q', \Sigma, q'_I, \delta', F' \,\rangle$, where

$$\begin{array}{rcl} Q' & = & Q \times ([r] \to \mathcal{D}_b + \{\#\}) \\ \Sigma & = & \mathcal{D}_b \\ q'_I & = & (q_I, \tau_I) \\ F' & = & F \times ([r] \to \mathcal{D}_b + \{\#\}) \end{array}$$

and $\delta'$ is defined to be the subset of the successor relation for $\mathcal{A}$

$$(q_1, \tau_1) \xrightarrow{d} (q_2, \tau_2)$$

where $d \in \mathcal{D}_b$ and $\tau_1, \tau_2 \in ([r] \to \mathcal{D}_b + \{\#\})$.

FA emptiness amounts to a simple reachability test (of a final state from the initial state). Thus, RA emptiness is decidable.

# Universality

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

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

**Proof idea** Reduction from Post Correspondence Problem:

given $x_i, y_i \in \{a, b\}^*$ $(i = 1, \cdots, n)$ do there exist $k \geq 1$ and $1 \leq i_1, \cdots, i_k \leq n$ such that

$$x_{i_1} \cdots x_{i_k} = y_{i_1} \cdots y_{i_k}.$$

# Representation scheme

Fix $\mathcal{D}_{\text{aux}} = \{\&, \#, 1, 2, \cdots, n, a, b\} \subseteq \mathcal{D}$. A solution $i_1, \cdots, i_k$ to a PCP instance can be represented by a string of the form

$$u \# v$$

where $u, v$ have the following shape.

- $u$ consists of $k$ segments of the form

$$\& \; id_j^u \; i_j^u \; \delta_1^u \; c_1^u \; \cdots \; \delta_{p_j^u}^u \; c_{p_j^u}^u,$$

  where $id_j^u, \delta_h^u \in \mathcal{D} \setminus \mathcal{D}_{\text{aux}}$, $i_j^u \in \{1, \cdots, n\}$ and $c_h^u \in \{a, b\}$.
- $v$ consists of $k$ segments of the form

$$\& \; id_j^v \; i_j^v \; \delta_1^v \; c_1^v \; \cdots \; \delta_{p_j^v}^v \; c_{p_j^v}^v,$$

  where $id_j^v, \delta_h^v \in \mathcal{D} \setminus \mathcal{D}_{\text{aux}}$, $i_j^v \in \{1, \cdots, n\}$ and $c_h^v \in \{a, b\}$.

# Correctness conditions

$$\cdots \& \; id_j^u \; i_j^u \; \delta_1^u \; c_1^u \; \cdots \; \delta_{p_j^u}^u \; c_{p_j^u}^u \cdots \# \cdots \& id_j^v \; i_j^v \; \delta_1^v \; c_1^v \; \cdots \; \delta_{p_j^v}^v \; c_{p_j^v}^v \cdots$$

- $x_{i_j} = c_1^u \cdots c_{p_j^u}^u$ and $y_{i_j} = c_1^v \cdots c_{p_j^v}^v$
- $id_j^u$'s and $\delta_h^u$'s are pairwise different.
- Ditto for $id_j^v$'s and $\delta_h^v$'s.
- $u \restriction id = v \restriction id$
- $u \restriction i_j^u = v \restriction i_j^v$
- $u \restriction \delta_h^u = v \restriction \delta_h^v$
- $u \restriction c_h^u = v \restriction c_h^v$

**Key idea**

For each condition above, we can construct a register automaton that will detect a violation (use nondeterminism).

# Universality

- Each kind of violations can be detected by an RA.
- Overall, because RA's are closed under union, there exists an RA $\mathcal{A}$ that can recognise all kinds of violations in the representation of a PCP solution.

## Key observation

If a PCP instance has no solution then all words are bound to contain a violation. Consequently

$$\mathcal{L}(\mathcal{A}) = \mathcal{D}^* \iff \text{PCP instance has no solution.}$$

Because PCP is not decidable, RA universality must be undecidable.

# Bibliography

- Register automata [KF94, NSV04]
- Pushdown register automata [CK98, Seg06, MRT14]
- More [NSV04, Seg06, BS07, BKL14]


[BKL14] Mikolaj Bojanczyk, Bartek Klin, and Slawomir Lasota. Automata theory in nominal sets. *Logical Methods in Computer Science*, 10(3), 2014.

[BS07] H. Björklund and T. Schwentick. On notions of regularity for data languages. In *Proceedings of FCT*, volume 4639 of *Lecture Notes in Computer Science*, pages 88–99. Springer, 2007.

[CK98] E. Y. C. Cheng and M. Kaminski. Context-free languages over infinite alphabets. *Acta Inf.*, 35(3):245–267, 1998.

[KF94] M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[MRT14] A. S. Murawski, S. J. Ramsay, and N. Tzevelekos. Reachability in pushdown register automata. In *Proceedings of MFCS*, LNCS, pages 464–473. Springer, 2014.

[NSV04] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Log.*, 5(3):403–435, 2004.

[Seg06] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Proceedings of CSL*, volume 4207 of *Lecture Notes in Computer Science*. Springer, 2006.