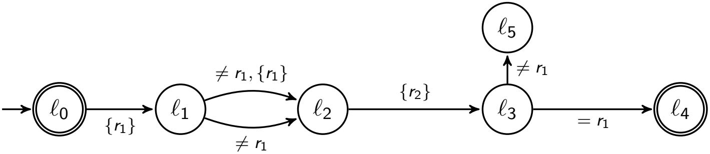
The Containment Problem for Unambiguous Register Automata

Antoine Mottet, Karin Quaas

QuantLA Workshop 2018

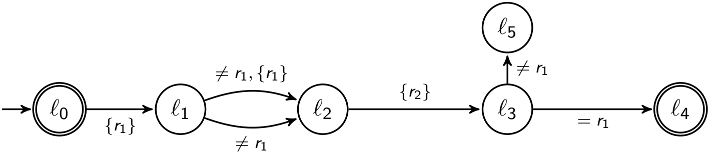
- Extension of finite automata to infinite alphabets ($\Sigma \times \mathbb{N}$)

- Extension of finite automata to infinite alphabets ($\Sigma \times \mathbb{N}$)



Register Automata

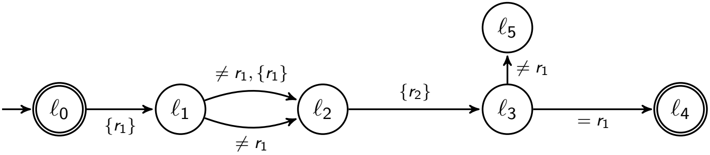
- Extension of finite automata to infinite alphabets ($\Sigma \times \mathbb{N}$)



$$(\ell_0, \bot, \bot) \xrightarrow{\binom{a}{0}} (\ell_0, 0, \bot) \xrightarrow{\binom{a}{1}} (\ell_2, 0, \bot) \xrightarrow{\binom{a}{3}} (\ell_3, 0, 3) \xrightarrow{\binom{a}{0}} (\ell_4, 0, 3)$$

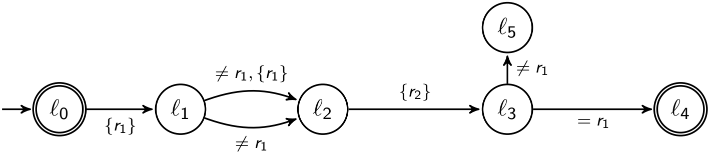
Register Automata

- Extension of finite automata to infinite alphabets ($\Sigma \times \mathbb{N}$)



$$
\begin{array}{ccccccccc}
(\ell_0, \bot, \bot) & \xrightarrow{\binom{a}{0}} & (\ell_0, 0, \bot) & \xrightarrow{\binom{a}{1}} & (\ell_2, 0, \bot) & \xrightarrow{\binom{a}{3}} & (\ell_3, 0, 3) & \xrightarrow{\binom{a}{0}} & (\ell_4, 0, 3) \\
 & & & \longrightarrow & (\ell_2, 1, \bot) & \longrightarrow & (\ell_3, 1, 3) & \longrightarrow & (\ell_5, 1, 3)
\end{array}
$$

- Extension of finite automata to infinite alphabets ($\Sigma \times \mathbb{N}$)



$$\begin{array}{ccccccccc} (\ell_0, \perp, \perp) & \xrightarrow{\binom{a}{0}} & (\ell_0, 0, \perp) & \xrightarrow{\binom{a}{1}} & (\ell_2, 0, \perp) & \xrightarrow{\binom{a}{3}} & (\ell_3, 0, 3) & \xrightarrow{\binom{a}{0}} & (\ell_4, 0, 3) \\ & & & \longrightarrow & (\ell_2, 1, \perp) & \longrightarrow & (\ell_3, 1, 3) & \longrightarrow & (\ell_5, 1, 3) \end{array}$$

- Recognizers of orbits:

$$\binom{a}{0}\binom{a}{1}\binom{a}{3}\binom{a}{0} \sim \binom{a}{4}\binom{a}{3}\binom{a}{1}\binom{a}{4}.$$

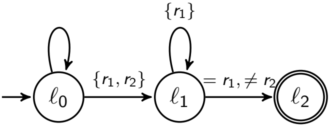
- Projection of $L \subseteq (\Sigma \times \mathbb{N})^*$ onto Σ^*: set of words $w \in \Sigma^*$ such that $(w_1, d_1) \dots (w_n, d_n) \in L$ for some $d_1, \dots, d_n \in \mathbb{N}$.

Projection and Emptiness

- Projection of $L \subseteq (\Sigma \times \mathbb{N})^*$ onto Σ^*: set of words $w \in \Sigma^*$ such that $(w_1, d_1) \dots (w_n, d_n) \in L$ for some $d_1, \dots, d_n \in \mathbb{N}$.
- Projection of recognizable L is regular (rec. by orbit automaton).

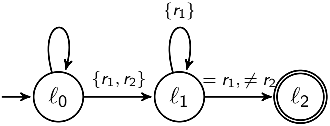
Projection and Emptiness

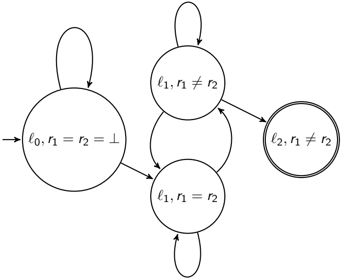
- Projection of $L \subseteq (\Sigma \times \mathbb{N})^*$ onto Σ^*: set of words $w \in \Sigma^*$ such that $(w_1, d_1) \dots (w_n, d_n) \in L$ for some $d_1, \dots, d_n \in \mathbb{N}$.
- Projection of recognizable L is regular (rec. by orbit automaton).



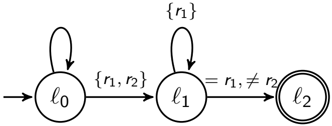
Projection and Emptiness

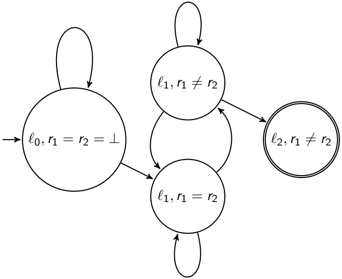
- Projection of $L \subseteq (\Sigma \times \mathbb{N})^*$ onto Σ^*: set of words $w \in \Sigma^*$ such that $(w_1, d_1) \ldots (w_n, d_n) \in L$ for some $d_1, \ldots, d_n \in \mathbb{N}$.
- Projection of recognizable L is regular (rec. by orbit automaton).





- Projection of $L \subseteq (\Sigma \times \mathbb{N})^*$ onto Σ^*: set of words $w \in \Sigma^*$ such that $(w_1, d_1) \dots (w_n, d_n) \in L$ for some $d_1, \dots, d_n \in \mathbb{N}$.
- Projection of recognizable L is regular (rec. by orbit automaton).
- Emptiness of L is decidable: $L = \emptyset \Leftrightarrow$ its projection is empty.





Definition

An automaton is unambiguous if every word has at most 1 accepting run.

Definition

An automaton is unambiguous if every word has at most 1 accepting run.

- Deterministic $\subseteq$ Unambiguous $\subseteq$ Non-deterministic,

Definition

An automaton is unambiguous if every word has at most 1 accepting run.

- Deterministic $\subseteq$ Unambiguous $\subseteq$ Non-deterministic,
- Ambiguity as a resource (STAA?),

Definition

An automaton is unambiguous if every word has at most 1 accepting run.

- Deterministic $\subseteq$ Unambiguous $\subseteq$ Non-deterministic,
- Ambiguity as a resource (STAA?),
- Collapses and non-collapses depending on model of computation,

Definition

An automaton is unambiguous if every word has at most 1 accepting run.

- Deterministic $\subseteq$ Unambiguous $\subseteq$ Non-deterministic,
- Ambiguity as a resource (STAA?),
- Collapses and non-collapses depending on model of computation,
- Succinctness,

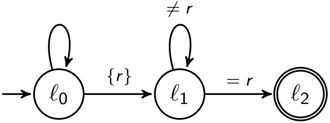
Unambiguity

Definition

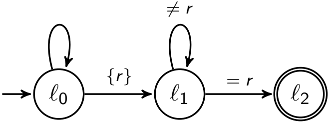
An automaton is unambiguous if every word has at most 1 accepting run.

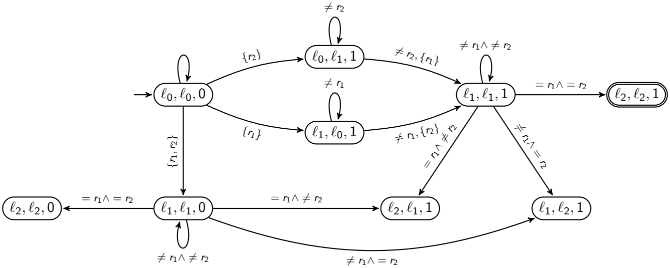
- Deterministic $\subseteq$ Unambiguous $\subseteq$ Non-deterministic,
- Ambiguity as a resource (STAA?),
- Collapses and non-collapses depending on model of computation,
- Succinctness,
- Important problems related to unambiguity (parity games in $UP \setminus P$?).

Given a RA $\mathcal{A}$, decide if $\mathcal{A}$ is unambiguous:



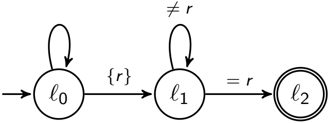
Given a RA $\mathcal{A}$, decide if $\mathcal{A}$ is unambiguous:





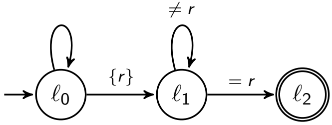
Non-closure under complement

- $L = \{d_1 \dots d_n \in \mathbb{N}^* \mid \exists i \in \{1, \dots, n-1\} : d_i = d_n\}$



Non-closure under complement

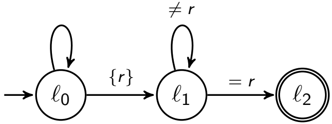
- $L = \{d_1 \dots d_n \in \mathbb{N}^* \mid \exists i \in \{1, \dots, n-1\} : d_i = d_n\}$



- $\overline{L}$ not recognizable (even by nondeterministic RA):

Non-closure under complement

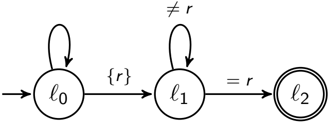
- $L = \{d_1 \ldots d_n \in \mathbb{N}^* \mid \exists i \in \{1, \ldots, n-1\} : d_i = d_n\}$



- $\overline{L}$ not recognizable (even by nondeterministic RA):
  - say $\mathcal{A}$ is a k-register RA recognizing $\overline{L}$,

Non-closure under complement

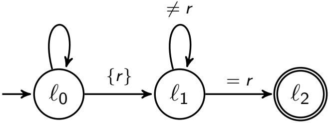
- $L = \{d_1 \dots d_n \in \mathbb{N}^* \mid \exists i \in \{1, \dots, n-1\} : d_i = d_n\}$



- $\overline{L}$ not recognizable (even by nondeterministic RA):
  - say $\mathcal{A}$ is a k-register RA recognizing $\overline{L}$,
  - $(0)(1)\dots(k)(k+1)(k+2) \in \overline{L}$, so $\exists$ accepting run of $\mathcal{A}$.

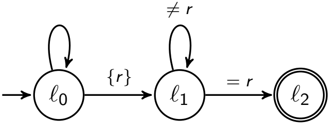
Non-closure under complement

- $L = \{d_1 \dots d_n \in \mathbb{N}^* \mid \exists i \in \{1, \dots, n-1\} : d_i = d_n\}$



- $\overline{L}$ not recognizable (even by nondeterministic RA):
  - say $\mathcal{A}$ is a k-register RA recognizing $\overline{L}$,
  - $(0)(1)\dots(k)(k+1)(k+2) \in \overline{L}$, so $\exists$ accepting run of $\mathcal{A}$.
  - Let $d \in \{1, \dots, k+1\}$ be one of the forgotten values.

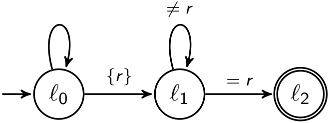
- $L = \{d_1 \dots d_n \in \mathbb{N}^* \mid \exists i \in \{1, \dots, n-1\} : d_i = d_n\}$



- $\overline{L}$ not recognizable (even by nondeterministic RA):
  - say $\mathcal{A}$ is a k-register RA recognizing $\overline{L}$,
  - $(0)(1)\dots(k)(k+1)(k+2) \in \overline{L}$, so $\exists$ accepting run of $\mathcal{A}$.
  - Let $d \in \{1, \dots, k+1\}$ be one of the forgotten values.
  - $(0)(1)\dots(k)(k+1)(d) \notin \overline{L}$ has the same accepting run.

Non-closure under complement

- $L = \{d_1 \dots d_n \in \mathbb{N}^* \mid \exists i \in \{1, \dots, n-1\} : d_i = d_n\}$

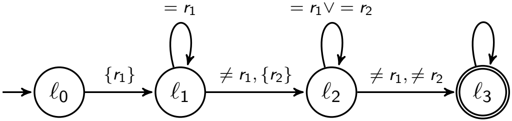


- $\overline{L}$ not recognizable (even by nondeterministic RA):
  - say $\mathcal{A}$ is a k-register RA recognizing $\overline{L}$,
  - $(0)(1)\dots(k)(k+1)(k+2) \in \overline{L}$, so $\exists$ accepting run of $\mathcal{A}$.
  - Let $d \in \{1, \dots, k+1\}$ be one of the forgotten values.
  - $(0)(1)\dots(k)(k+1)(d) \notin \overline{L}$ has the same accepting run.
- In particular L not recognizable by deterministic RA.

$$\{d_1 \cdots d_n \in \mathbb{N}^* \mid \#\{d_1, \ldots, d_n\} \geq 3\}$$

$$\{d_1 \cdots d_n \in \mathbb{N}^* \mid \#\{d_1, \ldots, d_n\} \geq 3\}$$

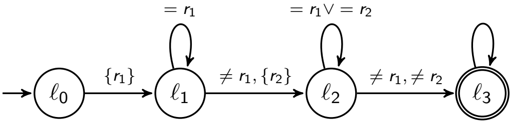
- Recognizable by deterministic RA:



"Succinctness"

$$\{d_1 \cdots d_n \in \mathbb{N}^* \mid \#\{d_1, \ldots, d_n\} \geq 3\}$$

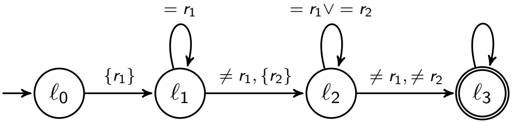
- Recognizable by deterministic RA:



- Needs 2 registers.

$$\{d_1 \cdots d_n \in \mathbb{N}^* \mid \#\{d_1, \ldots, d_n\} \geq 3\}$$

- Recognizable by deterministic RA:



- Needs 2 registers.
- Exists a 1-register unambiguous RA.

- Universality: Given $\mathcal{B}$, determine if $L(\mathcal{B}) = (\Sigma \times \mathbb{N})^*$.

$\mathcal{B}$	DRA	URA	NRA
1 register	NL-complete	?	Ackermann-complete
≥ 2 registers	NL-complete	?	Undecidable
*	PSPACE-complete	?	Undecidable

- Universality: Given $\mathcal{B}$, determine if $L(\mathcal{B}) = (\Sigma \times \mathbb{N})^*$.

$\mathcal{B}$	DRA	URA	NRA
1 register	NL-complete	?	Ackermann-complete
≥ 2 registers	NL-complete	?	Undecidable
*	PSPACE-complete	?	Undecidable

- Containment: Given $\mathcal{A}, \mathcal{B}$, determine if $L(\mathcal{A}) \subseteq L(\mathcal{B})$.

$\mathcal{B}$	DRA	URA	NRA
1 register	PSPACE-complete	?	Ackermann-complete
*	PSPACE-complete	?	Undecidable

- Universality: Given $\mathcal{B}$, determine if $L(\mathcal{B}) = (\Sigma \times \mathbb{N})^*$.

$\mathcal{B}$	DRA	URA	NRA
1 register	NL-complete	?	Ackermann-complete
≥ 2 registers	NL-complete	?	Undecidable
*	PSPACE-complete	?	Undecidable

- Containment: Given $\mathcal{A}, \mathcal{B}$, determine if $L(\mathcal{A}) \subseteq L(\mathcal{B})$.

$\mathcal{B}$	DRA	URA	NRA
1 register	PSPACE-complete	?	Ackermann-complete
*	PSPACE-complete	?	Undecidable

- $L(\mathcal{A}) \subseteq L(\mathcal{B}) \Leftrightarrow L(\mathcal{A}) \cap \overline{L(\mathcal{B})} = \emptyset$

- Universality: Given $\mathcal{B}$, determine if $L(\mathcal{B}) = (\Sigma \times \mathbb{N})^*$.

$\mathcal{B}$	DRA	URA	NRA
1 register	NL-complete	?	Ackermann-complete
≥ 2 registers	NL-complete	?	Undecidable
*	PSPACE-complete	?	Undecidable

- Containment: Given $\mathcal{A}, \mathcal{B}$, determine if $L(\mathcal{A}) \subseteq L(\mathcal{B})$.

$\mathcal{B}$	DRA	URA	NRA
1 register	PSPACE-complete	?	Ackermann-complete
*	PSPACE-complete	?	Undecidable

- $L(\mathcal{A}) \subseteq L(\mathcal{B}) \Leftrightarrow L(\mathcal{A}) \cap \overline{L(\mathcal{B})} = \emptyset$
  $\rightsquigarrow$ "on-the-fly" complementation.

- Configuration C of n-register $\mathcal{B}$: set of tuples $(\ell^{\mathcal{B}}, d_1, \ldots, d_n)$.

Synchronized configurations

- Configuration C of n-register $\mathcal{B}$: set of tuples $(\ell^{\mathcal{B}}, d_1, \ldots, d_n)$.
- Synchronized configuration of $\mathcal{A}$ and $\mathcal{B}$: $((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C)$.

- Configuration C of n-register $\mathcal{B}$: set of tuples $(\ell^{\mathcal{B}}, d_1, \ldots, d_n)$.
- Synchronized configuration of $\mathcal{A}$ and $\mathcal{B}$: $((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C)$.
- (Infinitely branching) transition system $(\mathbb{S}, \rightarrow)$ on synchronized configurations:

- Configuration C of n-register $\mathcal{B}$: set of tuples $(\ell^{\mathcal{B}}, d_1, \ldots, d_n)$.
- Synchronized configuration of $\mathcal{A}$ and $\mathcal{B}$: $((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C)$.
- (Infinitely branching) transition system $(\mathbb{S}, \rightarrow)$ on synchronized configurations:

$$((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C) \rightarrow ((\ell'^{\mathcal{A}}, e_1, \ldots, e_m), C')$$

if $(\ell^{\mathcal{A}}, d_1, \ldots, d_m) \xrightarrow{\binom{\sigma}{d}} (\ell'^{\mathcal{A}}, e_1, \ldots, e_m)$ and $C \xrightarrow{\binom{\sigma}{d}} C'$ for some $(\sigma, d) \in \Sigma \times \mathbb{N}$.

- Configuration C of n-register $\mathcal{B}$: set of tuples $(\ell^{\mathcal{B}}, d_1, \ldots, d_n)$.
- Synchronized configuration of $\mathcal{A}$ and $\mathcal{B}$: $((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C)$.
- (Infinitely branching) transition system $(\mathbb{S}, \to)$ on synchronized configurations:

$$((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C) \to ((\ell'^{\mathcal{A}}, e_1, \ldots, e_m), C')$$

  if $(\ell^{\mathcal{A}}, d_1, \ldots, d_m) \xrightarrow{\binom{\sigma}{d}} (\ell'^{\mathcal{A}}, e_1, \ldots, e_m)$ and $C \xrightarrow{\binom{\sigma}{d}} C'$ for some $(\sigma, d) \in \Sigma \times \mathbb{N}$.
- $((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C)$ bad if $\ell^{\mathcal{A}}$ accepting and C not accepting.

- Configuration C of n-register $\mathcal{B}$: set of tuples $(\ell^{\mathcal{B}}, d_1, \ldots, d_n)$.
- Synchronized configuration of $\mathcal{A}$ and $\mathcal{B}$: $((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C)$.
- (Infinitely branching) transition system $(\mathbb{S}, \to)$ on synchronized configurations:

$$((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C) \to ((\ell'^{\mathcal{A}}, e_1, \ldots, e_m), C')$$

  if $(\ell^{\mathcal{A}}, d_1, \ldots, d_m) \xrightarrow{\binom{\sigma}{d}} (\ell'^{\mathcal{A}}, e_1, \ldots, e_m)$ and $C \xrightarrow{\binom{\sigma}{d}} C'$ for some $(\sigma, d) \in \Sigma \times \mathbb{N}$.
- $((\ell^{\mathcal{A}}, d_1, \ldots, d_m), C)$ bad if $\ell^{\mathcal{A}}$ accepting and C not accepting.
- $L(\mathcal{A}) \not\subseteq L(\mathcal{B}) \Leftrightarrow \exists$bad reachable configuration in $(\mathbb{S}, \to)$.

Dealing with infinities

Dealing with infinities

The non-deterministic case:

- Infinite branching: only consider "essentially different" successors.

Dealing with infinities

The non-deterministic case:

- Infinite branching: only consider "essentially different" successors.
- Infinite depth: ...

Dealing with infinities

The non-deterministic case:

- Infinite branching: only consider "essentially different" successors.
- Infinite depth: ...
  - $\preceq$ is a well-quasi-order if for every infinite sequence $S_0, S_1, \ldots,$ there exist $i < j$ such that $S_i \preceq S_j$.
  - For 1 register: define a WQO $S \preceq S'$ on synchronized configurations such that if S' reaches a bad configuration in k steps, then S reaches bad in k steps.

Dealing with infinities

The non-deterministic case:

- Infinite branching: only consider "essentially different" successors.
- Infinite depth: ...
  - $\preceq$ is a well-quasi-order if for every infinite sequence $S_0, S_1, \dots$, there exist $i < j$ such that $S_i \preceq S_j$.
  - For 1 register: define a WQO $S \preceq S'$ on synchronized configurations such that if S' reaches a bad configuration in k steps, then S reaches bad in k steps.
  - For ≥ 2 registers: no such WQO exists (because of undecidability).

The non-deterministic case:

- Infinite branching: only consider "essentially different" successors.
- Infinite depth: ...
  - $\preceq$ is a well-quasi-order if for every infinite sequence $S_0, S_1, \ldots,$ there exist $i < j$ such that $S_i \preceq S_j$.
  - For 1 register: define a WQO $S \preceq S'$ on synchronized configurations such that if S' reaches a bad configuration in k steps, then S reaches bad in k steps.
  - For ≥ 2 registers: no such WQO exists (because of undecidability).

The unambiguous case: try to bound size of configurations.

Types...

Definition

An *n-type* is a satisfiable conjunction $\varphi(x_1, \ldots, x_n)$ of $=$ and $\neq$ that is maximal (any formula containing φ is equivalent to φ or insatisfiable).

Types...

Definition

An *n-type* is a satisfiable conjunction $\varphi(x_1, \ldots, x_n)$ of $=$ and $\neq$ that is maximal (any formula containing φ is equivalent to φ or insatisfiable).

- $x_1 = x_2$ and $x_1 \neq x_2$ are the only 2-types,

Definition

An *n-type* is a satisfiable conjunction $\varphi(x_1, \ldots, x_n)$ of $=$ and $\neq$ that is maximal (any formula containing φ is equivalent to φ or insatisfiable).

- $x_1 = x_2$ and $x_1 \neq x_2$ are the only 2-types,
- $x_1 = x_2 \wedge x_2 \neq x_3$ and $x_1 \neq x_2 \wedge x_1 = x_3$ are 3-types (there are 5 in total),

Types...

Definition

An *n-type* is a satisfiable conjunction $\varphi(x_1, \ldots, x_n)$ of $=$ and $\neq$ that is maximal (any formula containing φ is equivalent to φ or insatisfiable).

- $x_1 = x_2$ and $x_1 \neq x_2$ are the only 2-types,
- $x_1 = x_2 \wedge x_2 \neq x_3$ and $x_1 \neq x_2 \wedge x_1 = x_3$ are 3-types (there are 5 in total),
- In general, there are at most $n^n = O(2^{n^2})$ types with n variables (Bell numbers).

Definition

An *n-type* is a satisfiable conjunction $\varphi(x_1, \ldots, x_n)$ of $=$ and $\neq$ that is maximal (any formula containing φ is equivalent to φ or insatisfiable).

- $x_1 = x_2$ and $x_1 \neq x_2$ are the only 2-types,
- $x_1 = x_2 \wedge x_2 \neq x_3$ and $x_1 \neq x_2 \wedge x_1 = x_3$ are 3-types (there are 5 in total),
- In general, there are at most $n^n = O(2^{n^2})$ types with n variables (Bell numbers).
- Every tuple $(d_1, \ldots, d_n) \in \mathbb{N}^n$ has a type $\mathrm{tp}(d_1, \ldots, d_n)$.

Types...

Definition

An *n*-type is a satisfiable conjunction $\varphi(x_1, \dots, x_n)$ of $=$ and $\neq$ that is maximal (any formula containing φ is equivalent to φ or insatisfiable).

- $x_1 = x_2$ and $x_1 \neq x_2$ are the only 2-types,
- $x_1 = x_2 \wedge x_2 \neq x_3$ and $x_1 \neq x_2 \wedge x_1 = x_3$ are 3-types (there are 5 in total),
- In general, there are at most $n^n = O(2^{n^2})$ types with n variables (Bell numbers).
- Every tuple $(d_1, \dots, d_n) \in \mathbb{N}^n$ has a type $\mathrm{tp}(d_1, \dots, d_n)$.
- $\mathrm{tp}(d_1, \dots, d_n) = \mathrm{tp}(e_1, \dots, e_n) \Leftrightarrow$ $\exists$ permutation α of $\mathbb{N}$ s.t. $\alpha(d_i) = e_i$.

Consider

$$C = \{(\ell, 1, 2), (\ell'', 1, 2), (\ell', 3, 4), (\ell', 2, 5), (\ell'', 4, 5), (\ell, 1, 3), (\ell'', 1, 3)\}$$

Consider

$$C = \{(\ell, 1, 2), (\ell'', 1, 2), (\ell', 3, 4), (\ell', 2, 5), (\ell'', 4, 5), (\ell, 1, 3), (\ell'', 1, 3)\}$$

- Pick $\varphi(x_1, x_2, x_3, x_4)$ a 4-type.

Consider

$$C = \{(\ell, 1, 2), (\ell'', 1, 2), (\ell', 3, 4), (\ell', 2, 5), (\ell'', 4, 5), (\ell, 1, 3), (\ell'', 1, 3)\}$$

- Pick $\varphi(x_1, x_2, x_3, x_4)$ a 4-type.
- For (d_1, d_2), compute

$$L_\varphi(d_1, d_2) := \{\ell \mid \exists e_1, e_2 : (\ell, e_1, e_2) \in C \text{ and } \mathbb{N} \models \varphi(d_1, d_2, e_1, e_2)\}.$$

Consider

$$C = \{(\ell, 1, 2), (\ell'', 1, 2), (\ell', 3, 4), (\ell', 2, 5), (\ell'', 4, 5), (\ell, 1, 3), (\ell'', 1, 3)\}$$

- Pick $\varphi(x_1, x_2, x_3, x_4)$ a 4-type.
- For (d_1, d_2), compute

$$L_\varphi(d_1, d_2) := \{\ell \mid \exists e_1, e_2 : (\ell, e_1, e_2) \in C \text{ and } \mathbb{N} \models \varphi(d_1, d_2, e_1, e_2)\}.$$

- $\varphi := (x_1 = x_3 \neq x_2 = x_4)$
- $L_\varphi(1, 2) = \{\ell, \ell''\}$
- $L_\varphi(2, 5) = \{\ell'\}$,
- $L_\varphi(3, 4) = \{\ell'\}$,
- $L_\varphi(1, 3) = \{\ell, \ell''\}$.

- $\psi := \{x_2, x_3\}, \{x_1\}, \{x_4\}$
- $L_\psi(1, 2) = \{\ell'\}$
- $L_\psi(2, 5) = \emptyset$,
- $L_\psi(3, 4) = \{\ell''\}$,
- $L_\psi(1, 3) = \{\ell'\}$.

Consider

$$C = \{(\ell, 1, 2), (\ell'', 1, 2), (\ell', 3, 4), (\ell', 2, 5), (\ell'', 4, 5), (\ell, 1, 3), (\ell'', 1, 3)\}$$

- Pick $\varphi(x_1, x_2, x_3, x_4)$ a 4-type.
- For (d_1, d_2), compute

$$L_\varphi(d_1, d_2) := \{\ell \mid \exists e_1, e_2 : (\ell, e_1, e_2) \in C \text{ and } \mathbb{N} \models \varphi(d_1, d_2, e_1, e_2)\}.$$

- $\varphi := (x_1 = x_3 \neq x_2 = x_4)$
- $L_\varphi(1, 2) = \{\ell, \ell''\}$
- $L_\varphi(2, 5) = \{\ell'\}$,
- $L_\varphi(3, 4) = \{\ell'\}$,
- $L_\varphi(1, 3) = \{\ell, \ell''\}$.

- $\psi := \{x_2, x_3\}, \{x_1\}, \{x_4\}$
- $L_\psi(1, 2) = \{\ell'\}$
- $L_\psi(2, 5) = \emptyset$,
- $L_\psi(3, 4) = \{\ell''\}$,
- $L_\psi(1, 3) = \{\ell'\}$.

- $\overline{d} \equiv_C \overline{e}$ if for every $2n$-type φ, $L_\varphi(\overline{d}) = L_\varphi(\overline{e})$.

Consider

$$C = \{(\ell, 1, 2), (\ell'', 1, 2), (\ell', 3, 4), (\ell', 2, 5), (\ell'', 4, 5), (\ell, 1, 3), (\ell'', 1, 3)\}$$

- Pick $\varphi(x_1, x_2, x_3, x_4)$ a 4-type.
- For (d_1, d_2), compute

$$L_\varphi(d_1, d_2) := \{\ell \mid \exists e_1, e_2 : (\ell, e_1, e_2) \in C \text{ and } \mathbb{N} \models \varphi(d_1, d_2, e_1, e_2)\}.$$

- $\varphi := (x_1 = x_3 \neq x_2 = x_4)$
- $L_\varphi(1, 2) = \{\ell, \ell''\}$
- $L_\varphi(2, 5) = \{\ell'\}$,
- $L_\varphi(3, 4) = \{\ell'\}$,
- $L_\varphi(1, 3) = \{\ell, \ell''\}$.

- $\psi := \{x_2, x_3\}, \{x_1\}, \{x_4\}$
- $L_\psi(1, 2) = \{\ell'\}$
- $L_\psi(2, 5) = \emptyset$,
- $L_\psi(3, 4) = \{\ell''\}$,
- $L_\psi(1, 3) = \{\ell'\}$.

- $\overline{d} \equiv_C \overline{e}$ if for every $2n$-type φ, $L_\varphi(\overline{d}) = L_\varphi(\overline{e})$.
- $(1, 2) \equiv_C (1, 3)$.

Consider

$$C = \{(\ell, 1, 2), (\ell'', 1, 2), (\ell', 3, 4), (\ell', 2, 5), (\ell'', 4, 5), (\ell, 1, 3), (\ell'', 1, 3)\}$$

- Pick $\varphi(x_1, x_2, x_3, x_4)$ a 4-type.
- For (d_1, d_2), compute

$$L_\varphi(d_1, d_2) := \{\ell \mid \exists e_1, e_2 : (\ell, e_1, e_2) \in C \text{ and } \mathbb{N} \models \varphi(d_1, d_2, e_1, e_2)\}.$$

- $\varphi := (x_1 = x_3 \neq x_2 = x_4)$
- $L_\varphi(1, 2) = \{\ell, \ell''\}$
- $L_\varphi(2, 5) = \{\ell'\}$,
- $L_\varphi(3, 4) = \{\ell'\}$,
- $L_\varphi(1, 3) = \{\ell, \ell''\}$.

- $\psi := \{x_2, x_3\}, \{x_1\}, \{x_4\}$
- $L_\psi(1, 2) = \{\ell'\}$
- $L_\psi(2, 5) = \emptyset$,
- $L_\psi(3, 4) = \{\ell''\}$,
- $L_\psi(1, 3) = \{\ell'\}$.

- $\overline{d} \equiv_C \overline{e}$ if for every $2n$-type φ, $L_\varphi(\overline{d}) = L_\varphi(\overline{e})$.
- $(1, 2) \equiv_C (1, 3)$.
- Generalize $\equiv_C$ to synchronized configurations.

Collapsing

Proposition (M-Quaas '18)

C reachable. $\overline{a}, \overline{b}$ such that $\overline{a} \equiv_C \overline{b}$.
Let $C' = C \setminus \{(\ell, \overline{b}) \in C\}$.
C reaches a bad configuration in k steps iff
C' reaches a bad configuration in k steps.

- C coverable if $\exists C' \supseteq C$ reachable.

Proposition (M-Quaas '18)

C coverable configuration. $\overline{a}, \overline{b}$ such that $\overline{a} \equiv_C \overline{b}$.
Let $C' = C \setminus \{(\ell, \overline{b}) \in C\}$.
C reaches a bad configuration in k steps iff
C' reaches a bad configuration in k steps.

- C coverable if $\exists C' \supseteq C$ reachable.

Proposition (M-Quaas '18)

C coverable configuration. $\overline{a}, \overline{b}$ such that $\overline{a} \equiv_C \overline{b}$.
Let $C' = C \setminus \{(\ell, \overline{b}) \in C\}$.
C reaches a bad configuration in k steps iff
C' reaches a bad configuration in k steps.

- If C is coverable, C' is coverable.

- C coverable if $\exists C' \supseteq C$ reachable.

Proposition (M-Quaas '18)

C coverable configuration. $\overline{a}, \overline{b}$ such that $\overline{a} \equiv_C \overline{b}$.
Let $C' = C \setminus \{(\ell, \overline{b}) \in C\}$.
C reaches a bad configuration in k steps iff
C' reaches a bad configuration in k steps.

- If C is coverable, C' is coverable.
- Given C coverable, one can decide in exponential time if the proposition applies.

- C coverable if $\exists C' \supseteq C$ reachable.

Proposition (M-Quaas '18)

C coverable configuration. $\overline{a}, \overline{b}$ such that $\overline{a} \equiv_C \overline{b}$.
Let $C' = C \setminus \{(\ell, \overline{b}) \in C\}$.
C reaches a bad configuration in k steps iff
C' reaches a bad configuration in k steps.

- If C is coverable, C' is coverable.
- Given C coverable, one can decide in exponential time if the proposition applies.
- If collapsing does not apply to C, number of data in $C \leq 2^{|\mathcal{B}| \times B_{2n+m}} \leq 2^{|\mathcal{B}| \times 2^{(2n+m)^2}}$.

- C coverable if $\exists C' \supseteq C$ reachable.

Proposition (M-Quaas '18)

C coverable configuration. $\overline{a}, \overline{b}$ such that $\overline{a} \equiv_C \overline{b}$.
Let $C' = C \setminus \{(\ell, \overline{b}) \in C\}$.
C reaches a bad configuration in k steps iff
C' reaches a bad configuration in k steps.

- If C is coverable, C' is coverable.
- Given C coverable, one can decide in exponential time if the proposition applies.
- If collapsing does not apply to C, number of data in $C \leq 2^{|\mathcal{B}| \times B_{2n+m}} \leq 2^{|\mathcal{B}| \times 2^{(2n+m)^2}}$.
- $\rightsquigarrow$ number of collapsed configurations $\leq 2^{2^{2^{poly(|\mathcal{A}|,|\mathcal{B}|)}}}$.

Algorithm

Decide $L(\mathcal{A}) \subseteq L(\mathcal{B})$:

Decide $L(\mathcal{A}) \subseteq L(\mathcal{B})$:

- Start exploring reachable synchronized configurations, starting from $((\ell_{\mathsf{in}}^{\mathcal{A}}, \perp), \{(\ell_{\mathsf{in}}^{\mathcal{B}}, \perp)\})$.

Decide $L(\mathcal{A}) \subseteq L(\mathcal{B})$:

- Start exploring reachable synchronized configurations, starting from $((\ell_{\mathsf{in}}^{\mathcal{A}}, \perp), \{(\ell_{\mathsf{in}}^{\mathcal{B}}, \perp)\})$.
- When reaching S and S can be collapsed to S', pretend we reached S'. If S' is bad, reject.

Decide $L(\mathcal{A}) \subseteq L(\mathcal{B})$:

- Start exploring reachable synchronized configurations, starting from $((\ell_{\mathsf{in}}^{\mathcal{A}}, \bot), \{(\ell_{\mathsf{in}}^{\mathcal{B}}, \bot)\})$.
- When reaching S and S can be collapsed to S', pretend we reached S'. If S' is bad, reject.
- When everything has been reached, accept.

Decide $L(\mathcal{A}) \subseteq L(\mathcal{B})$:

- Start exploring reachable synchronized configurations, starting from $((\ell_{\mathsf{in}}^{\mathcal{A}}, \perp), \{(\ell_{\mathsf{in}}^{\mathcal{B}}, \perp)\})$.
- When reaching S and S can be collapsed to S', pretend we reached S'. If S' is bad, reject.
- When everything has been reached, accept.
- At most $2^{2^{2^{poly(|\mathcal{A}|,|\mathcal{B}|)}}}$ collapsed configurations.
  $\rightsquigarrow$ 2-EXPSPACE algorithm.

Decide $L(\mathcal{A}) \subseteq L(\mathcal{B})$:

- Start exploring reachable synchronized configurations, starting from $((\ell_{\text{in}}^{\mathcal{A}}, \perp), \{(\ell_{\text{in}}^{\mathcal{B}}, \perp)\})$.
- When reaching S and S can be collapsed to S', pretend we reached S'. If S' is bad, reject.
- When everything has been reached, accept.
- At most $2^{2^{2^{poly(|\mathcal{A}|,|\mathcal{B}|)}}}$ collapsed configurations.
  $\rightsquigarrow$ 2-EXPSPACE algorithm.

$\mathcal{B}$	DRA	URA	NRA
1 register	PSPACE-comp.	EXPSPACE	Ackermann-comp.
*	PSPACE-comp.	2-EXPSPACE	Undecidable

Open problems

- For register automata:
  - Lower bounds,
  - Length of shortest witnesses for $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$?
  - Minimal number of data in witness for $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$?
  - Bounded amount of ambiguity?

Open problems

- For register automata:
  - Lower bounds,
  - Length of shortest witnesses for $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$?
  - Minimal number of data in witness for $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$?
  - Bounded amount of ambiguity?
- For RAs over ordered domain: decidability for ≥ 2 registers?

Open problems

- For register automata:
  - Lower bounds,
  - Length of shortest witnesses for $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$?
  - Minimal number of data in witness for $L(\mathcal{A}) \not\subseteq L(\mathcal{B})$?
  - Bounded amount of ambiguity?
- For RAs over ordered domain: decidability for ≥ 2 registers?
- Timed automata: decidability for ≥ 2 clocks?