On the use of guards for logics with data

Gabriele Puppis

LaBRI / CNRS

based on joint works with

Thomas Colcombet
and Clemens Ley

What is this talk about?

Data languages: sets of finite words over an **infinite alphabet invariant under permutations** of the letters.

Lifting of some classical results to data languages, e.g.

- translations between monoids and logic
- characterization of first-order definability

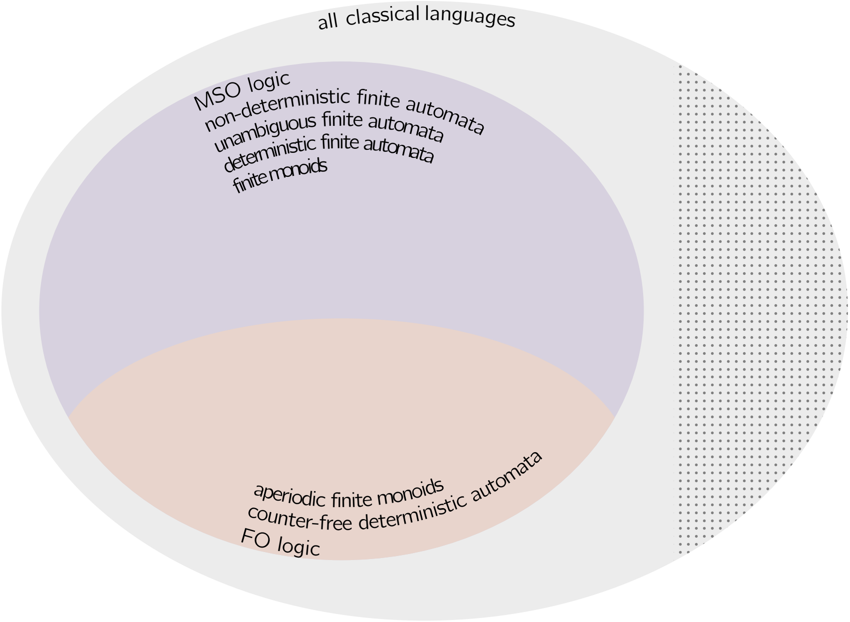
Example

$$L = \{w \in \mathbb{N}^\star \ : \ \textit{at most 2 distinct values in } w\}$$

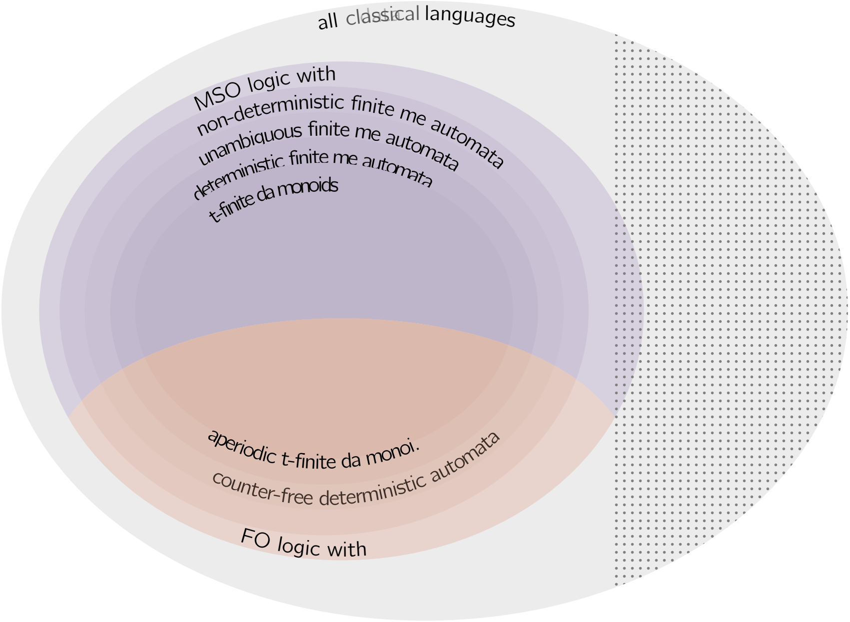
$= \{\varepsilon,$  $, \ \ldots\}$

$= \{\varepsilon,$  $, \ \ldots\}$

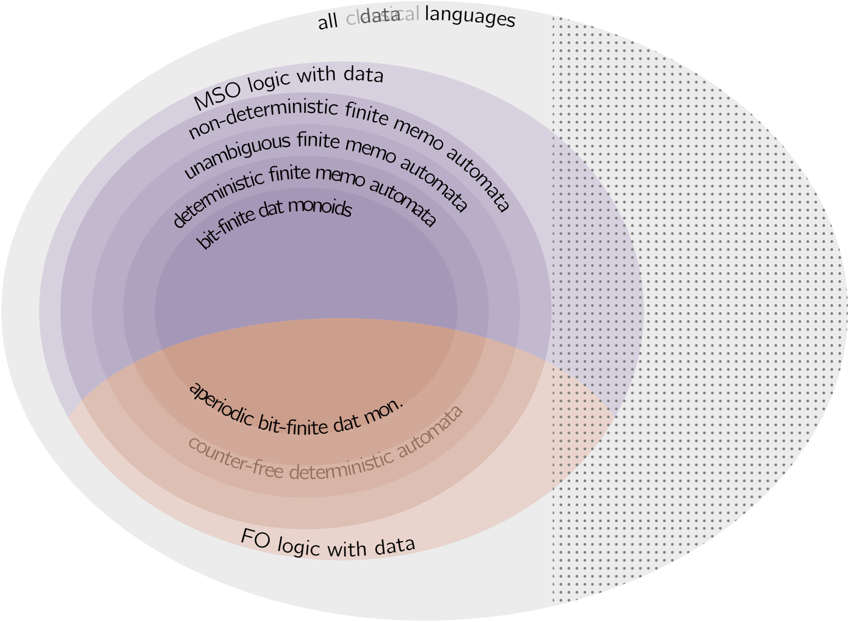
Logics, automata, and monoids over finite alphabets:



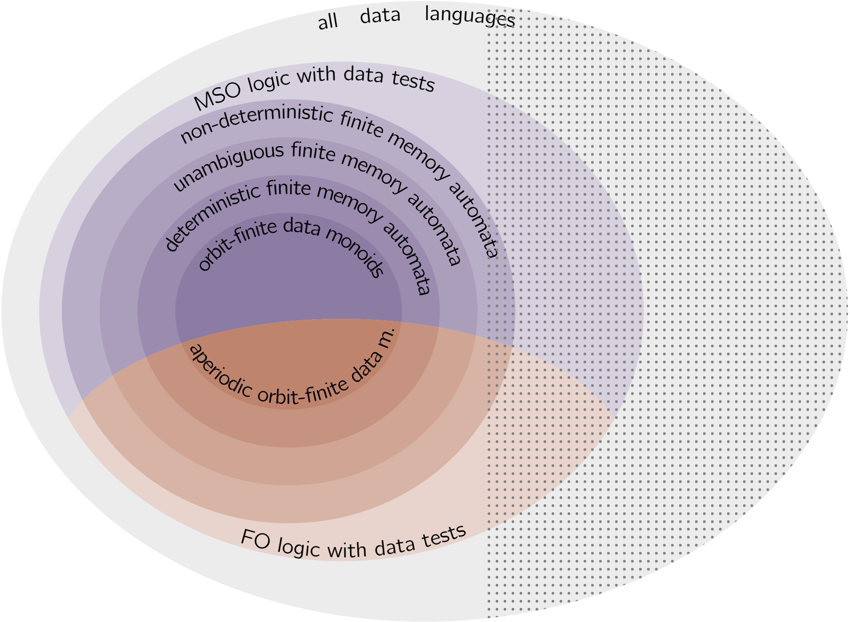
Logics, automata, and monoids over infinite alphabets:



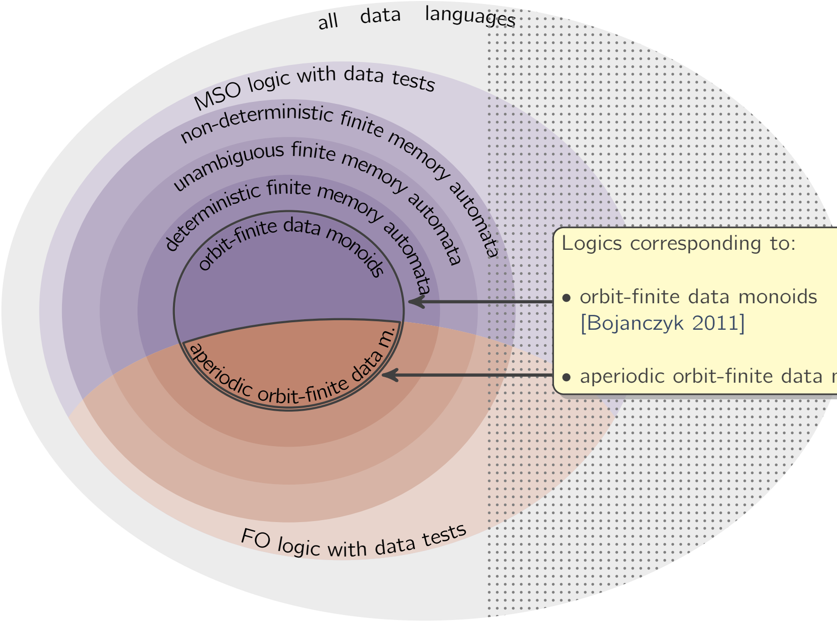
Logics, automata, and monoids over infinite alphabets:



Logics, automata, and monoids over infinite alphabets:



Logics, automata, and monoids over infinite alphabets:



Logics corresponding to:

- orbit-finite data monoids [Bojanczyk 2011]
- aperiodic orbit-finite data m.

Finite alphabet:

- **MSO logic**

$$\exists X.\ \ \text{first} \in X \ \wedge\ \text{last} \notin X$$
$$\wedge\ \ \forall y.\ y \in X \leftrightarrow y+1 \notin X$$

Infinite alphabet:

- **MSO logic with data tests**

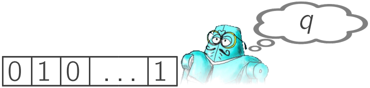
$$\exists X.\ \ \text{first} \in X \ \wedge\ \text{last} \notin X$$
$$\wedge\ \ \forall y.\ y \in X \leftrightarrow y+1 \notin X$$
$$\wedge\ \ \forall y, z \in X\ \rightarrow\ y \sim z$$

Finite alphabet:

- **MSO logic**

$$\exists X.\ \ \text{first} \in X \ \wedge\ \text{last} \notin X$$
$$\wedge\ \ \forall y.\ y \in X \leftrightarrow y+1 \notin X$$

- **Finite Automata**

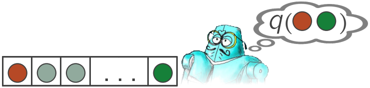


Infinite alphabet:

- **MSO logic with data tests**

$$\exists X.\ \ \text{first} \in X \ \wedge\ \text{last} \notin X$$
$$\wedge\ \ \forall y.\ y \in X \leftrightarrow y+1 \notin X$$
$$\wedge\ \ \forall y, z \in X \ \rightarrow\ {\color{red} y \sim z}$$

- **Finite Memory Automata**

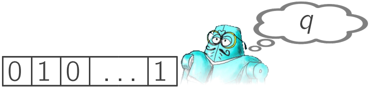


Finite alphabet:

- **MSO logic**

$$\exists X.\ \ \text{first} \in X \ \wedge\ \text{last} \notin X \ \wedge\ \forall y.\ y \in X \leftrightarrow y+1 \notin X$$

- **Finite Automata**



- **Finite monoids**

$$s \cdot t = s$$

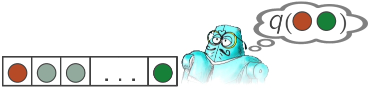
$$t \cdot s = t$$

Infinite alphabet:

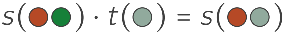
- **MSO logic with data tests**

$$\exists X.\ \ \text{first} \in X \ \wedge\ \text{last} \notin X \ \wedge\ \forall y.\ y \in X \leftrightarrow y+1 \notin X \ \wedge\ \forall y,z \in X \rightarrow {\color{red} y \sim z}$$

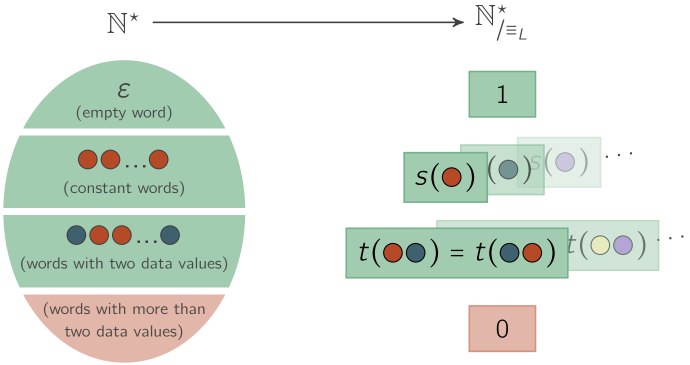
- **Finite Memory Automata**



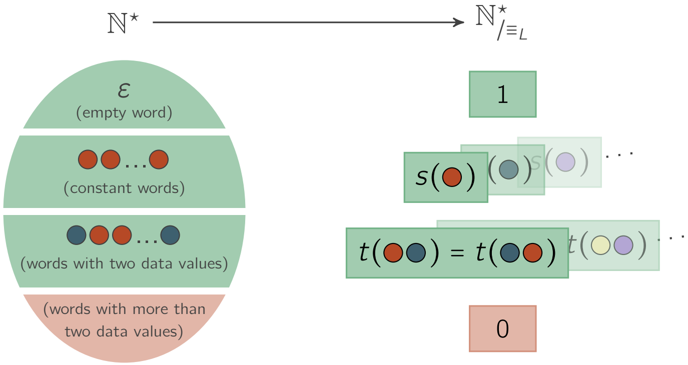
- **Orbit-finite Data Monoids**



Consider the Myhill-Nerode equivalence $\equiv_L$ for the data language
$L = \{w \in \mathbb{N}^\star : \textit{at most 2 distinct values in } w\}$



Consider the Myhill-Nerode equivalence $\equiv_L$ for the data language
$L = \{w \in \mathbb{N}^\star : \textit{at most 2 distinct values in } w\}$

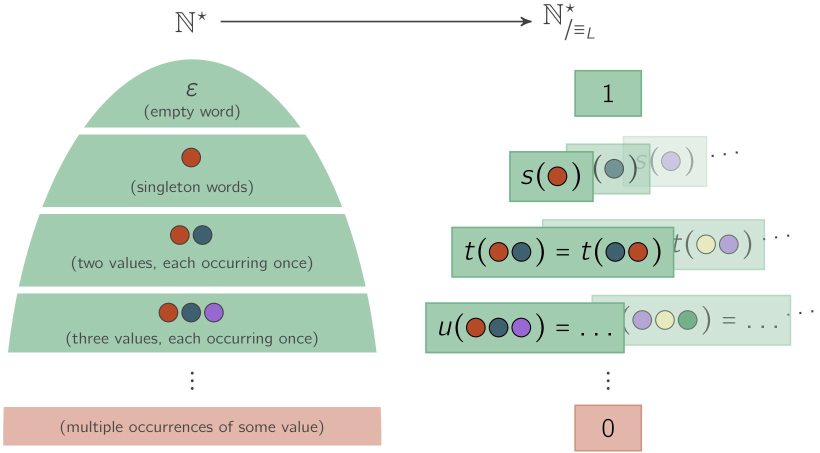


This gives a data monoid with products defined by equations like

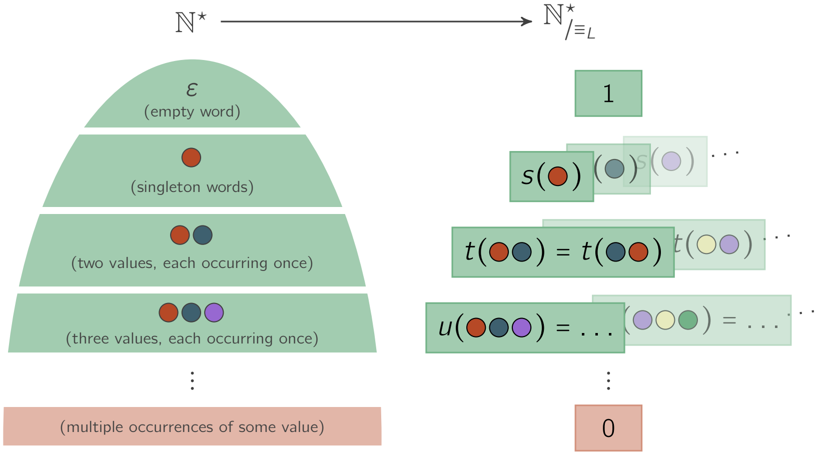
$$s({\color{red}\bullet}) \cdot s({\color{teal}\bullet}) = t({\color{red}\bullet}{\color{teal}\bullet}) \qquad t({\color{red}\bullet}{\color{teal}\bullet}) \cdot s({\color{violet}\bullet}) = 0 \qquad \ldots$$

☞ The data monoid is **orbit-finite** because
it has **finitely many elements up to renamings**

Consider the Myhill-Nerode equivalence $\equiv_L$ for the data language $L = \{w \in \mathbb{N}^\star : w \textit{ contains at most one occurrence of each value}\}$



Consider the Myhill-Nerode equivalence $\equiv_L$ for the data language
$L = \{w \in \mathbb{N}^\star : w \textit{ contains at most one occurrence of each value}\}$



☞ This syntactic data monoid is **not orbit-finite**
(unbounded "memory" → infinitely many orbits!)

Languages recognized by orbit-finite data monoids:

- ✓ Closed under all boolean operations
- ✓ *First value = last value*

- ✓ *At least two distinct values*

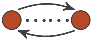
- ✓ *At least three distinct values*

- ✗ *First value reappears later*
- ✗ *Some value appears twice*
- ✗ *Every value appears at most once*

Languages recognized by orbit-finite data monoids:

✓ Closed under all boolean operations

✓ *First value = last value*



$\exists x, y.\ \ x \sim y \ \wedge \ x = \text{first} \ \wedge \ y = \text{last}$

✓ *At least two distinct values*

✓ *At least three distinct values*

✗ *First value reappears later*

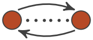
✗ *Some value appears twice*

✗ *Every value appears at most once*

Languages recognized by orbit-finite data monoids:

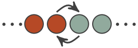
✓ Closed under all boolean operations

✓ *First value = last value*



$$\exists x, y.\ \ x \sim y \ \wedge\ x = \text{first} \ \wedge\ y = \text{last}$$

✓ *At least two distinct values*



$$\exists x, y.\ \ x \not\sim y \ \wedge\ y = x{+}1$$

✓ *At least three distinct values*

✗ *First value reappears later*

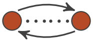
✗ *Some value appears twice*

✗ *Every value appears at most once*

Languages recognized by orbit-finite data monoids:

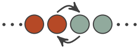
✓ Closed under all boolean operations

✓ *First value = last value*



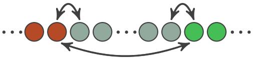
$$\exists x, y.\ \ x \sim y \ \wedge\ x = \text{first} \ \wedge\ y = \text{last}$$

✓ *At least two distinct values*



$$\exists x, y.\ \ x \nsim y \ \wedge\ y = x+1$$

✓ *At least three distinct values*



$$\exists x, y.\ \ x \nsim y \ \wedge\ (x \nsim x+1) \ \wedge\ (y-1 \nsim y) \ \wedge\ \nexists z \in [x+1, y-2].\ (z \nsim z+1)$$

✗ *First value reappears later*

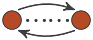
✗ *Some value appears twice*

✗ *Every value appears at most once*

Languages recognized by orbit-finite data monoids:

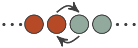
✓ Closed under all boolean operations

✓ *First value = last value*



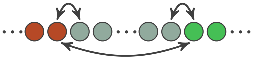
$\exists x, y.\ \ x \sim y \ \wedge\ x = \mathrm{first} \ \wedge\ y = \mathrm{last}$

✓ *At least two distinct values*



$\exists x, y.\ \ x \nsim y \ \wedge\ y = x{+}1$

✓ *At least three distinct values*



$\exists x, y.\ \ x \nsim y \ \wedge\ (x \nsim x{+}1) \wedge (y{-}1 \nsim y)$
$\wedge\ \nexists z \in [x{+}1, y{-}2].\ (z \nsim z{+}1)$

✗ *First value reappears later*

✗ *Some value appears twice*

✗ *Every value appears at most once*

All data tests are **guarded** by formulas defining **bijections**!

Definition

Rigidly guarded MSO$^{\sim}$ is the fragment of MSO with data tests defined by the following grammar:

$$\varphi \;\mapsto\; x < y \;\mid\; x \in Y \;\mid\; \neg\varphi \;\mid\; \varphi \wedge \varphi \;\mid\; \exists x.\, \varphi \;\mid\; \exists Y.\, \varphi \;\mid\; \boldsymbol{x \sim y \;\wedge\; \alpha(x,y)}$$

where $\alpha(x,y)$ is generated by the same grammar and is **rigid** i.e. in every word, it **determines** x **from** y **and vice versa**.

Definition

Rigidly guarded MSO$^{\sim}$ is the fragment of MSO with data tests defined by the following grammar:

$$\varphi \;\mapsto\; x < y \;\mid\; x \in Y \;\mid\; \neg\varphi \;\mid\; \varphi \wedge \varphi \;\mid\; \exists x.\, \varphi \;\mid\; \exists Y.\, \varphi \;\mid\; \boldsymbol{x \sim y \;\wedge\; \alpha(x,y)}$$

where $\alpha(x,y)$ is generated by the same grammar and is **rigid** i.e. in every word, it **determines** x **from** y **and vice versa**.

- Rigidity can be checked, or enforced syntactically like in
  $$\alpha^{\min}(x,y) \;=\; \alpha(x,y) \;\wedge\; \nexists x, y'.\, [x', y'] \subsetneq [x,y] \;\wedge\; \alpha(x', y')$$

- We can use shorthands like
  $$\begin{aligned} \alpha(x,y) \;\wedge\; x \nsim y &\;=\; \alpha(x,y) \;\wedge\; \neg\big(\alpha(x,y) \;\wedge\; x \sim y\big) \\ \alpha(x,y) \;\rightarrow\; x \sim y &\;=\; \neg\alpha(x,y) \;\vee\; \big(\alpha(x,y) \;\wedge\; x \sim y\big) \end{aligned}$$

Theorem 1

Rigidly guarded MSO~

=

Orbit-finite data monoids.

...and as in the Schützenberger-McNaughton-Papert's theorem:

Theorem 2

Rigidly guarded **FO~**

=

Aperiodic orbit-finite data monoids.

From logic to monoids: induction and closure properties

1. Negation: easy, by definition of recognizability
2. Conjunction: product of orbit-finite data monoids

From logic to monoids: induction and closure properties

1. Negation: easy, by definition of recognizability
2. Conjunction: product of orbit-finite data monoids
3. Existential quantification: **powerset construction**

   i.e. given $\underbrace{h : (\mathbb{N} \times \mathbb{B})^\star \to M}_{\text{for a formula } \varphi(X)}$, construct $\underbrace{h' : \mathbb{N}^\star \to \wp(M)}_{\text{for the quantified formula } \exists X.\, \varphi(X)}$:

   - elements of $\wp(M)$ are sets of elements of M
   - product is naturally defined by $S \cdot T = \{s \cdot t \mid s \in S,\ t \in T\}$

From logic to monoids: induction and closure properties

1. Negation: easy, by definition of recognizability
2. Conjunction: product of orbit-finite data monoids
3. Existential quantification: **powerset construction**

   i.e. given $\underbrace{h:\ (\mathbb{N}\times\mathbb{B})^\star \to M}_{\text{for a formula } \varphi(X)}$, construct $\underbrace{h':\ \mathbb{N}^\star \to \wp(M)}_{\text{for the quantified formula } \exists X.\,\varphi(X)}$:

   - elements of $\wp(M)$ are sets of **pairwise orbit-distinct** elements of M
   - product is naturally defined by $S \cdot T = \{s \cdot t \mid s \in S,\ t \in T\}$
   - **stronger invariant** (**projectability**) that forbids the following case

$$h(w, X) = s(\bullet) \qquad \textit{and} \qquad h(w, X') = s(\bullet)$$

From logic to monoids: induction and closure properties

1. Negation: easy, by definition of recognizability

2. Conjunction: product of orbit-finite data monoids

3. Existential quantification: **powerset construction**

   i.e. given $\underbrace{h: (\mathbb{N} \times \mathbb{B})^* \to M}_{\text{for a formula } \varphi(X)}$, construct $\underbrace{h': \mathbb{N}^* \to \wp(M)}_{\text{for the quantified formula } \exists X.\, \varphi(X)}$:

   - elements of $\wp(M)$ are sets of **pairwise orbit-distinct** elements of M
   - product is naturally defined by $S \cdot T = \{s \cdot t \mid s \in S,\ t \in T\}$
   - **stronger invariant** (**projectability**) that forbids the following case

   $$h(w, X) = s(\bullet) \quad \textit{and} \quad h(w, X') = s(\bullet)$$

4. Rigidly guarded data tests: product with **non-projectable** morphism

   i.e. given $\underbrace{h: (\mathbb{N} \times \mathbb{B} \times \mathbb{B})^* \to M}_{\text{for a rigid guard } \alpha(x,y)}$, construct $\underbrace{h': (\mathbb{N} \times \mathbb{B} \times \mathbb{B})^* \to M'}_{\text{for the data test } \alpha(x,y) \wedge x \sim y}$

From monoids to logic: generalization of Schützenberger's proof

Given a morphism $h : \mathbb{N}^\star \to M$, logically define the language $h^{-1}(s)$ by induction on the size of the **infix-closed set** $s^\uparrow = \{t \mid s \in M \cdot t \cdot M\}$.

From monoids to logic: generalization of Schützenberger's proof

Given a morphism $h : \mathbb{N}^\star \to M$, logically define the language $h^{-1}(s)$ by induction on the size of the **infix-closed set** $s^\uparrow = \{t \mid s \in M \cdot t \cdot M\}$.

Key ingredients in the classical aperiodic case:

1. $s \;=\; (s \cdot M) \;\cap\; (M \cdot s) \;\cap\; s^\uparrow$

From monoids to logic: generalization of Schützenberger's proof

Given a morphism $h : \mathbb{N}^\star \to M$, logically define the language $h^{-1}(s)$ by induction on the size of the **infix-closed set** $s^\uparrow = \{t \mid s \in M \cdot t \cdot M\}$.

Key ingredients in the classical aperiodic case:

1. $s \;=\; (s \cdot M) \;\cap\; (M \cdot s) \;\cap\; s^\uparrow$

2. $h(w) \in (s \cdot M) \;\leftrightarrow\; \exists$ *prefix* $u \cdot a$ *of* w *such that* $h(u) \cdot h(a) \in (s \cdot M)$

   (w.l.o.g. let u *be* ***maximal*** *such that* $h(u)^\uparrow \subsetneq s^\uparrow$*)*

From monoids to logic: generalization of Schützenberger's proof

Given a morphism $h : \mathbb{N}^\star \to M$, logically define the language $h^{-1}(s)$ by induction on the size of the **infix-closed set** $s^\uparrow = \{t \mid s \in M \cdot t \cdot M\}$.

Key ingredients in the classical aperiodic case:

1. $s \;=\; (s \cdot M) \;\cap\; (M \cdot s) \;\cap\; s^\uparrow$

2. $h(w) \in (s \cdot M) \;\leftrightarrow\; \exists$ *prefix* $u \cdot a$ *of* w *such that* $h(u) \cdot h(a) \in (s \cdot M)$

   (w.l.o.g. let u be **maximal** *such that $h(u)^\uparrow \subsetneq s^\uparrow$)*

3. $h(w) \in s^\uparrow \;\leftrightarrow\; \forall$ *infixes* $a \cdot u \cdot b$ *of* w, $h(a) \cdot h(u) \cdot h(b) \in s^\uparrow$

   (w.l.o.g. let u be **maximal** *such that $h(u)^\uparrow \subsetneq s^\uparrow$)*

From monoids to logic: generalization of Schützenberger's proof

Given a morphism $h : \mathbb{N}^\star \to M$, logically define the language $h^{-1}(s)$ by induction on the size of the **infix-closed set** $s^\uparrow = \{t \mid s \in M \cdot t \cdot M\}$.

Key ingredients in the classical aperiodic case:

1. $s \;=\; (s \cdot M) \;\cap\; (M \cdot s) \;\cap\; s^\uparrow$

2. $h(w) \in (s \cdot M) \;\leftrightarrow\; \exists$ *prefix* $u \cdot a$ *of* w *such that* $h(u) \cdot h(a) \in (s \cdot M)$

   (w.l.o.g. let u *be* **maximal** *such that* $h(u)^\uparrow \subsetneq s^\uparrow$*)*

3. $h(w) \in s^\uparrow \;\leftrightarrow\; \forall$ *infixes* $a \cdot u \cdot b$ *of* w, $h(a) \cdot h(u) \cdot h(b) \in s^\uparrow$

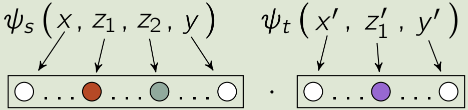
   (w.l.o.g. let u *be* **maximal** *such that* $h(u)^\uparrow \subsetneq s^\uparrow$*)*

Additional difficulties with data monoids:

- products depend on data values
- data tests must be performed under rigid guards
- "data groups" must be considered to keep track of data values

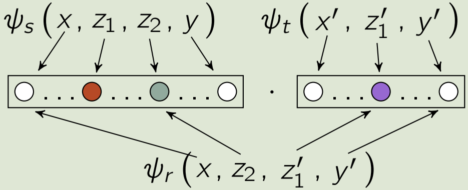
Example

Consider the product $s(\bigcirc\bigcirc) \cdot t(\bigcirc) = r(\bigcirc\bigcirc)$ of two elements



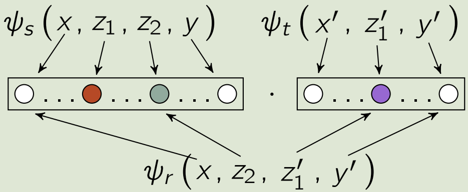
Example

Consider the product $s(\bullet\bullet) \cdot t(\bullet) = r(\bullet\bullet)$ of two elements



Example

Consider the product $s(\bullet\bullet) \cdot t(\bullet) = r(\bullet\bullet)$ of two elements



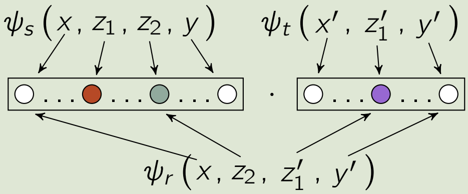
Stronger inductive hypothesis

Given a morphism $h : \mathbb{N}^\star \to M$ and an **orbit** $s(\bigcirc, \ldots, \bigcirc)$, one can construct the following objects by induction on $s^\uparrow$:

1. a formula $\varphi_s^\uparrow(x, y)$ that defines the infixes $w[x, y] \in h^{-1}(s^\uparrow)$

Example

Consider the product $s(\bullet\bullet) \cdot t(\bullet) = r(\bullet\bullet)$ of two elements



Stronger inductive hypothesis

Given a morphism $h : \mathbb{N}^\star \to M$ and an **orbit** $s(\circ, \ldots, \circ)$, one can construct the following objects by induction on $s^\uparrow$:

1. a formula $\varphi_s^\uparrow(x, y)$ that defines the infixes $w[x, y] \in h^{-1}(s^\uparrow)$
2. for each rigid guard $\alpha(x, y)$ that entails $\varphi_s^\uparrow(x, y)$, a rigid formula $\psi_s^\alpha(x, z_1, \ldots, z_k, y)$ such that

$$w \vDash \psi_s^\alpha\big(x, z_1, \ldots, z_k, y\big) \;\to\; h\big(w[x, y]\big) = s\big(w(z_1), \ldots, w(z_k)\big)$$

From monoids to logic: non-aperiodic case

1. Check that $h(w) \in (M \cdot s) \cap (s \cdot M) \cap s^{\uparrow}$

From monoids to logic: non-aperiodic case

1. Check that $h(w) \in (M \cdot s) \cap (s \cdot M) \cap s^{\uparrow}$

2. Factorize $w = \overbrace{u_1 \, a_1 \, \overleftrightarrow{v_1} \, b_1}^{w_1} \, \overbrace{u_2 \, a_2 \, \overleftrightarrow{v_2} \, b_2}^{w_2} \cdots$

   with $h(u_i)^{\uparrow} \not\subseteq s^{\uparrow}$ and v_i **maximal** such that $h(v_i)^{\uparrow} \not\subseteq s^{\uparrow}$

From monoids to logic: non-aperiodic case

1. Check that $h(w) \in (M \cdot s) \cap (s \cdot M) \cap s^{\uparrow}$
2. Factorize $w = \overbrace{u_1\ a_1 \leftarrow v_1 \rightarrow b_1}^{w_1}\ \overbrace{u_2\ a_2 \leftarrow v_2 \rightarrow b_2}^{w_2} \cdots$

   with $h(u_i)^{\uparrow} \subsetneq s^{\uparrow}$ and v_i **maximal** such that $h(v_i)^{\uparrow} \subsetneq s^{\uparrow}$
3. Guess elements $s_1, s_2, \ldots$ (over a bounded data domain) and using i.h. and products, check that each $\boldsymbol{s_i}$ **is a renaming of** $\boldsymbol{h(w_i)}$

From monoids to logic: non-aperiodic case

1. Check that $h(w) \in (M \cdot s) \;\cap\; (s \cdot M) \;\cap\; s^{\uparrow}$

2. Factorize $w = \overbrace{\boxed{u_1}\,\boxed{a_1}\,\boxed{\leftarrow\ v_1\ \rightarrow}\,\boxed{b_1}}^{w_1}\ \overbrace{\boxed{u_2}\,\boxed{a_2}\,\boxed{\leftarrow\ v_2\ \rightarrow}\,\boxed{b_2}}^{w_2} \cdots$

   with $h(u_i)^{\uparrow} \subsetneq s^{\uparrow}$ and v_i **maximal** such that $h(v_i)^{\uparrow} \subsetneq s^{\uparrow}$

3. Guess elements $s_1, s_2, \ldots$ (over a bounded data domain) and using i.h. and products, check that each s_i **is a renaming of** $h(w_i)$

4. Check that each pair s_i, s_{i+1} **is a renaming of** $h(w_i), h(w_{i+1})$ and, similarly, that s_1, s_n **is a renaming of** $h(w_1), h(w_n)$

From monoids to logic: non-aperiodic case

1. Check that $h(w) \in (M \cdot s) \cap (s \cdot M) \cap s^{\uparrow}$

2. Factorize $w = \overbrace{u_1 \; a_1 \; \overleftrightarrow{\;v_1\;} \; b_1}^{w_1} \; \overbrace{u_2 \; a_2 \; \overleftrightarrow{\;v_2\;} \; b_2}^{w_2} \cdots$

   with $h(u_i)^{\uparrow} \subsetneq s^{\uparrow}$ and v_i **maximal** such that $h(v_i)^{\uparrow} \subsetneq s^{\uparrow}$

3. Guess elements $s_1, s_2, \ldots$ (over a bounded data domain) and using i.h. and products, check that each $\boldsymbol{s_i}$ **is a renaming of** $\boldsymbol{h(w_i)}$

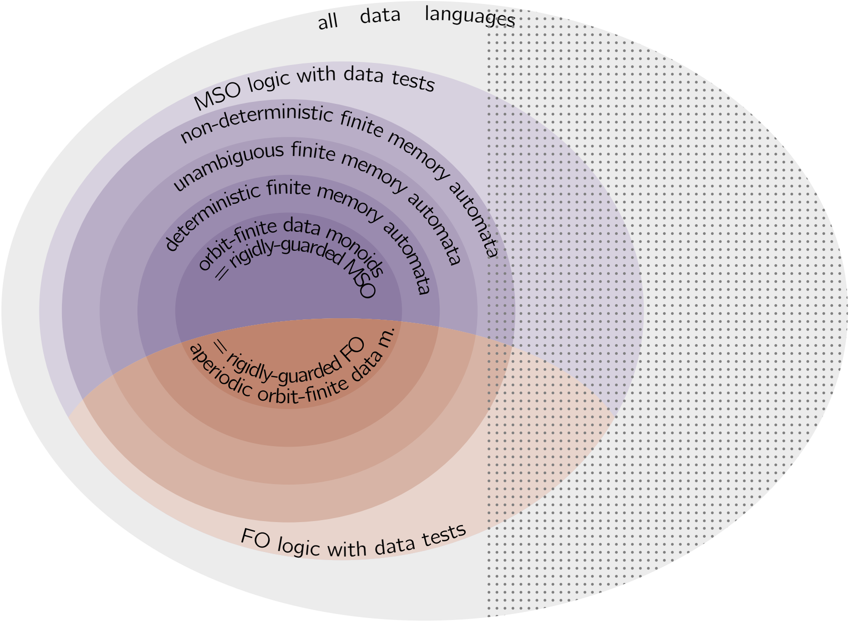
4. Check that each pair $\boldsymbol{s_i, s_{i+1}}$ **is a renaming of** $\boldsymbol{h(w_i), h(w_{i+1})}$ and, similarly, that $\boldsymbol{s_1, s_n}$ **is a renaming of** $\boldsymbol{h(w_1), h(w_n)}$

5. Inductively compute the partial products $s_1 \cdot \ldots \cdot s_i$, for $i = 1, \ldots, n$:

Lemma

$s_1 \cdot \ldots \cdot s_n$ is a renaming of $h(w) = h(w_1) \cdot \ldots \cdot h(w_n)$.

Logics, automata, and monoids over infinite alphabets:



Logics, automata, and monoids over infinite alphabets: