

# On the use of guards for logics with data

Thomas Colcombet, Clemens Ley and Gabriele Puppis

MFCS 2011

## What is this talk about?

Generalizations of classical results about **regular languages** from finite-alphabet case to **infinite-alphabet case**:

- correspondence between logics, automata, and monoids
- characterizations of first-order definability
- decidability of logics

Applications of languages over infinite alphabets (**data languages**):

- Databases: XML documents with text/attributes
- Verification: programs with variables over an infinite domain

A **data language** is a set of words/trees over a fixed infinite alphabet  $D$  (e.g.  $D = \{0, 1, 2, \dots\}$ ).

To make life easier, we enforce some restrictions:

- 1 Only finite words! (no infinite words, no finite/infinite trees)
- 2 Languages are invariant under permutations of data values  
(e.g.  $\boxed{1} \boxed{2} \boxed{1} \boxed{1} \boxed{3} \in L$  iff  $\boxed{5} \boxed{3} \boxed{5} \boxed{5} \boxed{7} \in L$  )

👉 we focus on **properties concerning equalities of data values**

### An example of data language

$$\begin{aligned} L &= \{w \in D^* : \text{at most 2 distinct values in } w\} \\ &= \{\varepsilon, \boxed{\color{red}\bullet}, \boxed{\color{blue}\bullet}, \boxed{\color{red}\bullet} \boxed{\color{blue}\bullet}, \boxed{\color{green}\bullet} \boxed{\color{magenta}\bullet}, \boxed{\color{blue}\bullet} \boxed{\color{red}\bullet} \boxed{\color{blue}\bullet}, \boxed{\color{green}\bullet} \boxed{\color{magenta}\bullet} \boxed{\color{magenta}\bullet} \boxed{\color{green}\bullet}, \dots\} \end{aligned}$$

## Languages over finite alphabets:

- **MSO logic**

$\exists X. \text{first} \in X$

$\wedge \text{last} \notin X$

$\wedge \forall y. (y \in X \leftrightarrow y + 1 \notin X)$

- **automata**

1 0 1 1 1 ... 0 1



- **finite monoids**

$s \cdot t = s$

$t \cdot s = t$

Languages over  
finite alphabets:

- **MSO logic**

$$\begin{aligned} \exists X. \text{first} \in X \\ \wedge \text{last} \notin X \\ \wedge \forall y. (y \in X \leftrightarrow y + 1 \notin X) \end{aligned}$$

- **automata**

1 0 1 1 1 ... 0 1



- **finite monoids**

$$s \cdot t = s$$

$$t \cdot s = t$$

Languages over  
infinite alphabets:

- **MSO logic with data tests**

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

- **register automata**

● ● ● ● ... ● ●

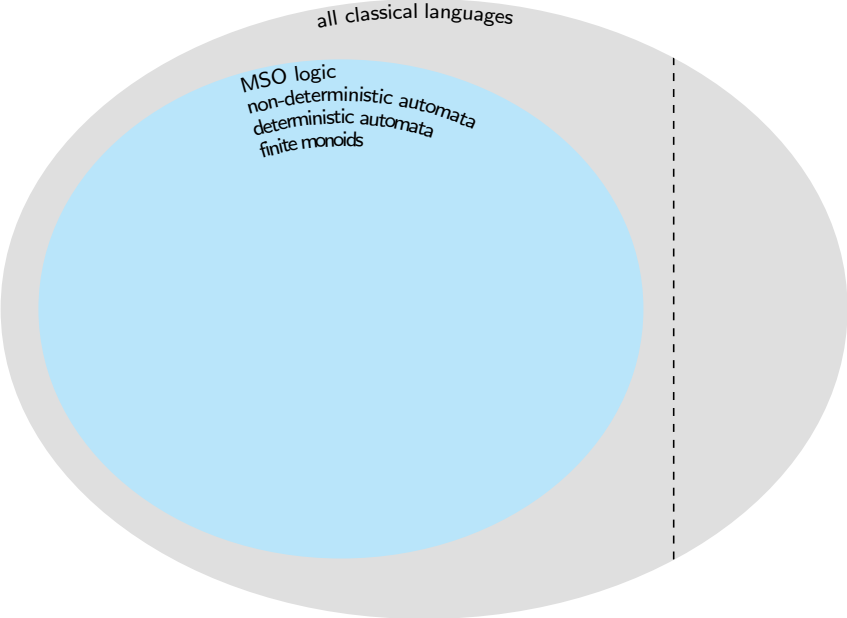


- **orbit finite data monoids**

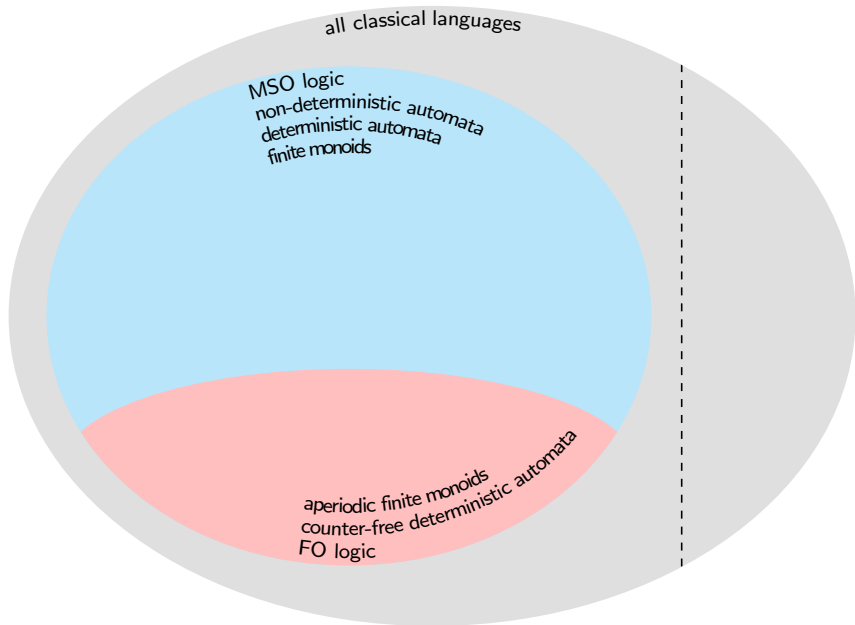
$$s(\text{red}, \text{blue}) \cdot t(\text{green}) = s(\text{blue}, \text{green})$$

$$t(\text{blue}) \cdot s(\text{blue}, \text{red}) = t(\text{red})$$

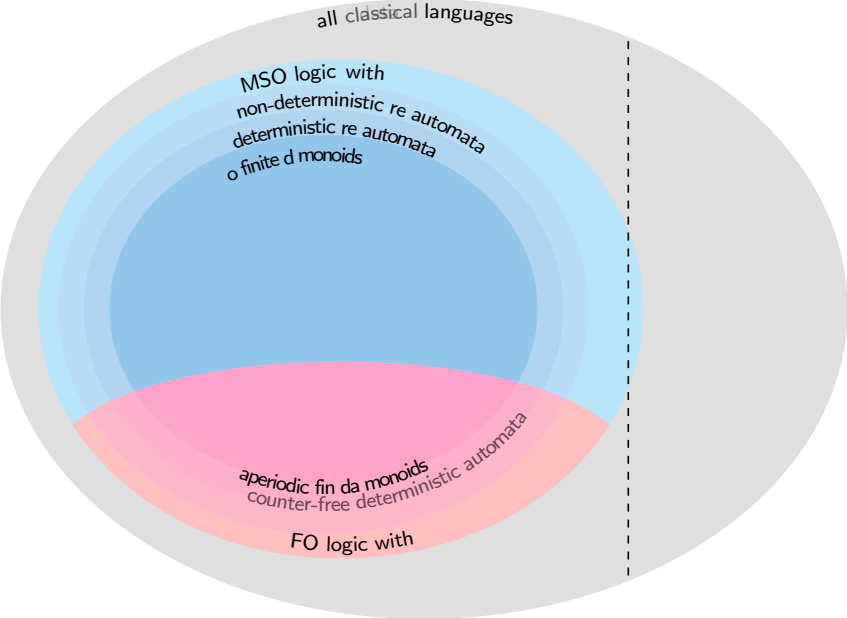
Expressiveness of logics, automata, monoids over **finite alphabets**:



Expressiveness of logics, automata, monoids over **finite alphabets**:

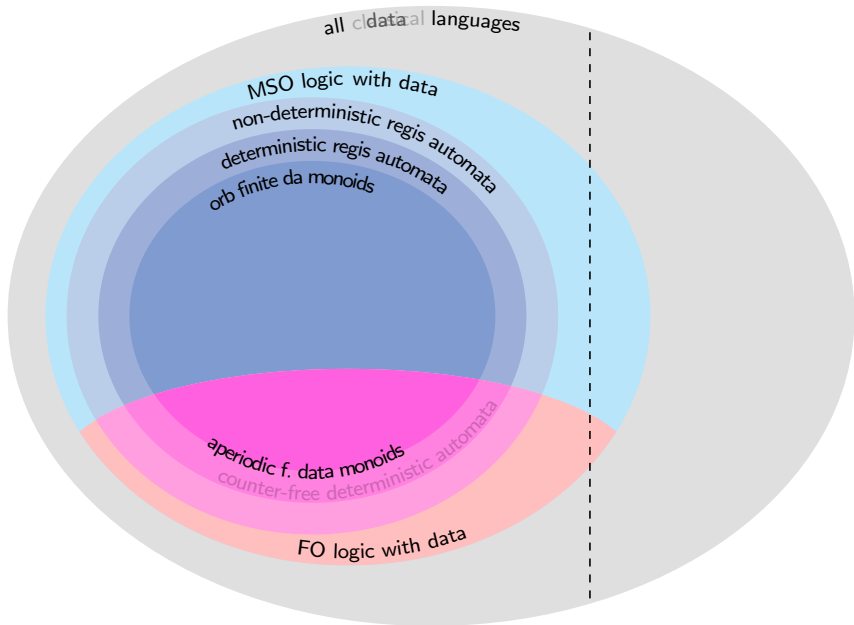


# Expressiveness of logics, automata, monoids over infinite alphabets:

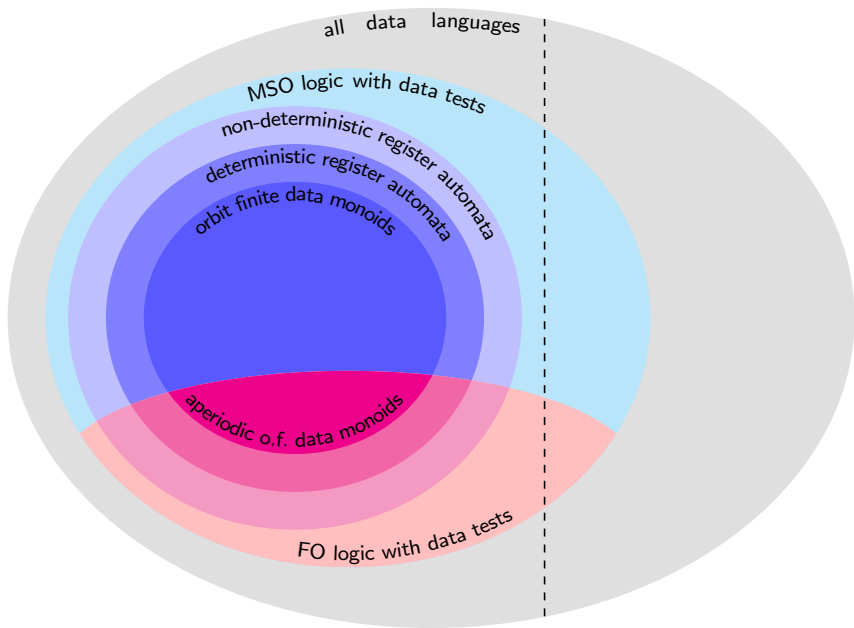




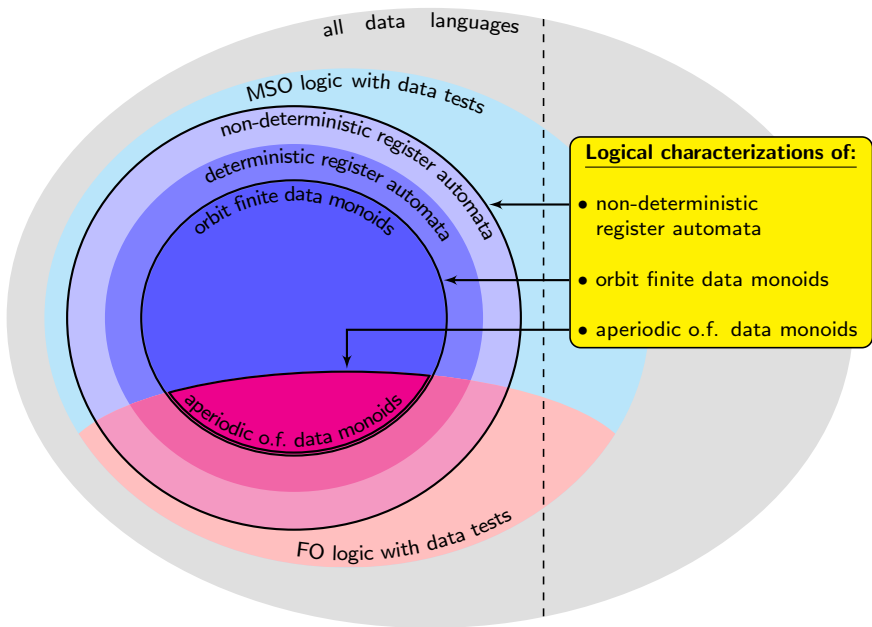
# Expressiveness of logics, automata, monoids over infinite alphabets:



# Expressiveness of logics, automata, monoids over infinite alphabets:

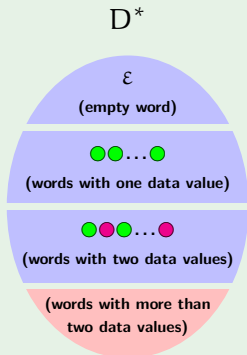


# Expressiveness of logics, automata, monoids over infinite alphabets:



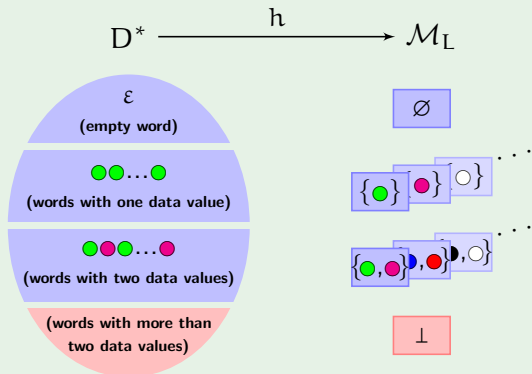
## What is an orbit finite data monoid?

Consider the syntactic monoid of the language  
 $L = \{w \in D^* : \text{at most 2 distinct values in } w\}$ :



## What is an orbit finite data monoid?

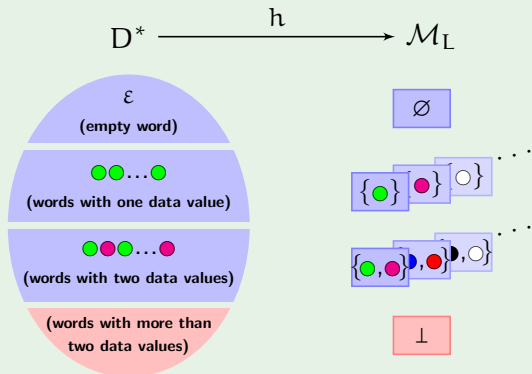
Consider the syntactic monoid of the language  
 $L = \{w \in D^* : \text{at most 2 distinct values in } w\}$ :



The product of  $\mathcal{M}_L$  is the union of sets, up to cardinality 2.

## What is an orbit finite data monoid?

Consider the syntactic monoid of the language  
 $L = \{w \in D^* : \text{at most 2 distinct values in } w\}$ :



The product of  $\mathcal{M}_L$  is the union of sets, up to cardinality 2.

👉 Each permutation  $\pi$  on  $D$  induces a permutation  $\hat{\pi}$  on  $\mathcal{M}_L$   
e.g., if  $\pi = \{\text{green} \leftrightarrow \text{blue}\}$ , then  $\hat{\pi}(\{\text{green}, \text{pink}\}) = \{\text{blue}, \text{pink}\}$ .

Examples of languages recognized by orbit finite data monoids:

- ✓ *Exactly two/three/... distinct values*
  - ✓ *Any two consecutive values are different*
  - ✓ *First value equals last value*
  - ✓ *"Lifting" by permutations of any classical regular language*
  - ✗ *First value reappears*
  - ✗ *Some value appears twice*
  - ✗ *All values appears at most once*
- ⊕ Closure under all boolean operations!

Consider some languages recognized by orbit finite data monoids:

- words where **first value equals last value**:

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

- words with **at least two distinct values** (e.g. ...●●...):

$$\exists x, y. (y = x + 1) \wedge (x \not\sim y)$$

...and some languages not recognized by orbit finite data monoids:

- words where **first value reappears**:

$$\exists x, y. (x = \text{first} \wedge x < y) \wedge (x \sim y)$$

- words where **all values appear at most once**:

$$\neg \exists x, y. (x < y) \wedge (x \sim y)$$



Consider some languages recognized by orbit finite data monoids:

- words where **first value equals last value**:

$$\exists x, y. (\mathbf{x = first} \wedge \mathbf{y = last}) \wedge (x \sim y)$$

- words with **at least two distinct values** (e.g. ...●●...):

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

...and some languages not recognized by orbit finite data monoids:

- words where **first value reappears**:

$$\exists x, y. (\mathbf{x = first} \wedge \mathbf{x < y}) \wedge (x \sim y)$$

- words where **all values appear at most once**:

$$\neg \exists x, y. (\mathbf{x < y}) \wedge (x \sim y)$$

Consider some languages recognized by orbit finite data monoids:

- words where **first value equals last value**:

$$\exists x, y. (\mathbf{x = first} \wedge \mathbf{y = last}) \wedge (x \sim y)$$

- words with **at least two distinct values** (e.g. ...●●...):

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

- words with **at least three distinct values** (e.g. ...●●●●...):

$$\exists x, y. ((\mathbf{x \not\sim x+1}) \wedge (\mathbf{y \not\sim y+1}) \wedge \forall z. (\mathbf{x < z < y} \rightarrow \mathbf{z \sim z+1})) \\ \wedge (x \not\sim y+1)$$

...and some languages not recognized by orbit finite data monoids:

- words where **first value reappears**:

$$\exists x, y. (\mathbf{x = first} \wedge \mathbf{x < y}) \wedge (x \sim y)$$

- words where **all values appear at most once**:

$$\neg \exists x, y. (\mathbf{x < y}) \wedge (x \sim y)$$

## Definition

**Rigidly guarded MSO** is the fragment of MSO with data tests, defined by the following grammar:

$$\varphi := x < y \mid x \in Y \mid \neg \varphi \mid \varphi \wedge \varphi \mid \exists x. \varphi \mid \exists Y. \varphi \mid \\ \varphi_{\text{rigid}}(x, y) \wedge x \sim y$$

where  $\varphi_{\text{rigid}}(x, y)$  is a rigid guard (generated by the same grammar)

( $\varphi(x, y)$  is **rigid** if, in every word,  $x$  determines  $y$  and vice versa).


## Definition

**Rigidly guarded MSO** is the fragment of MSO with data tests, defined by the following grammar:

$$\varphi := x < y \mid x \in Y \mid \neg \varphi \mid \varphi \wedge \varphi \mid \exists x. \varphi \mid \exists Y. \varphi \mid \\ \varphi_{\text{rigid}}(x, y) \wedge x \sim y$$


where  $\varphi_{\text{rigid}}(x, y)$  is a rigid guard (generated by the same grammar)

( $\varphi(x, y)$  is **rigid** if, in every word,  $x$  determines  $y$  and vice versa).

 Rigidity is a semantical restriction.

However, it can be enforced syntactically, e.g.,

$$\varphi_{\text{rigid}} = \varphi(x, y) \wedge (\forall x', y'. [x', y'] \not\subseteq [x, y] \rightarrow \neg \varphi(x', y'))$$

 Is  $\varphi_{\text{rigid}}(x, y) \wedge x \not\sim y$  needed?

$$\text{No: } \varphi_{\text{rigid}}(x, y) \wedge \neg(\varphi_{\text{rigid}}(x, y) \wedge x \sim y)$$

## Main theorem (1)

Languages defined in rigidly guarded MSO

||

Languages recognized by orbit finite data monoids.

### Main theorem (1)

Languages defined in rigidly guarded MSO

||

Languages recognized by orbit finite data monoids.

...and as in the [Schützenberger-McNaughton-Papert's](#) theorem:

### Main theorem (2)

Languages defined in rigidly guarded FO

||

Languages recognized by aperiodic orbit finite data monoids.

(A data monoid is **aperiodic** if all its sub-groups are trivial)

## Proof idea (rigidly guarded MSO $\rightarrow$ orbit finite data monoid)

By induction on formulas, using closure properties of data monoids:

- negation of a formula  $\Rightarrow$  easy, by definition of recognizability
- conjunction of formulas  $\Rightarrow$  product of orbit finite data monoids
- existential quantification  $\Rightarrow$  powerset of an orbit finite data monoid

## Proof idea (rigidly guarded MSO $\rightarrow$ orbit finite data monoid)

By induction on formulas, using closure properties of data monoids:

- negation of a formula  $\Rightarrow$  easy, by definition of recognizability
- conjunction of formulas  $\Rightarrow$  product of orbit finite data monoids
- existential quantification  $\Rightarrow$  powerset of an orbit finite data monoid

Given a formula

$\varphi(X)$

construct  $\exists X. \varphi(X)$



## Proof idea (rigidly guarded MSO $\rightarrow$ orbit finite data monoid)

By induction on formulas, using closure properties of data monoids:

- negation of a formula  $\Rightarrow$  easy, by definition of recognizability
- conjunction of formulas  $\Rightarrow$  product of orbit finite data monoids
- existential quantification  $\Rightarrow$  powerset of an orbit finite data monoid

Given a formula  $\varphi(X)$  construct  $\exists X. \varphi(X)$

Given a morphism  $h : (D \times \{0, 1\})^* \rightarrow \mathcal{M}$  construct  $h' : D^* \rightarrow 2^{\mathcal{M}}$   
where  $h'(w) = \{h(\langle w, X \rangle) : X \subseteq \text{dom}(w)\}$

## Proof idea (rigidly guarded MSO $\rightarrow$ orbit finite data monoid)

By induction on formulas, using closure properties of data monoids:

- negation of a formula  $\Rightarrow$  easy, by definition of recognizability
- conjunction of formulas  $\Rightarrow$  product of orbit finite data monoids
- existential quantification  $\Rightarrow$  powerset of an orbit finite data monoid

Given a formula  $\varphi(X)$  construct  $\exists X. \varphi(X)$

Given a morphism  $h : (D \times \{0, 1\})^* \rightarrow \mathcal{M}$  construct  $h' : D^* \rightarrow 2^{\mathcal{M}}$   
where  $h'(w) = \{h(\langle w, X \rangle) : X \subseteq \text{dom}(w)\}$

Given a monoid  $\mathcal{M} = (M, \cdot, \hat{\cdot})$  construct  $2^{\mathcal{M}} = (2^M, \odot, \hat{\cdot})$   
where  $S \odot T = \{s \cdot t : s \in S, t \in T\}$   
 $\hat{\pi}(S) = \{\hat{\pi}(s) : s \in S\}$

## Proof idea (rigidly guarded MSO $\rightarrow$ orbit finite data monoid)

By induction on formulas, using closure properties of data monoids:

- negation of a formula  $\Rightarrow$  easy, by definition of recognizability
- conjunction of formulas  $\Rightarrow$  product of orbit finite data monoids
- existential quantification  $\Rightarrow$  powerset of an orbit finite data monoid

Given a formula  $\varphi(X)$  construct  $\exists X. \varphi(X)$

Given a morphism  $h : (D \times \{0, 1\})^* \rightarrow \mathcal{M}$  construct  $h' : D^* \rightarrow 2^{\mathcal{M}}$   
where  $h'(w) = \{h(\langle w, X \rangle) : X \subseteq \text{dom}(w)\}$

Given a monoid  $\mathcal{M} = (M, \cdot, \hat{\cdot})$  construct  $2^{\mathcal{M}} = (2^M, \odot, \hat{\cdot})$   
where  $S \odot T = \{s \cdot t : s \in S, t \in T\}$   
 $\hat{\pi}(S) = \{\hat{\pi}(s) : s \in S\}$



Technical problem: this does not preserve orbit finiteness...

Proof idea (aperiodic o.f. data monoid  $\rightarrow$  rigidly guarded FO)

Follow the same induction as in the Schützenberger's proof:

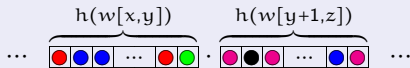
“ Given a morphism  $h : D^* \rightarrow \mathcal{M}$ ,  
construct formulas computing  $h(w[x, y])$   
for larger and larger infixes  $w[x, y]$  of words. ”

Proof idea (aperiodic o.f. data monoid  $\rightarrow$  rigidly guarded FO)

Follow the same induction as in the Schützenberger's proof:

“ Given a morphism  $h: D^* \rightarrow \mathcal{M}$ ,  
construct formulas computing  $h(w[x, y])$   
for larger and larger infixes  $w[x, y]$  of words. ”

👉 Technical problem: in order to let the induction go through,  
we need to simulate products of the monoid with formulas...

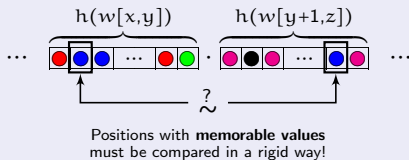


## Proof idea (aperiodic o.f. data monoid $\rightarrow$ rigidly guarded FO)

Follow the same induction as in the Schützenberger's proof:

“ Given a morphism  $h: D^* \rightarrow \mathcal{M}$ ,  
construct formulas computing  $h(w[x, y])$   
for larger and larger infixes  $w[x, y]$  of words. ”

👉 Technical problem: in order to let the induction go through,  
we need to simulate products of the monoid with formulas...

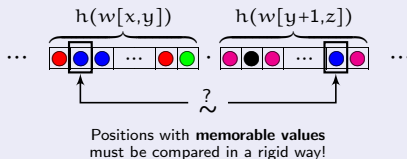


Proof idea (aperiodic o.f. data monoid  $\rightarrow$  rigidly guarded ~~FO~~/MSO)

Follow the same induction as in the Schützenberger's proof:

“ Given a morphism  $h: D^* \rightarrow \mathcal{M}$ ,  
construct formulas computing  $h(w[x, y])$   
for larger and larger infixes  $w[x, y]$  of words. ”

👉 Technical problem: in order to let the induction go through,  
we need to simulate products of the monoid with formulas...



If we drop the assumption of aperiodicity,  
we need MSO formulas to compute elements of the monoid.

👉 Unlike in the classical case, we cannot simulate runs of automata  
(instead, we need to further generalize Schützenberger's proof).

We also considered a relaxation of the rigidity constraints:

### Definition

**Semi-rigidly guarded MSO** is defined by the grammar

$$\psi := \exists Z_1, \dots, Z_k. \varphi(Z_1, \dots, Z_k)$$

$$\begin{aligned} \varphi(Z_1, \dots, Z_k) := & x < y \mid x \in Y \mid x \in Z_i \mid \\ & \neg \varphi \mid \varphi \wedge \varphi \mid \exists x. \varphi \mid \exists Y. \varphi \mid \\ & \varphi_{\text{semi-rigid}}(Z_1, \dots, Z_k, x, y) \wedge x \sim y \end{aligned}$$

where  $\varphi_{\text{semi-rigid}}(Z_1, \dots, Z_k, x, y)$  determines  $y$  from  $Z_1, \dots, Z_k, x$ .

### Example

The formula below defines the language of all words where some value reappears at the last even position:

$$\begin{aligned} \psi = & \exists Z. \forall z. (z \in Z \leftrightarrow \text{Even}(z)) \\ & \wedge \exists x, y. (x < y \wedge y = \text{last}(Z)) \wedge x \sim y. \end{aligned}$$



### Theorem (3)

Languages of data words defined in semi-rigidly guarded MSO

||

Languages recognized by non-deterministic register word automata.

### Theorem (3)

trees

Languages of data ~~words~~ defined in semi-rigidly guarded MSO

||

Languages recognized by non-deterministic register ~~word~~ automata.

tree

### Theorem (3)

trees

Languages of data ~~words~~ defined in semi-rigidly guarded MSO

||

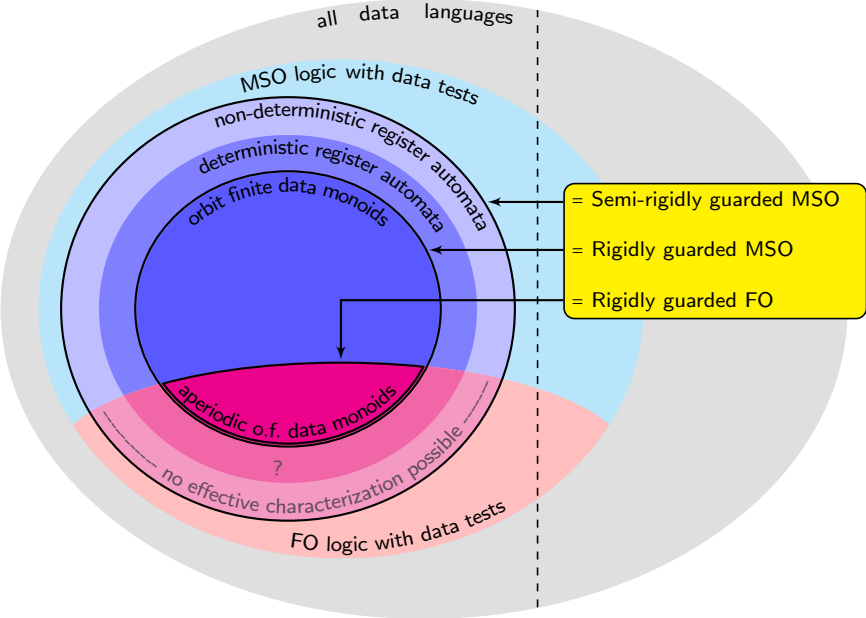
Languages recognized by non-deterministic register ~~word~~ automata.

tree

### Corollary

Satisfiability of semi-rigidly guarded MSO is decidable.

Back to our picture...

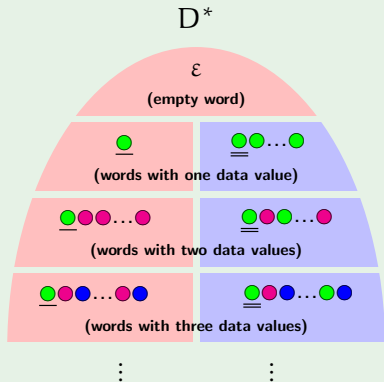




## A data monoid with infinitely many orbits

Consider the syntactic monoid of the language

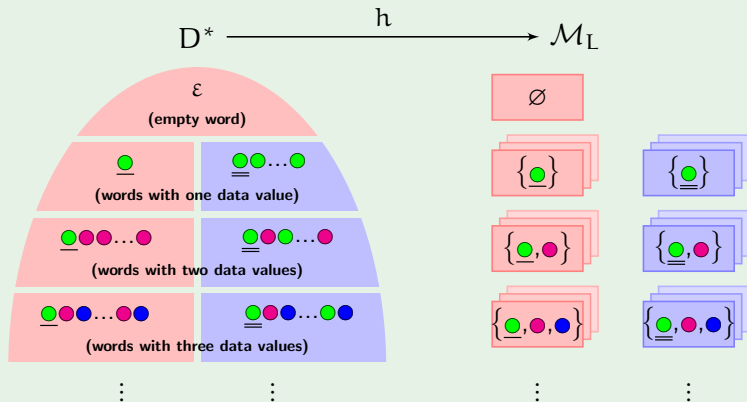
$L = \{w \in D^* : \text{first value of } w \text{ reappears}\}$ :



## A data monoid with infinitely many orbits

Consider the syntactic monoid of the language

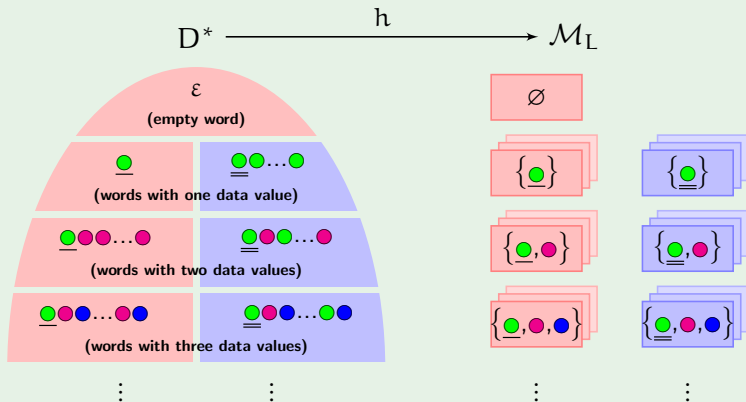
$L = \{w \in D^* : \text{first value of } w \text{ reappears}\}$ :



## A data monoid with infinitely many orbits

Consider the syntactic monoid of the language

$$L = \{w \in D^* : \text{first value of } w \text{ reappears}\}:$$



$L$  cannot be recognized with finitely many orbits

(but it is recognized by a deterministic register automaton).