

# Algebraic Data Language Theory

Thomas Colcombet, Clemens Ley, Gabriele Puppis

# Algebraic Language Theory

Thomas Colcombet, Clemens Ley, Gabriele Puppis

# Data Language

Thomas Colcombet, Clemens Ley, Gabriele Puppis

# Algebraic Data Language Theory

Thomas Colcombet, Clemens Ley, Gabriele Puppis

# Overview

Algebraic Language Theory

Algebraic Data Language Theory

# Recall

A language is regular iff

it is defined by a regular expression,

it is accepted by a finite automaton,

it has a Myhill-Nerode equivalence of finite index,

it is defined by monadic second order logic,

it is recognized by a finite monoid.

# Monoids

A monoid  $(M, \cdot, \mathbf{1})$  consists of

- ♦ a set  $M$ ,
- ♦ an associative operation  $\cdot$  on  $M$ , i.e.  $(x \cdot y) \cdot z = x \cdot (y \cdot z)$  for all  $x, y, z \in M$ ,
- ♦ an identity element  $\mathbf{1}$  of  $M$ , i.e.  $\mathbf{1} \cdot x = x \cdot \mathbf{1} = x$  for all  $x \in M$ .

Example: The ‘free monoid’ over an alphabet  $\Sigma$  is a monoid  $(\Sigma^*, \circ, \mathbf{1})$  where

- ♦  $\Sigma^*$  is the set of all finite words over  $\Sigma$ ,
- ♦ the product is string concatenation (i.e.  $u \circ v = uv$ ),
- ♦ the empty word  $\varepsilon$  is the identity  $\mathbf{1}$ .

# The Syntactic Monoid

Def. The two-sided Myhill-Nerode equivalence  $\equiv_L$  of language  $L$ :

$$w \equiv_L w' \text{ iff for all words } u, v: uwv \in L \leftrightarrow uw'v \in L.$$

Example:  $\equiv_L$ -classes of  $ba^*$ :  $a^*$ ,  $ba^*$ ,  $\perp$

The ‘syntactic monoid’ of a language  $L$  is the monoid  $(M_L, \cdot, I)$  where

- ♦  $M_L$  is the set of equivalence classes of  $\equiv_L$
- ♦ product is defined by  $\llbracket u \rrbracket_{\equiv_L} \cdot \llbracket v \rrbracket_{\equiv_L} = \llbracket uv \rrbracket_{\equiv_L}$
- ♦  $\llbracket \varepsilon \rrbracket_{\equiv_L}$  is the identity  $I$

# Acceptance by a Monoid

Def. A homomorphism  $h$  from  $(N, \circ, e)$  to  $(M, \cdot, \mathbf{1})$  is a mapping from  $N$  to  $M$  such that  $h(x \circ y) = h(x) \cdot h(y)$  and  $h(e) = \mathbf{1}$ .

Def. A language  $L \subseteq \Sigma^*$  is accepted by a monoid  $(M, \cdot, \mathbf{1})$  if there is a homomorphism  $h$  from the free monoid  $\Sigma^*$  to  $M$  and a subset  $F$  of  $M$  such that:  $w \in L$  iff  $h(w) \in F$ .

Thm. A language  $L \subseteq \Sigma^*$  is regular iff  $L$  is accepted by a finite monoid.

# Algebraic Language Theory

Which subsets of regular languages are accepted  
by which subsets of finite monoids?

# Schützenberger et al.

Def. A language is star-free if it can be defined by a regular expression without Kleene-star.

Def. A monoid is aperiodic if  $x^n = x^{n+1}$  for all monoid elements  $x$  and “sufficiently large”  $n$ .

# Schützenberger et al.

Def. A language is star-free if it can be defined by a regular expression without Kleene-star.

Def. A monoid is aperiodic if  $x^n = x^{n+1}$  for all monoid elements  $x$  and “sufficiently large”  $n$ .

Thm [[Schützenberger, 1965; McNaughton, Papert, 1971]]. Let  $L$  be a language.

Then

- ♦  $L$  is accepted by an aperiodic finite monoid iff
- ♦  $L$  is star-free iff
- ♦  $L$  can be defined in first order logic.

# Other results

regular expressions	$\text{MSO}(+I)$	finite monoids
star-free expressions	$\text{FO}(<)$	aperiodic finite monoids
turtle expressions	$\text{FO}^2(<)$	DA
boolean combinations of local expressions	$\text{FO}(+I)$	V
...	...	...

# Question

Which of these results extend to languages over an infinite alphabet?

# Overview

Algebraic Language Theory

Algebraic Data Language Theory

# Data Languages

A *data language* is a language over an infinite alphabet  $D$  that is closed under bijections on  $D$  (that is,  $d_1 \dots d_n \in L$  iff  $\pi(d_1) \dots \pi(d_n) \in L$  for all bijections  $\pi$  on  $D$ ).

*Examples:*

- ▶ There is only one data value.
- ▶ There are exactly  $k$  distinct data values.
- ▶ No two consecutive data values are equal.
- ▶ All the data values are pairwise distinct.
- ▶ The first and last data values are equal.
- ▶ Isomorphism closure of a language over finite alphabet

# Which Language Class?

# Which Language Class?

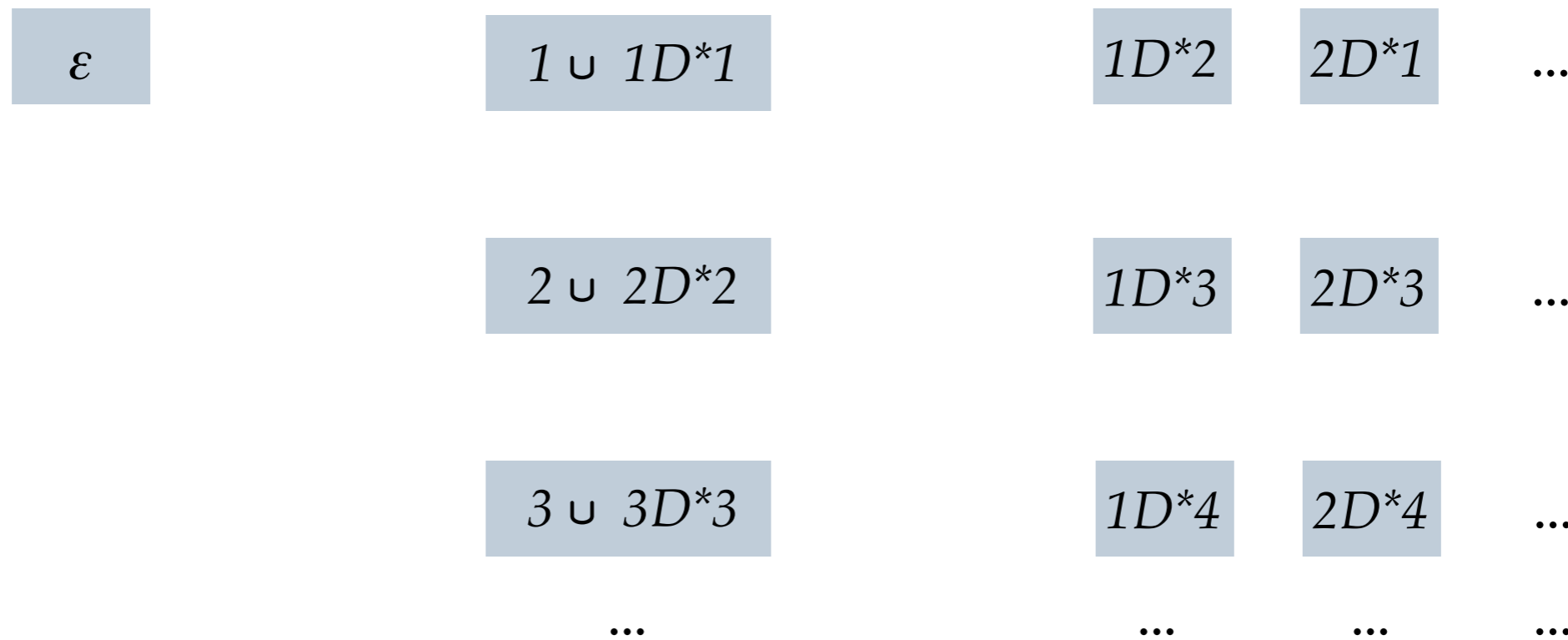
- A data language  $L$  is accepted by a finite monoid iff membership in  $L$  depends only on the length of the word.

# Which Language Class?

- A data language  $L$  is accepted by a finite monoid iff membership in  $L$  depends only on the length of the word.
- Every language is accepted by an infinite monoid.

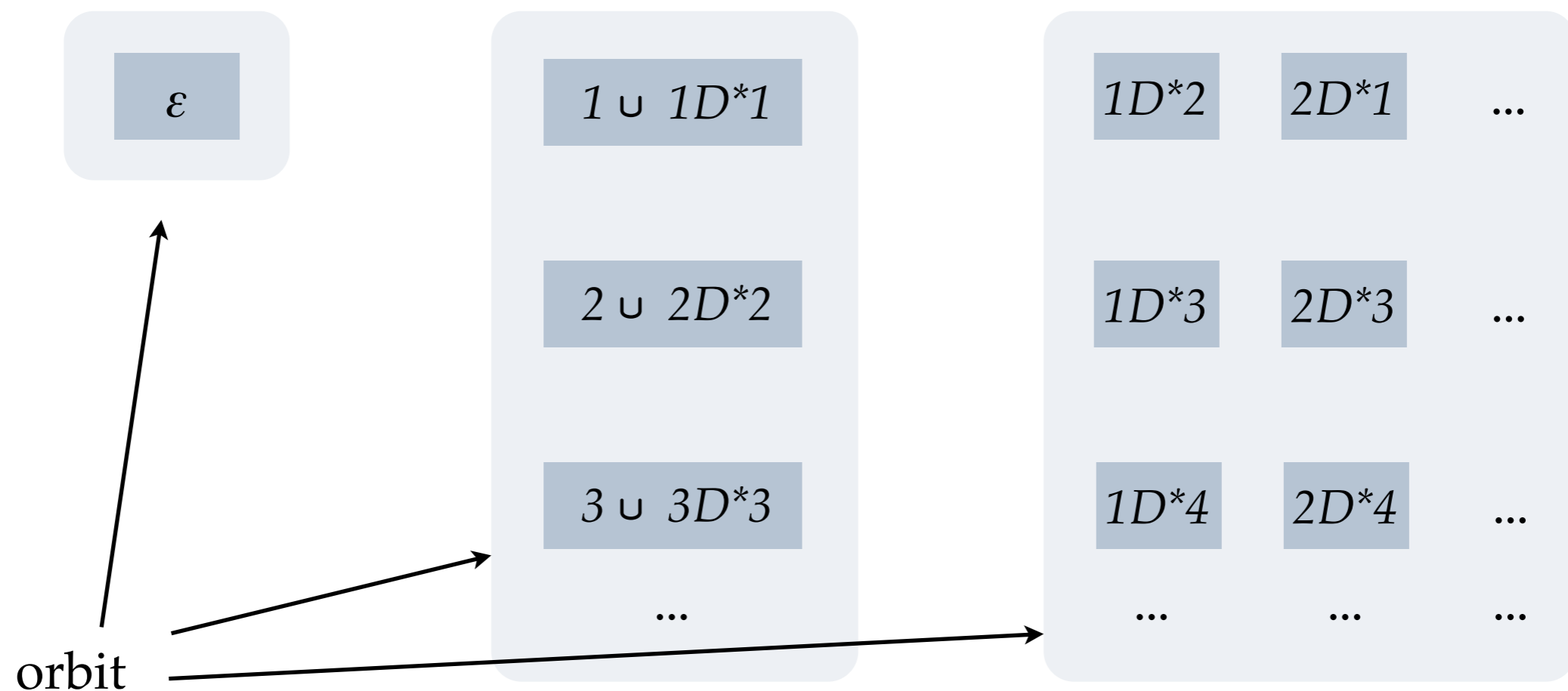
# Which Language Class?

- ▶ A data language  $L$  is accepted by a finite monoid iff membership in  $L$  depends only on the length of the word.
- ▶ Every language is accepted by an infinite monoid.
- ▶ Consider the syntactic monoid of the language  $L = \{d_1 \dots d_n : d_1 = d_n\}$ :



# Which Language Class?

- ▶ A data language  $L$  is accepted by a finite monoid iff membership in  $L$  depends only on the length of the word.
- ▶ Every language is accepted by an infinite monoid.
- ▶ Consider the syntactic monoid of the language  $L = \{d_1 \dots d_n : d_1 = d_n\}$ :



# Group Actions

A *group* is a structure  $(M, +, 1)$  with addition, identity, and inverse as usual.

A *group-action* of a group  $G$  on a set  $X$  maps every  $g \in G$  to an function  $[g]$  on  $X$  such that

- $[g \cdot h](x) = [g]([h](x))$  for all  $g, h \in G$  and  $x \in X$
- $[e](x) = x$  for the identity  $e$  of  $G$

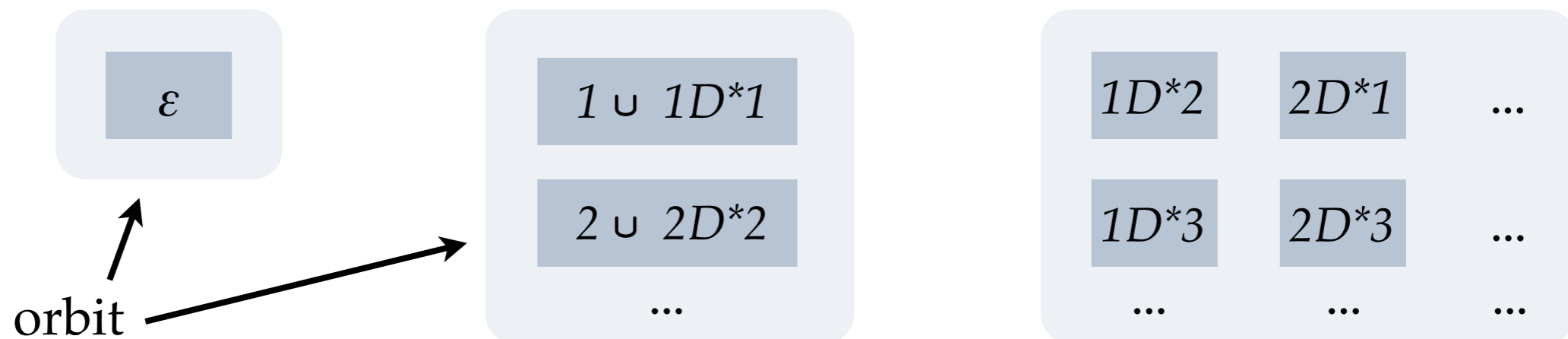
*Example:* Let  $\Pi$  be the group of bijections on  $D$  and let  $D^*$  be the set of finite strings over  $D$ . Then  $\Pi$  acts on  $D^*$  by  $\pi \rightarrow [\pi]$  where  $[\pi](w) = \pi(w)$ .

# Data Monoids

A *data-monoid* is a monoid  $(M, \cdot, 1)$  that is acted upon by the group of bijections  $\Pi$  on  $D$ , that is

- $[\pi \cdot \tau](x) = [\pi]([\tau](x))$  for all  $\pi, \tau \in \Pi$  and  $x \in M$
- $[id](x) = x$  for the identity renaming  $id$  of  $\Pi$
- $[\pi](x \cdot y) = [\pi](x) \cdot [\pi](y)$
- $[\pi](1) = 1$

The *orbit* of  $x \in M$  is  $\{[\pi](x) : \pi \in \Pi\}$ .



# Orbit Finite Data Monoids

# Orbit Finite Data Monoids

Straightforward definitions for:

- free data monoid
- syntactic data monoid
- homomorphisms between data monoids
- acceptance by a data monoid

# Orbit Finite Data Monoids

Straightforward definitions for:

- free data monoid
- syntactic data monoid
- homomorphisms between data monoids
- acceptance by a data monoid

**We study finite-orbit data monoids**

(that are homomorphic images of the free data monoid).

# Orbit Finite Data Monoids

Straightforward definitions for:

- free data monoid
- syntactic data monoid
- homomorphisms between data monoids
- acceptance by a data monoid

**We study finite-orbit data monoids**

(that are homomorphic images of the free data monoid).

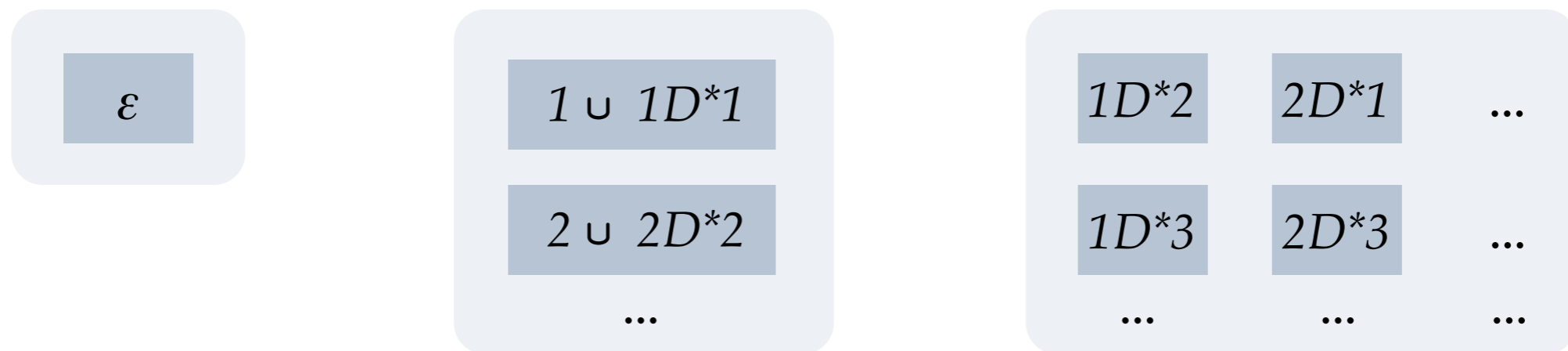
Languages accepted by finite-orbit data monoids:

- No two consecutive data values are equal. ✓
- There are exactly  $k$  distinct data values. ✓
- The first and last data values are equal. ✓
- Isomorphism closures of regular languages. ✓
- All the data values are pairwise distinct. ✗

# Memory

A set  $C \subseteq D$  is a *memory* of  $x \in M$  if for all permutations  $\pi$  on  $D$ ,  
 $\pi(d)=d$  for all  $d \in C$  then  $[\pi](x)=x$ .

*Proposition: Every element  $x$  of an finite-orbit data monoid has a minimal memory.*



If  $x$  and  $y$  are in the same orbit then their memory has the same size.

Thus, 'all values are different' is not accepted by an orbit finite data monoid.

# Monoids vs. Logics

Recall: Schützenberger et. al. (over finite alphabets)

finite monoids = MSO

aperiodic finite monoids = FO

# Monoids vs. Logics

Recall: Schützenberger et. al. (over finite alphabets)

finite monoids = MSO

aperiodic finite monoids = FO

Over infinite alphabets:

finite-orbit data monoids

aperiodic orbit-finite data monoids

MSO

FO

all values  
are different



# Rigidly-guarded Logics

Syntax of *rigidly-guarded MSO (RMSO)*:

$$\varphi := \varphi_{\text{rigid}}(x,y) \wedge x \sim y \mid x = y+1 \mid x \in X \mid$$

$$\neg\varphi \mid \varphi \wedge \varphi \mid \exists X . \varphi \mid \exists x . \varphi$$

where  $\varphi_{\text{rigid}} \in \text{RMSO}$  is rigid. Here, a formula  $\varphi(x,y)$  is *rigid* if it defines a bijection.

*Example:* Consecutive positions have a different value:

$$\forall x \forall y . x = y+1 \rightarrow x \neq y$$

# Theorem

Theorem[Colcombet, L., Puppis]. *Let  $L$  be a data language. Then*

- *$L$  can be defined in RMSO iff  $L$  is accepted by a finite-orbit data monoid.*
- *$L$  can be defined in RFO iff  $L$  is accepted by an aperiodic finite-orbit data monoid.*

Corollary: *RMSO is decidable.*

orbit-finite data monoids = RMSO

aperiodic orbit-finite data monoids = RFO

# From Logic to Monoids

Lemma (Inductive Invariant): *For every RMSO formula  $\varphi(X_1, \dots, X_n)$  the language  $L(\varphi(X_1, \dots, X_n))$  is recognized by an orbit finite data monoid.*

*Proof.*

- By structural induction on the formula.
- To maintain the invariant we need to show a stronger statement:  
`... recognized by a finite-orbit data monoid via a projectable morphism.'
- Special construction for existential quantifier and for guarded quantification
- Our construction is effective
- Proof is 6 pages

# From Monoids to Logic

Lemma (Inductive Invariant). *Let  $h: D^* \rightarrow M$  be a homomorphism recognizing  $L$ . For each orbit  $o$ , word  $w$ , and (almost) each infix  $w[i,j]$  there is a formula  $\phi_o(x,y)$  s.t.*

$$w \models \phi_o(i,j) \text{ iff } h(w[i,j]) \in o.$$

*In addition, the memorable values of  $w[i,j]$  can be defined in a rigid way.*

# From Monoids to Logic

Lemma (Inductive Invariant). *Let  $h: D^* \rightarrow M$  be a homomorphism recognizing  $L$ . For each orbit  $o$ , word  $w$ , and (almost) each infix  $w[i,j]$  there is a formula  $\phi_o(x,y)$  s.t.*

$$w \models \phi_o(i,j) \text{ iff } h(w[i,j]) \in o.$$

*In addition, the memorable values of  $w[i,j]$  can be defined in a rigid way.*

Definition [Some of Green's Relations]. *Let  $M$  be a monoid. Then for all  $x,y \in M$*

$$x \leq_{\mathcal{J}} y \text{ iff there are } z, z' \in M \text{ such that } x = zyz'$$

$$x =_{\mathcal{J}} y \text{ iff } x \leq_{\mathcal{J}} y \text{ and } y \leq_{\mathcal{J}} x$$

*Proof of the Lemma.*

- Induction on  $\leq_{\mathcal{J}}$ : Start with 'short' words, move to 'long' ones.
- Induction step: Orbit of an infix can be determined by multiplying 'shorter' infixes.
- Product  $h(u) \cdot h(v)$  depends only on orbits of  $h(u)$ ,  $h(v)$  and their memorable values.
- Proof is 13 pages.

# From Monoids to Logic

Lemma (Inductive Invariant). *Let  $h: D^* \rightarrow M$  be a homomorphism recognizing  $L$ . For each orbit  $o$ , word  $w$ , and (almost) each infix  $w[i,j]$  there is a formula  $\phi_o(x,y)$  s.t.*

$$w \models \phi_o(i,j) \text{ iff } h(w[i,j]) \in o.$$

*In addition, the memorable values of  $w[i,j]$  can be defined in a rigid way.*

Def.  $\mathcal{R}$ -decomposition of  $d_1 \dots d_{21}$ :

$\mathcal{J}_9: d_1 d_2 d_3 d_4 d_5$

$\mathcal{J}_5: d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9 d_{10} d_{11}$

$\mathcal{J}_3: d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_8 d_9 d_{10} d_{11} d_{12} d_{13} d_{14} d_{15} d_{16} d_{17} d_{18} d_{19} d_{20} d_{21}$

Lemma: *Every memorable value is on a step of an  $\mathcal{R}$ - or  $\mathcal{L}$ -decomposition.*

# Conclusion

Theorem[Colcombet, L., Puppis]. *Let  $L$  be a data language. Then*

- *$L$  can be defined in RMSO iff  $L$  is accepted by a finite-orbit data monoid.*
- *$L$  can be defined in RFO iff  $L$  is accepted by an aperiodic finite-orbit data monoid.*

Next steps:

- Characterize fragments of FO.
- Extend algebraic approach to a bigger class of data languages.
- Investigate relationship with Freeze-LTL, 1ARA, ...

# Conclusion

Theorem[Colcombet, L., Puppis]. *Let  $L$  be a data language. Then*

- *$L$  can be defined in RMSO iff  $L$  is accepted by a finite-orbit data monoid.*
- *$L$  can be defined in RFO iff  $L$  is accepted by an aperiodic finite-orbit data monoid.*

Next steps:

- Characterize fragments of FO.
- Extend algebraic approach to a bigger class of data languages.
- Investigate relationship with Freeze-LTL, 1ARA, ...

# Thank You

Theorem[Colcombet, L., Puppis]. *Let  $L$  be a data language. Then*

- *$L$  can be defined in RMSO iff  $L$  is accepted by a finite-orbit data monoid.*
- *$L$  can be defined in RFO iff  $L$  is accepted by an aperiodic finite-orbit data monoid.*

Next steps:

- Characterize fragments of FO.
- Extend algebraic approach to a bigger class of data languages.
- Investigate relationship with Freeze-LTL, 1ARA, ...

# Appendix

# Register Automata

A register automaton is an automaton with a finite set of registers that store values.

On each transition,

- the automaton checks which registers contain the input symbol,
- it updates some registers with the input symbol,
- it deletes some registers.