LANGUAGES OF COUNTABLE WORDS

Lorem ipsum (Don't read this! Better listen to the talk) [illegible] Still reading?) ullam corporis suscipit laboriosam, nisi...

Gabriele Puppis

LaBRI / CNRS

based on joint works with

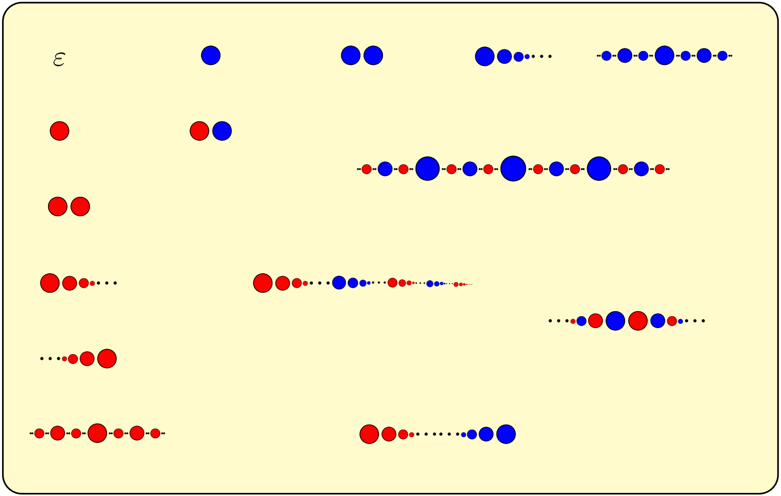
Olivier Carton,
Thomas Colcombet

Given an alphabet $A = \{{\color{red}\bullet}, {\color{blue}\bullet}\}$, let

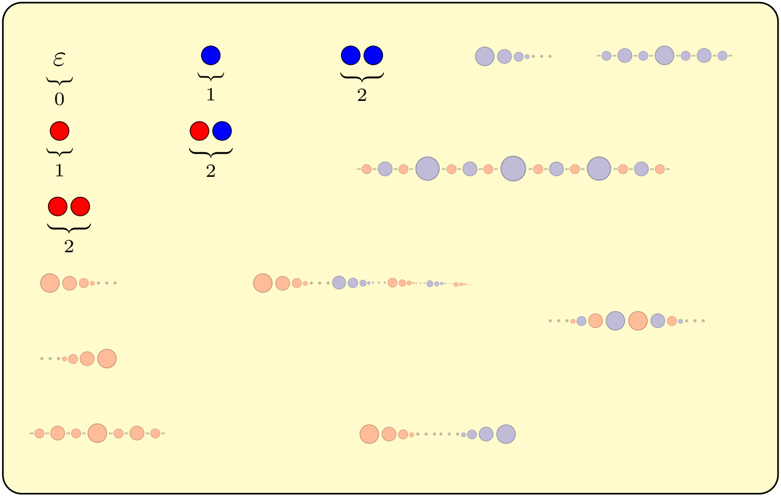
A° = $\Big\{$all **countable words** on $A\Big\}$

Given an alphabet $A = \{\ ,\ \}$, let

$A^\circ =$

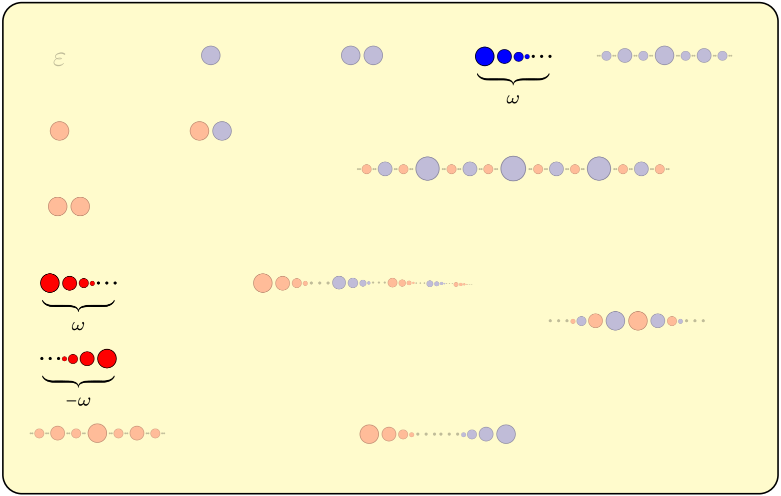


Given an alphabet $A = \{$●, ●$\}$, let

$A^\circ =$ 

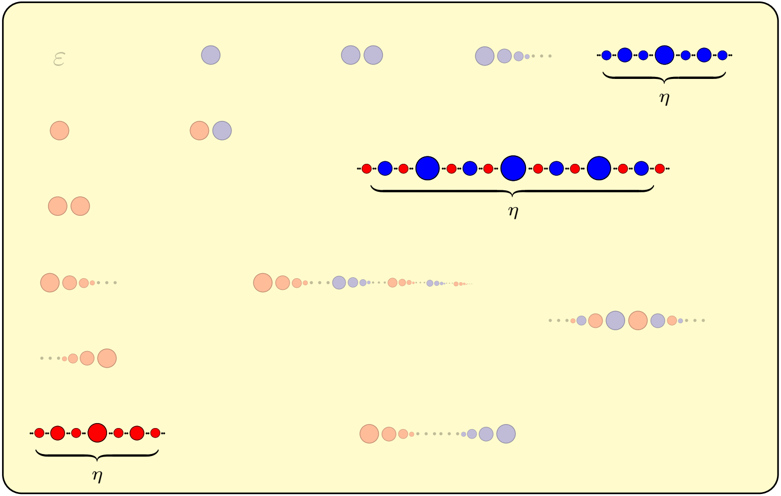
Given an alphabet $A = \{\bullet, \bullet\}$, let

A° =



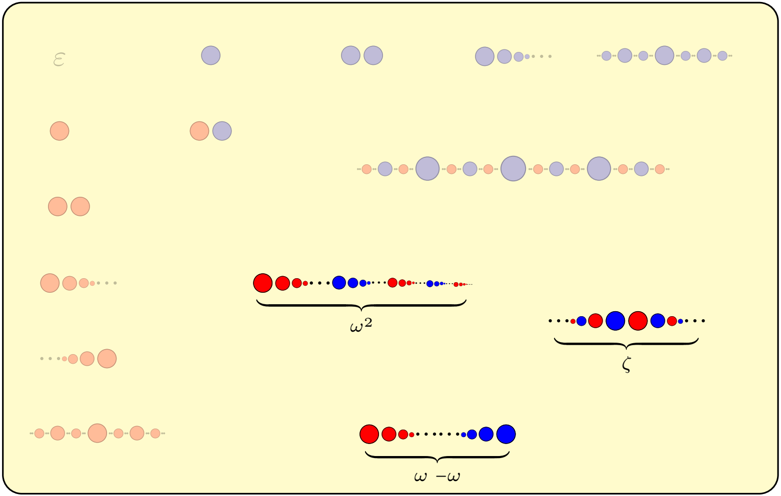
Given an alphabet $A = \{$, $\}$, let

A° =



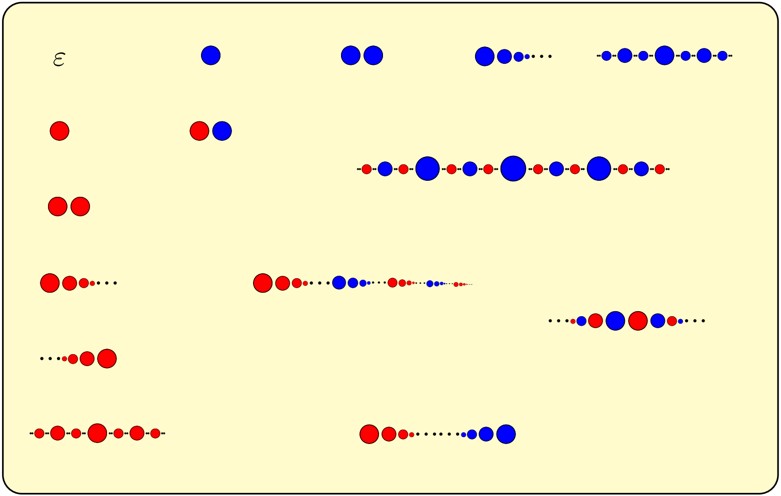
Given an alphabet $A = \{$🔴, 🔵$\}$, let

$A^\circ =$



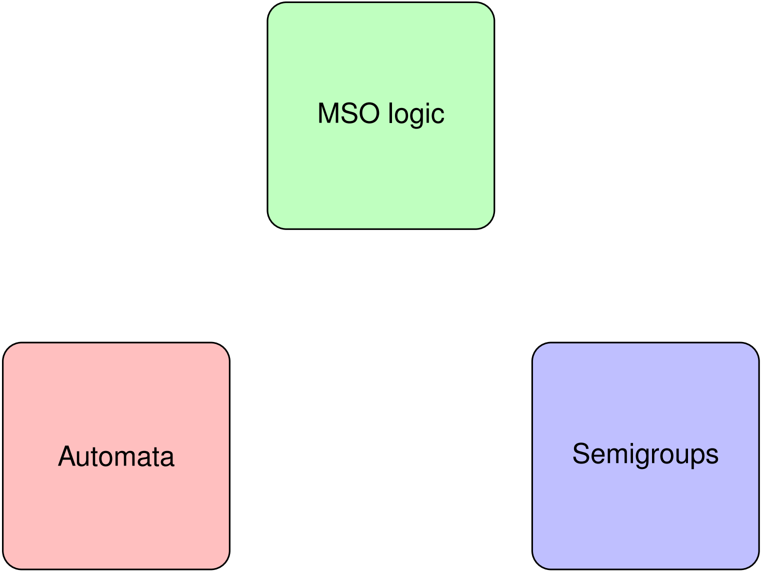
Given an alphabet $A = \{$●, ●$\}$, let

A° =



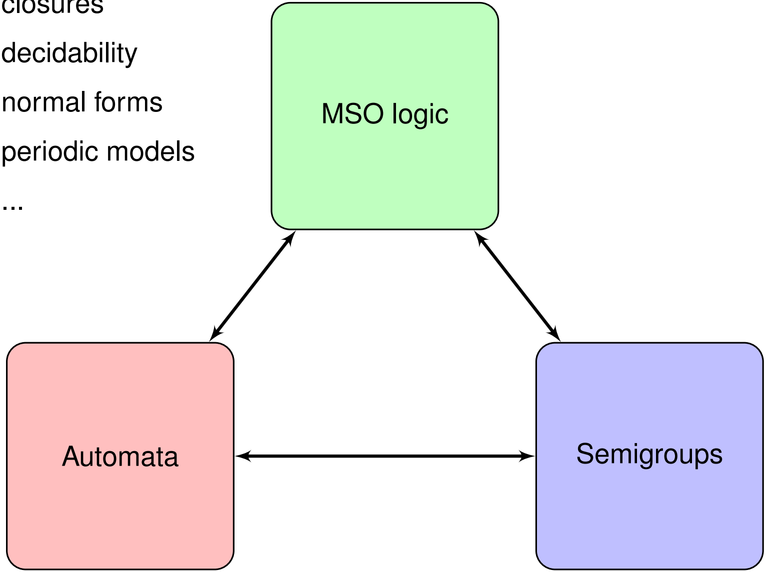
Interest on “**regular**” (= robust & decidable) languages $L \subseteq A^\circ$

Formalisms for classical regular languages

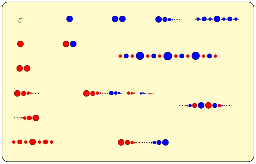


Formalisms for classical regular languages

- ✓ closures
- ✓ decidability
- ✓ normal forms
- ✓ periodic models
- ✓ ...

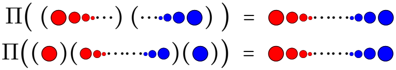


Recognizability of languages via semigroups

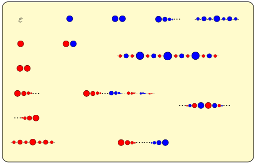
$A^\circ =$ 

associative product $\Pi : (A^\circ)^\circ \to A^\circ$

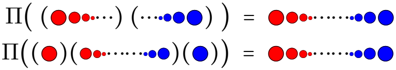
e.g.

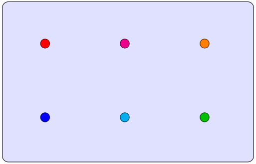


Recognizability of languages via semigroups

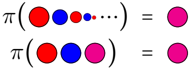
$A^\circ =$ 

associative product $\Pi : (A^\circ)^\circ \to A^\circ$

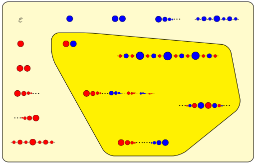
e.g. 

$S =$ 

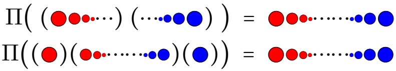
associative product $\pi : S^\circ \to S$

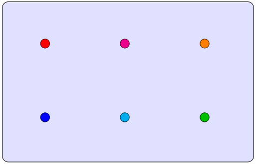
e.g. 

Recognizability of languages via semigroups

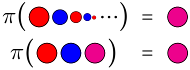
$A^\circ =$ 

associative product $\Pi : (A^\circ)^\circ \to A^\circ$

e.g. 

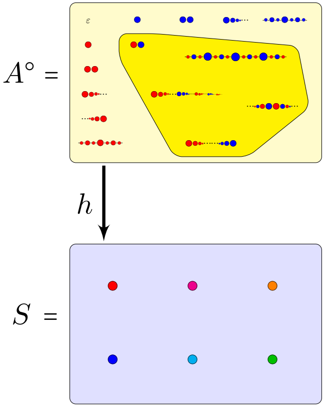
$S =$ 

associative product $\pi : S^\circ \to S$

e.g. 

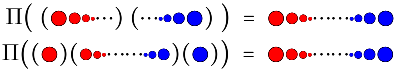
$L \subseteq A^\circ$ **recognized by** (S, π) $\Leftrightarrow$ ...

Recognizability of languages via semigroups



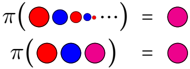
associative product $\Pi : (A^\circ)^\circ \to A^\circ$

e.g.



associative product $\pi : S^\circ \to S$

e.g.

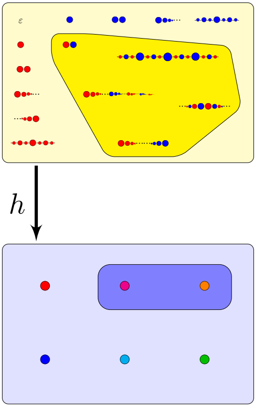


$L \subseteq A^\circ$ **recognized by** (S, π) $\Leftrightarrow$ $\exists\, h : (A^\circ, \Pi) \to (S, \pi)$

...

Recognizability of languages via semigroups

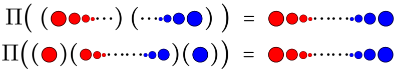
$A^\circ =$



$S =$

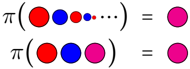
associative product $\Pi : (A^\circ)^\circ \to A^\circ$

e.g.



associative product $\pi : S^\circ \to S$

e.g.

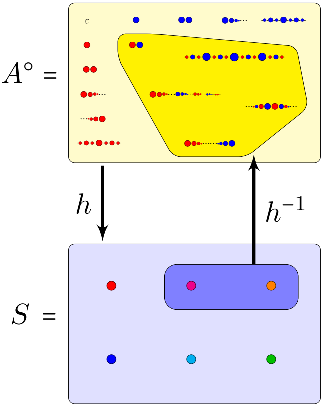


$L \subseteq A^\circ$ **recognized by** (S, π) $\Leftrightarrow$ $\exists\, h : (A^\circ, \Pi) \to (S, \pi)$

$\exists\, F \subseteq S$

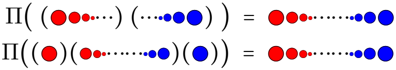
...

Recognizability of languages via semigroups



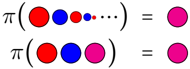
associative product $\Pi : \ (A^\circ)^\circ \to A^\circ$

e.g.



associative product $\pi : \ S^\circ \to S$

e.g.



$L \subseteq A^\circ$ **recognized by** (S, π) $\Leftrightarrow$ $\exists\, h : \ (A^\circ, \Pi) \to (S, \pi)$
$\exists\, F \subseteq S$
$L = h^{-1}(F)$

Properties of recognizable languages

Closure under complementations, unions, projections, ...

e.g. if $L = h^{-1}(F)$ then $A^{\circ} \smallsetminus L = h^{-1}(S \smallsetminus F)$

Properties of recognizable languages

- Closure under complementations, unions, projections, ...

  e.g. if $L = h^{-1}(F)$ then $A^\circ \smallsetminus L = h^{-1}(S \smallsetminus F)$

- Capture all languages definable in MSO

  negations → complementations
  disjunctions → unions
  existential quantifications → projections

Properties of recognizable languages

Closure under complementations, unions, projections, ...

e.g. if $L = h^{-1}(F)$ then $A^{\circ} \setminus L = h^{-1}(S \setminus F)$

Capture all languages definable in MSO

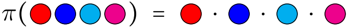
negations $\rightarrow$ complementations
disjunctions $\rightarrow$ unions
existential quantifications $\rightarrow$ projections

Algorithms for emptiness, universality, ... ?

PROBLEM: need to finitely represent countable products!

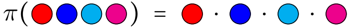
Algebras as representations of countable products

We use the same approch as in classical semigroups

i.e. 

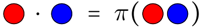
Algebras as representations of countable products

We use the same approch as in classical semigroups

i.e. 

Given a semigroup (S, π), the **induced algebra** consists of

- binary product $\cdot \; : \; S \times S \;\to\; S$



Algebras as representations of countable products

We use the same approch as in classical semigroups

i.e. $\pi(\textcolor{red}{\bullet}\textcolor{blue}{\bullet}\textcolor{cyan}{\bullet}\textcolor{magenta}{\bullet}) = \textcolor{red}{\bullet} \cdot \textcolor{blue}{\bullet} \cdot \textcolor{cyan}{\bullet} \cdot \textcolor{magenta}{\bullet}$

Given a semigroup (S, π), the **induced algebra** consists of

- binary product $\cdot \; : \; S \times S \to S$

  $\textcolor{red}{\bullet} \cdot \textcolor{blue}{\bullet} = \pi(\textcolor{red}{\bullet}\textcolor{blue}{\bullet})$

- $\pm\omega$-powers $\pm\omega \; : \; S \to S$

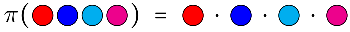
  $\textcolor{red}{\bullet}^{\omega} = \pi(\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\cdots)$

  $\textcolor{red}{\bullet}^{-\omega} = \pi(\cdots\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet})$

Algebras as representations of countable products

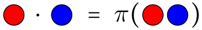
We use the same approch as in classical semigroups

i.e.

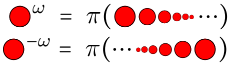


Given a semigroup (S, π), the **induced algebra** consists of

- binary product $\cdot \; : \; S \times S \to S$



- $\pm\omega$-powers $\pm\omega \; : \; S \to S$



- perfect shuffle $\eta \; : \; \mathscr{P}(S) \to S$

Algebras as representations of countable products

We use the same approch as in classical semigroups

i.e. $\pi(\textcolor{red}{\bullet}\textcolor{blue}{\bullet}\textcolor{cyan}{\bullet}\textcolor{magenta}{\bullet}) = \textcolor{red}{\bullet}\cdot\textcolor{blue}{\bullet}\cdot\textcolor{cyan}{\bullet}\cdot\textcolor{magenta}{\bullet}$

Given a semigroup (S,π), the **induced algebra** consists of

- binary product $\cdot \;:\; S \times S \to S$

  $$\textcolor{red}{\bullet}\cdot\textcolor{blue}{\bullet} = \pi(\textcolor{red}{\bullet}\textcolor{blue}{\bullet})$$

- $\pm\omega$-powers $\pm\omega \;:\; S \to S$

  $$\textcolor{red}{\bullet}^{\omega} = \pi(\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\cdots)$$

  $$\textcolor{red}{\bullet}^{-\omega} = \pi(\cdots\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet}\textcolor{red}{\bullet})$$

- perfect shuffle $\eta \;:\; \mathscr{P}(S) \to S$

  $$\{\textcolor{red}{\bullet},\textcolor{blue}{\bullet},\textcolor{green}{\bullet}\}^{\eta} = \pi(\cdots\textcolor{red}{\bullet}\textcolor{blue}{\bullet}\textcolor{green}{\bullet}\textcolor{red}{\bullet}\textcolor{blue}{\bullet}\textcolor{green}{\bullet}\textcolor{red}{\bullet}\textcolor{blue}{\bullet}\textcolor{green}{\bullet}\textcolor{red}{\bullet}\textcolor{blue}{\bullet}\textcolor{green}{\bullet}\textcolor{red}{\bullet}\textcolor{blue}{\bullet}\textcolor{green}{\bullet}\cdots)$$

Equations derived from associativity

e.g. if $\{\textcolor{red}{\bullet},\textcolor{blue}{\bullet},\textcolor{green}{\bullet}\}^{\eta} = \textcolor{magenta}{\bullet}$ then $\textcolor{magenta}{\bullet}\cdot\textcolor{blue}{\bullet}\cdot\textcolor{magenta}{\bullet} = \textcolor{magenta}{\bullet}$

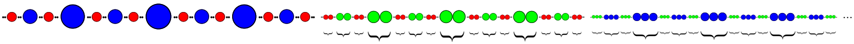
$\{\textcolor{blue}{\bullet},\textcolor{magenta}{\bullet}\}^{\eta} = \textcolor{magenta}{\bullet}$

Recovering countable products from algebras

$w =$ 

To define $\pi(w)$ repeatedly simplify w by evaluating infixes

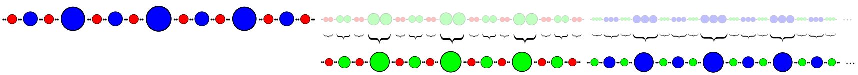
Recovering countable products from algebras

$w =$ 

To define $\pi(w)$ repeatedly simplify w by evaluating infixes

- finite words: use operator $\cdot$

Recovering countable products from algebras

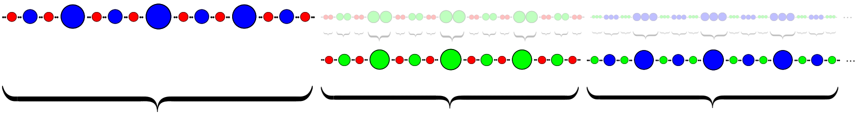
$w =$ 

To define $\pi(w)$ repeatedly simplify w by evaluating infixes

- finite words: use operator $\cdot$

Recovering countable products from algebras

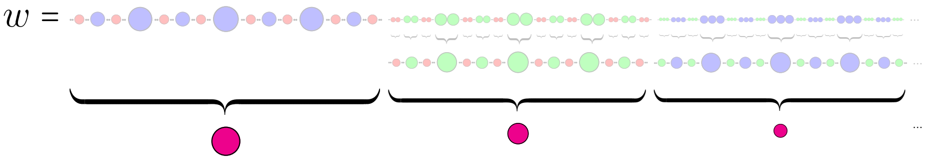
$w =$



To define $\pi(w)$ repeatedly simplify w by evaluating infixes

- finite words: use operator $\cdot$
- perfect shuffles: use operator η

Recovering countable products from algebras

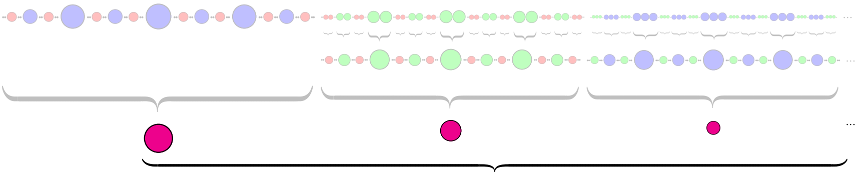


To define $\pi(w)$ repeatedly simplify w by evaluating infixes

- finite words: use operator $\cdot$
- perfect shuffles: use operator η

Recovering countable products from algebras

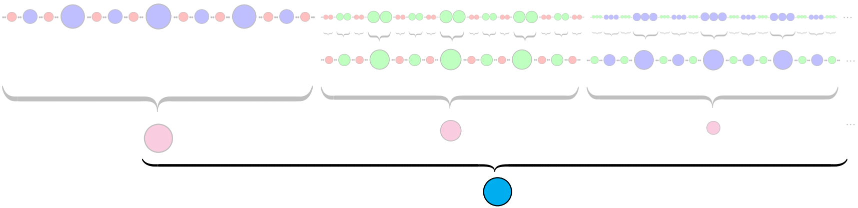
$w =$



To define $\pi(w)$ repeatedly simplify w by evaluating infixes

- finite words: use operator $\cdot$
- perfect shuffles: use operator η
- $\pm\omega$-iterations: use operators ω and $-\omega$

Recovering countable products from algebras

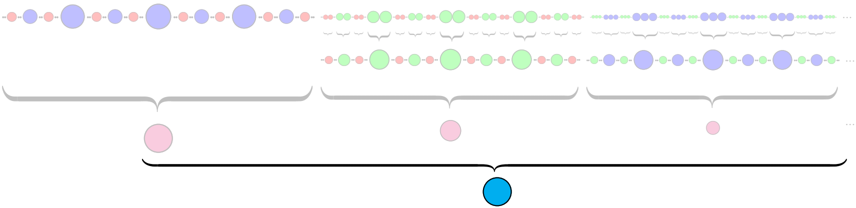
$w =$ 

To define $\pi(w)$ repeatedly simplify w by evaluating infixes

- finite words: use operator $\cdot$
- perfect shuffles: use operator η
- $\pm\omega$-iterations: use operators ω and $-\omega$

Recovering countable products from algebras

$w =$



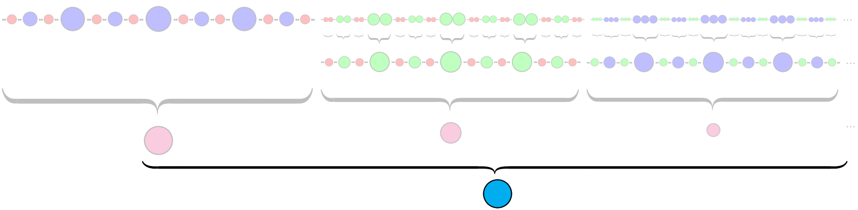
To define $\pi(w)$ repeatedly simplify w by evaluating infixes

- finite words: use operator $\cdot$
- perfect shuffles: use operator η
- $\pm\omega$-iterations: use operators ω and $-\omega$

evaluation strategy = well-founded tree where siblinghoods can be easily evaluated using $\cdot, \eta, \omega, -\omega$

Recovering countable products from algebras

$w =$



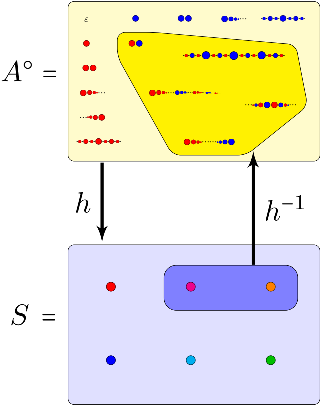
To define $\pi(w)$ repeatedly simplify w by evaluating infixes

- finite words: use operator $\cdot$
- perfect shuffles: use operator η
- $\pm\omega$-iterations: use operators ω and $-\omega$

evaluation strategy = well-founded tree where siblinghoods can be easily evaluated using $\cdot, \eta, \omega, -\omega$

- existence: Theorems a-la Ramsey + Axiom of Choice
- well-definedness: Equations for associativity + Induction

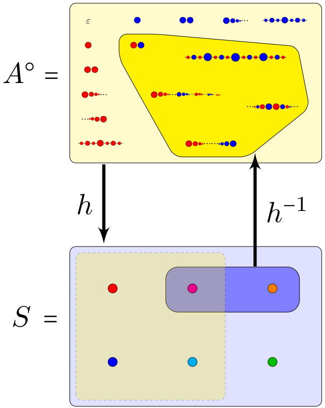
Deciding emptiness of recognizable languages



$$L = h^{-1}(F)$$

$$h(L) = h\big(h^{-1}(F)\big)$$

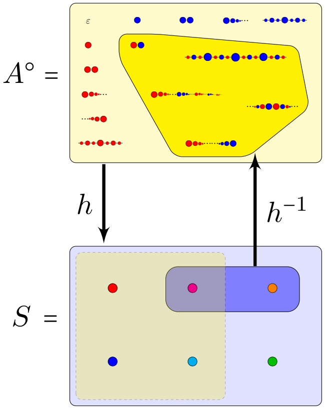
Deciding emptiness of recognizable languages



$$L = h^{-1}(F)$$

$$\begin{aligned} h(L) &= h\big(h^{-1}(F)\big) \\ &= F \cap h\big(A^\circ\big) \end{aligned}$$

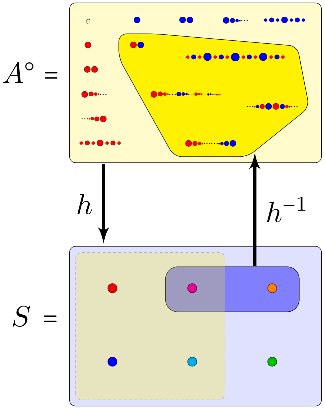
Deciding emptiness of recognizable languages



$$L \;=\; h^{-1}(F)$$

$$\begin{aligned} h(L) \;&=\; h\big(h^{-1}(F)\big) \\ &=\; F \,\cap\, h\big(A^\circ\big) \\ &=\; F \,\cap\, \underbrace{\langle\, h(A) \,\rangle}_{\substack{\text{closure of } h(A) \\ \text{under } \cdot,\,\omega,\,-\omega,\,\eta}} \end{aligned}$$

Deciding emptiness of recognizable languages

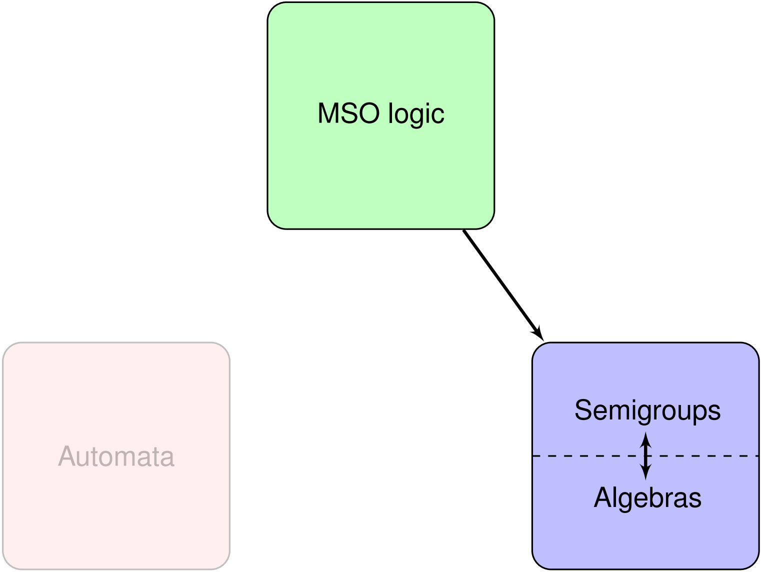


$$L = h^{-1}(F)$$

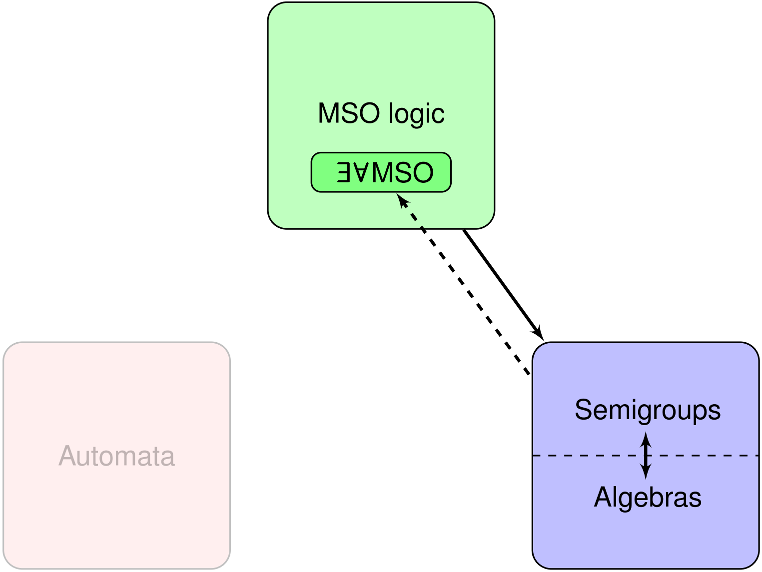
$$\begin{aligned} h(L) &= h\big(h^{-1}(F)\big) \\ &= F \cap h\big(A^\circ\big) \\ &= F \cap \underbrace{\langle\, h(A) \,\rangle}_{\substack{\text{closure of } h(A) \\ \text{under } \cdot,\, \omega,\, -\omega,\, \eta}} \end{aligned}$$

$L \neq \varnothing$ iff $F \cap \langle\, h(A) \,\rangle \neq \varnothing$

Translations between formalisms



Translations between formalisms



From semigroups to MSO in normal form

☞ Recognizable languages can be defined in **∃∀MSO**

i.e. by formulas $\exists\, \bar{X}.\ \forall\, \bar{Y}.\ \underbrace{\varphi(\bar{X}, \bar{Y})}_{\text{FO formula}}$

From semigroups to MSO in normal form

☞ Recognizable languages can be defined in **∃∀MSO**

i.e. by formulas $\exists\, \bar{X}.\ \forall\, \bar{Y}.\ \underbrace{\varphi(\bar{X},\bar{Y})}_{\text{FO formula}}$

To check whether $w \in L$, one needs again to evaluate $\pi(w)$

PROBLEM: evaluation strategy must be guessed in MSO!

From semigroups to MSO in normal form

Recognizable languages can be defined in **∃∀MSO**

i.e. by formulas $\exists \bar{X}.\ \forall \bar{Y}.\ \underbrace{\varphi(\bar{X},\bar{Y})}_{\text{FO formula}}$

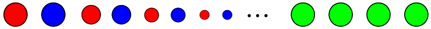
To check whether $w \in L$, one needs again to evaluate $\pi(w)$

PROBLEM: evaluation strategy must be guessed in MSO!

new evaluation strategy = Factorization Forest [Simon '90] [Colcombet '10]

= tree of small (bounded) height that eases evaluation of subwords via FO

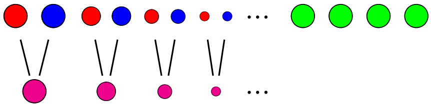
Factorization forests

$w =$ 

Internal nodes of a factorization forest can have:

- **2 children** with arbitrary values
- **several children** with **same idempotent** $(e \cdot e = e)$

Factorization forests

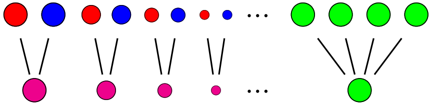
$w =$ 

Internal nodes of a factorization forest can have:

- **2 children** with arbitrary values
- **several children** with **same idempotent** $(e \cdot e = e)$

Factorization forests

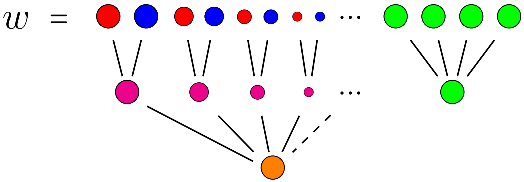
w =



Internal nodes of a factorization forest can have:

- **2 children** with arbitrary values
- **several children** with **same idempotent** $(e \cdot e = e)$

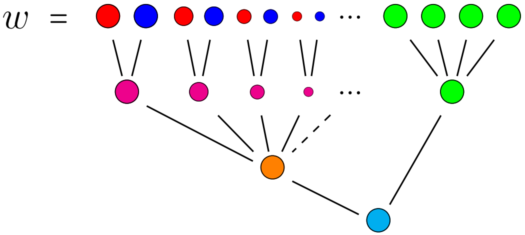
Factorization forests



Internal nodes of a factorization forest can have:

- **2 children** with arbitrary values
- **several children** with **same idempotent** $(e \cdot e = e)$

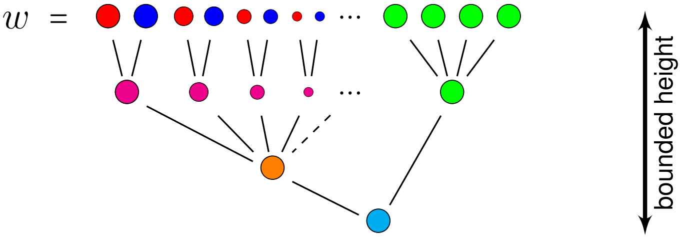
Factorization forests



Internal nodes of a factorization forest can have:

- **2 children** with arbitrary values
- **several children** with **same idempotent** $(e \cdot e = e)$

Factorization forests

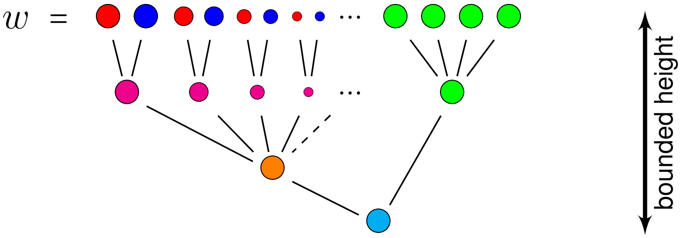


Internal nodes of a factorization forest can have:

- **2 children** with arbitrary values
- **several children** with **same idempotent** $(e \cdot e = e)$

There always exist a factorization forest of height $\leq k|S|$

Factorization forests



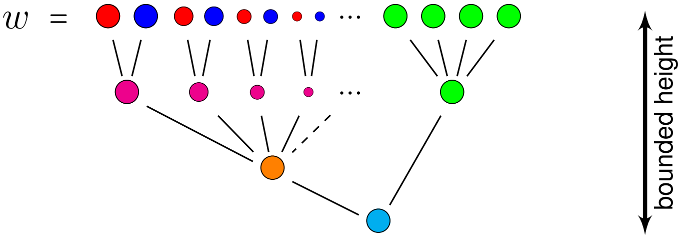
Internal nodes of a factorization forest can have:

- **2 children** with arbitrary values
- **several children** with **same idempotent** $(e \cdot e = e)$

There always exist a factorization forest of height $\leq k|S|$

$w \in L \;\Leftrightarrow\; \exists$ factorization forest $\bar{X}$. value$(w) \in F$

Factorization forests



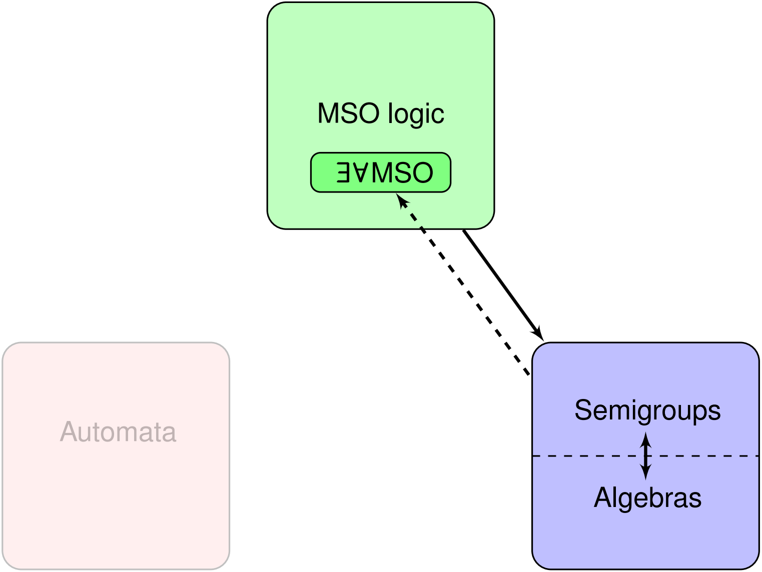
Internal nodes of a factorization forest can have:

- **2 children** with arbitrary values
- **several children** with **same idempotent** $(e \cdot e = e)$

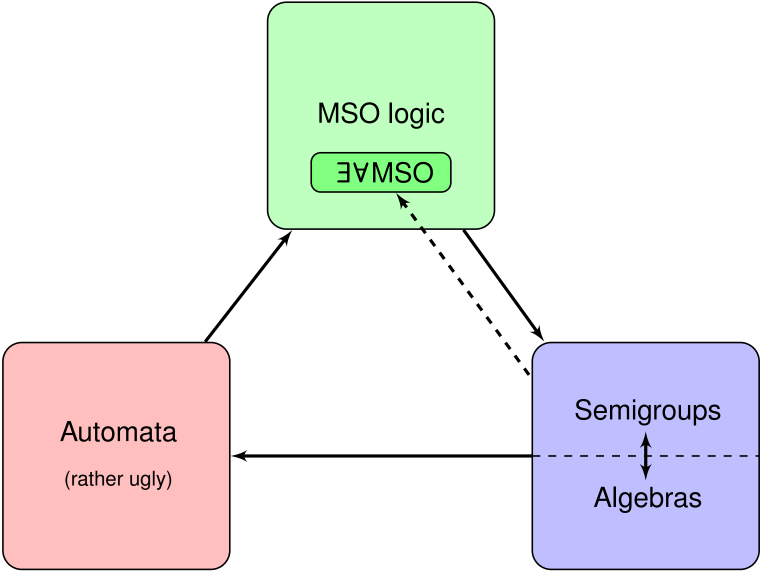
There always exist a factorization forest of height $\leq k|S|$

$$w \in L \iff \exists \text{ factorization forest } \bar{X}.\ \ \text{value}(w) \in F$$
$$\wedge\ \ \forall \text{ subword } Y.\ \forall \text{ factorization } Z.\ \text{value}(Y) = \prod_{Y_i \text{ factor of } Z} \text{value}(Y_i)$$

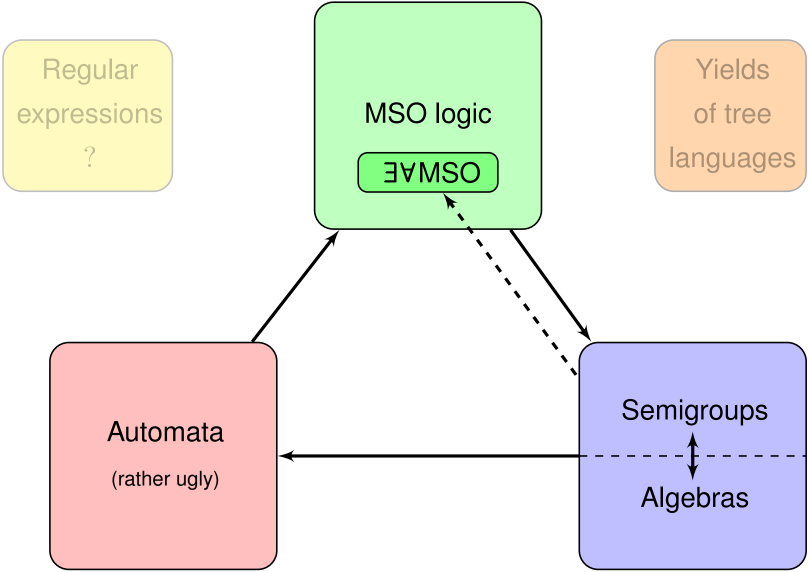
Translations between formalisms (cont'd)



Translations between formalisms (cont'd)

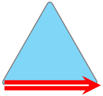


Translations between formalisms (cont'd)



Other applications

- **Yields of trees**



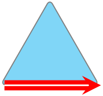
L regular language of countable words

$$\Updownarrow$$

$T = \left\{ t \;:\; \text{yield}(t) \in L \right\}$ regular language of trees

Other applications

- **Yields of trees**

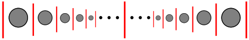


L regular language of countable words

$$\Updownarrow$$

$T = \{t \,:\, \mathrm{yield}(t) \in L\}$ regular language of trees

- **Logics with cuts in the background** [Gurevitch & Rabinovitch]

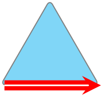


variables $x, X, \ldots$ for positions
$\hat{x}, \hat{X}, \ldots$ for cuts

MSO$[\mathbb{Q}, \hat{\mathbb{Q}}]$ is undecidable (like MSO$[\mathbb{R}]$)
MSO$[\mathbb{Q}, \hat{\mathbb{Q}}]$ defines same predicates over $\mathbb{Q}$ as MSO$[\mathbb{Q}]$

Other applications

- **Yields of trees**

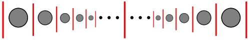


L regular language of countable words

$\Updownarrow$

$T = \{t \ : \ \mathrm{yield}(t) \in L\}$ regular language of trees

- **Logics with cuts in the background** [Gurevitch & Rabinovitch]



variables $x, X, \ldots$ for positions
$\hat{x}, \hat{X}, \ldots$ for cuts

$\mathrm{MSO}[\mathbb{Q}, \hat{\mathbb{Q}}]$ is undecidable (like $\mathrm{MSO}[\mathbb{R}]$)
$\mathrm{MSO}[\mathbb{Q}, \hat{\mathbb{Q}}]$ defines same predicates over $\mathbb{Q}$ as $\mathrm{MSO}[\mathbb{Q}]$

- **Characterizations of FO-definable languages...**