The expressiveness of recognizability by orbite finite nominal monoids

Thomas Colcombet
{Ley, Puppis}

{symmetry, logic, computation}
Simons Institute
November 10, 2016

Fondamental theorems

Fondamental theorems

Fondamental theorem of regular languages:

monadic second-order logic $\overset{=}{\text{eff}}$ recognizability by monoids $\overset{=}{\text{eff}}$ deterministic automata $\overset{=}{\text{eff}}$ NDA = Rat = altA = …

Fondamental theorems

Fondamental theorem of regular languages:

monadic second-order logic $\overset{=}{\text{eff}}$ recognizability by monoids $\overset{=}{\text{eff}}$ deterministic automata $\overset{=}{\text{eff}}$ NDA = Rat = altA = …

Fondamental theorem of aperiodic regular languages:

first-order logic $\overset{=}{\text{eff}}$ aperiodic monoids $\overset{=}{\text{eff}}$ counter-free automata $\overset{=}{\text{eff}}$ star free = LTL = …

Fondamental theorems

Fondamental theorem of regular languages:

monadic second-order logic $\overset{=}{\text{eff}}$ recognizability by monoids $\overset{=}{\text{eff}}$ deterministic automata $\overset{=}{\text{eff}}$ NDA = Rat = altA = …

Fondamental theorem of aperiodic regular languages:

first-order logic $\overset{=}{\text{eff}}$ aperiodic monoids $\overset{=}{\text{eff}}$ counter-free automata $\overset{=}{\text{eff}}$ star free = LTL = …

VARIATIONS

languages of words of length ω

languages of words of countable length

languages finite trees

languages of infinite trees

cost functions (functions modulo relative boundedness)

Fondamental theorems

Fondamental theorem of regular languages:

monadic second-order logic $\overset{=}{\text{eff}}$ recognizability by monoids $\overset{=}{\text{eff}}$ deterministic automata $\overset{=}{\text{eff}}$ NDA = Rat = altA = …

Fondamental theorem of aperiodic regular languages:

first-order logic $\overset{=}{\text{eff}}$ aperiodic monoids $\overset{=}{\text{eff}}$ counter-free automata $\overset{=}{\text{eff}}$ star free = LTL = …

VARIATIONS

- languages of words of length ω
- languages of words of countable length
- languages finite trees
- languages of infinite trees
- cost functions (functions modulo relative boundedness)

What happens in the **nominal world**?

Regular data languages?

Regular data languages?

Idea: Extend language theory with the ability to talk about:

- names
- identifiers
- stamps of origin

Regular data languages?

Idea: Extend language theory with the ability to talk about:

- names
- identifiers
- stamps of origin

Data/nominal words $u \in D^*$ for **D** infinite.

Regular data languages?

Idea: Extend language theory with the ability to talk about:

- names
- identifiers
- stamps of origin

Data/nominal words $u \in D^*$ for **D** infinite.

Datas can only be compared using equality.

Regular data languages?

Idea: Extend language theory with the ability to talk about:

- names
- identifiers
- stamps of origin

Data/nominal words $u \in D^*$ for **D** infinite.

Datas can only be compared using equality.

Basic automata use registers (**{Kaminsky, Francez}**…)

Regular data languages?

Idea: Extend language theory with the ability to talk about:

- names
- identifiers
- stamps of origin

Data/nominal words $u \in D^*$ for **D** infinite.
Datas can only be compared using equality.

Basic automata use registers (**{Kaminsky, Francez}**…)

Things are not nice anymore:

- many notions of recognizers now have distinct expressiveness
  DRA ≠ **NRA** ≠ **NRA'** ≠ **Monoid** ≠ **Alternating automata** …
- many problems usually decidable, are not anymore
  universality of **NFA** / SAT of **FO** / SAT of **MSO**, …

Regular data languages?

Idea: Extend language theory with the ability to talk about:
- names
- identifiers
- stamps of origin

Data/nominal words $u \in D^*$ for **D** infinite.
Datas can only be compared using equality.

Basic automata use registers (**{Kaminsky, Francez}**…)

Things are not nice anymore:
- many notions of recognizers now have distinct expressiveness
  DRA ≠ **NRA** ≠ **NRA’** ≠ **Monoid** ≠ **Alternating automata** …
- many problems usually decidable, are not anymore
  universality of **NFA** / SAT of **FO** / SAT of **MSO**, …

Decidability results (for instance):
Decidability of satisfiability of **LTLfreeze** / **FO2** / **Alt1reg** / **walking logics** / **class automata**.

Regular data languages?

Idea: Extend language theory with the ability to talk about:

- names
- identifiers
- stamps of origin

Data/nominal words $u \in D^*$ for **D** infinite.
Datas can only be compared using equality.

Basic automata use registers (**{Kaminsky, Francez}**…)

Things are not nice anymore:

- many notions of recognizers now have distinct expressiveness
  DRA ≠ **NRA** ≠ **NRA'** ≠ **Monoid** ≠ **Alternating automata** …
- many problems usually decidable, are not anymore
  universality of **NFA** / SAT of **FO** / SAT of **MSO**, …

Decidability results (for instance):
Decidability of satisfiability of **LTLfreeze** / **FO2** / **Alt1reg** / **walking logics** / **class automata**.

There is no convincing notion of regular language of data words.

In this talk

{C., Ley, Puppis}2011

For languages of data words:

definable in rigid monadic second-order logic	$=_{\text{eff}}$	recognizability by orbit finite nominal monoids
definable in rigid first-order logic	$=_{\text{eff}}$	recognizability by aperiodic orbit finite nominal monoids

In this talk

{C., Ley, Puppis}2011

For languages of data words:

definable in rigid monadic second-order logic	$=_{\text{eff}}$	recognizability by orbit finite nominal monoids
definable in rigid first-order logic	$=_{\text{eff}}$	recognizability by aperiodic orbit finite nominal monoids

What is an orbit finite nominal monoid?

What are logics for data words?

What is rigidity?

Other consequences?

Data / nominal world

Data / nominal world

Data/nominal words $u \in \mathbf{D}^*$ for **D** infinite of names/datas.

Datas can only be compared using equality.

Things are FO-definable in (D,=).

Data / nominal world

Data/nominal words $u \in \mathbf{D}^*$ for **D** infinite of names/datas.

Datas can only be compared using equality.

Things are FO-definable in (D,=).

Nominal sets offer a framework for talking about these phenomena.

Data / nominal world

Data/nominal words $u \in \mathbf{D}^*$ for **D** infinite of names/datas.

Datas can only be compared using equality.

Things are FO-definable in (D,=).

Nominal sets offer a framework for talking about these phenomena.

A **nominal set** is a set on which acts permutations of **D** (+ support …).
Examples: $\mathbf{D}$, $\mathbf{D}\times\mathbf{D}$, $\mathbf{D}^*$, sets.

Data / nominal world

Data/nominal words $u \in \mathbf{D}^*$ for **D** infinite of names/datas.

Datas can only be compared using equality.

Things are FO-definable in (D,=).

Nominal sets offer a framework for talking about these phenomena.

A **nominal set** is a set on which acts permutations of **D** (+ support …).
Examples: **D**, **D**×**D**, **D***, sets.

An **orbit finite set** is one that is finite up to the action of permutations.
Examples: **D**, **D**×**D**, finite sets, but not **D*** or infinite sets.

Data / nominal world

Data/nominal words $u \in \mathbf{D}^*$ for **D** infinite of names/datas.

Datas can only be compared using equality.

Things are FO-definable in (D,=).

Nominal sets offer a framework for talking about these phenomena.

A **nominal set** is a set on which acts permutations of **D** (+ support …).
Examples: **D**, **D**×**D**, **D***, sets.

An **orbit finite set** is one that is finite up to the action of permutations.
Examples: **D**, **D**×**D**, finite sets, but not **D*** or infinite sets.

An **equivariant** set/relation/function is one that is invariant under permutations of **D**.

Equivalently, it is definable using only the equality between **D** objects.

Languages recognized by an orbit finite nominal monoid

A is an **orbit finite alphabet** (**D** or B×**D**)

(M, ▪ ,1) is a **monoid** where M is orbit finite
and 1 and ▪ : M×M → M are equivariant.

h : A* → **M** is an equivariant **monoid morphism**.

F⊆M is an equivariant **set of accepting elements**.

The **recognized language** is L(M,h,F) = { u∈A* : f(u)∈F }

Languages recognized by an orbit finite nominal monoid

A is an **orbit finite alphabet** (**D** or B×**D**)

(M, · ,1) is a **monoid** where M is orbit finite
and 1 and · : M×M → M are equivariant.

h : A* → **M** is an equivariant **monoid morphism**.

F⊆M is an equivariant **set of accepting elements**.

The **recognized language** is L(M,h,F) = { u∈A* : f(u)∈F }

Extremities: **E** = 'words such that the first and the last datas coincide'

elements	1, (d,e) for d,e datas (three orbits)
product	(d,e) · (d',e')=(d,e')
morphism	h(d) = (d,d) for all datas d (and generated)
accepting set	F = { (d,d) : d∈**D** }

Examples of nominal monoids

Examples of nominal monoids

Sdistinct: **SD** = ‘words such that every consecutive datas are different’

elements	1, 0, (d,e) for d,e datas (four orbits)
product	$(d,e)\cdot(d',e') = \begin{cases} (d,e') & \text{if } e \neq d' \\ 0 & \text{otherwise} \end{cases}$
morphism	h(d) = (d,d) for all datas $d \in \mathbf{D}$
accepting set	$F = \{1\} \cup \{ (d,e) : d,e \in \mathbf{D} \}$

Examples of nominal monoids

Sdistinct: **SD** = 'words such that every consecutive datas are different'

elements	1, 0, (d,e) for d,e datas (four orbits)
product	$(d,e)\cdot(d',e') = \begin{cases} (d,e') & \text{if } e \neq d' \\ 0 & \text{otherwise} \end{cases}$
morphism	h(d) = (d,d) for all datas $d \in \mathbf{D}$
accepting set	$F = \{1\} \cup \{ (d,e) : d,e \in \mathbf{D} \}$

Counting: **C(k)** = 'words that contain exactly k distinct data values'

elements	sets of values of cardinality at most k, and 0
product	union, 0 when size k is exceeded
morphism	$h(d) = \{ d \}$
accepting set	$F = \{ X : \|X\|=k\}$

Examples of nominal monoids

Sdistinct: **SD** = 'words such that every consecutive datas are different'

elements	1, 0, (d,e) for d,e datas (four orbits)
product	$(d,e)\cdot(d',e')=\begin{cases}(d,e') & \text{if } e\neq d' \\ 0 & \text{otherwise}\end{cases}$
morphism	h(d) = (d,d) for all datas $d\in\mathbf{D}$
accepting set	$F = \{1\} \cup \{ (d,e) : d,e \in \mathbf{D} \}$

Counting: **C(k)** = 'words that contain exactly k distinct data values'

elements	sets of values of cardinality at most k, and 0
product	union, 0 when size k is exceeded
morphism	h(d) = { d }
accepting set	$F = \{ X : \lvert X\rvert=k\}$

Distinct: **D** = 'words such that all datas are distinct'

Impossible: the Myhill-Nerode congruence for this language has infinitely many orbits…

{C., Ley, Puppis}

For languages of data words:

definable in rigid monadic second-order logic	$=_{\text{eff}}$	recognizability by orbit finite nominal monoids
definable in rigid first-order logic	$=_{\text{eff}}$	recognizability by aperiodic orbit finite nominal monoids

What is an orbit finite nominal monoid?

What are logics for data words?

What is rigidity?

Other consequences?

Logics for data words

Logics for data words

Alphabet **B×D** or simply **D**

Word (a,1) (b,124) (a,3) (b,2) (a,5) (a,124)

Logics for data words

Alphabet **B×D** or simply **D**

Word (a,1) (b,124) (a,3) (b,2) (a,5) (a,124)

First-order logic FO[<,~]:

- (first-order) variables range over positions in the word
- predicates are
  - **x<y** = ‘position x is to the left of position y’
  - **a(x)** = ‘the finite part of the letter at position x is **a**’
  - **x~y** = ‘x and y carry the same data’

Logics for data words

Alphabet **B×D** or simply **D**

Word (a,1) (b,124) (a,3) (b,2) (a,5) (a,124)

First-order logic FO[<,~]:

- (first-order) variables range over positions in the word
- predicates are
  - **x<y** = 'position x is to the left of position y'
  - **a(x)** = 'the finite part of the letter at position x is **a**'
  - **x~y** = 'x and y carry the same data'

Monadic second-order logic MSO[<,~]:

- FO[<,~] +
- (monadic) variables that range over set of positions
- membership predicates **x∈Y**

Logics for data words

Alphabet **B×D** or simply **D**

Word (a,1) (b,124) (a,3) (b,2) (a,5) (a,124)

First-order logic FO[<,~]:

- (first-order) variables range over positions in the word
- predicates are
  - **x<y** = 'position x is to the left of position y'
  - **a(x)** = 'the finite part of the letter at position x is **a**'
  - **x~y** = 'x and y carry the same data'

Monadic second-order logic MSO[<,~]:

- FO[<,~] +
- (monadic) variables that range over set of positions
- membership predicates **x∈Y**

Examples:

- The first and last datas coincide
- Some data reappears
- Some data appears an odd number of times

Logics for data words

Alphabet **B×D** or simply **D**

Word (a,1) (b,124) (a,3) (b,2) (a,5) (a,124)

First-order logic FO[<,~]:

- (first-order) variables range over positions in the word
- predicates are
  - **x<y** = ‘position x is to the left of position y’
  - **a(x)** = ‘the finite part of the letter at position x is **a**’
  - **x~y** = ‘x and y carry the same data’

Monadic second-order logic MSO[<,~]:

- FO[<,~] +
- (monadic) variables that range over set of positions
- membership predicates **x∈Y**

Examples:

- The first and last datas coincide
- Some data reappears
- Some data appears an odd number of times

Automatically, definable languages are equivariant

Undecidability of FO[<,~]

Undecidability of FO[<,~]

Over the alphabet $\{a,b\}\times\mathbf{D}$, finite graphs can be interpreted in a data word:

- A **vertex** is a maximal interval of a-labeled positions
- There is an **edge** between two vertices if these use a common value

Undecidability of FO[<,~]

Over the alphabet {a,b}×**D**, finite graphs can be interpreted in a data word:

- A **vertex** is a maximal interval of a-labeled positions
- There is an **edge** between two vertices if these use a common value

(a,1)(a,2) (b,0) **(a,1)(a,3)** (b,0) **(a,2)(a,3)**

Undecidability of FO[<,~]

Over the alphabet $\{a,b\}\times\mathbf{D}$, finite graphs can be interpreted in a data word:

- A **vertex** is a maximal interval of a-labeled positions
- There is an **edge** between two vertices if these use a common value

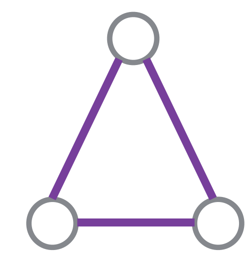
(a,1)(a,2) (b,0) **(a,1)(a,3)** (b,0) **(a,2)(a,3)**

Undecidability of FO[<,~]

Over the alphabet {a,b}×**D**, finite graphs can be interpreted in a data word:

- A **vertex** is a maximal interval of a-labeled positions
- There is an **edge** between two vertices if these use a common value

(a,1)(a,2) (b,0) **(a,1)(a,3)** (b,0) **(a,2)(a,3)**

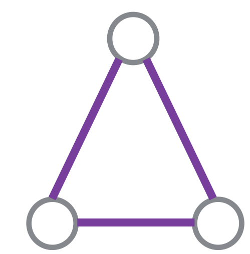


Undecidability of FO[<,~]

Over the alphabet {a,b}×**D**, finite graphs can be interpreted in a data word:

- A **vertex** is a maximal interval of a-labeled positions
- There is an **edge** between two vertices if these use a common value

(a,1)(a,2) (b,0) **(a,1)(a,3)** (b,0) **(a,2)(a,3)**



Fact: For every graph, there is a data-word in which it is interpreted.

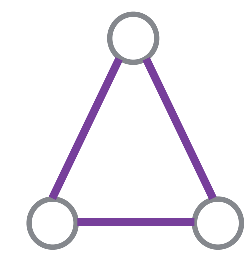
Corollary: Satisfiability of **FO[<,~]** is undecidable over data words.

Undecidability of FO[<,~]

Over the alphabet {a,b}×**D**, finite graphs can be interpreted in a data word:

- A **vertex** is a maximal interval of a-labeled positions
- There is an **edge** between two vertices if these use a common value

(a,1)(a,2) (b,0) **(a,1)(a,3)** (b,0) **(a,2)(a,3)**



Fact: For every graph, there is a data-word in which it is interpreted.

Corollary: Satisfiability of **FO[<,~]** is undecidable over data words.

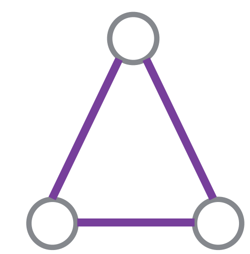
How to reduce the power of these logics and become decidable?

Undecidability of FO[<,~]

Over the alphabet {a,b}×**D**, finite graphs can be interpreted in a data word:

- A **vertex** is a maximal interval of a-labeled positions
- There is an **edge** between two vertices if these use a common value

(a,1)(a,2) (b,0) **(a,1)(a,3)** (b,0) **(a,2)(a,3)**



Fact: For every graph, there is a data-word in which it is interpreted.

Corollary: Satisfiability of **FO[<,~]** is undecidable over data words.

How to reduce the power of these logics and become decidable?

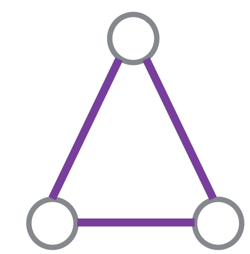
Go to decidable logics over the class of all graphs (e.g. μ-calculus)
→ walking logics, (can express 'a value appears twice')
these are incomparable to e.g. monoids / automata

Undecidability of FO[<,~]

Over the alphabet {a,b}×**D**, finite graphs can be interpreted in a data word:

- A **vertex** is a maximal interval of a-labeled positions
- There is an **edge** between two vertices if these use a common value

(a,1)(a,2) (b,0) **(a,1)(a,3)** (b,0) **(a,2)(a,3)**



Fact: For every graph, there is a data-word in which it is interpreted.

Corollary: Satisfiability of **FO[<,~]** is undecidable over data words.

How to reduce the power of these logics and become decidable?

Go to decidable logics over the class of all graphs (e.g. μ-calculus)
→ walking logics, (can express ‘a value appears twice’)
these are incomparable to e.g. monoids / automata

Use a syntactic restrictions specially tailored for the expressiveness of recognizability by orbit finite monoids → **rigidity**

Known results

Known results

{Bojanczyk}2010:
Every data language recognized by an orbit finite monoid is MSO[<,~] definable.

Every FO[<,~] definable data languages is recognized by an aperiodic syntactic monoid (not orbit finite in general).

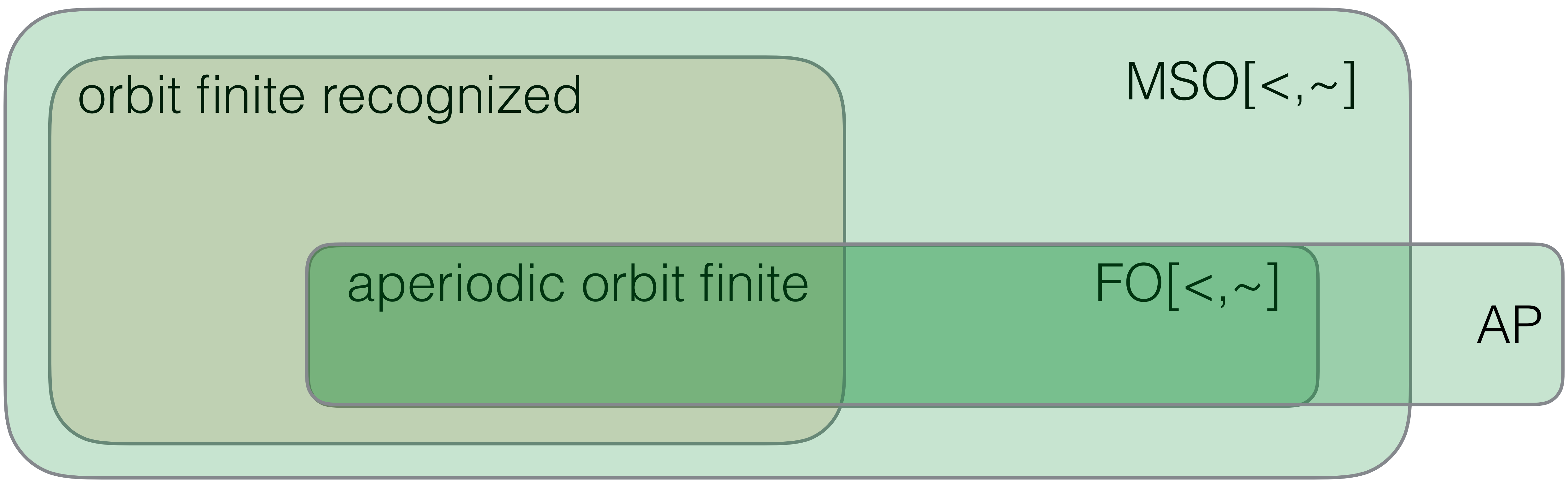
If a data language is recognizable by an orbit finite aperiodic monoid, then it is FO[<,~] definable.

Known results

{Bojanczyk}2010:
Every data language recognized by an orbit finite monoid is MSO[<,~] definable.

Every FO[<,~] definable data languages is recognized by an aperiodic syntactic monoid (not orbit finite in general).

If a data language is recognizable by an orbit finite aperiodic monoid, then it is FO[<,~] definable.

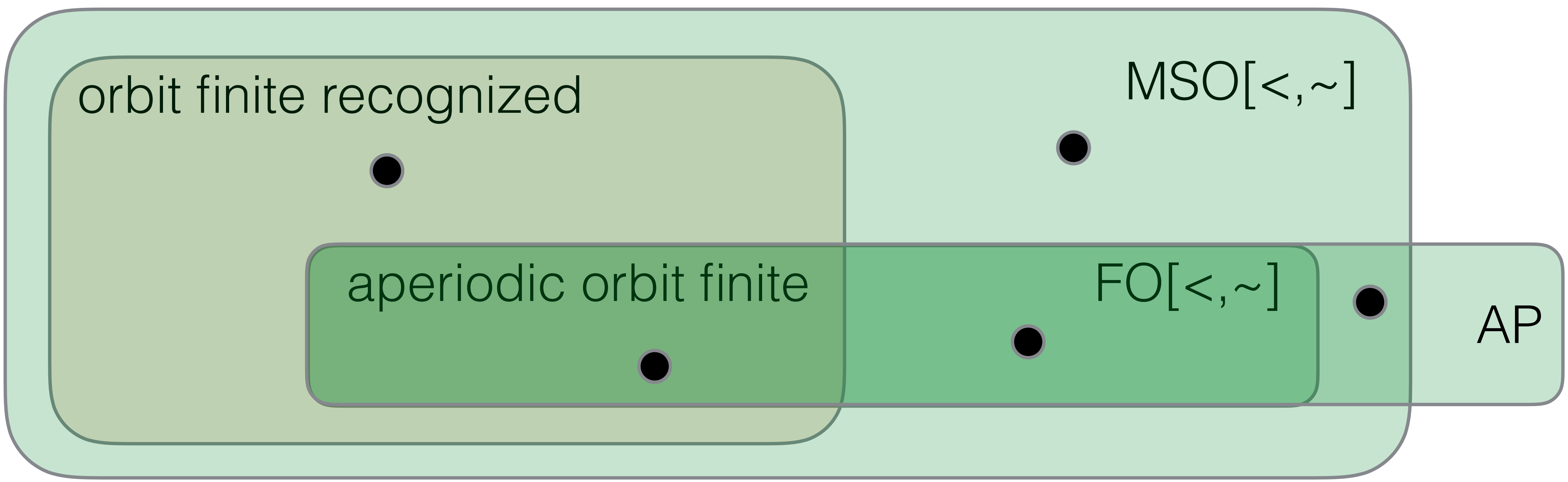


Known results

{Bojanczyk}2010:
Every data language recognized by an orbit finite monoid is MSO[<,~] definable.

Every FO[<,~] definable data languages is recognized by an aperiodic syntactic monoid (not orbit finite in general).

If a data language is recognizable by an orbit finite aperiodic monoid, then it is FO[<,~] definable.

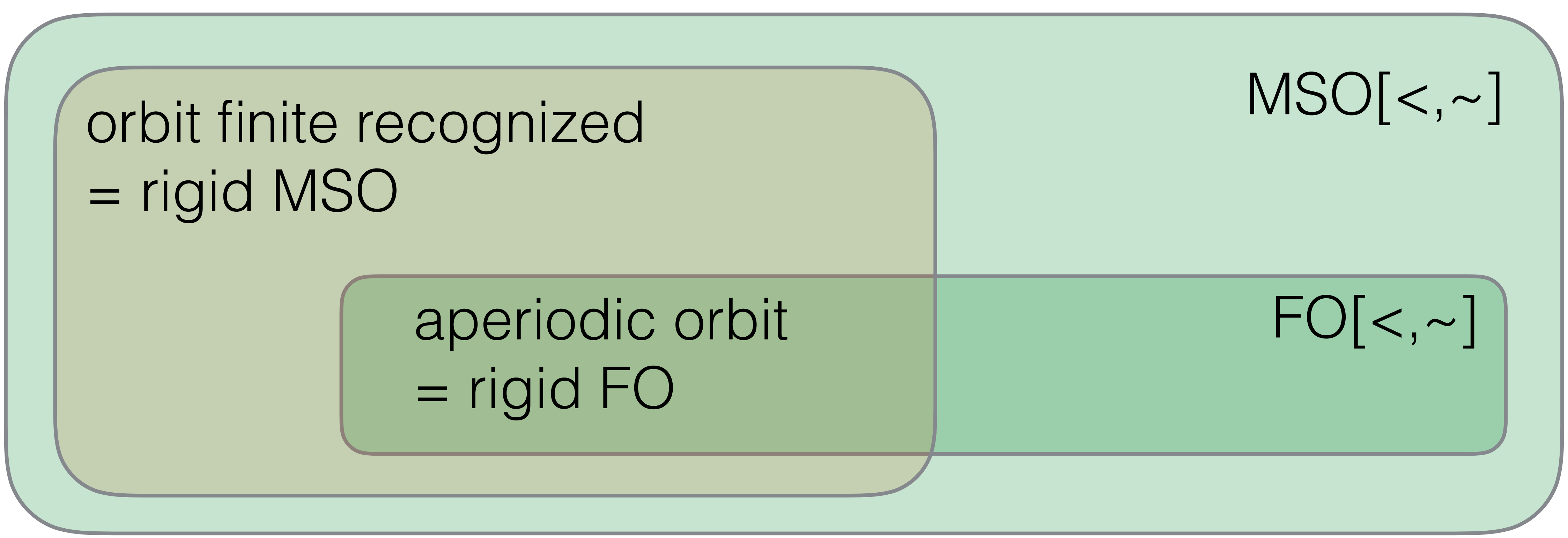


{C., Ley, Puppis}

For languages of data words:

definable in rigid monadic second-order logic $=_{\text{eff}}$ recognizability by orbit finite nominal monoids

definable in rigid first-order logic $=_{\text{eff}}$ recognizability by aperiodic orbit finite nominal monoids



{C., Ley, Puppis}

For languages of data words:

definable in rigid monadic second-order logic	$=_{\text{eff}}$	recognizability by orbit finite nominal monoids
definable in rigid first-order logic	$=_{\text{eff}}$	recognizability by aperiodic orbit finite nominal monoids

What is an orbit finite nominal monoid?

What are logics for data words?

What is rigidity?

Other consequences?

Rigid guards

Rigid guards

A formula **$\phi(x,y)$** is **rigid** if it defines a partial one-to-one map over positions.

Rigid guards

A formula $\boldsymbol{\phi(x,y)}$ is **rigid** if it defines a partial one-to-one map over positions.

Examples: '$\mathbf{first(x)} \wedge \mathbf{last(y)}$', '$\mathbf{succ(x,y)}$',

'$\mathbf{a(x) \wedge a(y) \wedge x<y \wedge \forall z(x<z<y \rightarrow \neg a(x))}$'

Rigid guards

A formula **$\phi(x,y)$** is **rigid** if it defines a partial one-to-one map over positions.

Examples: '**$\mathrm{first}(x) \wedge \mathrm{last}(y)$**', '**$\mathrm{succ}(x,y)$**',
'**$a(x) \wedge a(y) \wedge x<y \wedge \forall z (x<z<y \rightarrow \neg a(x))$**'

Remark: Rigidity can be first-order enforced as follows

$$\phi'(x,y) := \phi(x,y) \wedge \forall x' (\phi(x',y) \rightarrow x'=x) \wedge \forall y' (\phi(x,y') \rightarrow y'=y)$$

Rigid guards

A formula **$\phi(x,y)$** is **rigid** if it defines a partial one-to-one map over positions.

Examples: '**$first(x) \wedge last(y)$**', '**$succ(x,y)$**',
'**$a(x) \wedge a(y) \wedge x<y \wedge \forall z(x<z<y \rightarrow \neg a(x))$**'

Remark: Rigidity can be first-order enforced as follows

$$\phi'(x,y) := \phi(x,y) \wedge \forall x' (\phi(x',y) \rightarrow x'=x) \wedge \forall y' (\phi(x,y') \rightarrow y'=y)$$

Rigid FO (resp. **rigid MSO**): all equivalence tests are of the form
$\phi(x,y) \wedge x \sim y$ in which $\phi(x,y)$ is rigid (no other free variables).

Rigid guards

A formula **$\phi(x,y)$** is **rigid** if it defines a partial one-to-one map over positions.

Examples: '**$\mathrm{first}(x) \wedge \mathrm{last}(y)$**', '**$\mathrm{succ}(x,y)$**',
'**$a(x) \wedge a(y) \wedge x<y \wedge \forall z (x<z<y \rightarrow \neg a(x))$**'

Remark: Rigidity can be first-order enforced as follows

$$\phi'(x,y) := \phi(x,y) \wedge \forall x' (\phi(x',y) \rightarrow x'=x) \wedge \forall y' (\phi(x,y') \rightarrow y'=y)$$

Rigid FO (resp. **rigid MSO**): all equivalence tests are of the form
$\phi(x,y) \wedge x \sim y$ in which $\phi(x,y)$ is rigid (no other free variables).

Remark: Another derived construct is possible:

$$\boldsymbol{\phi(x,y) \rightarrow x \sim y} \quad = \quad (\boldsymbol{\phi(x,y)} \wedge \boldsymbol{x \sim y}) \vee \boldsymbol{\neg \phi(x,y)}$$

Rigid guards

A formula **$\phi(x,y)$** is **rigid** if it defines a partial one-to-one map over positions.

Examples: '**$\mathrm{first}(x) \wedge \mathrm{last}(y)$**', '**$\mathrm{succ}(x,y)$**',
'**$a(x) \wedge a(y) \wedge x<y \wedge \forall z (x<z<y \rightarrow \neg a(x))$**'

Remark: Rigidity can be first-order enforced as follows

$$\phi'(x,y) := \phi(x,y) \wedge \forall x' (\phi(x',y) \rightarrow x'=x) \wedge \forall y' (\phi(x,y') \rightarrow y'=y)$$

Rigid FO (resp. **rigid MSO**): all equivalence tests are of the form
$\phi(x,y) \wedge x \sim y$ in which $\phi(x,y)$ is rigid (no other free variables).

Remark: Another derived construct is possible:

$$\phi(x,y) \rightarrow x \sim y \quad = \quad (\phi(x,y) \wedge x \sim y) \vee \neg\phi(x,y)$$

Examples:

- The first and last values coincide
- Every odd position letters carry the same value
- There are two consecutive letters that carry the same value
- non-example: some value appears twice

{C., Ley, Puppis}

For languages of data words:

definable in rigid monadic second-order logic $\underset{\text{eff}}{=}$ recognizability by orbit finite nominal monoids

definable in rigid first-order logic $\underset{\text{eff}}{=}$ recognizability by aperiodic orbit finite nominal monoids

{C., Ley, Puppis}

For languages of data words:

definable in rigid monadic second-order logic $=_{\text{eff}}$ recognizability by orbit finite nominal monoids

definable in rigid first-order logic $=_{\text{eff}}$ recognizability by aperiodic orbit finite nominal monoids

?

Projection and powerset

How to translate MSO into monoids?

Projection and powerset

How to translate MSO into monoids?

Types (compositionality)

Projection and powerset

How to translate MSO into monoids?

Types (compositionality)

Language theoretic closure operations: Regular languages being closed under union, intersection, complement, projection and having suitable constants, captures MSO.

Projection and powerset

How to translate MSO into monoids?

Types (compositionality)

Language theoretic closure operations: Regular languages being closed under union, intersection, complement, projection and having suitable constants, captures MSO.

Let L be recognized by **M**,h,F

$h : (\{\mathbf{t},\mathbf{f}\}\times A)^* \rightarrow M$

truth value of variable X

Projection and powerset

How to translate MSO into monoids?

Types (compositionality)

Language theoretic closure operations: Regular languages being closed under union, intersection, complement, projection and having suitable constants, captures MSO.

Let L be recognized by **M**,h,F

$h : (\{\mathbf{t},\mathbf{f}\}\times A)^* \rightarrow M$

truth value of variable X

M,h,F recognizes $L(\phi(x))$

Projection and powerset

How to translate MSO into monoids?

Types (compositionality)

Language theoretic closure operations: Regular languages being closed under union, intersection, complement, projection and having suitable constants, captures MSO.

Let L be recognized by $\mathbf{M}$,h,F

$h : (\{\mathbf{t},\mathbf{f}\}\times A)^* \rightarrow M$

truth value of variable X

$\mathbf{M}$,h,F recognizes

$L(\phi(x))$

$\mathbf{P}(\mathbf{M}) = (\mathcal{P}(M), \circ, \{1\})$

$X \circ Y = \{(a \cdot b) : a \in X,\ b \in Y\}$

$\text{proj-h} : A^* \rightarrow \mathbf{P}(\mathbf{M})$

$a \longmapsto \{\ h(\mathbf{t},a),\ h(\mathbf{f},a)\ \}$

(freely completed)

Projection and powerset

How to translate MSO into monoids?

Types (compositionality)

Language theoretic closure operations: Regular languages being closed under union, intersection, complement, projection and having suitable constants, captures MSO.

Let L be recognized by **M**,h,F

$h : (\{\mathbf{t},\mathbf{f}\}\times A)^* \to M$

truth value of variable X

$\mathbf{P}(\mathbf{M}) = (\mathcal{P}(M),\circ,\{1\})$

$X\circ Y = \{(a\cdot b) : a\in X,\ b\in Y\}$

$\text{proj-h} : A^* \to \mathbf{P}(\mathbf{M})$

$a \longmapsto \{\ h(\mathbf{t},a),\ h(\mathbf{f},a)\ \}$

(freely completed)

M,h,F recognizes $L(\phi(x))$ $\Rightarrow$ P(M), proj-h, F' recognizes $L(\exists x.\phi(x))$

Projection and powerset

How to translate MSO into monoids?

Types (compositionality)

Language theoretic closure operations: Regular languages being closed under union, intersection, complement, projection and having suitable constants, captures MSO.

Let L be recognized by **M**,h,F

$h : (\{\mathbf{t},\mathbf{f}\}\times A)^* \to M$

truth value of variable X

$\mathbf{P}(\mathbf{M}) = (\mathcal{P}(M),\circ,\{1\})$

$X\circ Y = \{(a\cdot b) : a\in X,\ b\in Y\}$

$\text{proj-h} : A^* \to \mathbf{P}(\mathbf{M})$

$a \longmapsto \{\ h(\mathbf{t},a),\ h(\mathbf{f},a)\ \}$

(freely completed)

M,h,F recognizes $L(\phi(x))$ $\Rightarrow$ P(M), proj-h, F' recognizes $L(\exists x.\phi(x))$

Problem: The powerset does not preserve orbit finite sets.

Projectability

Projectability

nominal

nominal orbit finite

A equivariant map $h : A{\times}B \to C$ is **projectable** (w.r.t A) if whenever $a,a' \in A$,

$h(a,b)$ and $h(a',b)$ are in the same orbit, then $h(a,b)=h(a',b)$

Projectability

nominal

nominal orbit finite

A equivariant map $h : A \times B \to C$ is **projectable** (w.r.t A) if whenever $a, a' \in A$,

$h(a,b)$ and $h(a',b)$ are in the same orbit, then $h(a,b)=h(a',b)$

Lemma: If $h : A \times B \to C$ is projectable, then the map

$$B \longrightarrow \mathcal{P}(C)$$

$$b \longmapsto \{h(a,b) : a \in A\}$$

has an orbit finite image.

Projectability

nominal

nominal orbit finite

A equivariant map $h : A \times B \to C$ is **projectable** (w.r.t A) if whenever $a, a' \in A$,

$h(a,b)$ and $h(a',b)$ are in the same orbit, then $h(a,b)=h(a',b)$

Lemma: If $h : A \times B \to C$ is projectable, then the map

$$B \longrightarrow \mathcal{P}(C)$$

$$b \longmapsto \{h(a,b) : a \in A\}$$

has an orbit finite image.

Translating from Rigid MSO to nominal sets uses the standard construction, paying attention that all **morphisms** are projectable with respect to the variable dimensions.

Projectability

nominal

nominal orbit finite

A equivariant map $h : A \times B \to C$ is **projectable** (w.r.t A) if whenever $a, a' \in A$,

$h(a,b)$ and $h(a',b)$ are in the same orbit, then $h(a,b)=h(a',b)$

Lemma: If $h : A \times B \to C$ is projectable, then the map

$$B \longrightarrow \mathcal{P}(C)$$

$$b \longmapsto \{h(a,b) : a \in A\}$$

has an orbit finite image.

Translating from Rigid MSO to nominal sets uses the standard construction, paying attention that all **morphisms** are projectable with respect to the variable dimensions.

Rigidly guarded test operation produces projectable monoids.

{C., Ley, Puppis}

For languages of data words:

definable in rigid monadic second-order logic	$=_{\mathrm{eff}}$	recognizability by orbit finite nominal monoids
definable in rigid first-order logic	$=_{\mathrm{eff}}$	recognizability by aperiodic orbit finite nominal monoids

{C., Ley, Puppis}

For languages of data words:

definable in rigid monadic second-order logic $=_{\text{eff}}$ recognizability by orbit finite nominal monoids

definable in rigid first-order logic $=_{\text{eff}}$ recognizability by aperiodic orbit finite nominal monoids

?

Nested use of guards

Nested use of guards

Counting: **C(k)** = ‘words that contain exactly k distinct data values’

Nested use of guards

Counting: **C(k)** = ‘words that contain exactly k distinct data values’

value1(x,y) =

$x \leq y$

and for all positions $x', y' \in [x,y]$ **successor**$(x,y) \rightarrow x \sim y$

The word assumes exactly one data value in the interval $[x,y]$

Nested use of guards

Counting: **C(k)** = ‘words that contain exactly k distinct data values’

value1(x,y) =
$x \leq y$
and for all positions $x', y' \in [x,y]$ **successor**$(x,y) \rightarrow x \sim y$

The word assumes exactly one data value in the interval $[x,y]$

successor1(x,y) =
$x \leq y$
and **value1**$(x+1, y-1)$
and ¬**value1**$(x, y-1)$
and ¬**value1**$(x+1, y)$

Nested use of guards

Counting: **C(k)** = ‘words that contain exactly k distinct data values’

value1(x,y) =
$x \leq y$
and for all positions $x', y' \in [x,y]$ **successor**$(x,y) \rightarrow x \sim y$

The word assumes exactly one data value in the interval [x,y]

successor1(x,y) =
$x \leq y$
and **value1**$(x+1,y-1)$
and $\neg$**value1**$(x,y-1)$
and $\neg$**value1**$(x+1,y)$

x y
111 2 11111 3 11111

Nested use of guards

Counting: **C(k)** = ‘words that contain exactly k distinct data values’

value1(x,y) =
x≤y
and for all positions x’,y’∈[x,y] **successor**(x,y) → x~y

The word assumes exactly one data value in the interval [x,y]

successor1(x,y) =
x≤y
and **value1**(x+1,y-1)
and ¬**value1**(x,y-1)
and ¬**value1**(x+1,y)

x y
111 2 11111 3 11111

successor1 is **rigid**

Nested use of guards

Counting: **C(k)** = ‘words that contain exactly k distinct data values’

value1(x,y) =
x≤y
and for all positions x’,y’∈[x,y] **successor**(x,y) → x~y

The word assumes exactly one data value in the interval [x,y]

successor1(x,y) =
x≤y
and **value1**(x+1,y-1)
and ¬**value1**(x,y-1)
and ¬**value1**(x+1,y)

x y
111 2 11111 3 11111

successor1 is **rigid**

value2(x,y) =
x≤y
¬**value1**(x,y)
and for all positions x’,y’∈[x,y] **successor1**(x,y) → x~y

Nested use of guards

Counting: **C(k)** = ‘words that contain exactly k distinct data values’

value1(x,y) =
x≤y
and for all positions x’,y’∈[x,y] **successor**(x,y) ➞ x~y

The word assumes exactly one data value in the interval [x,y]

successor1(x,y) =
x≤y
and **value1**(x+1,y-1)
and ¬**value1**(x,y-1)
and ¬**value1**(x+1,y)

x y
111 2 11111 3 11111

successor1 is **rigid**

value2(x,y) =
x≤y
¬**value1**(x,y)
and for all positions x’,y’∈[x,y] **successor1**(x,y) ➞ x~y

And so on.

Nested use of guards

Counting: **C(k)** = ‘words that contain exactly k distinct data values’

value1(x,y) =
x≤y
and for all positions x’,y’∈[x,y] **successor**(x,y) ➞ x~y

The word assumes exactly one data value in the interval [x,y]

successor1(x,y) =
x≤y
and **value1**(x+1,y-1)
and ¬**value1**(x,y-1)
and ¬**value1**(x+1,y)

x y
111 2 11111 3 11111

successor1 is **rigid**

value2(x,y) =
x≤y
¬**value1**(x,y)
and for all positions x’,y’∈[x,y] **successor1**(x,y) ➞ x~y

And so on. This requires nesting of rigid guards.

Nested use of guards

Counting: **C(k)** = ‘words that contain exactly k distinct data values’

value1(x,y) =
x≤y
and for all positions x’,y’∈[x,y] **successor**(x,y) ➞ x~y

The word assumes exactly one data value in the interval [x,y]

successor1(x,y) =
x≤y
and **value1**(x+1,y-1)
and ¬**value1**(x,y-1)
and ¬**value1**(x+1,y)

x y
111 2 11111 3 11111

successor1 is **rigid**

value2(x,y) =
x≤y
¬**value1**(x,y)
and for all positions x’,y’∈[x,y] successor1(x,y) ➞ x~y

And so on. This requires nesting of rigid guards.

➔ The nominal monoid to rigid logic translation cannot be as ‘flat’ as usual for word languages.

Green’s relations

Green's relations

Fondamental theorem of aperiodic regular languages:

first-order logic $\underset{\text{eff}}{=}$ aperiodic monoids $\underset{\text{eff}}{=}$ counter-free automata $\underset{\text{eff}}{=}$ star free = LTL = …

Green's relations

Fondamental theorem of aperiodic regular languages:

first-order logic $\overset{=}{\text{eff}}$ aperiodic monoids $\overset{=}{\text{eff}}$ counter-free automata $\overset{=}{\text{eff}}$ star free = LTL = …

{Schützenberger}: Works by understanding the structure of ideals of a monoid: Green's relations.

Green's relations

Fondamental theorem of aperiodic regular languages:

first-order logic $\underset{\text{eff}}{=}$ aperiodic monoids $\underset{\text{eff}}{=}$ counter-free automata $\underset{\text{eff}}{=}$ star free = LTL = …

{Schützenberger}: Works by understanding the structure of ideals of a monoid: Green's relations.

{C., Ley, Puppis}
For languages of data words:

definable in rigid monadic second-order logic $\underset{\text{eff}}{=}$ recognizability by orbit finite nominal monoids

definable in rigid first-order logic $\underset{\text{eff}}{=}$ recognizability by aperiodic orbit finite nominal monoids

Green's relations

Fondamental theorem of aperiodic regular languages:

first-order logic $\underset{\text{eff}}{=}$ aperiodic monoids $\underset{\text{eff}}{=}$ counter-free automata $\underset{\text{eff}}{=}$ star free = LTL = …

{Schützenberger}: Works by understanding the structure of ideals of a monoid: Green's relations.

{C., Ley, Puppis}

For languages of data words:

definable in rigid monadic second-order logic $\underset{\text{eff}}{=}$ recognizability by orbit finite nominal monoids

definable in rigid first-order logic $\underset{\text{eff}}{=}$ recognizability by aperiodic orbit finite nominal monoids

Uses the same technique, and requires to understand Green's relations in a nominal orbit finite monoid (even more for MSO!).

Conclusion

Conclusion

It is possible to recover part of the **fundamental theorem** of regular languages in the **nominal setting**. This is the only known case of equality of classes.

Conclusion

It is possible to recover part of the **fundamental theorem** of regular languages in the **nominal setting**. This is the only known case of equality of classes.

This requires to properly **guard** which data values can be compared. The notion of **rigidity** (definable bijective maps) is the correct one.

Conclusion

It is possible to recover part of the **fundamental theorem** of regular languages in the **nominal setting**. This is the only known case of equality of classes.

This requires to properly **guard** which data values can be compared. The notion of **rigidity** (definable bijective maps) is the correct one.

At a technical level, the notion of **projectability of maps** captures the essence of the interest of such guards.

Conclusion

It is possible to recover part of the **fundamental theorem** of regular languages in the **nominal setting**. This is the only known case of equality of classes.

This requires to properly **guard** which data values can be compared. The notion of **rigidity** (definable bijective maps) is the correct one.

At a technical level, the notion of **projectability of maps** captures the essence of the interest of such guards.

This study requires the development of results concerning **Green's relations** in orbit finite nominal set.

Conclusion

It is possible to recover part of the **fundamental theorem** of regular languages in the **nominal setting**. This is the only known case of equality of classes.

This requires to properly **guard** which data values can be compared. The notion of **rigidity** (definable bijective maps) is the correct one.

At a technical level, the notion of **projectability of maps** captures the essence of the interest of such guards.

This study requires the development of results concerning **Green's relations** in orbit finite nominal set.

A consequence is that every orbit finite monoid is the quotient of one that has a permutation free presentation.

Thank you!

Thanks to the organizers of the workshop.

Thanks to the organizers of the program.

Thanks to Simons Institute.