Positive Data Languages

28th August 2023

Florian Frank, Stefan Milius, and Henning Urbat

48th International Symposium on Mathematical Foundations of Computer Science

Chair for Theoretical Computer Science
Friedrich-Alexander-Universität Erlangen-Nürnberg

Friedrich-Alexander-Universität
Faculty of Engineering

What is 'Positivity'?

$\mathbb{D}$: Admissible User IDs for a Server ($\rightsquigarrow$ *Infinite Set*)

What is ‘Positivity’?

'last user has not logged in before'

$$L_0 = \{ \, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n \, \}$$

$\mathbb{D}$: Admissible User IDs for a Server ($\rightsquigarrow$ *Infinite Set*)

What is ‘Positivity’?

$\mathbb{D}$: Admissible User IDs for a Server ($\rightsquigarrow$ *Infinite Set*)

‘last user has not logged in before’

$$L_0 = \{\, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n \,\}$$

$$L_1 = \{\, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j \,\}$$

‘some user has logged in twice’

What is 'Positivity'?

'last user has not logged in before'

$$L_0 = \{\, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n \,\}$$

$\mathbb{D}$: Admissible User IDs for a Server ($\rightsquigarrow$ *Infinite Set*)

$\rightsquigarrow$ Both languages involve assertions of (in-)equality of data values!

$$L_1 = \{\, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j \,\}$$

'some user has logged in twice'

What is ‘Positivity’?

$\mathbb{D}$: Admissible User IDs for a Server ($\rightsquigarrow$ *Infinite Set*)

‘last user has not logged in before’

$$L_0 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n\ \}$$

What happens if we restrict this to just equalities?

$\rightsquigarrow$ Both languages involve assertions of (in-)equality of data values!

$$L_1 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j\ \}$$

‘some user has logged in twice’

What is ‘Positivity’?

Definition (*Positive Data Language* **)**

A *positive data language* is closed under arbitrary renamings $\rho\colon \mathbb{D} \to \mathbb{D}$ of data values.

$\rightsquigarrow$ Intuitively, positive data languages should model properties of data words which are expressible by assertions of equality, yet not inequality.

What is ‘Positivity’?

Definition (***Positive Data Language*** **)**

A *positive data language* is closed under arbitrary renamings $\rho\colon \mathbb{D} \to \mathbb{D}$ of data values.

⇝ Intuitively, positive data languages should model properties of data words which are expressible by assertions of equality, yet not inequality.

$$L_0 = \{\, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n \,\}$$

not positive

What is 'Positivity'?

Definition (*Positive Data Language* **)**

A *positive data language* is closed under arbitrary renamings $\rho\colon \mathbb{D} \to \mathbb{D}$ of data values.

⇝ Intuitively, positive data languages should model properties of data words which are expressible by assertions of equality, yet not inequality.

$$L_0 = \{\, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n \,\}$$

not positive

$$L_1 = \{\, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j \,\}$$

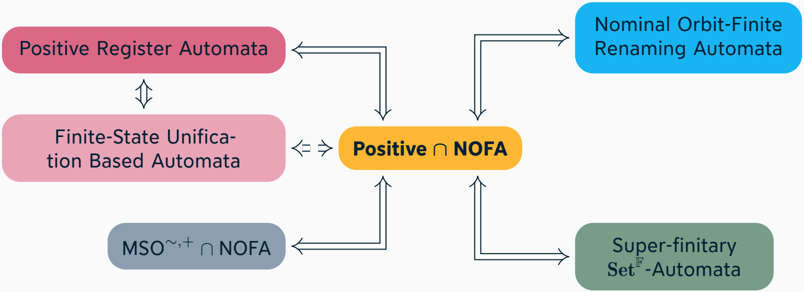
positive

What is ‘Positivity’?

Definition (*Positive Data Language* **)**

A *positive data language* is closed under arbitrary renamings $\rho\colon \mathbb{D} \to \mathbb{D}$ of data values.

$\rightsquigarrow$ Intuitively, positive data languages should model properties of data words which are expressible by assertions of equality, yet not inequality.



Nominal Sets (Talking about Infinite Data with Finite Means)

T.CS

$\rightsquigarrow$ Fix a (countably infinite) set $\mathbb{D}$ of 'names'. ⟵ Data Values

Definition (***Nominal Sets*** **)** **Gabbay, Pitts '99**

A *nominal set* is a set whose elements depend on a *finite* number of these names.
$\to \mathrm{supp}(x)$

$\rightsquigarrow$ We can change the names of an element using permutations $\pi \colon \mathbb{D} \xrightarrow{\simeq} \mathbb{D}$ which act upon these elements.

```
<book id="bk007">
   <author lstname="Doe"
      fstname="John"/>
   <title value="Biggy"/>
   <price cur="USD"
      amount="12.95"/>
</book>
```

Nominal Sets (Talking about Infinite Data with Finite Means)

- ⇝ Fix a (countably infinite) set $\mathbb{D}$ of ‘names’. ⟵ Data Values

Definition (*Nominal Sets*) Gabbay, Pitts ’99

A *nominal set* is a set whose elements depend on a *finite* number of these names.
$\to \operatorname{supp}(x)$

- ⇝ We can change the names of an element using permutations $\pi\colon \mathbb{D} \xrightarrow{\simeq} \mathbb{D}$ which act upon these elements.

Support

```
<book id="bk007">
   <author lstname="Doe"
      fstname="John"/>
   <title value="Biggy"/>
   <price cur="USD"
      amount="12.95"/>
</book>
```

Nominal Sets (Talking about Infinite Data with Finite Means)

$\rightsquigarrow$ Fix a (countably infinite) set $\mathbb{D}$ of 'names'. ⟵ Data Values

Definition (*Nominal Sets*) Gabbay, Pitts '99

A *nominal set* is a set whose elements depend on a *finite* number of these names. $\to \mathrm{supp}(x)$

$\rightsquigarrow$ We can change the names of an element using permutations $\pi\colon \mathbb{D} \xrightarrow{\simeq} \mathbb{D}$ which act upon these elements.

```
<book id="bk007">
   <author lstname="Doe"
      fstname="John"/>
   <title value="Biggy"/>
   <price cur="USD"
      amount="12.95"/>
</book>
```

- Proper 'finiteness' is now replaced by finiteness up to such permutations. $\rightsquigarrow$ *Orbit-Finiteness*
- Nominal Sets and permutation-preserving maps form a category $\mathbf{Nom}$.

$\rightsquigarrow$ Fix a (countably infinite) set $\mathbb{D}$ of 'names'. ← Data Values

Definition (*Nominal Sets*) Gabbay, Pitts '99

A *nominal set* is a set whose elements depend on a *finite* number of these names.
$\to \mathrm{supp}(x)$

$\rightsquigarrow$ We can change the names of an element using permutations $\pi\colon \mathbb{D} \xrightarrow{\simeq} \mathbb{D}$ which act upon these elements.

```
<book id="bk007">
   <author lstname="Doe"
      fstname="John"/>
   <title value="Biggy"/>
   <price cur="USD"
      amount="12.95"/>
</book>
```

- Proper 'finiteness' is now replaced by finiteness up to such permutations. $\rightsquigarrow$ *Orbit-Finiteness*
- Nominal Sets and permutation-preserving maps form a category $\mathbf{Nom}$.

$\rightsquigarrow$ Nominal Sets can also be understood as specific presheaves. $(\mathbf{Nom} \leftrightsquigarrow \mathbf{Set}^{\mathbb{I}})$

Nominal Sets (Talking about Infinite Data with Finite Means)

$\rightsquigarrow$ Fix a (countably infinite) set $\mathbb{D}$ of ‘names’. ⟵ Data Values

Definition (***Nominal Sets*** **)** — **Gabbay, Pitts ’99**

A *nominal set* is a set whose elements depend on a *finite* number of these names.
$\rightarrow \mathrm{supp}(x)$

$\rightsquigarrow$ We can change the names of an element using permutations $\pi\colon \mathbb{D} \xrightarrow{\simeq} \mathbb{D}$ which act upon these elements.

```
<book id="bk007">
  <title value="Biggy"/>
  <price cur="USD"
     amount="12.95"/>
</book>
```

- Proper ‘finiteness’ is now replaced

What happens if we give up injectivity of these permutations like before?
$\rightarrow$ $\mathbf{RnNom}$ (Renaming Nominal Sets, Gabbay, Hofmann ’08)

- Nominal Sets and permutation-preserving maps form a category $\mathbf{Nom}$.

$\rightsquigarrow$ Nominal Sets can also be understood as specific presheaves. $(\mathbf{Nom} \rightsquigarrow \mathbf{Set}^{\mathbb{I}})$

Definition (*NOF A*) **Bojańczyk, Klin, Lasota '14**

A *nondeterministic orbit-finite* *automaton* $A = (Q, \delta, I, F)$ consists of:

- an orbit-finite nominal set $Q \in \mathbf{Nom}$ specifying *states*;
- an equivariant *transition* relation $\delta \subseteq Q \times \mathbb{D} \times Q$;
- equivariant sets $I \subseteq Q$ and $F \subseteq Q$ specifying *initial* and *final* states.

Acceptance of words $w \in \mathbb{D}^\star$ is defined classically over runs.

$$L_0 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n\ \}$$

$$L_1 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j\ \}$$

Nominal Automata (NOFAs & NOFRAs)

Definition (*NOF A*) **Bojańczyk, Klin, Lasota '14**

A *nondeterministic orbit-finite automaton* $A = (Q, \delta, I, F)$ consists of:

- an orbit-finite nominal set $Q \in \mathbf{Nom}$ specifying *states*;
- an equivariant *transition relation* $\delta \subseteq Q \times \mathbb{D} \times Q$;
- equivariant sets $I \subseteq Q$ and $F \subseteq Q$ specifying *initial*

Acceptance of words $w \in \mathbb{D}^\star$ is defined classically

closed under
arbitrary permutations
$\pi : \mathbb{D} \xrightarrow{\cong} \mathbb{D}$

$$L_0 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n\ \}$$

$$L_1 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j\ \}$$

Nominal Automata (NOFAs & NOFRAs)

Definition (*NOF A*) Bojańczyk, Klin, Lasota '14

A *nondeterministic orbit-finite automaton* $A = (Q, \delta, I, F)$ consists of:

- an orbit-finite nominal set $Q \in \mathbf{Nom}$ specifying *states*;
- an equivariant *transition* relation $\delta \subseteq Q \times \mathbb{D} \times Q$;
- equivariant sets $I \subseteq Q$ and $F \subseteq Q$ specifying *initial* and *final* states.

Acceptance of words $w \in \mathbb{D}^\star$ is defined classically over runs.

$$L_0 = \{ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n \}$$

L_0 and L_1 are both NOFA-recognizable.

$$L_1 = \{ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j \}$$

Nominal Automata (NOFAs & NOFRAs)

Definition (*NOFRA*)

A *nondeterministic orbit-finite renaming automaton* $A = (Q, \delta, I, F)$ consists of:

- an orbit-finite renaming nominal set $Q \in \mathbf{RnNom}$ specifying *states*;
- an equivariant *transition* relation $\delta \subseteq Q \times \mathbb{D} \times Q$;
- equivariant sets $I \subseteq Q$ and $F \subseteq Q$ specifying *initial* and *final* states.

Acceptance of words $w \in \mathbb{D}^\star$ is defined classically over runs.

$$L_0 = \{ \ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n \ \}$$

$$L_1 = \{ \ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j \ \}$$

Nominal Automata (NOFAs & NOFRAs)

Definition (*NOFRA*)

A *nondeterministic orbit-finite renaming automaton* $A = (Q, \delta, I, F)$ consists of:

- an orbit-finite renaming nominal set $Q \in \mathbf{RnNom}$ specifying *states*;
- an equivariant *transition relation* $\delta \subseteq Q \times \mathbb{D} \times Q$;
- equivariant sets $I \subseteq Q$ and $F \subseteq Q$ specifying *initial* a

Acceptance of words $w \in \mathbb{D}^\star$ is defined classically ov

closed under
arbitrary renamings

$$L_0 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n\ \}$$

$$L_1 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j\ \}$$

Nominal Automata (NOFAs & NOFRAs)

Definition (*NOFRA*)

A *nondeterministic orbit-finite renaming automaton* $A = (Q, \delta, I, F)$ consists of:

- an orbit-finite renaming nominal set $Q \in \mathbf{RnNom}$ specifying *states*;
- an equivariant *transition* relation $\delta \subseteq Q \times \mathbb{D} \times Q$;
- equivariant sets $I \subseteq Q$ and $F \subseteq Q$ specifying *initial* and *final* states.

Acceptance of words $w \in \mathbb{D}^\star$ is defined classically over runs.

$$\cancel{L_0 = \{\, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n \,\}}$$

Only L_1 is NOFRA-recognizable.

$$L_1 = \{\, d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j \,\}$$

Definition (*NOFRA*)

A *nondeterministic orbit-finite renaming automaton* $A = (Q, \delta, I, F)$ consists of:

- an orbit-finite renaming nominal set $Q \in \mathbf{RnNom}$ specifying *states*;
- an equivariant *transition* relation $\delta \subseteq Q \times \mathbb{D} \times Q$;
- equivariant sets $I \subseteq Q$ and $F \subseteq Q$ specifying *initial* and *final* states.

Acceptance of words $w \in \mathbb{D}^\star$ is defined classically over runs.

Proposition (*Positive Languages*)

Every NOFRA accepts a positive language.

Nominal Automata (NOFAs & NOFRAs)

Definition (*NOFRA*)

A *nondeterministic orbit-finite renaming automaton* $A = (Q, \delta, I, F)$ consists of:

- an orbit-finite renaming nominal set $Q \in \mathbf{RnNom}$ specifying *states*;
- an equivariant *transition* relation $\delta \subseteq Q \times \mathbb{D} \times Q$;
- equivariant sets $I \subseteq Q$ and $F \subseteq Q$ specifying *initial* and *final* states.

Acceptance of words $w \in \mathbb{D}^{\star}$ is defined classically over runs.

Proposition (*Positive Languages*)

Every NOFRA accepts a positive language.

Theorem (*First Equivalence*)

A language is positive and NOFA-recognizable iff it is recognized by a NOFRA.

A Slight Problem

The state set of NOFRAs is not truly finite, but just *orbit-finite*.

Store the names of states now explicitly in a finite amount of registers.

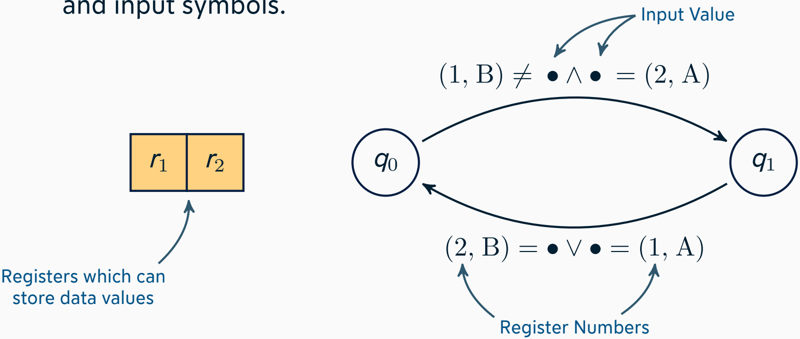
$\rightsquigarrow$ **Idea (**Kaminski, Francez '94, Bojańczyk, Klin, Lasota '14**):**
Change transitions to Boolean formulae of equations of register values and input symbols.

A Slight Problem

The state set of NOFRAs is not truly finite, but just *orbit-finite*.

Store the names of states now explicitly in a finite amount of registers.

⇝ **Idea (Kaminski, Francez '94, Bojańczyk, Klin, Lasota '14):** Change transitions to Boolean formulae of equations of register values and input symbols.

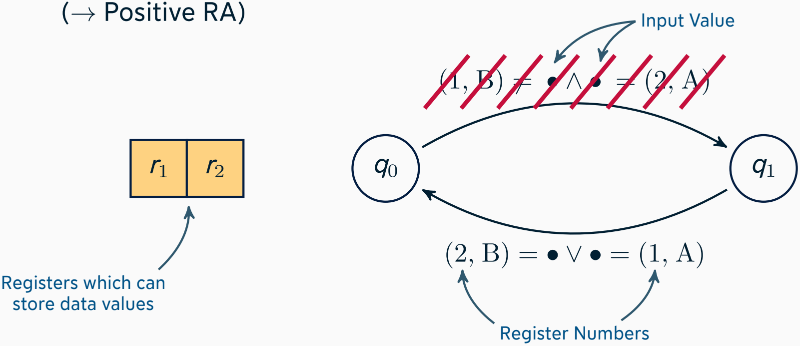


A Slight Problem

The state set of NOFRAs is not truly finite, but just *orbit-finite*.

Store the names of states now explicitly in a finite amount of registers.

$\rightsquigarrow$ **Our Idea:**
Let the transition formulae be positive, i.e. without negation.
($\to$ Positive RA)



Theorem (***Second Equivalence*** **)**

A language is positive and NOFA-recognizable iff it is accepted by a positive RA.

Theorem (***Third Equivalence*** **)**

Positive register automata are equivalent to *finite-state unification based automata*.

Introduced by
Tal '99 and Kaminski, Tan '06

$\mathsf{MSO}^{\sim}$ Neven, Schwentick, Vianu '04

$$\phi, \psi \quad := \quad x < y \mid x \sim y \mid X(x) \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \exists x.\, \phi \mid \exists X.\, \phi \mid \forall x.\, \phi \mid \forall X.\, \phi$$

- Formulae are interpreted over a fixed data word $w = d_1 \cdots d_n \in \mathbb{D}^\star$.

MSO$^\sim$ Neven, Schwentick, Vianu '04

$$\phi, \psi \quad := \quad x < y \mid x \sim y \mid X(x) \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \exists x.\, \phi \mid \exists X.\, \phi \mid \forall x.\, \phi \mid \forall X.\, \phi$$

- Formulae are interpreted over a fixed data word $w = d_1 \cdots d_n \in \mathbb{D}^\star$.

$$L_0 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n\ \}$$

describes the last position

$$\varphi_0 = \forall y.\ \mathrm{last}(y) \Rightarrow (\forall x.\ x < y \Rightarrow \neg(x \sim y))$$

MSO$^{\sim}$ — Neven, Schwentick, Vianu '04

$$\phi,\psi \;:=\; x < y \mid x \sim y \mid X(x) \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \exists x.\,\phi \mid \exists X.\,\phi \mid \forall x.\,\phi \mid \forall X.\,\phi$$

- Formulae are interpreted over a fixed data word $w = d_1 \cdots d_n \in \mathbb{D}^\star$.

$$L_0 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i \neq d_n \text{ for all } i < n\ \}$$

describes the last position

$$\varphi_0 = \forall y.\ \mathrm{last}(y) \Rightarrow (\forall x.\ x < y \Rightarrow \neg(x \sim y))$$

$$L_1 = \{\ d_1 \cdots d_n \in \mathbb{D}^\star \mid d_i = d_j \text{ for some } i \neq j\ \}$$

$$\varphi_1 = \exists x.\ \exists y.\ x < y \wedge x \sim y$$

$\mathsf{MSO}^{\sim}$ — Neven, Schwentick, Vianu '04

$$\phi, \psi \quad := \quad x < y \mid x \sim y \mid X(x) \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \exists x.\, \phi \mid \exists X.\, \phi \mid \forall x.\, \phi \mid \forall X.\, \phi$$

- Formulae are interpreted over a fixed data word $w = d_1 \cdots d_n \in \mathbb{D}^\star$.

Definition ($MSO^{\sim,+}$)

We restrict $\mathsf{MSO}^{\sim}$ formulae to those whose NNF contains no subformula of the form $\neg(x \sim y)$.

MSO$^\sim$ — Neven, Schwentick, Vianu '04

$$\phi, \psi \quad := \quad x < y \mid x \sim y \mid X(x) \mid \neg\phi \mid \phi \vee \psi \mid \phi \wedge \psi \mid \exists x.\, \phi \mid \exists X.\, \phi \mid \forall x.\, \phi \mid \forall X.\, \phi$$

- Formulae are interpreted over a fixed data word $w = d_1 \cdots d_n \in \mathbb{D}^\star$.

Definition (*$MSO^{\sim,+}$* **)**

We restrict MSO$^\sim$ formulae to those whose NNF contains no subformula of the form $\neg(x \sim y)$.

Theorem (*Fourth Equivalence* **)**

A NOFA-recognizable language is positive iff it is definable within MSO$^{\sim,+}$.

There are equivalences of categories for both $\mathbf{Nom}$ and $\mathbf{RnNom}$:

Pullback-Preserving Presheaves

$$\mathbf{Nom} \simeq \mathsf{Sh}(\mathbf{Set}^{\mathbb{I}}) \qquad \text{and} \qquad \mathbf{RnNom} \simeq \mathsf{Sh}(\mathbf{Set}^{\mathbb{F}})$$

Pullback-Preserving Presheaves

Categorical Automata

There are equivalences of categories for both $\mathbf{Nom}$ and $\mathbf{RnNom}$:

Pullback-Preserving Presheaves

$$\mathbf{Nom} \simeq \mathsf{Sh}(\mathbf{Set}^{\mathbb{I}}) \qquad \text{and} \qquad \mathbf{RnNom} \simeq \mathsf{Sh}(\mathbf{Set}^{\mathbb{F}})$$

Pullback-Preserving Presheaves

$\hookrightarrow$ We developed a notion of NA on those presheaf categories as well.

There are equivalences of categories for both $\mathbf{Nom}$ and $\mathbf{RnNom}$:

Pullback-Preserving Presheaves

$$\mathbf{Nom} \simeq \mathsf{Sh}(\mathbf{Set}^{\mathbb{I}}) \qquad \text{and} \qquad \mathbf{RnNom} \simeq \mathsf{Sh}(\mathbf{Set}^{\mathbb{F}})$$

Pullback-Preserving Presheaves

$\hookrightarrow$ We developed a notion of NA on those presheaf categories as well.

Definition (***Nondeterministic $\mathscr{C}$-Automata*** **)**

A *nondeterministic $\mathscr{C}$-Automaton* $A = (Q, \Sigma, \delta, I, F)$ consists of:

- objects $Q \in \mathscr{C}$ (*states*) and $\Sigma \in \mathscr{C}$ (*input alphabet*);
- a subobject $m_\delta : \delta \rightarrowtail Q \times \Sigma \times Q$ specifying *transitions*; and
- subobjects $m_I : I \rightarrowtail Q$ and $m_F : F \rightarrowtail Q$ for *initial* and *final* states.

The accepted language is then a family of subobjects of Σ^n for each $n \in \mathbb{N}$ defined over generalized runs.

Example (*Instances of Categorical Automata*)

Classical NFA, NOFA, and NOFRA with $\Sigma = \mathbb{D}$ are all instances of categorical automata for $\mathscr{C} = \mathbf{Set}, \mathbf{Nom}, \mathbf{RnNom}.$

Example (*Instances of Categorical Automata* **)**

Classical NFA, NOFA, and NOFRA with $\Sigma = \mathbb{D}$ are all instances of categorical automata for $\mathscr{C} = \mathbf{Set}, \mathbf{Nom}, \mathbf{RnNom}.$

- Finiteness is expressed through the use of *finitely presentable objects*.

Example (*Instances of Categorical Automata* **)**

Classical NFA, NOFA, and NOFRA with $\Sigma = \mathbb{D}$ are all instances of categorical automata for $\mathscr{C} = \mathbf{Set}, \mathbf{Nom}, \mathbf{RnNom}$.

- Finiteness is expressed through the use of *finitely presentable objects*.
- While presheaf automata accept presheaf languages, we can also look at the accepted *word languages* (subsets of $\mathbb{D}^\star$).

Categorical Automata (Presheaf Automata)

Example (*Instances of Categorical Automata* **)**

Classical NFA, NOFA, and NOFRA with $\Sigma = \mathbb{D}$ are all instances of categorical automata for $\mathscr{C} = \mathbf{Set}, \mathbf{Nom}, \mathbf{RnNom}$.

- Finiteness is expressed through the use of *finitely presentable objects*.
- While presheaf automata accept presheaf languages, we can also look at the accepted *word languages* (subsets of $\mathbb{D}^\star$).

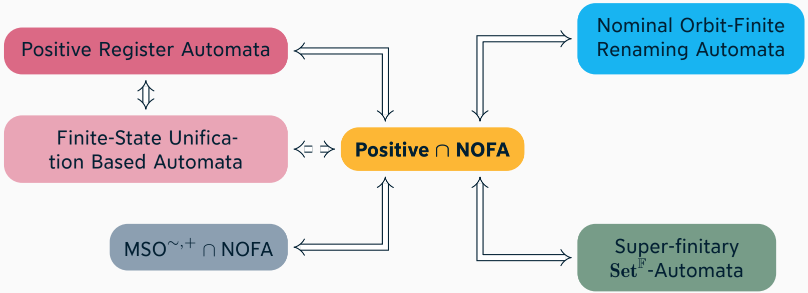
Theorem (*Fifth Equivalence* **)**

A word language is NOFA-recognizable iff it is accepted by a finitely presentable $\mathbf{Set}^{\mathbb{I}}$-automaton.

A word language is positive and NOFA-recognizable iff it is accepted by a finitely presentable $\mathbf{Set}^{\mathbb{F}}$-automaton.

Conclusion

- We looked at a restricted subclass of data languages, which has a rich theory and many equivalent perspectives: ($\rightsquigarrow$ Regular Languages)



- There are still open problems left:
  - Identifying a Suitable Fragment of MSO$^{\sim,+}$
  - Decidability Results
  - Expressive Power of MSO$^{\sim,+}$
  - Equivalences of Different Automata Models

References (1)

- Bojańczyk, Mikołaj, Bartek Klin, Sławomir Lasota. 'Automata theory in nominal sets'. *Log. Methods Comput. Sci.* 10.3 (2014).
- Gabbay, Murdoch J., Martin Hofmann. 'Nominal Renaming Sets'. *Proc. 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2008).* Springer, 2008, pp. 158–173. isbn: 9783540894384.
- Gabbay, Murdoch J., Andrew M. Pitts. 'A new approach to abstract syntax involving binders'. *Proc. 14th Annual IEEE Symposium on Logic in Computer Science (LICS 1999).* IEEE Computer Society, 1999, pp. 214–224.
- Kaminski, Michael, Nissim Francez. 'Finite-Memory Automata'. *Theor. Comput. Sci.* 134.2 (1994), pp. 329–363.
- Kaminski, Michael, Tony Tan. 'Regular Expressions for Languages over Infinite Alphabets'. *Fundam. Informaticae* 69.3 (2006), pp. 301–318.

References (2)

- Neven, Frank, Thomas Schwentick, Victor Vianu. 'Finite State Machines for Strings over Infinite Alphabets'. *ACM Trans. Comput. Logic* 5.3 (2004), pp. 403–435.
- Tal, A. 'Decidability of inclusion for unification based automata'. MA thesis. Department of Computer Science, Technion – Israel Institute of Technology, 1999.

Register Automata (Make Infinite Things Finite Again)

T.CS

Definition (*Register Automata*) Bojańczyk, Klin, Lasota '14

A *register automaton* $A = (C, m, \delta, I, F)$ consists of:

- a *finite* set C of control states;
- a number $m \in \mathbb{N}$ of registers;
- a transition relation $\delta \subseteq C \times \mathbb{B}(\Phi) \times C$, where

Boolean formulae with Φ as atoms.

$$\Phi = (\{1, \ldots, m\} \times \{\mathrm{BEF}, \mathrm{AFT}\} \cup \{\bullet\})^2 \text{; and}$$

Equations: Compare register values with one another or the input value ($\bullet$).

- sets $I \subseteq C$ and $F \subseteq C$ of *initial* and *final* states.

Configurations: (c, r) with $c \in C$ and $r \in (\mathbb{D} \cup \{\perp\})^m$ (partial assigments to registers)
A move $(c, r) \xrightarrow{a} (c', r')$ is defined iff it is *consistent* with some transition $c \xrightarrow{\varphi} c'$.
Acceptance is defined over runs of moves.

T.CS

Definition (*Positive Register Automata*)

A *positive register automaton* $A = (C, m, \delta, I, F)$ consists of:

- a *finite* set C of control states;
- a number $m \in \mathbb{N}$ of registers;
- a transition relation $\delta \subseteq C \times \mathbb{B}^+(\Phi) \times C$, where

Positive Boolean formulae (i.e. *no* negations) with Φ as atoms.

$$\Phi = (\{ 1, \dots, m \} \times \{ \mathrm{BEF}, \mathrm{AFT} \} \cup \{ \bullet \})^2 \text{ ; and}$$

Equations: Compare register values with one another or the input value ($\bullet$).

- sets $I \subseteq C$ and $F \subseteq C$ of *initial* and *final* states.

Configurations: (c, r) with $c \in C$ and $r \in (\mathbb{D} \cup \{ \perp \})^m$ (partial assigments to registers)
A move $(c, r) \xrightarrow{a} (c', r')$ is defined iff it is *consistent* with some transition $c \xrightarrow{\varphi} c'$.
Acceptance is defined over runs of moves.