

Investigations into Automata for Data Languages

Presentation of a Dissertation Plan

20th September 2023

Florian Frank

Cybercrime Workshop 2023 in Waischenfeld

Research and Training Group 2475 — 'Cybercrime and Forensic Computing'
Friedrich-Alexander-Universität Erlangen-Nürnberg



Cybercrime and
Forensic Computing
Research Training Group 2475



Friedrich-Alexander-Universität
Faculty of Engineering



INVESTIGATIONS INTO AUTOMATA FOR DATA LANGUAGES

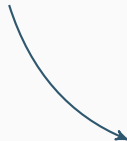


INVESTIGATIONS INTO AUTOMATA FOR DATA LANGUAGES





INVESTIGATIONS INTO AUTOMATA FOR DATA LANGUAGES





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



'last user has not logged in before'



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

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



'last user has not logged in before'

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

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

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

'some user has logged in twice'



'last user has not logged in before'



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

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

\rightsquigarrow Both languages involve assertions of (IN-)EQUALITY of data values!

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



'some user has logged in twice'



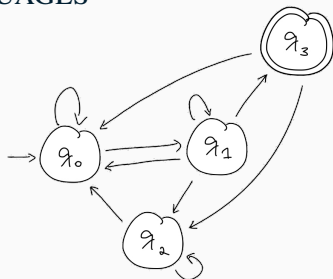
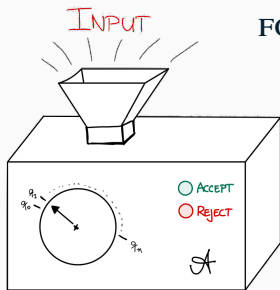
INVESTIGATIONS INTO AUTOMATA FOR DATA LANGUAGES



Nominal Techniques

INVESTIGATIONS INTO **AUTOMATA**

FOR DATA LANGUAGES





↪ 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.
 $\rightarrow \text{supp}(x)$

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

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



↪ 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.

→ $\text{supp}(x)$

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

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

Support



↪ 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.
→ $\text{supp}(x)$

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

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

» Proper 'finiteness' is now replaced by finiteness up to such permutations.

↪ *Orbit-Finiteness*



INVESTIGATIONS INTO AUTOMATA FOR DATA LANGUAGES



01

MFCS '23: Positive Data Languages

What happens with a restrictive class of data languages?



Positive Data Languages (Florian Frank, Stefan Milius and Henning Urbat)
In: *Proc. 48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*, Volume 272, pp. 48:1–48:15

DOI: 10.4230/LIPIcs.MFCS.2023.48



01

MFCS '23: Positive Data Languages

What happens with a restrictive class of data languages?

02

'Model Checking' for Data Languages with α -Equiv.

→ Streamlining finite and developing infinite words



01

MFCS '23: Positive Data Languages

What happens with a restrictive class of data languages?

02

'Model Checking' for Data Languages with α -Equiv.

→ Streamlining finite and developing infinite words

03

Unspecified Future Work

(see next slide for details)



01

MFCS '23: Positive Data Languages

What happens with a restrictive class of data languages?

02

'Model Checking' for Data Languages with α -Equiv.

→ Streamlining finite and developing infinite words

03

Unspecified Future Work

(see next slide for details)

04

Submission in Sept '25



» Extending results to *non-standard symmetries*.

Rather than performing equality checks, we may examine the order of data values.



»» Extending results to *non-standard symmetries*.

Rather than performing equality checks, we may examine the order of data values.

»» **Problems with Model Checking:** Current Algorithms (Inclusion and Emptiness Checks) are terribly inefficient.



»» Extending results to *non-standard symmetries*.

Rather than performing equality checks, we may examine the order of data values.

»» **Problems with Model Checking:** Current Algorithms (Inclusion and Emptiness Checks) are terribly inefficient.

- » There are solutions to make classical model checking algorithms more efficient.

→ *Transferring these strategies to nominal automata might be a viable course of action.*



»» Extending results to *non-standard symmetries*.

Rather than performing equality checks, we may examine the order of data values.

»» **Problems with Model Checking:** Current Algorithms (Inclusion and Emptiness Checks) are terribly inefficient.

- » There are solutions to make classical model checking algorithms more efficient.

→ *Transferring these strategies to nominal automata might be a viable course of action.*

- » However, the techniques are INCOMPLETE.

→ *We might be able to use those for more general language classes.*



» Extending results to *non-standard symmetries*.

Rather than performing equality checks, we may examine the order of data values.

» **Problems with Model Checking:** Current Algorithms (Inclusion and Emptiness Checks) are terribly inefficient.

- » There are solutions to make classical model checking algorithms more efficient.

→ *Transferring these strategies to nominal automata might be a viable course of action.*

- » However, the techniques are INCOMPLETE.

→ *We might be able to use those for more general language classes.*

- » What happens when looking at 'positive data languages'? Are there efficient algorithms for these languages?



01

MFCS '23: Positive Data Languages

What happens with a restrictive class of data languages?

02

'Model Checking' for Data Languages with α -Equiv.

→ Streamlining finite and developing infinite words

03

'Unspecified' Future Work


Domine, quo vadis?

04

Submission in Sept '25



 Frank, Florian, Stefan Milius, Henning Urbat. **'Positive Data Languages'**. *Proc. 48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*. Ed. by Jérôme Leroux, Sylvain Lombardy, David Peleg. Vol. 272. LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Aug. 2023, 48:1–48:15. ISBN: 978-3-95977-292-1. DOI: 10.4230/LIPIcs.MFCS.2023.48. URL: <https://drops.dagstuhl.de/opus/volltexte/2023/18582>.

 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.