Alternating Nominal Automata with Name Allocation

07th January 2025

Florian Frank, Daniel Hausmann, Stefan Milius, Lutz Schröder and Henning Urbat

Oberseminar WS2024/25

Lehrstuhl für Theoretische Informatik 8
Friedrich-Alexander-Universität Erlangen-Nürnberg

Friedrich-Alexander-Universität
Faculty of Engineering

Motivation

Friedrich-Alexander-Universität
Faculty of Engineering

Motivation

- Consider sequences of user logins within a given time period on a server.

  Forming finite *data* words $a_1 \cdots a_n \in \mathbb{A}^\star$

$\mathbb{A}$: admissible user IDs for a server ($\rightsquigarrow$ *infinite set*)

Motivation

- Consider sequences of user logins within a given time period on a server.

  Forming finite *data* words $a_1 \cdots a_n \in \mathbb{A}^\star$

- Behaviour patterns can be modelled as data languages over $\mathbb{A}$:

$\mathbb{A}$: admissible user IDs for a server ($\rightsquigarrow$ *infinite set*)

- Consider sequences of user logins within a given time period on a server.

  Forming finite *data* words $a_1 \cdots a_n \in \mathbb{A}^\star$

- Behaviour patterns can be modelled as data languages over $\mathbb{A}$:

$\mathbb{A}$: admissible user IDs for a server ($\rightsquigarrow$ *infinite set*)

$$L_1 = \{a_1 \cdots a_n \in \mathbb{A}^\star \,:\, a_i = a_j \text{ for some } i \neq j\}$$

'some user has logged in twice'

'first pair of users is equal to last pair with only different users in between'

$$L_2 = \left\{a_1 \cdots a_n \in \mathbb{A}^\star \,:\, \left(\begin{array}{c} a_1 = a_{n-1} \wedge a_2 = a_n \wedge \\ \forall 2 \leqslant i < n-1.\, a_1 \neq a_i \wedge a_2 \neq a_{i+1} \end{array}\right)\right\}$$

Motivation

- Consider sequences of user logins within a given time period on a server.

  Forming finite *data* words $a_1 \cdots a_n \in \mathbb{A}^\star$

- Behaviour patterns can be modelled as data languages over $\mathbb{A}$:

$\mathbb{A}$: admissible user IDs for a server ($\rightsquigarrow$ *infinite set*)

$$L_1 = \{a_1 \cdots a_n \in \mathbb{A}^\star \ : \ a_i = a_j \text{ for some } i \neq j\}$$

'some user has logged in twice'

'first pair of users is equal to last pair with only different users in between'

$$L_2 = \left\{a_1 \cdots a_n \in \mathbb{A}^\star \ : \ \left(\begin{array}{c} a_1 = a_{n-1} \wedge a_2 = a_n \wedge \\ \forall 2 \leqslant i < n-1.\, a_1 \neq a_i \wedge a_2 \neq a_{i+1} \end{array}\right)\right\}$$

- **Now:** Model these patterns with explicit 'name binding' (*bar languages*)

- Introduce ‘variables’ to words/strings:

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{\mathsf{l}a \,:\, a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

Motivation (Name Binding /Allocation for Automata)

T.CS

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|a \,:\, a \in \mathbb{A}\} \quad\cong\quad \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

Motivation (Name Binding /Allocation for Automata)

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{ | a : a \in \mathbb{A} \} \cong \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \; := \; \mathbb{A} \; \cup \; \{|a : a \in \mathbb{A}\} \;\; \cong \;\; \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable'
with name a

- Examples:

Motivation (Name Binding /Allocation for Automata)

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{ |a : a \in \mathbb{A} \} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Examples:

$$\underbrace{|a}_{} \; b \; a \; \underbrace{|b}_{} \; a \; b \; \underbrace{|a}_{} \; b \; a$$

'variable' intros

Motivation (Name Binding /Allocation for Automata)

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{ |a : a \in \mathbb{A} \} \cong \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Examples:

$$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$$

Motivation (Name Binding /Allocation for Automata)

- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|a : a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name a

- Examples:

$$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$$

$$n\ b\ n\ d\ n\ d\ a\ d\ a$$

$$h\ b\ h\ a\ h\ a\ a\ a\ a$$

- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name a

- Examples:

$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$

$n\ b\ n\ d\ n\ d\ a\ d\ a$ $\quad \boxed{a}\boxed{}$

$h\ b\ h\ a\ h\ a\ a\ a\ a$ $\quad \boxed{a}\boxed{}$

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|a : a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Examples:

$$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$$

$n\ b\ n\ d\ n\ d\ a\ d\ a$ $\quad$ $\boxed{a \mid n}$

$h\ b\ h\ a\ h\ a\ a\ a\ a$ $\quad$ $\boxed{a \mid h}$

Motivation (Name Binding /Allocation for Automata)

T.CS

- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{{|}a \;:\; a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name *a*

- Examples:

$${|}a \;\; b \;\; a \;\; {|}b \;\; a \;\; b \;\; {|}a \;\; b \;\; a$$

$$n \;\; \boxed{b} \;\; n \;\; d \;\; n \;\; d \;\; a \;\; d \;\; a$$

a	n

$$h \;\; \boxed{b} \;\; h \;\; a \;\; h \;\; a \;\; a \;\; a \;\; a$$

a	h

Motivation (Name Binding /Allocation for Automata)

» Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name *a*

» Examples:

$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$

$n\ b\ n\ d\ n\ d\ a\ d\ a$ $\quad \boxed{a \mid n}$

$h\ b\ h\ a\ h\ a\ a\ a\ a$ $\quad \boxed{a \mid h}$

Motivation (Name Binding /Allocation for Automata)

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{{|a} : a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Examples:

$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$

$n\ b\ n\ d\ n\ d\ a\ d\ a$ | a n | b

$h\ b\ h\ a\ h\ a\ a\ a\ a$ | a h | b

Motivation (Name Binding /Allocation for Automata)

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|a \,:\, a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Examples:

$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$

$n\ b\ n\ d\ n\ d\ a\ d\ a$

$h\ b\ h\ a\ h\ a\ a\ a\ a$

a	n
a	h

b	d
b	a

Motivation (Name Binding / Allocation for Automata)

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|a \,:\, a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name *a*

- Examples:

$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$

$n\ b\ n\ d\ n\ d\ a\ d\ a$

$h\ b\ h\ a\ h\ a\ a\ a\ a$

a	n
a	h

b	d
b	a

Motivation (Name Binding / Allocation for Automata)

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{{|}a \,:\, a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Examples:

$$|a \; b \; a \; |b \; a \; b \; |a \; b \; a$$

$$n \; b \; n \; d \; n \; d \; a \; d \; a$$

$$h \; b \; h \; a \; h \; a \; a \; a \; a$$

b	d

b	a

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Examples:

$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$

$n\ b\ n\ d\ n\ d\ a\ d\ a$ $\qquad$ | b | d |

$h\ b\ h\ a\ h\ a\ a\ a\ a$ $\qquad$ | b | a |

- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|a : a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name *a*

- Examples:

$$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$$

$n\ b\ n\ d\ n\ d\ a\ d\ a$ | a | | b | d

$h\ b\ h\ a\ h\ a\ a\ a\ a$ | a | | b | a

Motivation (Name Binding /Allocation for Automata)

- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name *a*

- Examples:

$$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$$

$$n\ b\ n\ d\ n\ d\ a\ d\ a$$

$$\cancel{n}\ \cancel{b}\ \cancel{n}\ \cancel{a}\ \cancel{n}\ \cancel{a}\ \cancel{a}\ \cancel{a}\ \cancel{a}$$

a	a
a	

b	d
b	a

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|a \,:\, a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name *a*

- Examples:

$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$

$n\ b\ n\ d\ n\ d\ a\ \boxed{d}\ a$

~~$n\ b\ n\ a\ n\ a\ a\ a\ a$~~

a	a

b	d

a	

b	a

Motivation (Name Binding /Allocation for Automata)

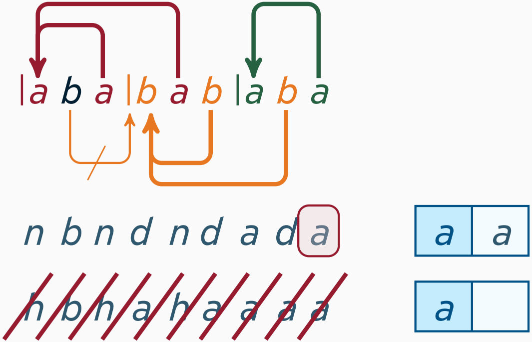
- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name *a*

- Examples:



Motivation (Name Binding /Allocation for Automata)

- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name a

- Examples:

$$|a\ b\ a\ |b\ a\ b\ |a\ b\ a \qquad\qquad |c\ b\ c\ |d\ c\ d\ |e\ d\ e$$

Motivation (Name Binding /Allocation for Automata)

- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name *a*

- Examples:

$|a\ b\ a\ |b\ a\ b\ |a\ b\ a$

$|c\ b\ c\ |d\ c\ d\ |e\ d\ e$

‘variable’ intros

Motivation (Name Binding /Allocation for Automata)

- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name a

- Examples:

$$|a\ b\ a\ |b\ a\ b\ |a\ b\ a \qquad\qquad |c\ b\ c\ |d\ c\ d\ |e\ d\ e$$

Motivation (Name Binding /Allocation for Automata)

- Introduce ‘variables’ to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or ‘variable names’

introduces ‘variable’ with name *a*

- Examples:

$$|a\ b\ a\ |b\ a\ b\ |a\ b\ a \equiv_\alpha |c\ b\ c\ |d\ c\ d\ |e\ d\ e$$

Motivation (Name Binding /Allocation for Automata)

» Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{ |a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name *a*

» Examples:

$$|a\; b\; a\; |b\; a\; b\; |a\; b\; a \quad \equiv_\alpha \quad |c\; b\; c\; |d\; c\; d\; |e\; d\; e$$

» These 'bar strings', i.e. words with 'variables', result in different kinds of languages:

» *Data Languages* (over $\mathbb{A}^\star$)

» *Literal Languages* (over $\overline{\mathbb{A}}^\star$)

» *Bar Languages* (over $\overline{\mathbb{A}}^\star / \equiv_\alpha$)

Motivation (Name Binding /Allocation for Automata)

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|a : a \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Examples:

$$|a\; b\; a\; |b\; a\; b\; |a\; b\; a \;\equiv_\alpha\; |c\; b\; c\; |d\; c\; d\; |e\; d\; e$$

- These 'bar strings', i.e. words with 'variables', result in different kinds of languages:

Conversion via Global/Local Freshness

- *Data Languages* (over $\mathbb{A}^\star$)
- *Literal Languages* (over $\overline{\mathbb{A}}^\star$)
- *Bar Languages* (over $\overline{\mathbb{A}}^\star / \equiv_\alpha$)

Motivation (Name Binding / Allocation for Automata)

T.CS

- Introduce 'variables' to words/strings:

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|a : a \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

data values or 'variable names'

introduces 'variable' with name a

- Examples:

$$|a\ b\ a\ |b\ a\ b\ |a\ b\ a \;\equiv_\alpha\; |c\ b\ c\ |d\ c\ d\ |e\ d\ e$$

- These 'bar strings', i.e. words with 'variables', result in different kinds of languages:

Conversion via Global/Local Freshness

- *Data Languages* (over $\mathbb{A}^\star$)
- *Literal Languages* (over $\overline{\mathbb{A}}^\star$)
- *Bar Languages* (over $\overline{\mathbb{A}}^\star / \equiv_\alpha$)

$\approx$

$\rightsquigarrow$ Fix a (countably infinite) set $\mathbb{A}$ 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. $\rightarrow \mathsf{supp}(x)$

$\rightsquigarrow$ We can change the names of an element using permutations $\pi\colon \mathbb{A} \xrightarrow{\simeq} \mathbb{A}$ which act upon these elements. ($\rightsquigarrow$ Group Actions $\triangleright\colon \mathsf{Perm}(\mathbb{A}) \times X \to X$)

```
<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 $\mathbb{A}$ 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 $\pi\colon \mathbb{A} \xrightarrow{\simeq} \mathbb{A}$ which act upon these elements. ($\rightsquigarrow$ Group Actions $\triangleright\colon \mathrm{Perm}(\mathbb{A}) \times X \to X$)

Support

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

⇝ Fix a (countably infinite) set $\mathbb{A}$ 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. $\to \mathrm{supp}(x)$

⇝ We can change the names of an element using permutations $\pi\colon \mathbb{A} \xrightarrow{\simeq} \mathbb{A}$ which act upon these elements. ($\rightsquigarrow$ Group Actions $\triangleright\colon \mathrm{Perm}(\mathbb{A}) \times X \to X$)

Support

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

Example: Let $\pi = (\text{John} \leftrightarrows \text{Jane}) \circ (\text{Doe} \leftrightarrows \text{Biggy})$ act on the document.

Nominal Sets (Talking about Infinite Data with Finite Means)

- $\rightsquigarrow$ Fix a (countably infinite) set $\mathbb{A}$ 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 $\pi\colon \mathbb{A} \xrightarrow{\simeq} \mathbb{A}$ which act upon these elements. ($\rightsquigarrow$ Group Actions $\triangleright\colon \mathrm{Perm}(\mathbb{A}) \times X \to X$)

Support

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

Example: Let $\pi = (\text{John}\ \text{Jane}) \circ (\text{Doe}\ \text{Biggy})$ act on the document.

Nominal Sets (Talking about Infinite Data with Finite Means)

TCS

⇝ Fix a (countably infinite) set $\mathbb{A}$ 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.
$\to \mathsf{supp}(x)$

⇝ We can change the names of an element using permutations $\pi\colon \mathbb{A} \xrightarrow{\simeq} \mathbb{A}$ which act upon these elements. (⇝ Group Actions $\triangleright\colon \mathsf{Perm}(\mathbb{A}) \times X \to X$)

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

⋙ 'Freshness': $\mathbb{A} \ni a\#x$ iff $a \notin \mathsf{supp}(x)$.

⋙ Proper 'finiteness' is now replaced by finiteness up to such permutations. ⇝ *Orbit-Finiteness*

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

$$\forall \pi.\ \forall x.\ f(\pi \triangleright x) = \pi \triangleright f(x)$$

Definition (*Abstraction*) Gabbay, Pitts '99

Given a nominal set X, define the equivalence relation $\approx_\alpha$ on $\mathbb{A} \times X$ as follows:

$$(a, x) \approx_\alpha (b, y) \text{ iff } \exists c \# (a, b, x, y).\ (a\ c) \triangleright x = (b\ c) \triangleright y. \tag{1}$$

With this, define the nominal set $[\mathbb{A}]X$ as the quotient $(\mathbb{A} \times X)/\approx_\alpha$, and denote the equivalence classes by $\langle a \rangle x$.

Bound Data Values (Name Abstraction)

Definition (*Abstraction*) — Gabbay, Pitts '99

Given a nominal set X, define the equivalence relation $\approx_\alpha$ on $\mathbb{A} \times X$ as follows:

$$(a, x) \approx_\alpha (b, y) \text{ iff } \exists c \# (a, b, x, y).\ (a\ c) \triangleright x = (b\ c) \triangleright y. \tag{1}$$

With this, define the nominal set $[\mathbb{A}]X$ as the quotient $(\mathbb{A} \times X)/\approx_\alpha$, and denote the equivalence classes by $\langle a \rangle x$.

Behaves similarly to '$\lambda a.\ x$', i.e. binds the name a in the 'term' x.

Bound Data Values (Name Abstraction)

Definition (*Abstraction*) **Gabbay, Pitts '99**

Given a nominal set X, define the equivalence relation $\approx_\alpha$ on $\mathbb{A} \times X$ as follows:

$$(a, x) \approx_\alpha (b, y) \text{ iff } \exists c \# (a, b, x, y).\ (a\ c) \triangleright x = (b\ c) \triangleright y. \tag{1}$$

With this, define the nominal set $[\mathbb{A}]X$ as the quotient $(\mathbb{A} \times X)/\approx_\alpha$, and denote the equivalence classes by $\langle a \rangle x$.

Behaves similarly to '$\lambda a.\ x$', i.e. binds the name a in the 'term' x.

⋙ $[\mathbb{A}]X$ extends to a endofunctor $[\mathbb{A}]-$ on $\mathbf{Nom}$.

T.CS

Definition (*Abstraction*) **Gabbay, Pitts '99**

Given a nominal set X, define the equivalence relation $\approx_\alpha$ on $\mathbb{A} \times X$ as follows:

$$(a, x) \approx_\alpha (b, y) \text{ iff } \exists c \mathrel{\#} (a, b, x, y).\ (a\ c) \triangleright x = (b\ c) \triangleright y. \tag{1}$$

With this, define the nominal set $[\mathbb{A}]X$ as the quotient $(\mathbb{A} \times X)/\approx_\alpha$, and denote the equivalence classes by $\langle a \rangle x$.

Behaves similarly to '$\lambda a.\ x$', i.e. binds the name a in the 'term' x.

- $[\mathbb{A}]X$ extends to a endofunctor $[\mathbb{A}]-$ on $\mathbf{Nom}$.
- This results in an 'easier' definition of α-equivalence:
  Substitution is pushed back to the definition of the nominal set.

Definition (*Abstraction* **)** **Gabbay, Pitts '99**

Given a nominal set X, define the equivalence relation $\approx_\alpha$ on $\mathbb{A} \times X$ as follows:

$$(a, x) \approx_\alpha (b, y) \text{ iff } \exists c \# (a, b, x, y).\ (a\ c) \triangleright x = (b\ c) \triangleright y. \tag{1}$$

With this, define the nominal set $[\mathbb{A}]X$ as the quotient $(\mathbb{A} \times X)/\approx_\alpha$, and denote the equivalence classes by $\langle a \rangle x$.

Behaves similarly to '$\lambda a.\ x$', i.e. binds the name a in the 'term' x.

- $[\mathbb{A}]X$ extends to a endofunctor $[\mathbb{A}]-$ on $\mathbf{Nom}$.

- This results in an 'easier' definition of α-equivalence:

  Substitution is pushed back to the definition of the nominal set.

Definition ($\equiv_\alpha$ *on Bar Strings* **)** **Schröder, Kozen, Milius, Wißmann '17**

$\equiv_\alpha$ is the equivalence generated by $w|av \equiv_\alpha w|bu$ if $\langle a \rangle v = \langle b \rangle u$ holds in $[\mathbb{A}]\overline{\mathbb{A}}^\star$

Bar Languages (With corresponding Data Languages)

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|d \,:\, d \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

- A name a is *free* in a bar string $w \in \overline{\mathbb{A}}^{\star}$, if a occurs to the left of any occurence of $|a$.

$$\overline{\mathbb{A}} := \mathbb{A} \cup \{|d : d \in \mathbb{A}\} \cong \mathbb{A} + \mathbb{A}$$

- A name a is *free* in a bar string $w \in \overline{\mathbb{A}}^\star$, if a occurs to the left of any occurence of $|a$.
- A bar string $w \in \overline{\mathbb{A}}^\star$ is *clean* if its bound letters $|a$ are mutually distinct and distinct from all its free names.

Bar Languages (With corresponding Data Languages)

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|d \,:\, d \in \mathbb{A}\} \quad\cong\quad \mathbb{A} + \mathbb{A}$$

- A name a is *free* in a bar string $w \in \overline{\mathbb{A}}^{\star}$, if a occurs to the left of any occurence of $|a$.
- A bar string $w \in \overline{\mathbb{A}}^{\star}$ is *clean* if its bound letters $|a$ are mutually distinct and distinct from all its free names.
- Bar Languages ($L \subseteq \overline{\mathbb{A}}^{\star}/\equiv_{\alpha}$) may also be understood as data languages via two conversions:

Bar Languages (With corresponding Data Languages)

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|d \,:\, d \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

- A name a is *free* in a bar string $w \in \overline{\mathbb{A}}^{\star}$, if a occurs to the left of any occurence of $|a$.
- A bar string $w \in \overline{\mathbb{A}}^{\star}$ is *clean* if its bound letters $|a$ are mutually distinct and distinct from all its free names.
- Bar Languages ($L \subseteq \overline{\mathbb{A}}^{\star}/\equiv_\alpha$) may also be understood as data languages via two conversions:

Global Freshness: $\mathrm{GF}(L) = \{\mathrm{ub}(w) \,:\, w \text{ clean},\; w \equiv_\alpha w' \in L\}$

Local Freshness: $\mathrm{LF}(L) = \{\mathrm{ub}(w) \,:\, w \equiv_\alpha w' \in L\}$

Bar Languages (With corresponding Data Languages)

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|d \;:\; d \in \mathbb{A}\} \;\;\cong\;\; \mathbb{A} + \mathbb{A}$$

- A name a is *free* in a bar string $w \in \overline{\mathbb{A}}^\star$, if a occurs to the left of any occurence of $|a$.
- A bar string $w \in \overline{\mathbb{A}}^\star$ is *clean* if its bound letters $|a$ are mutually distinct and distinct from all its free names.
- Bar Languages ($L \subseteq \overline{\mathbb{A}}^\star / \equiv_\alpha$) may also be understood as data languages via two conversions:

Global Freshness: $\mathrm{GF}(L) = \{\mathrm{ub}(w) \;:\; w \text{ clean}, \;\; w \equiv_\alpha w' \in L\}$

Local Freshness: $\mathrm{LF}(L) = \{\mathrm{ub}(w) \;:\; w \equiv_\alpha w' \in L\}$

$\mathrm{ub}(w)$ removes all '$|$'s in w by replacing all $|a$ in w with a.

Bar Languages (With corresponding Data Languages)

$$\overline{\mathbb{A}} \;:=\; \mathbb{A} \;\cup\; \{|d \;:\; d \in \mathbb{A}\} \;\cong\; \mathbb{A} + \mathbb{A}$$

- A name a is *free* in a bar string $w \in \overline{\mathbb{A}}^\star$, if a occurs to the left of any occurence of $|a$.
- A bar string $w \in \overline{\mathbb{A}}^\star$ is *clean* if its bound letters $|a$ are mutually distinct and distinct from all its free names.
- Bar Languages ($L \subseteq \overline{\mathbb{A}}^\star / \equiv_\alpha$) may also be understood as data languages via two conversions:

Global Freshness: $\mathrm{GF}(L) = \{\mathrm{ub}(w) \;:\; w \text{ clean}, \;\; w \equiv_\alpha w' \in L\}$

Local Freshness: $\mathrm{LF}(L) = \{\mathrm{ub}(w) \;:\; w \equiv_\alpha w' \in L\}$

$\mathrm{ub}(w)$ **removes all '|'s in w by replacing all $|a$ in w with a.**

Motivation

- Behaviour patterns can be modelled as data languages over $\mathbb{A}$ (or as bar languages):

$\mathbb{A}$: admissible user IDs for a server ($\rightsquigarrow$ *infinite set*)

Motivation

- Behaviour patterns can be modelled as data languages over $\mathbb{A}$ (or as bar languages):

$\mathbb{A}$: admissible user IDs for a server ($\rightsquigarrow$ *infinite set*)

Motivation

T.CS

- Behaviour patterns can be modelled as data languages over $\mathbb{A}$ (or as bar languages):

$\mathbb{A}$: admissible user IDs for a server ($\rightsquigarrow$ *infinite set*)

$$\mathsf{LF}(L_1) = \{a_1 \cdots a_n \in \mathbb{A}^\star \ : \ a_i = a_j \text{ for some } i \neq j\}$$

$$L_1 = [(|b)^*|a(|b)^* a(|b)^*]_\alpha$$

'some user has logged in twice'

'first pair of users is equal to last pair with only different users in between'

$$\mathsf{LF}(L_2) = \left\{a_1 \cdots a_n \in \mathbb{A}^\star \ : \ \left(\begin{array}{c} a_1 = a_{n-1} \wedge a_2 = a_n \wedge \\ \forall 2 \leqslant i < n-1.\ a_1 \neq a_i \wedge a_2 \neq a_{i+1} \end{array}\right)\right\}$$

$$L_2 = [|a|b(|c)^* ab]_\alpha$$

Alternation

Friedrich-Alexander-Universität
Faculty of Engineering

Motivation

Model checking with fixed-point/temporal logics over (in-)finite words usually uses alternating automata.

Motivation

Model checking with fixed-point/temporal logics over (in-)finite words usually uses alternating automata.

- **Goal:** Introduce *alternation* using transition formulae.

Motivation

Model checking with fixed-point/temporal logics over (in-)finite words usually uses alternating automata.

⋙ **Goal:** Introduce *alternation* using transition formulae.

Definition (*Boolean Formulae*)

Let X be a set of *atoms*. Then, $\mathcal{B}_{\mathsf{n}}(X)$ denotes the set of *Boolean formulae over X* defined by the grammar

$$\varphi, \psi ::= \top \mid \bot \mid x \mid \neg\varphi \mid \varphi \vee \psi \mid \varphi \wedge \psi. \qquad (x \in X)$$

Denote by $\mathcal{B}_{+}(X)$ the subset of *positive Boolean formulae over X*, i.e. formulae that do not contain any negation.

Motivation

Model checking with fixed-point/temporal logics over (in-)finite words usually uses alternating automata.

» **Goal:** Introduce *alternation* using transition formulae.

Definition (*Boolean Formulae*)

Let X be a set of *atoms*. Then, $\mathcal{B}_{\mathsf{n}}(X)$ denotes the set of *Boolean formulae over* X defined by the grammar

$$\varphi, \psi ::= \top \mid \bot \mid x \mid \neg\varphi \mid \varphi \vee \psi \mid \varphi \wedge \psi. \qquad (x \in X)$$

Denote by $\mathcal{B}_{+}(X)$ the subset of *positive Boolean formulae over* X, i.e. formulae that do not contain any negation.

» If X is a nominal set, then we regard $\mathcal{B}_{\mathsf{n}}(X)$ and $\mathcal{B}_{+}(X)$ also as nominal sets with the obvious group action.

Definition (*RANA*)

A *regular alternating nominal automaton* (*RANA*) $A = (Q, \delta, q_0)$ consists of:

- an orbit-finite set Q specifying *states*;
- an equivariant *initial state* $q_0 \in Q$; and
- an equivariant *transition function* $\delta\colon Q \to \mathcal{B}_{\mathsf{n}}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$.

Definition (*RANA*)

A *regular alternating nominal automaton* (*RANA*) $A = (Q, \delta, q_0)$ consists of:

- an orbit-finite set Q specifying *states*;
- an equivariant *initial state* $q_0 \in Q$; and
- an equivariant *transition function* $\delta\colon Q \to \mathcal{B}_{\mathsf{n}}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$.

Definition (*Positive RANA*)

A RANA is *positive* if the transition function corestricts to $\mathcal{B}_{+}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$.

An *extended regular nondeterministic nominal automaton* (*ERNNA*) is a positive RANA in which non of the transition formulae uses a conjunction ($\wedge$).

Definition (*RANA*)

A *regular alternating nominal automaton* (*RANA*) $A = (Q, \delta, q_0)$ consists of:

- an orbit-finite set Q specifying *states*;
- an equivariant *initial state* $q_0 \in Q$; and
- an equivariant *transition function* $\delta\colon\ Q \to \mathcal{B}_{\mathsf{n}}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$.

Definition (*Positive RANA*)

A RANA is *positive* if the transition function corestricts to $\mathcal{B}_{+}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$.

An *extended regular nondeterministic nominal automaton* (*ERNNA*) is a positive RANA in which non of the transition formulae uses a conjunction ($\wedge$).

Notation

- Denote the unique atom in 1 by ε.
- Denote the atoms (a, q) and $\langle a \rangle q$ by $\Diamond_a q$ and $\Diamond_{|a} q$, respectively.

Definition (*Semantics*)

Define $w \models \varphi$ for $w \in \overline{\mathbb{A}}^\star$, and $\varphi \in \mathcal{B}_\mathsf{n}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$ recursively by:

- $\vee$, $\wedge$, $\neg$, $\top$, and $\bot$ have the conventional interpretation.
- The interpretation of atoms $x \in \mathbb{A} \times Q + [\mathbb{A}]Q$ is given by the following clauses:

$$
\begin{aligned}
w \models \varepsilon &:\Longleftrightarrow\ w = \varepsilon \\
w \models \Diamond_a q &:\Longleftrightarrow\ \exists v \in \overline{\mathbb{A}}^\star.\ w = av \text{ and } v \models \delta(q) \\
w \models \Diamond_{|a} q &:\Longleftrightarrow\ \exists v, v' \in \overline{\mathbb{A}}^\star,\ b, c \in \mathbb{A},\ q' \in Q.\ w = {|bv} \equiv_\alpha {|cv'}, \\
&\qquad \langle a \rangle q = \langle c \rangle q',\ \text{and } v' \models \delta(q')
\end{aligned}
$$

Define the accepted languages as follows:

- **Literal Language:** $L_0(A) := \left\{ w \in \overline{\mathbb{A}}^\star\ :\ w \text{ is closed and } q_0 \text{ accepts } w \right\}$.
- **Bar Language:** $L_\alpha(A) := L_0(A)/\equiv_\alpha$.

RANA (Small Remarks)

- Our choice for transitions functions is deliberate:
  ($\delta\colon Q \to \mathcal{B}_{\mathsf{n}}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$ instead of final states & $\delta\colon Q \times \overline{\mathbb{A}} \to \mathcal{B}_{\mathsf{n}}(Q)$)
  - Encoding transition 'letters' into formulae makes proofs and constructions easier later on. Additionally, there'd be multiple extra conditions on δ.
  - Encoding 'finality' into formulae makes transition formulae more align to their logical counterpart Bar-μTL and our modalities match more closely to their logical counterparts.
    $\rightsquigarrow$ Otherwise, diamonds would need to accept ε depending on finality.

- Our choice for transitions functions is deliberate:
  $(\delta\colon Q \to \mathcal{B}_{\mathsf{n}}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$ instead of final states & $\delta\colon Q \times \overline{\mathbb{A}} \to \mathcal{B}_{\mathsf{n}}(Q))$
  - Encoding transition 'letters' into formulae makes proofs and constructions easier later on. Additionally, there'd be multiple extra conditions on δ.
  - Encoding 'finality' into formulae makes transition formulae more align to their logical counterpart Bar-μTL and our modalities match more closely to their logical counterparts.
    $\rightsquigarrow$ Otherwise, diamonds would need to accept ε depending on finality.
- Why do we need a possibility of α-renaming bar strings for 'bar-modalities' $\Diamond_{|a}$?

$$w \models \Diamond_{|a} q :\Longleftrightarrow \exists v, v' \in \overline{\mathbb{A}}^{\star},\ b, c \in \mathbb{A},\ q' \in Q.\ w = |bv \equiv_\alpha |cv',\\ \langle a \rangle q = \langle c \rangle q',\ \text{and}\ v' \models \delta(q')$$

- Our choice for transitions functions is deliberate:
  ($\delta\colon Q \to \mathcal{B}_{\mathrm{n}}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$ instead of final states & $\delta\colon Q \times \overline{\mathbb{A}} \to \mathcal{B}_{\mathrm{n}}(Q)$)
  - Encoding transition 'letters' into formulae makes proofs and constructions easier later on. Additionally, there'd be multiple extra conditions on δ.
  - Encoding 'finality' into formulae makes transition formulae more align to their logical counterpart Bar-μTL and our modalities match more closely to their logical counterparts.
    $\rightsquigarrow$ Otherwise, diamonds would need to accept ε depending on finality.
- Why do we need a possibility of α-renaming bar strings for 'bar-modalities' $\Diamond_{|a}$?

$$w \models \Diamond_{|a} q :\Longleftrightarrow \exists v, v' \in \overline{\mathbb{A}}^{\star},\ b, c \in \mathbb{A},\ q' \in Q.\ w = |bv \equiv_{\alpha} |cv',\ \langle a \rangle q = \langle c \rangle q',\ \text{and } v' \models \delta(q')$$

  - Otherwise, negation *is not* α-invariant! (Example at blackboard)
  - This would contradict the expected complementation procedure!

Results I: Equivalence of Models

Friedrich-Alexander-Universität
Faculty of Engineering

Equivalence of Models

We showed that negation does not matter for expresivity of RANAs, in more detail:

Theorem (*Equivalence* **)**

Positive RANAs accept the same bar languages as ordinary RANAs do.

Hence, also the same languages under the local/global freshness semantics.

Explicit-Dual RANAs

- To untangle the proof and for the later equivalence to Bar-μTL, we introduce another variant of RANAs:

Explicit-Dual RANAs

- To untangle the proof and for the later equivalence to Bar-μTL, we introduce another variant of RANAs:

Definition (*Dualizable Boolean Formulae* **)**

Let X be a (nominal) set of atoms.
Put $\mathcal{B}_{\mathsf{d}}(X) := \mathcal{B}_{+}(X \cup X_{\mathsf{d}})$ with $X_{\mathsf{d}} = \left\{ x^{\mathsf{d}} \ : \ x \in X \right\}$ as a copy of X.

Explicit-Dual RANAs

- To untangle the proof and for the later equivalence to Bar-μTL, we introduce another variant of RANAs:

Definition (***Dualizable Boolean Formulae*** **)**

Let X be a (nominal) set of atoms.
Put $\mathcal{B}_{\mathsf{d}}(X) := \mathcal{B}_{+}(X \cup X_{\mathsf{d}})$ with $X_{\mathsf{d}} = \left\{x^{\mathsf{d}} \ : \ x \in X\right\}$ as a copy of X.

Definition (***Explicit-Dual RANA*** **)**

An explicit-dual RANA $A = (Q, \delta, q_0)$ is defined like a RANA but with an equivariant transition function $\delta\colon\ Q \to \mathcal{B}_{\mathsf{d}}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$.
We denote the copy $(\Diamond_\alpha q)^{\mathsf{d}}$ of atoms by $\Box_\alpha q$.

The additional atoms are interpreted as follows:

$$w \models \varepsilon^{\mathsf{d}} :\Longleftrightarrow\ w \neq \varepsilon$$

$$w \models \Box_a q :\Longleftrightarrow\ \forall v \in \overline{\mathbb{A}}^{\star}.\ w = av \implies v \models \delta(q)$$

$$w \models \Box_{|a} q :\Longleftrightarrow\ \forall b \in \mathbb{A}, v \in \overline{\mathbb{A}}^{\star}. w = |bv \implies w \models \Diamond_{|a} q.$$

Proposition (*Ordinary to Explicit-Dual* **)**

For every ordinary RANA A, there is an explicit-dual RANA A^{d} that accepts the *same* literal language, has twice as many orbits and the same degree as A.

Proposition (*Ordinary to Explicit-Dual* **)**

For every ordinary RANA A, there is an explicit-dual RANA A^{d} that accepts the *same* literal language, has twice as many orbits and the same degree as A.

- The resulting RANA A^{d} has two states q, q_{n} for each state q in A.

Proposition (*Ordinary to Explicit-Dual* **)**

For every ordinary RANA A, there is an explicit-dual RANA A^{d} that accepts the *same* literal language, has twice as many orbits and the same degree as A.

- The resulting RANA A^{d} has two states q, q_{n} for each state q in A.
- q in A^{d} accepts the same literal language as in A:

Proposition (*Ordinary to Explicit-Dual* **)**

For every ordinary RANA A, there is an explicit-dual RANA A^{d} that accepts the *same* literal language, has twice as many orbits and the same degree as A.

- The resulting RANA A^{d} has two states q, q_{n} for each state q in A.
- q in A^{d} accepts the same literal language as in A:
  - For transitions, only 'negations' change: $\neg\Diamond_\alpha q$ with $\alpha \in \overline{\mathbb{A}}$ becomes $\Box_\alpha q_{\mathsf{n}}$.
  - The 'negated epsilon' ($\neg\varepsilon$) is replaced by ε^{d}.

Equivalence (Ordinary to Explicit-Dual)

Proposition (*Ordinary to Explicit-Dual*)

For every ordinary RANA A, there is an explicit-dual RANA A^{d} that accepts the *same* literal language, has twice as many orbits and the same degree as A.

- The resulting RANA A^{d} has two states q, q_{n} for each state q in A.
- q in A^{d} accepts the same literal language as in A:
  - For transitions, only 'negations' change: $\neg\Diamond_\alpha q$ with $\alpha \in \overline{\mathbb{A}}$ becomes $\Box_\alpha q_{\mathsf{n}}$.
  - The 'negated epsilon' ($\neg\varepsilon$) is replaced by ε^{d}.
- q_{n} in A^{d} accepts the complement of q. Herefore, we 'negate' the transition formula and convert it as above.
- Acceptance is then shown easily by double induction (word length and size of transition formulae in NNF).

Equivalence (Explicit-Dual to Positive)

Proposition (*Explicit-Dual to Positive*)

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

Proposition (***Explicit-Dual to Positive*** **)**

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

- The idea is to make use of the following logical equivalence:

$$\Box_\alpha \varphi \equiv \Diamond_\alpha \varphi \vee \bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top .$$

Proposition (*Explicit-Dual to Positive* **)**

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

- The idea is to make use of the following logical equivalence:

$$\Box_\alpha \varphi \equiv \Diamond_\alpha \varphi \vee \bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top.$$

- The $\top$ is easily managed by a single additional $\top$-state $q_\top$ with transition formula $\top$.

T.CS

Proposition (***Explicit-Dual to Positive*** **)**

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

- The idea is to make use of the following logical equivalence:

$$\Box_\alpha \varphi \equiv \Diamond_\alpha \varphi \vee \bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top .$$

- The $\top$ is easily managed by a single additional $\top$-state $q_\top$ with transition formula $\top$.
- **Problem:** The disjunction $\bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top$ is infinite!

Equivalence (Explicit-Dual to Positive)

Proposition (*Explicit-Dual to Positive* **)**

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree as $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

» **Problem:** The disjunction $\bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top$ is infinite!

Equivalence (Explicit-Dual to Positive)

Proposition (*Explicit-Dual to Positive* **)**

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree as $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

» **Problem:** The disjunction $\bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top$ is infinite!

Definition (*Escape Letters* **)**

Given a bar string $w \in \overline{\mathbb{A}}^\star$ and a formula $\varphi \in \mathcal{B}_\mathsf{d}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$, a *free* name $a \in \mathsf{FN}(w)$ is an *escape letter for w at φ* if the satisfaction $w \models \varphi$ 'can end' at

- » some $av \models \varepsilon^\mathsf{d}$ or
- » $av \models \Box_\alpha q$ (then $\alpha \neq a$).

(precise definition uses evaluation DAGs)

Proposition (***Explicit-Dual to Positive*** **)**

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree as $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

- **Problem:** The disjunction $\bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top$ is infinite!

Definition (***Escape Letters*** **)**

Given a bar string $w \in \overline{\mathbb{A}}^\star$ and a formula $\varphi \in \mathcal{B}_{\mathsf{d}}(1 + \mathbb{A} \times Q + [\mathbb{A}]Q)$, a *free* name $a \in \mathrm{FN}(w)$ is an *escape letter for w at φ* if the satisfaction $w \models \varphi$ 'can end' at

- some $av \models \varepsilon^{\mathsf{d}}$ or
- $av \models \Box_\alpha q$ (then $\alpha \neq a$). (precise definition uses evaluation DAGs)

- Intuition: Processing of the input word ends immediately!

...but input is still accepted!

Proposition (***Explicit-Dual to Positive*** **)**

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree as $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

- **Problem:** The disjunction $\bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top$ is infinite!

Proposition (*Explicit-Dual to Positive*)

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree as $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

- **Problem:** The disjunction $\bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top$ is infinite!
- Fix this, by restricting the disjunction to *escape letters*.

(bound names occur only once by use of abstraction sets)

Proposition (*Explicit-Dual to Positive* **)**

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree as $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

- **Problem:** The disjunction $\bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top$ is infinite!
- Fix this, by restricting the disjunction to *escape letters*.
  (bound names occur only once by use of abstraction sets)
- **Note:** The set of escape letters for $w \in \overline{\mathbb{A}}^\star$ at φ is **always** contained in $\mathrm{supp}(\varphi) \cup \{a\}$ for some $a \in \mathbb{A}$!

Equivalence (Explicit-Dual to Positive)

Proposition (*Explicit-Dual to Positive* **)**

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree as $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

- **Problem:** The disjunction $\bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top$ is infinite!
- Fix this, by restricting the disjunction to *escape letters*.
  (bound names occur only once by use of abstraction sets)
- **Note:** The set of escape letters for $w \in \overline{\mathbb{A}}^\star$ at φ is **always** contained in $\mathrm{supp}(\varphi) \cup \{a\}$ for some $a \in \mathbb{A}$!
- Thus, A^+ has pairs (q, S) with $q \in Q$ and $S \subseteq_\mathsf{f} \mathbb{A}$ as states, where S is the current set of escape letters. ($|S| \leqslant k+1$)

Proposition (*Explicit-Dual to Positive*)

For every explicit-dual RANA A of degree k and with n orbits, there is a positive RANA A^+ that accepts the *same* literal language, has degree as $2k+1$, and at most $n \cdot (k+2) \cdot (2k+1)^{2k+1} + 1$ orbits.

- **Problem:** The disjunction $\bigvee_{\sigma \neq \alpha} \Diamond_\sigma \top$ is infinite!
- Fix this, by restricting the disjunction to *escape letters*.
  (bound names occur only once by use of abstraction sets)
- **Note:** The set of escape letters for $w \in \overline{\mathbb{A}}^\star$ at φ is **always** contained in $\mathrm{supp}(\varphi) \cup \{a\}$ for some $a \in \mathbb{A}$!
- Thus, A^+ has pairs (q, S) with $q \in Q$ and $S \subseteq_\mathrm{f} \mathbb{A}$ as states, where S is the current set of escape letters. ($|S| \leqslant k+1$)
- We change transition formulae accordingly and verify the equivalence of languages by induction.

Results II: Equivalence to Bar-μTL

Friedrich-Alexander-Universität
Faculty of Engineering

Bar-μTL and RANAs

- Transition formulae of RANAs look like modal formulae (especially with our notation for atoms).

Bar-μTL and RANAs

- Transition formulae of RANAs look like modal formulae (especially with our notation for atoms).
- Fix a countably infinite set Var of fixed-point variables.

Definition (*Bar Formulae* **)** **Hausmann, Milius, Schröder '21**

Bar formulae of Bar-μTL are defined by the grammar

$$\varphi, \psi ::= \varepsilon \mid \neg\varepsilon \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \heartsuit_\sigma \varphi \mid X \mid \mu X.\varphi\,. \qquad (\heartsuit \in \{\Diamond, \Box\}, \sigma \in \overline{\mathbb{A}}, X \in \mathsf{Var})$$

Additionally, $\top := \varepsilon \vee \neg\varepsilon$ and $\bot := \varepsilon \wedge \neg\varepsilon$.

Bar-μTL and RANAs

- Transition formulae of RANAs look like modal formulae (especially with our notation for atoms).
- Fix a countably infinite set Var of fixed-point variables.

Definition (*Bar Formulae*) **Hausmann, Milius, Schröder '21**

Bar formulae of Bar-μTL are defined by the grammar

$$\varphi, \psi ::= \varepsilon \mid \neg\varepsilon \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \heartsuit_\sigma \varphi \mid X \mid \mu X.\varphi\,. \qquad (\heartsuit \in \{\Diamond, \Box\}, \sigma \in \overline{\mathbb{A}}, X \in \mathsf{Var})$$

Additionally, $\top := \varepsilon \vee \neg\varepsilon$ and $\bot := \varepsilon \wedge \neg\varepsilon$.

- The semantics of bar formulae is defined like the semantics for transition formulae.

- Transition formulae of RANAs look like modal formulae (especially with our notation for atoms).
- Fix a countably infinite set Var of fixed-point variables.

Definition (***Bar Formulae*** **)** **Hausmann, Milius, Schröder '21**

Bar formulae of Bar-μTL are defined by the grammar

$$\varphi, \psi ::= \varepsilon \mid \neg\varepsilon \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \heartsuit_\sigma \varphi \mid X \mid \mu X.\varphi\,. \qquad (\heartsuit \in \{\Diamond, \Box\}, \sigma \in \overline{\mathbb{A}}, X \in \mathsf{Var})$$

Additionally, $\top := \varepsilon \vee \neg\varepsilon$ and $\bot := \varepsilon \wedge \neg\varepsilon$.

- The semantics of bar formulae is defined like the semantics for transition formulae.

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

For every bar formula φ, there is an explicit-dual RANA A_φ accepting the literal language of φ: $L_0(A) = \{w \in \mathsf{bs}(\emptyset) \,:\, w \models \varphi\}$.

$\rightsquigarrow$ makes use of the Fisher–Ladner closure of φ

Results III: De-Alternation

Friedrich-Alexander-Universität
Faculty of Engineering

Motivation

We have seen earlier that renaming is *necessary* for negation to be α-invariant. This *non-acceptance* of some α-equivalent bar strings w/o renaming was previously (ERNNAs/RNNAs) ameliorated by the use of *name-dropping*.

Definition (***Restricted Semantics*** **)**

Let $A = (Q, \delta, q_0)$ be a positive RANA. We define the *restricted satisfaction* $w \models^{\mathsf{r}} \varphi$ just as $\models$ for ε, $\Diamond_a$-modalities ($a \in \mathbb{A}$) and

$$w \models^{\mathsf{r}} \Diamond_{|a} q :\iff \exists v \in \overline{\mathbb{A}}^{\star}, b \in \mathbb{A}, q' \in Q.\ w = {|b}v, \langle a \rangle q = \langle b \rangle q' \text{ and } v \models^{\mathsf{r}} \delta(q').$$

Definition (*Restricted Semantics*)

Let $A = (Q, \delta, q_0)$ be a positive RANA. We define the *restricted satisfaction* $w \models^{\mathsf{r}} \varphi$ just as $\models$ for ε, $\lozenge_a$-modalities ($a \in \mathbb{A}$) and

$$w \models^{\mathsf{r}} \lozenge_{|a} q :\Longleftrightarrow \exists v \in \overline{\mathbb{A}}^{\star}, b \in \mathbb{A}, q' \in Q.\ w = |bv, \langle a \rangle q = \langle b \rangle q' \text{ and } v \models^{\mathsf{r}} \delta(q').$$

Theorem (*Name-Dropping for RANAs*)

For every positive RANA A with degree k and n orbits, there is a positive RANA A_{nd} (*the name-dropping modification*) accepting the same literal language, with degree k and at most $n \cdot 2^k$ orbits for which the restricted and ordinary semantics coincide.

Definition (*Restricted Semantics* **)**

Let $A = (Q, \delta, q_0)$ be a positive RANA. We define the *restricted satisfaction* $w \models^{\mathsf{r}} \varphi$ just as $\models$ for ε, $\Diamond_a$-modalities ($a \in \mathbb{A}$) and

$$w \models^{\mathsf{r}} \Diamond_{\mid a} q :\Longleftrightarrow \exists v \in \overline{\mathbb{A}}^{\star}, b \in \mathbb{A}, q' \in Q.\ w = \mid bv, \langle a \rangle q = \langle b \rangle q' \text{ and } v \models^{\mathsf{r}} \delta(q').$$

Theorem (*Name-Dropping for RANAs* **)**

For every positive RANA A with degree k and n orbits, there is a positive RANA A_{nd} (*the name-dropping modification*) accepting the same literal language, with degree k and at most $n \cdot 2^k$ orbits for which the restricted and ordinary semantics coincide.

- Thus, we can restrict ourself to the restricted semantics whenever necessary.

De-Alternation

Theorem (*De-Alternation* **)**

Every RANA with n orbits and degree k can be de-alternated into an ERNNA (RNNA with one single $\top$-state).

De-Alternation

Theorem (*De-Alternation* **)**

Every RANA with n orbits and degree k can be de-alternated into an ERNNA (RNNA with one single $\top$-state).

- **Classical Idea:** Use a power-set construction where a word is accepted by $S \subseteq Q$ iff all $q \in S$ accept the word: transitions built accordingly.

De-Alternation

Theorem (*De-Alternation* **)**

Every RANA with n orbits and degree k can be de-alternated into an ERNNA (RNNA with one single $\top$-state).

- **Classical Idea:** Use a power-set construction where a word is accepted by $S \subseteq Q$ iff all $q \in S$ accept the word: transitions built accordingly.
- **Problem:** The power-set construction yields a **non orbit-finite** set!
  $\rightsquigarrow$ *Restrict* the number of states tracked simultaneously:

De-Alternation

Theorem (*De-Alternation* **)**

Every RANA with n orbits and degree k can be de-alternated into an ERNNA (RNNA with one single $\top$-state).

- **Classical Idea:** Use a power-set construction where a word is accepted by $S \subseteq Q$ iff all $q \in S$ accept the word: transitions built accordingly.
- **Problem:** The power-set construction yields a **non orbit-finite** set!
  $\rightsquigarrow$ *Restrict* the number of states tracked simultaneously:
  - Suppose $q \neq q'$ are in the same orbit of Q, then:

Theorem (*De-Alternation*)

Every RANA with n orbits and degree k can be de-alternated into an ERNNA (RNNA with one single $\top$-state).

- **Classical Idea:** Use a power-set construction where a word is accepted by $S \subseteq Q$ iff all $q \in S$ accept the word: transitions built accordingly.
- **Problem:** The power-set construction yields a **non orbit-finite** set!
  $\rightsquigarrow$ *Restrict* the number of states tracked simultaneously:
  - Suppose $q \neq q'$ are in the same orbit of Q, then:
    (A) If $\mathrm{supp}(q) \neq \mathrm{supp}(q')$ and $A := \mathrm{supp}(q) \cap \mathrm{supp}(q')$:

    Either q *and* $q'|_A$ or $q|_A$ *and* q' accept w iff both q and q' accept w.

Theorem (*De-Alternation* **)**

Every RANA with n orbits and degree k can be de-alternated into an ERNNA (RNNA with one single $\top$-state).

- **Classical Idea:** Use a power-set construction where a word is accepted by $S \subseteq Q$ iff all $q \in S$ accept the word: transitions built accordingly.
- **Problem:** The power-set construction yields a **non orbit-finite** set!
  $\rightsquigarrow$ *Restrict* the number of states tracked simultaneously:
  - Suppose $q \neq q'$ are in the same orbit of Q, then:
    (A) If $\mathrm{supp}(q) \neq \mathrm{supp}(q')$ and $A := \mathrm{supp}(q) \cap \mathrm{supp}(q')$:

    Either q *and* $q'|_A$ or $q|_A$ *and* q' accept w iff both q and q' accept w.

    (B) If $\mathrm{supp}(q) = \mathrm{supp}(q')$, both q and q' must be checked.

De-Alternation

Theorem (*De-Alternation* **)**

Every RANA with n orbits and degree k can be de-alternated into an ERNNA (RNNA with one single $\top$-state).

- **Classical Idea:** Use a power-set construction where a word is accepted by $S \subseteq Q$ iff all $q \in S$ accept the word: transitions built accordingly.
- **Problem:** The power-set construction yields a **non orbit-finite** set!
  $\rightsquigarrow$ *Restrict* the number of states tracked simultaneously:
  - Suppose $q \neq q'$ are in the same orbit of Q, then:
    (A) If $\mathrm{supp}(q) \neq \mathrm{supp}(q')$ and $A := \mathrm{supp}(q) \cap \mathrm{supp}(q')$:

    Either q *and* $q'|_A$ or $q|_A$ *and* q' accept w iff both q and q' accept w.

    (B) If $\mathrm{supp}(q) = \mathrm{supp}(q')$, both q and q' must be checked.
- We restrict the power-set construction to sets of at *most* size $n \cdot k!$.

De-Alternation

Theorem (*De-Alternation* **)**

Every RANA with n orbits and degree k can be de-alternated into an ERNNA (RNNA with one single $\top$-state).

- **Classical Idea:** Use a power-set construction where a word is accepted by $S \subseteq Q$ iff all $q \in S$ accept the word: transitions built accordingly.
- **Problem:** The power-set construction yields a **non orbit-finite** set!
  $\rightsquigarrow$ *Restrict* the number of states tracked simultaneously:
  - Suppose $q \neq q'$ are in the same orbit of Q, then:
    (A) If $\mathrm{supp}(q) \neq \mathrm{supp}(q')$ and $A := \mathrm{supp}(q) \cap \mathrm{supp}(q')$:

      Either q *and* $q'|_A$ or $q|_A$ *and* q' accept w iff both q and q' accept w.

    (B) If $\mathrm{supp}(q) = \mathrm{supp}(q')$, both q and q' must be checked.
- We restrict the power-set construction to sets of at *most* size $n \cdot k!$.
- The resulting ERNNA has a degree of at most $n \cdot k \cdot k!$ and a number of orbits that is at most singly exponential in n and doubly exponential in k.

De-Alternation

Theorem (*De-Alternation* **)**

Every RANA with n orbits and degree k can be de-alternated into an ERNNA (RNNA with one single $\top$-state).

Under the local freshness semantics, RANAs can be completely de-alternated into RNNAs.

- A full de-alternation to RNNAs (w/o the $\top$-state) is impossible.

  (Example at blackboard)

- Similarly, the naïve power-set construction is impossible.

  (Example at blackboard)

Results IV: Finitisation & Model-Checking

Friedrich-Alexander-Universität
Faculty of Engineering

Finite Representability

- To simplify model checking, we desire a correspondence between classical AFAs and our RANAs:

Finite Representability

- To simplify model checking, we desire a correspondence between classical AFAs and our RANAs:

Theorem (*Equivalence* **)**

Every RANA is bar-language-equivalent to a bar AFA, that is a classical *alternating finite automaton* over a **finite** alphabet $\overline{\mathbb{A}}_0$ having a certain semantics.

- To simplify model checking, we desire a correspondence between classical AFAs and our RANAs:

Theorem (*Equivalence*)

Every RANA is bar-language-equivalent to a bar AFA, that is a classical *alternating finite automaton* over a **finite** alphabet $\overline{\mathbb{A}}_0$ having a certain semantics.

- If the RANA is of degree k with n orbits, the bar AFA has an alphabet of size $2k+1$ and at most $n \cdot k! = n \cdot 2^{k \log(k)}$ states.

Finite Representability

- To simplify model checking, we desire a correspondence between classical AFAs and our RANAs:

Theorem (*Equivalence* **)**

Every RANA is bar-language-equivalent to a bar AFA, that is a classical *alternating finite automaton* over a **finite** alphabet $\overline{\mathbb{A}}_0$ having a certain semantics.

- If the RANA is of degree k with n orbits, the bar AFA has an alphabet of size $2k + 1$ and at most $n \cdot k! = n \cdot 2^{k \log(k)}$ states.
- The semantics looks at bar strings and split them up into a pre-word and a suffix, where the pre-word is '*read up to α-equivalence*' by the Bar-AFA. If it results in just $\top$'s, any suffix may be added.

Finite Representability (Emptiness-Equivalence)

Problem

The previously mentioned equivalence will not help us directly, since standard algorithms for AFAs use the classical finite semantics.

Finite Representability (Emptiness-Equivalence)

Problem

The previously mentioned equivalence will not help us directly, since standard algorithms for AFAs use the classical finite semantics.

- For a bar AFA, let $L_0(A)$ be the literal language under our semantics and $L_{\mathsf{AFA}}(A)$ be the literal language under the classical finite semantics:

T.CS

Problem

The previously mentioned equivalence will not help us directly, since standard algorithms for AFAs use the classical finite semantics.

- For a bar AFA, let $L_0(A)$ be the literal language under our semantics and $L_{\mathsf{AFA}}(A)$ be the literal language under the classical finite semantics:

Theorem (*Emptiness-Equivalence*)

For every bar AFA, we have the following equivalence:

$$L_0(A) = \emptyset \quad \text{iff} \quad \underbrace{L_{\mathsf{AFA}}(A) \cap \mathsf{bs}(\emptyset)}_{\text{is recognizable by an AFA}} = \emptyset$$

Finite Representability (Emptiness-Equivalence)

Problem

The previously mentioned equivalence will not help us directly, since standard algorithms for AFAs use the classical finite semantics.

- For a bar AFA, let $L_0(A)$ be the literal language under our semantics and $L_{\mathsf{AFA}}(A)$ be the literal language under the classical finite semantics:

Theorem (*Emptiness-Equivalence*)

For every bar AFA, we have the following equivalence:

$$L_0(A) = \emptyset \quad \text{iff} \quad \underbrace{L_{\mathsf{AFA}}(A) \cap \mathsf{bs}(\emptyset)}_{\text{is recognizable by an AFA}} = \emptyset$$

- If A is a bar AFA with alphabet size k and n states, the AFA accepting $L_{\mathsf{AFA}}(A) \cap \mathsf{bs}(\emptyset)$ has alphabet size k and at most $n + 2^{k/2} + 1$ states.

Remark (*Complexities*)

Given any RANA of degree k and with n orbits, its equivalent name-dropping modification has at most $(2 \cdot n \cdot (k+2)+1) \cdot 2^{(2k+1)\cdot \log(4k+2)}$ orbits and a degree of $2k+1$.

Its de-alternation has a degree that is linear in n and exponential in k as well as a number of orbits that is exponential in n and doubly exponential in k.

Decidability Problems

Remark (*Complexities*)

Given any RANA of degree k and with n orbits, its equivalent name-dropping modification has at most $(2 \cdot n \cdot (k+2)+1) \cdot 2^{(2k+1)\cdot \log(4k+2)}$ orbits and a degree of $2k+1$.

Its de-alternation has a degree that is linear in n and exponential in k as well as a number of orbits that is exponential in n and doubly exponential in k.

Theorem (*Decidability Problems*)

Non-Emptiness for (name-dropping) RANAs is decidable in ExpSpace:

$\rightsquigarrow$ space linear in the number of orbits and exponential in the degree of the RANA

Decidability Problems

Remark (*Complexities* **)**

Given any RANA of degree k and with n orbits, its equivalent name-dropping modification has at most $(2 \cdot n \cdot (k+2)+1) \cdot 2^{(2k+1) \cdot \log(4k+2)}$ orbits and a degree of $2k+1$.

Its de-alternation has a degree that is linear in n and exponential in k as well as a number of orbits that is exponential in n and doubly exponential in k.

Theorem (*Decidability Problems* **)**

Non-Emptiness for (name-dropping) RANAs is decidable in ExpSpace:

$\rightsquigarrow$ space linear in the number of orbits and exponential in the degree of the RANA

Universality for RANAs is decidable in ExpSpace:

$\rightsquigarrow$ space linear in the number of orbits and exponential in the degree of the RANA.

Decidability Problems

Remark (*Complexities*)

Given any RANA of degree k and with n orbits, its equivalent name-dropping modification has at most $(2 \cdot n \cdot (k+2)+1) \cdot 2^{(2k+1) \cdot \log(4k+2)}$ orbits and a degree of $2k+1$.

Its de-alternation has a degree that is linear in n and exponential in k as well as a number of orbits that is exponential in n and doubly exponential in k.

Theorem (*Decidability Problems*)

Non-Emptiness for (name-dropping) RANAs is decidable in ExpSpace:

$\leadsto$ space linear in the number of orbits and exponential in the degree of the RANA

Universality for RANAs is decidable in ExpSpace:

$\leadsto$ space linear in the number of orbits and exponential in the degree of the RANA.

Inclusion-Checking for RANAs is decidable in ExpSpace:

$\leadsto$ space linear in the number of both orbits and exponential in the maximum degree of both RANAs.

Decidability Problems

Remark (*Complexities*)

Given any RANA of degree k and with n orbits, its equivalent name-dropping modification has at most $(2 \cdot n \cdot (k+2)+1) \cdot 2^{(2k+1) \cdot \log(4k+2)}$ orbits and a degree of $2k+1$.

Its de-alternation has a degree that is linear in n and exponential in k as well as a number of orbits that is exponential in n and doubly exponential in k.

Theorem (*Inclusion-Checking under Local Freshness*)

The inclusion problem for RANAs under local freshness is decidable in 2ExpSpace:

$\rightsquigarrow$ space exponential in both the number of orbits and the degree of both RANAs.

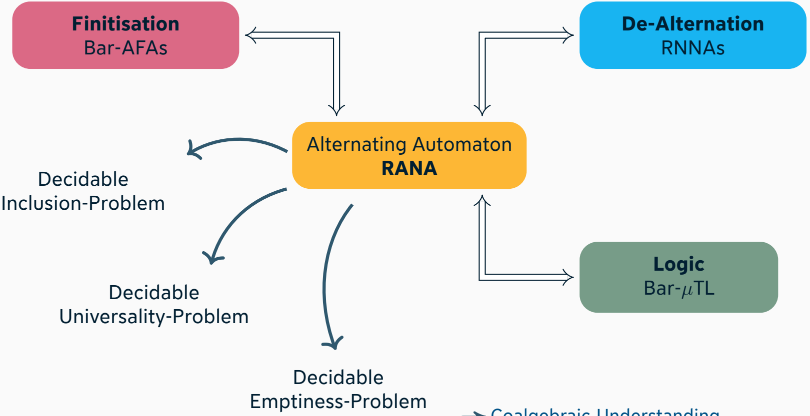
⋙ For local freshness, we need to de-alternate completely!

(Example at blackboard)

Conclusion

Conclusion & Future Work

- We looked at a variant of alternating automaton for data languages with inherent name binding, and found many nice properties:



- There are still open problems left:
  - Coalgebraic Understanding
  - Residuality/Learning RANAs
  - Extension to ω-Words

Questions?

References (1)

- 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.
- Hausmann, Daniel, Stefan Milius, Lutz Schröder. **'A Linear-Time Nominal μ-Calculus with Name Allocation'.** *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021).* Ed. by Filippo Bonchi, Simon J. Puglisi. Vol. 202. LIPIcs. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 58:1–58:18. ISBN: 978-3-95977-201-3. DOI: `10.4230/LIPIcs.MFCS.2021.58`. URL: `https://drops.dagstuhl.de/opus/volltexte/2021/14498`.
- Schröder, Lutz, Dexter Kozen, Stefan Milius, Thorsten Wißmann. **'Nominal Automata with Name Binding'.** *Proc. 20th International Conference on Foundations of Software Science and Computation Structures, (FOSSACS 2017).* Vol. 10203. Lect. Notes Comput. Sci. 2017, pp. 124–142.