On Regular Expressions and Nominal Automata

Tomoyuki Suzuki

Institute of Computer Science, Academy of Sciences of the Czech Republic

20 September, 2013 @ Paris, France

(joint work with Alexander Kurz and Emilio Tuosto)

Nominal automata and languages over infinite alphabets

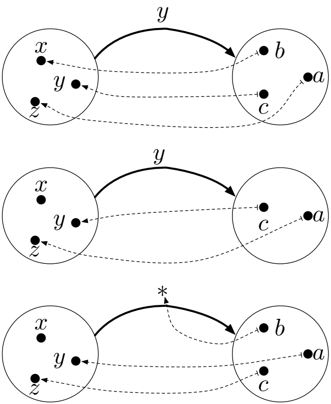
1. Finite-memory automata
2. Fresh-register automata
3. History-dependent automata
4. History-register automata
5. and so on...

So far, the main concern is their semantic aspects (of course, there are some interesting algebraic results e.g. Data monoids or Regular expressions).

Our approach

Target: discuss several resource-handling automata by means of nominal regular expressions from the general perspective.

Difficulties: for example, in history-dependent automata...



Automata $A^\sharp$

$\mathcal{H} = \langle Q, I, q_0, F, tr \rangle$

1. Q: (finite) named set (endowed with a function $\|\_\|: Q \to \mathbb{N}$) and we let $reg(q) := \{1, \ldots, \|q\|\}$
2. I: input function

$$I(q) := \Sigma \cup reg(q) \cup \{\star, \oslash\}$$

3. q_0: initial state with no memory cell ($reg(q_0) = 0$)
4. F: final states with no memory cell ($reg(q) = 0$ for $q \in F$)
5. tr: transition relations satisfying for $q, q' \in Q$ and $\alpha \in I(q) \cup \{\epsilon\}$,

$$q' \in tr(q, \alpha) \iff \begin{cases} \|q'\| = \|q\| + 1 & \alpha = \star \\ \|q'\| + 1 = \|q\| & \alpha = \oslash \\ \|q'\| = \|q\| & \text{otherwise} \end{cases}$$

Automata $A^\sharp$

$\mathcal{H} = \langle Q, I, q_0, F, tr \rangle$

1. Q: (finite) named set (endowed with a function $\|\_\| : Q \to \mathbb{N}$) and we let $reg(q) := \{1, \ldots, \|q\|\}$
2. I: input function

$$I(q) := \Sigma \cup reg(q) \cup \{\star, \oslash\}$$

3. q_0: initial state with no memory cell ($reg(q_0) = 0$)
4. F: final states with no memory cell ($reg(q) = 0$ for $q \in F$)
5. tr: transition relations satisfying for $q, q' \in Q$ and $\alpha \in I(q) \cup \{\epsilon\}$,

$$q' \in tr(q, \alpha) \iff \begin{cases} \|q'\| = \|q\| + 1 & \alpha = \star \\ \|q'\| + 1 = \|q\| & \alpha = \oslash \\ \|q'\| = \|q\| & \text{otherwise} \end{cases}$$

Automata $A^\sharp$

$\mathcal{H} = \langle Q, I, q_0, F, tr \rangle$

1. Q: (finite) named set (endowed with a function $\|\_\| \colon Q \to \mathbb{N}$) and we let $reg(q) := \{1, \ldots, \|q\|\}$
2. I: input function

$$I(q) := \Sigma \cup reg(q) \cup \{\star, \oslash\}$$

3. q_0: initial state with no memory cell ($reg(q_0) = 0$)
4. F: final states with no memory cell ($reg(q) = 0$ for $q \in F$)
5. tr: transition relations satisfying for $q, q' \in Q$ and $\alpha \in I(q) \cup \{\epsilon\}$,

$$q' \in tr(q, \alpha) \iff \begin{cases} \|q'\| = \|q\| + 1 & \alpha = \star \\ \|q'\| + 1 = \|q\| & \alpha = \oslash \\ \|q'\| = \|q\| & \text{otherwise} \end{cases}$$

Automata $A^\sharp$

$\mathcal{H} = \langle Q, I, q_0, F, tr \rangle$

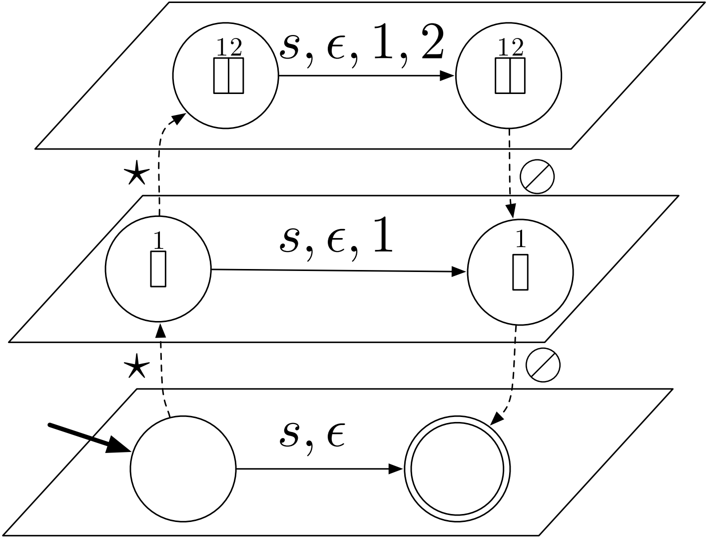
1. Q: (finite) named set (endowed with a function $\|\_\| \colon Q \to \mathbb{N}$) and we let $reg(q) := \{1, \ldots, \|q\|\}$
2. I: input function

$$I(q) := \Sigma \cup reg(q) \cup \{\star, \oslash\}$$

3. q_0: initial state with no memory cell ($reg(q_0) = 0$)
4. F: final states with no memory cell ($reg(q) = 0$ for $q \in F$)
5. tr: transition relations satisfying for $q, q' \in Q$ and $\alpha \in I(q) \cup \{\epsilon\}$,

$$q' \in tr(q, \alpha) \iff \begin{cases} \|q'\| = \|q\| + 1 & \alpha = \star \\ \|q'\| + 1 = \|q\| & \alpha = \oslash \\ \|q'\| = \|q\| & \text{otherwise} \end{cases}$$

Picture of layered automata



With binders or without?

On $A^\sharp$, we have two different notions of words:
Σ: a finite set of constants
$\mathcal{N}$: an infinite set of names.

With binders

$$w ::= \epsilon \mid s \in \Sigma \mid n \in \mathcal{N} \mid w \circ w \mid \langle n.w \rangle$$

Without binders

$$w \in (\Sigma \cup \mathcal{N})^*$$

With binders or without?

On $A^\sharp$, we have two different notions of words:
Σ: a finite set of constants
$\mathcal{N}$: an infinite set of names.

With binders

$$w ::= \epsilon \mid s \in \Sigma \mid n \in \mathcal{N} \mid w \circ w \mid \langle\!\langle n.w \rangle\!\rangle$$

Without binders

$$w \in (\Sigma \cup \mathcal{N})^*$$

With binders or without?

On $A^\sharp$, we have two different notions of words:
Σ: a finite set of constants
$\mathcal{N}$: an infinite set of names.

With binders

$$w ::= \epsilon \mid s \in \Sigma \mid n \in \mathcal{N} \mid w \circ w \mid \langle\!\langle n.w \rangle\!\rangle$$

Without binders

$$w \in (\Sigma \cup \mathcal{N})^*$$

Nominal regular expressions (b-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid ne + ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle$$

Theorem
$A^\sharp \iff$ (closed) b-NREs.

Sketch.
($\Leftarrow$). Induction.
($\Rightarrow$). ϵ-closure, the layer-wise powerset construction, the vertical powerset construction and the two-step inductions of paths. $\square$

Nominal regular expressions (b-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid ne + ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle$$

Theorem

$A^\sharp \iff$ (closed) b-NREs.

Sketch.

($\Leftarrow$). Induction.

($\Rightarrow$). ϵ-closure, the layer-wise powerset construction, the vertical powerset construction and the two-step inductions of paths. $\square$

Nominal regular expressions (b-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid ne + ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle$$

Theorem

$A^\sharp \iff$ (closed) b-NREs.

Sketch.

($\Leftarrow$). Induction.

($\Rightarrow$). ϵ-closure, the layer-wise powerset construction, the vertical powerset construction and the two-step inductions of paths. $\square$

Nominal regular expressions (b-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid ne + ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle$$

Theorem

$A^\sharp \iff$ (closed) b-NREs.

Sketch.

($\Leftarrow$). Induction.

($\Rightarrow$). ϵ-closure, the layer-wise powerset construction, the vertical powerset construction and the two-step inductions of paths. $\square$

$A^\sharp$ is not expressive enough

Hereinafter, we discuss languages over infinite alphabets with $\Sigma = \emptyset$ (words without binders) only.

$$L \subseteq \mathcal{N}^*$$

Two important languages:

1. $\mathcal{L}_{all} := \{n_1 \cdots n_k \mid \forall k \in \mathcal{N}, \forall 1 \leq i, j \leq k . n_i \neq n_j\}$
2. $\mathcal{L}_{two} := \{n_1 \cdots n_k \mid \forall 1 \leq i < k . n_i \neq n_{i+1}\}$

Theorem

Neither $\mathcal{L}_{all}$ nor $\mathcal{L}_{two}$ are recognisable on $A^\sharp$.

$A^\sharp$ is not expressive enough

Hereinafter, we discuss languages over infinite alphabets with $\Sigma = \emptyset$ (words without binders) only.

$$L \subseteq \mathcal{N}^*$$

Two important languages:

1. $\mathcal{L}_{all} := \{n_1 \cdots n_k \mid \forall k \in \mathcal{N}, \forall 1 \leq i, j \leq k . n_i \neq n_j\}$
2. $\mathcal{L}_{two} := \{n_1 \cdots n_k \mid \forall 1 \leq i < k . n_i \neq n_{i+1}\}$

Theorem

Neither $\mathcal{L}_{all}$ nor $\mathcal{L}_{two}$ are recognisable on $A^\sharp$.

$A^\sharp$ is not expressive enough

Hereinafter, we discuss languages over infinite alphabets with $\Sigma = \emptyset$ (words without binders) only.

$$L \subseteq \mathcal{N}^*$$

Two important languages:

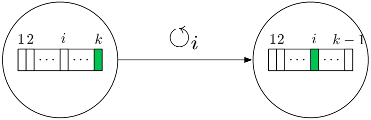
1. $\mathcal{L}_{all} := \{n_1 \cdots n_k \mid \forall k \in \mathcal{N}, \forall 1 \leq i, j \leq k . n_i \neq n_j\}$
2. $\mathcal{L}_{two} := \{n_1 \cdots n_k \mid \forall 1 \leq i < k . n_i \neq n_{i+1}\}$

Theorem

Neither $\mathcal{L}_{all}$ nor $\mathcal{L}_{two}$ are recognisable on $A^\sharp$.

Further extensions

Permutation $\circlearrowleft_i$



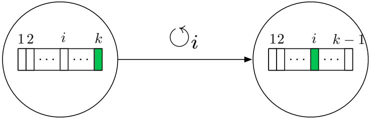
Underline $\underline{i}$

1. $DA^\sharp = A^\sharp +$ permutations
2. $CA^\sharp = A^\sharp +$ underlines
3. $CDA^\sharp = A^\sharp +$ permutations $+$ underlines

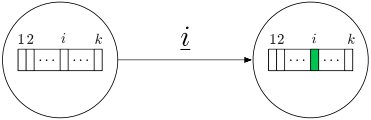
$I'(q) := \Sigma \cup reg(q) \cup \{\star, \epsilon\} \cup \{\circlearrowleft_i \mid i \in reg(q)\} \cup \{\underline{i} \mid i \in reg(q)\}$

Further extensions

Permutation $\circlearrowleft_i$



Underline $\underline{i}$

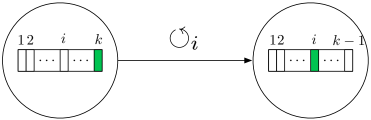


1. $DA^\sharp = A^\sharp +$ permutations
2. $CA^\sharp = A^\sharp +$ underlines
3. $CDA^\sharp = A^\sharp +$ permutations + underlines

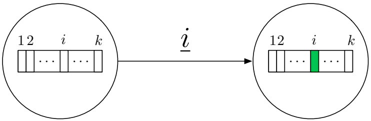
$$I'(q) := \Sigma \cup reg(q) \cup \{\star, \epsilon\} \cup \{\circlearrowleft_i \mid i \in reg(q)\} \cup \{\underline{i} \mid i \in reg(q)\}$$

Further extensions

Permutation $\circlearrowleft_i$



Underline $\underline{i}$



1. $DA^\sharp = A^\sharp +$ permutations
2. $CA^\sharp = A^\sharp +$ underlines
3. $CDA^\sharp = A^\sharp +$ permutations $+$ underlines

$$I'(q) := \Sigma \cup reg(q) \cup \{\star, \epsilon\} \cup \{\circlearrowleft_i \mid i \in reg(q)\} \cup \{\underline{i} \mid i \in reg(q)\}$$

Kleene-style results

Extensions of (basic) nominal regular expressions (b-NREs)

- Nominal regular expressions with permutations (p-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid ne + ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle_n^m$$

- Nominal regular expressions with underlines (u-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid \underline{n} \mid ne + ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle$$

- Nominal regular expressions with underlines and permutations (up-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid \underline{n} \mid ne + ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle_n^m$$

Theorem

1. $DA^\sharp \iff$ *p-NREs*
2. $CA^\sharp \iff$ *u-NREs*
3. $CDA^\sharp \iff$ *up-NREs*

Kleene-style results

Extensions of (basic) nominal regular expressions (b-NREs)

- Nominal regular expressions with permutations (p-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid ne + ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle_n^m$$

- Nominal regular expressions with underlines (u-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid \underline{n} \mid ne + ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle$$

- Nominal regular expressions with underlines and permutations (up-NREs)

$$ne ::= 1 \mid 0 \mid s \in \Sigma \mid n \in \mathcal{N} \mid \underline{n} \mid ne+ne \mid ne \circ ne \mid ne^* \mid \langle_n ne \rangle_n^m$$

Theorem

1. $DA^\sharp \iff$ *p-NREs*
2. $CA^\sharp \iff$ *u-NREs*
3. $CDA^\sharp \iff$ *up-NREs*

Simulation results

Theorem

1. *$DA^\sharp$ and $CA^\sharp$ can simulate $A^\sharp$.*
2. *$CDA^\sharp$ can simulate $DA^\sharp$ and $CA^\sharp$.*
3. *$DA^\sharp$ and $CA^\sharp$ cannot simulate each other.*

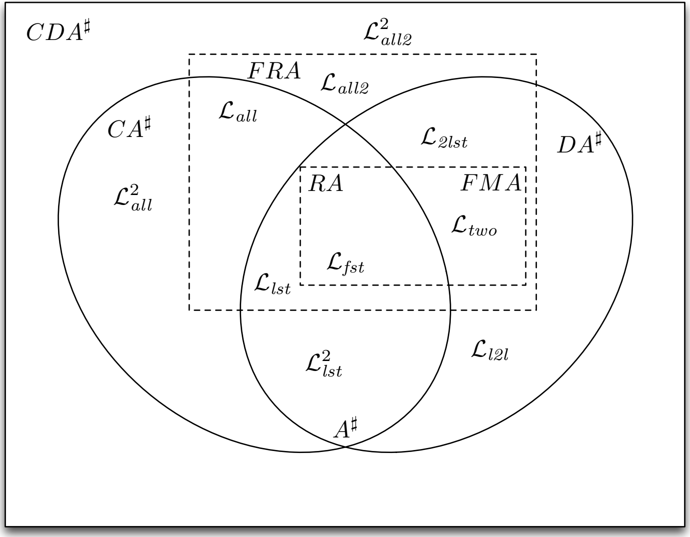
Theorem

$DA^\sharp$ can simulate finite-memory automata and register-automata. But, the converse does not hold.

Theorem

$CDA^\sharp$ can simulate fresh-register automata. But the converse does not hold.

Language comparisons



Conclusion and future work

1. Complete description of languages over infinite alphabets on nominal regular expressions
2. Algebraic properties on nominal regular expressions
3. Predicate characterisation of our automata and languages over infinite alphabets
4. History-register automata and parallel automata models
5. Concurrency and regular expressions