NOMINAL MODAL LOGICS FOR FRESH-REGISTER AUTOMATA
(WORK IN PROGRESS)

M.H.Bandukara, N.Tzevelekos

Queen Mary University of London

INFINITE ALPHABETS

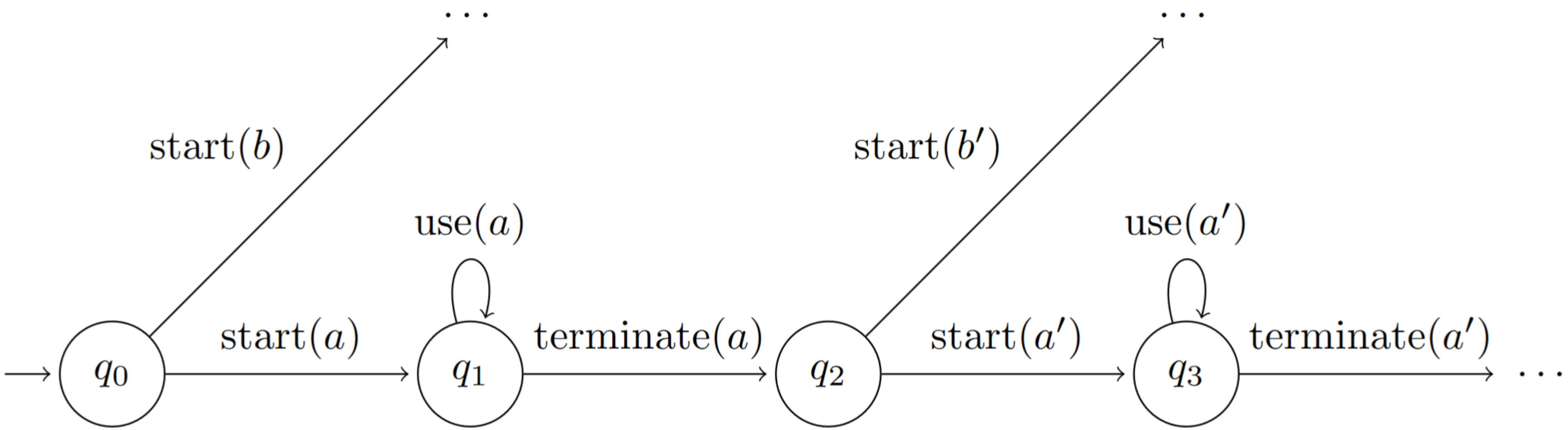
Systems that operate over ***infinite alphabets***

- Mobile processes, program semantics, dynamic resource allocation

INFINITE ALPHABETS

Systems that operate over ***infinite alphabets***

- Mobile processes, program semantics, dynamic resource allocation
- Example: Application usage session



FRESH-REGISTER AUTOMATA

Extension of the Finite-Memory Automata model [Kaminski & Francez, 1994]

Uses ***registers*** to store names (or atoms)

Captures ***global-freshness***

- Allows acceptance of a name that has not been seen in the current run

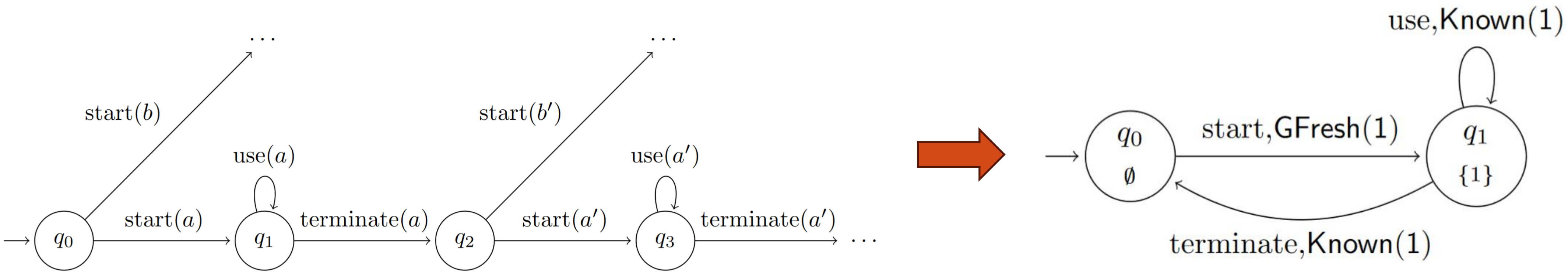
FRESH-REGISTER AUTOMATA

Extension of the Finite-Memory Automata model [Kaminski & Francez, 1994]

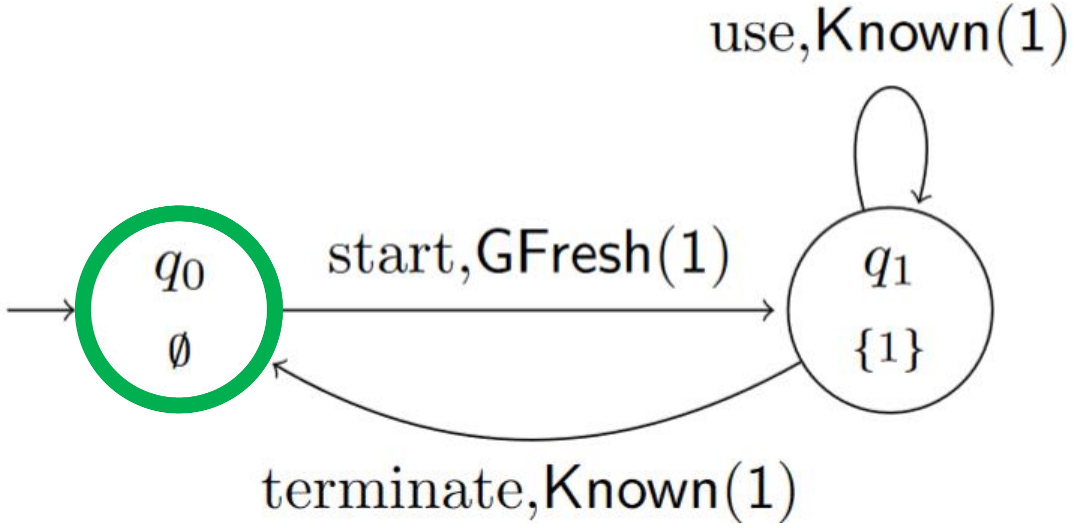
Uses ***registers*** to store names (or atoms)

Captures ***global-freshness***

- Allows acceptance of a name that has not been seen in the current run

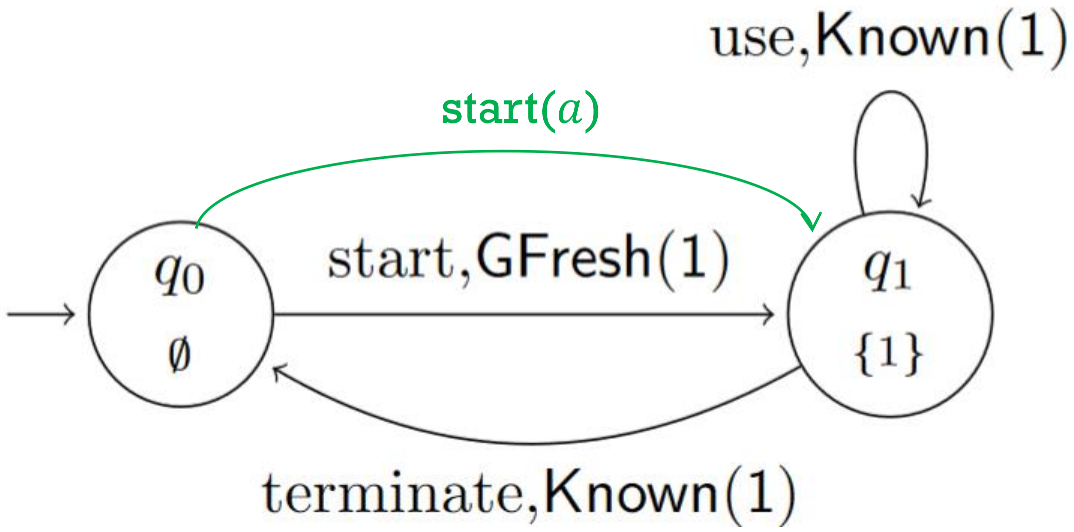


FRESH-REGISTER AUTOMATA



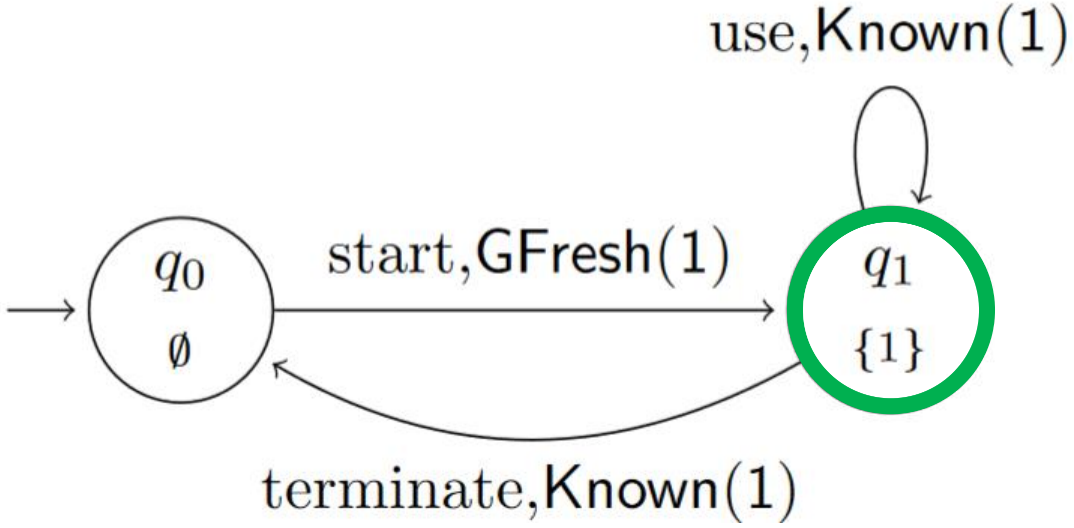
Run: $(q_0, \emptyset, \emptyset)$

FRESH-REGISTER AUTOMATA



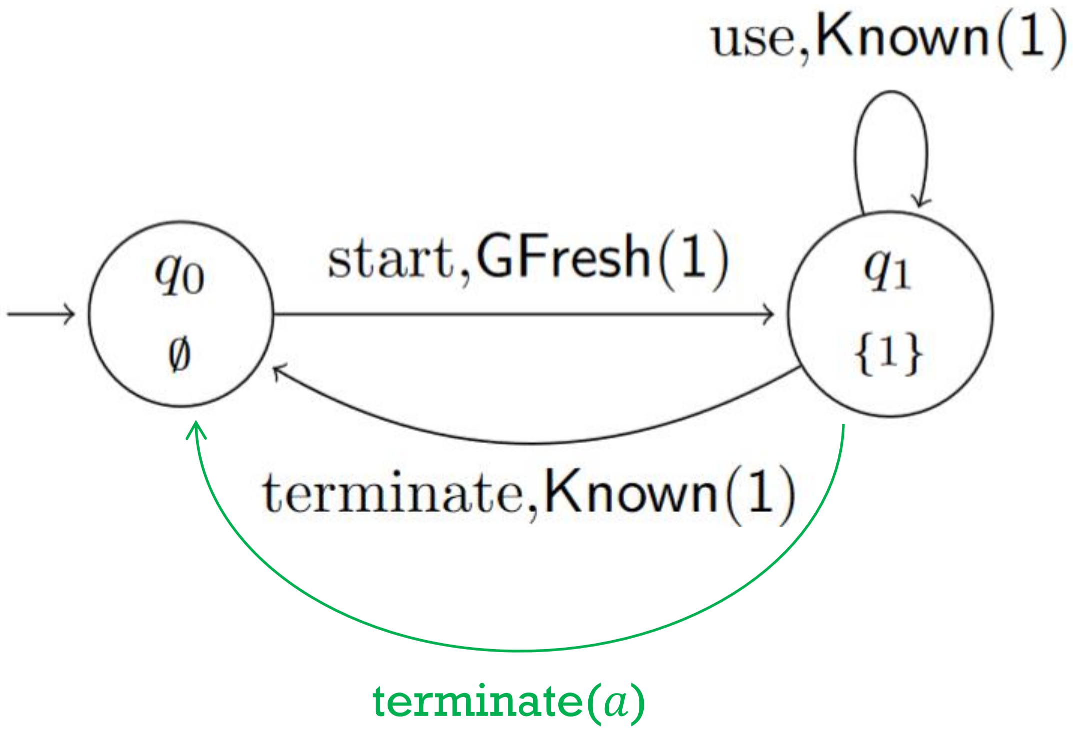
Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)}$

FRESH-REGISTER AUTOMATA



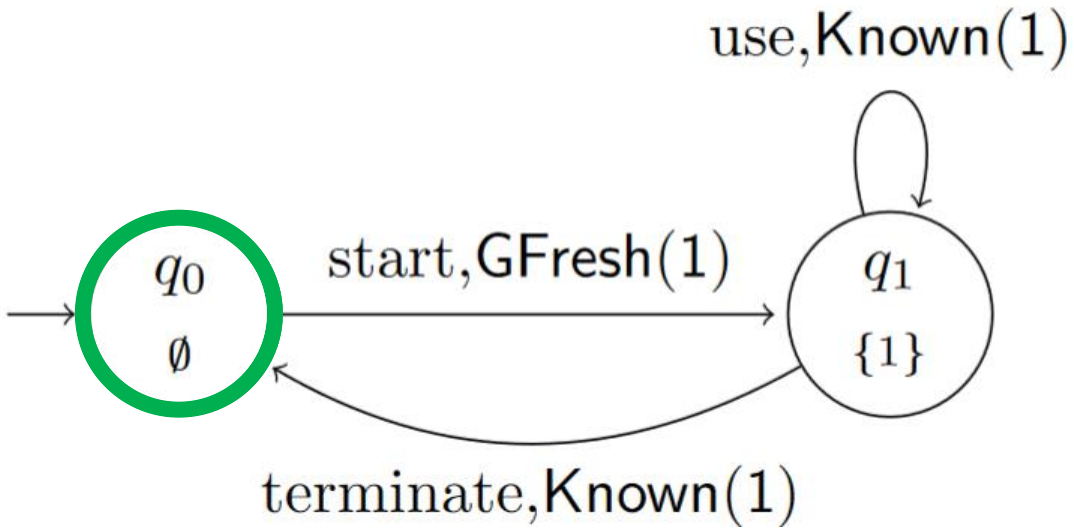
Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)} (q_1, \{1 \mapsto a\}, \{a\})$

FRESH-REGISTER AUTOMATA



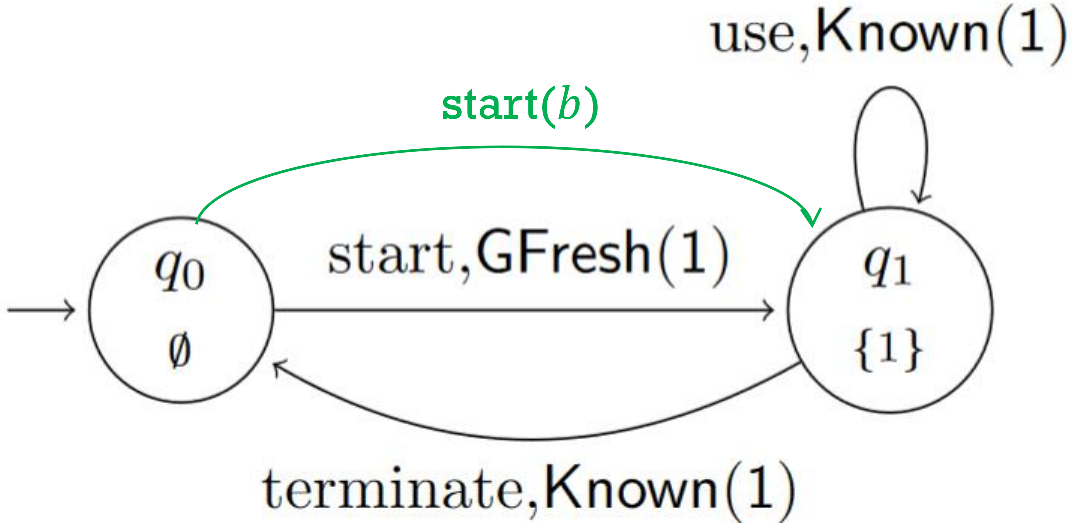
Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)} (q_1, \{1 \mapsto a\}, \{a\}) \xrightarrow{terminate(a)}$

FRESH-REGISTER AUTOMATA



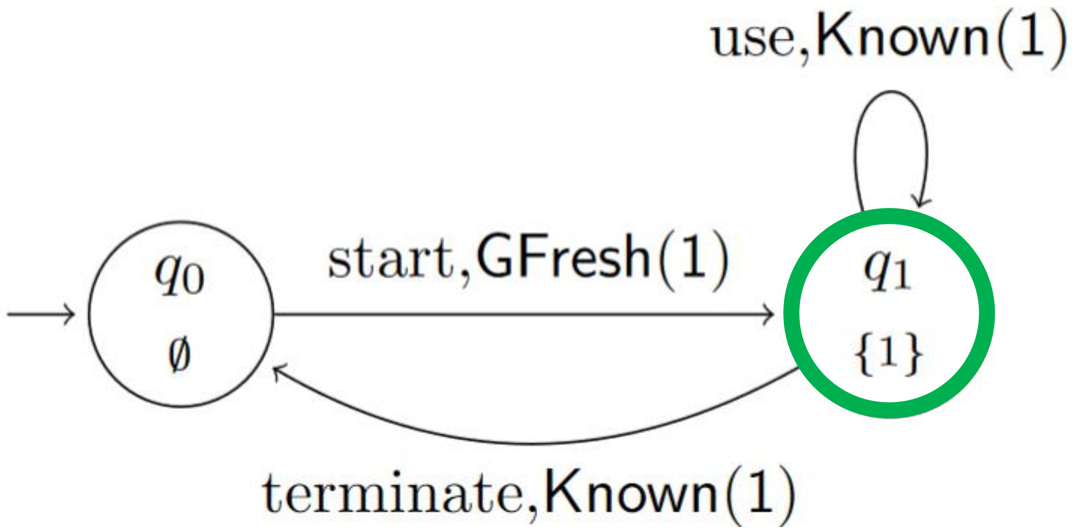
Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)} (q_1, \{1 \mapsto a\}, \{a\}) \xrightarrow{terminate(a)} (q_0, \emptyset, \{a\})$

FRESH-REGISTER AUTOMATA



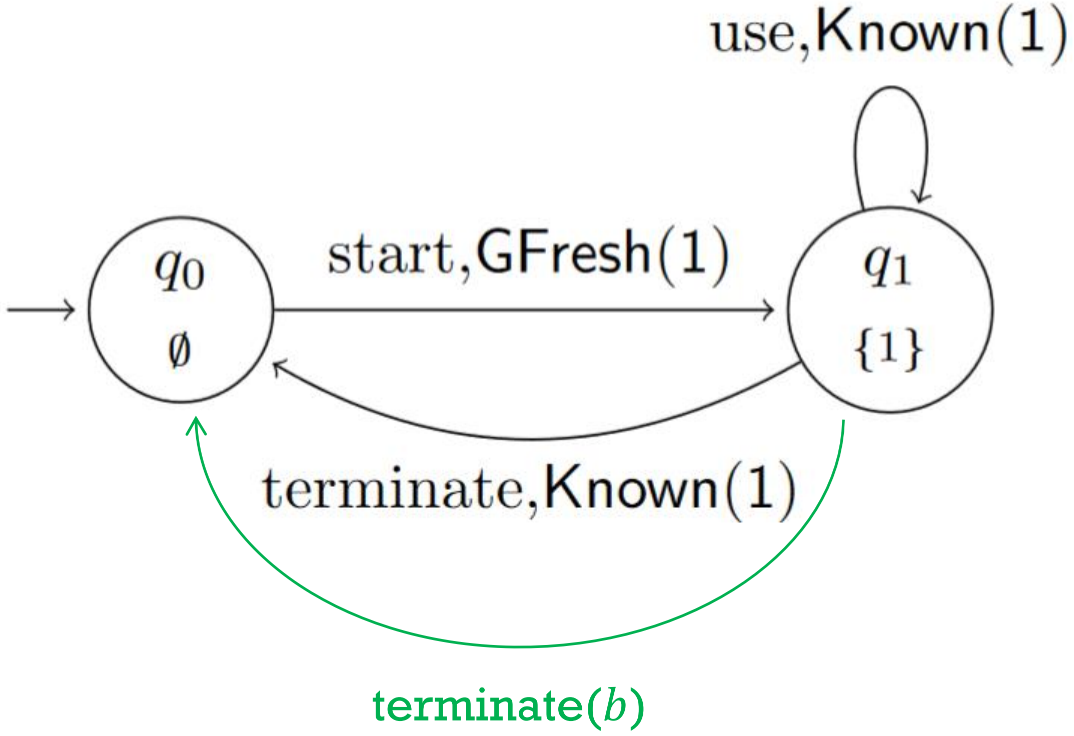
Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)} (q_1, \{1 \mapsto a\}, \{a\}) \xrightarrow{terminate(a)} (q_0, \emptyset, \{a\}) \xrightarrow{start(b)}$

FRESH-REGISTER AUTOMATA



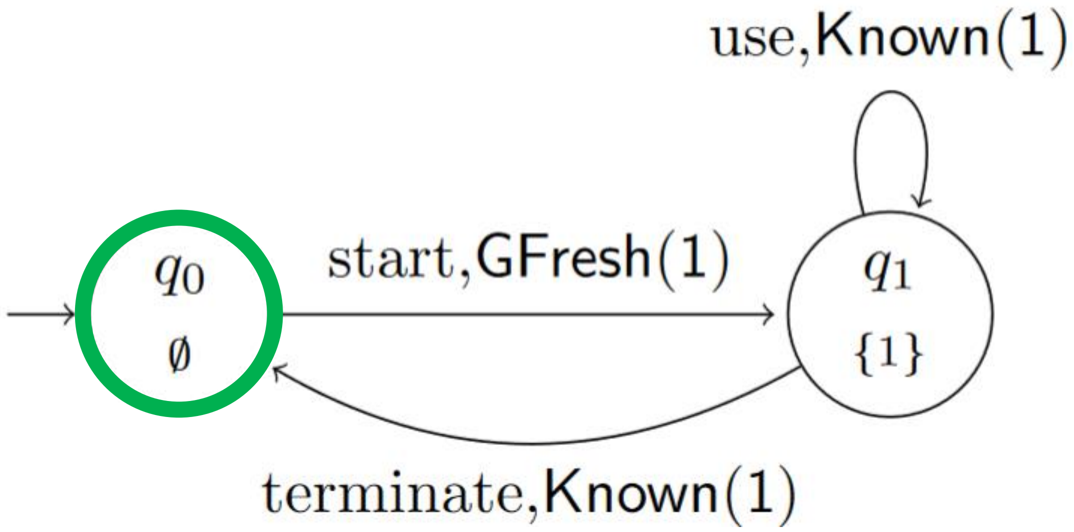
Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)} (q_1, \{1 \mapsto a\}, \{a\}) \xrightarrow{terminate(a)} (q_0, \emptyset, \{a\}) \xrightarrow{start(b)} (q_1, \{1 \mapsto b\}, \{a, b\})$

FRESH-REGISTER AUTOMATA



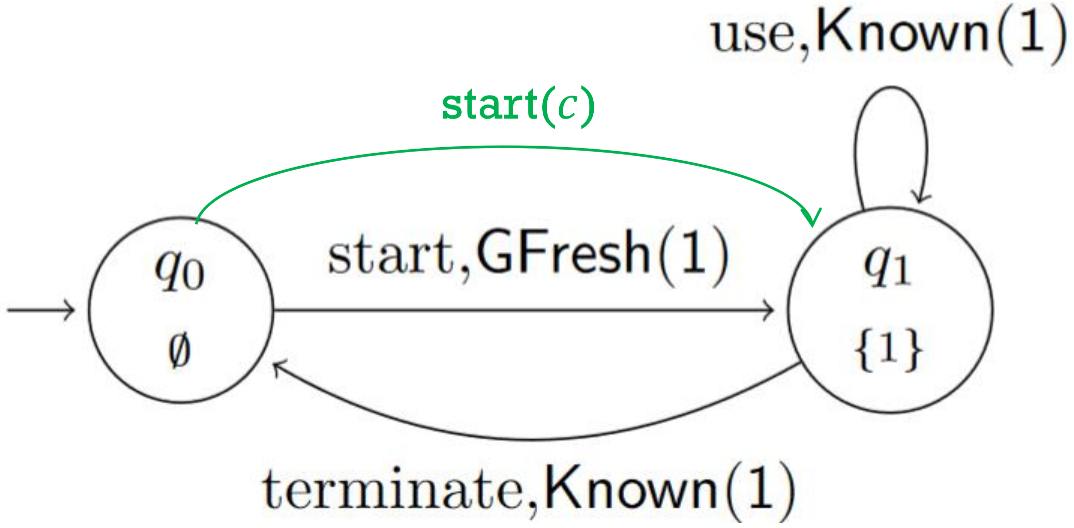
Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)} (q_1, \{1 \mapsto a\}, \{a\}) \xrightarrow{terminate(a)} (q_0, \emptyset, \{a\}) \xrightarrow{start(b)} (q_1, \{1 \mapsto b\}, \{a, b\}) \xrightarrow{terminate(b)}$

FRESH-REGISTER AUTOMATA



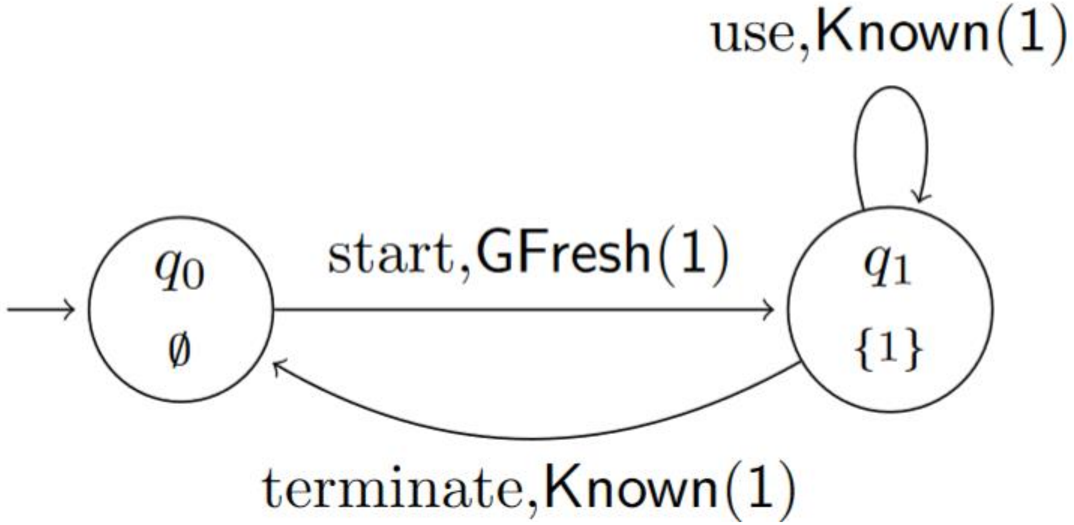
Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)} (q_1, \{1 \mapsto a\}, \{a\}) \xrightarrow{terminate(a)} (q_0, \emptyset, \{a\}) \xrightarrow{start(b)} (q_1, \{1 \mapsto b\}, \{a, b\}) \xrightarrow{terminate(b)} (q_0, \emptyset, \{a, b\})$

FRESH-REGISTER AUTOMATA



Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)} (q_1, \{1 \mapsto a\}, \{a\}) \xrightarrow{terminate(a)} (q_0, \emptyset, \{a\}) \xrightarrow{start(b)} (q_1, \{1 \mapsto b\}, \{a, b\}) \xrightarrow{terminate(b)} (q_0, \emptyset, \{a, b\}) \to \cdots$

FRESH-REGISTER AUTOMATA



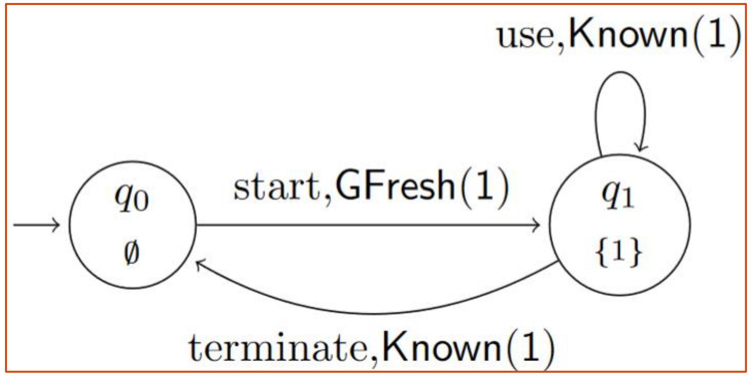
Run: $(q_0, \emptyset, \emptyset) \xrightarrow{start(a)} (q_1, \{1 \mapsto a\}, \{a\}) \xrightarrow{terminate(a)} (q_0, \emptyset, \{a\}) \xrightarrow{start(b)} (q_1, \{1 \mapsto b\}, \{a, b\}) \xrightarrow{terminate(b)} (q_0, \emptyset, \{a, b\}) \to \cdots$

Configurations

FRESH-REGISTER AUTOMATA – FORMAL DEFINITION

An r-Fresh-Register Automaton is a tuple $A = \langle \Sigma, Q, q_0, \mu, \delta, F \rangle$ that operates on a set of registers $\{1, \dots, r\}$ where:

- Σ is a finite set of tags
- Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the finite set of final states
- $\mu : Q \to P(\{1, \dots, r\})$ indicates which registers are filled at each state
- δ is the transition relation
  - $\delta \subseteq Q \times \{(t, X(i)) | t \in \Sigma, i \in \{1, \dots, r\}, X \; in \; \{Known, LFresh, GFresh\}\} \times Q$



RESULTS ON PROPERTIES OF FRESH-REGISTER AUTOMATA

What has been done:

Language equivalence undecidable [Neven et al, 2004]

- E.g., can encode computations of counter machines

Bisimulation equivalence decidable by use of *symbolic techniques* [Murawski et al, 2018]

- Language equivalence is decidable in the deterministic case

Translation from finitary π-calculus processes to fresh-register automata [Bandukara & Tzevelekos, 2022]

RESULTS ON PROPERTIES OF FRESH-REGISTER AUTOMATA

What has been done:

Language equivalence undecidable [Neven et al, 2004]

- E.g., can encode computations of counter machines

Bisimulation equivalence decidable by use of *symbolic techniques* [Murawski et al, 2018]

- Language equivalence is decidable in the deterministic case

Translation from finitary π-calculus processes to fresh-register automata [Bandukara & Tzevelekos, 2022]

What we are working on:

A nominal logic that can express fresh-register automata properties

MOTIVATING EXAMPLES

Suppose we wanted to check if the following properties holds for two FRAs:

$$P1\text{: } At\ every\ state, there\ is\ an\ infinite\ path\ a_0, a_1, \dots, a_n$$
$$Such\ that\ \forall a_i.\, a_i \neq a_{i-1}$$

$$P2\text{: } At\ each\ state, there\ is\ an\ infinite\ path\ a_0, a_1, \dots a_n$$
$$such\ that\ \forall a_i.\, a_i \notin \{a_0, \dots, a_{i-1}\}$$

NOMINAL MODAL μ-CALCULUS

Given a countably infinite set of variables $Var\ (x, y, etc.)$ and recursion variables $VAR\ (X, Y, etc.)$, we define:

- $Formulae \ni \phi ::= u = u \mid \phi \vee \phi \mid \neg\phi \mid \bigvee_{x \in \mathbb{A}} \phi \mid \langle \ell \rangle \phi \mid (\mu X(\vec{x}).\, \phi)(\vec{u}) \mid X(\vec{u})$
- $Values \ni u ::= x \mid a$
- $Labels \ni \ell ::= \tau \mid (t, u)$

Defined according to the specification of Register Automata

- countably infinite set $\mathbb{A}$ of names ranged over by a (and variants)
- finite set Σ of tags ranged over by t (and variants)

Built on previous works by Dam [Dam, 2003] and Klin [Klin & Łełyk, 2017].

Extension of HML with recursion (*modal μ-calculus*) was introduced by Kozen [Kozen, 1983]

NOMINAL SETS - DEFINITIONS

Nominal sets (Pitts, 2013)

- A set X with action $\cdot$ of the group of finite permutations of $\mathbb{A}$ such that all elements of X are finitely supported
- A set $S \subseteq \mathbb{A}$ of names supports an element $x \in X$ if for all $\pi \in Perm(\mathbb{A})$:
  - $(\forall a \in S. \pi \cdot a = a) \Rightarrow \pi \cdot x = x$

Equivariant (Pitts, 2013)

- A relation R over a nominal set X is *equivariant* when for all $x \in X$ and permutations π:
  - $x \in R$ iff $\pi \cdot x \in R$

NOMINAL SETS - DEFINITIONS

Orbit

- Given a nominal set X, the orbit of any element $x \in X$ is:

$$\mathcal{O}(x) = \left\{\pi \cdot \left(\vec{a}, x(\vec{a})\right) \middle| \, \vec{a} \in \mathbb{A}^n, \pi \in PERM(\mathbb{A})\right\}$$

Orbit-finite

- A nominal set X is *orbit-finite* if there is a finite subset $\{x_1, \dots, x_n\} \subseteq X$ such that:

$$X = \bigcup_i \{\pi \cdot x_i \mid \pi \text{ } is\ a\ permutation\}$$

NOMINAL LTS

Definition 2. *A* nominal Labelled-Transition System (nominal LTS) *is a tuple* $\mathcal{L} = \langle \mathcal{S}, L, \rightarrow \rangle$, *where* $\mathcal{S}$ *is a nominal set of states,* L *is a nominal set of actions and* $\rightarrow \subseteq \mathcal{S} \times L \times \mathcal{S}$ *is an equivariant transition relation.*
$\mathcal{L}$ *is called* orbit-finite *if* $\mathcal{S}$ *and* L *are orbit-finite nominal sets.*

Definition 3. *A normal-nominal LTS* $= \langle \mathcal{S}, L, \rightarrow \rangle$ *with* $L = \{\tau\} \cup (\Sigma \times \mathbb{A})$ *is a nominal LTS with the restriction that for every* $\kappa \xrightarrow{\ell} \kappa'$

$$\mathrm{supp}(\kappa') \subseteq \mathrm{supp}(\kappa) \cup \mathrm{supp}(\ell).$$

Moreover, let us set $\mathcal{U} = \mathcal{P}(\mathcal{S})$ *and for each* $n \in \mathbb{N}$

$$\mathcal{U}_n = \{f : \mathbb{A}^n \rightarrow \mathcal{U} \mid f \textit{ is equivariant}\}.$$

NOMINAL MODAL μ-CALCULUS: SEMANTICS

Defined on normal-nominal LTS where states are configurations of register automata

- Same as fresh-register automata configurations, without histories

Let $\mathcal{U}$ be the set of all configurations and $\xi: VAR \rightarrow \bigcup_{n\in\mathbb{N}} \mathcal{U}_n$ be a recursion variable assignment.

The semantics of a formula ϕ with respect to ξ, written $[\![\phi]\!]_\xi$ is given inductively by:

$$[\![a = b]\!]_\xi = \emptyset$$

$$[\![a = a]\!]_\xi = \mathcal{U}$$

$$[\![\phi_1 \vee \phi_2]\!]_\xi = [\![\phi_1]\!]_\xi \cup [\![\phi_2]\!]_\xi$$

$$[\![\neg\phi]\!]_\xi = \mathcal{U} \setminus [\![\phi]\!]_\xi$$

$$[\![\bigvee_{x\in\mathbb{A}} \phi]\!]_\xi = \bigcup_{a\in\mathbb{A}} [\![\phi\{a/x\}]\!]_\xi$$

$$[\![\langle\ell\rangle\phi]\!]_\xi = \{U \in \mathcal{U} \mid \exists U \xrightarrow{\ell} U'. U' \in [\![\phi]\!]_\xi\}$$

NOMINAL MODAL μ-CALCULUS: SEMANTICS

Defined on normal-nominal LTS where states are configurations of register automata

- Same as fresh-register automata configurations, without histories

Let $\mathcal{U}$ be the set of all configurations and $\xi : VAR \rightharpoonup \bigcup_{n\in\mathbb{N}} \mathcal{U}_n$ be a recursion variable assignment.

The semantics of a formula ϕ with respect to ξ, written $[\![\phi]\!]_\xi$ is given inductively by:

$$[\![(\mu X(\vec{x}).\phi)(\vec{a})]\!]_\xi = (\mathrm{lfp}(\lambda f.\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]}))(\vec{a})$$

$$[\![X(\vec{a})]\!]_\xi = \xi(X)(\vec{a})$$

UTILIZING OUR LOGIC

Suppose we want to check if the following property holds for an RA:

$$At\ every\ state, there\ is\ an\ infinite\ path\ a_0, a_1, \dots, a_n$$
$$Such\ that\ \forall a_i. a_i \neq a_{i-1}$$

Does the RA satisfy the following formula:

$$\bigvee_{x \in \mathbb{A}} \langle x \rangle . [\nu X(z). \bigvee_{y \in \mathbb{A}} z \neq y \wedge \langle y \rangle X(y)](x)$$

UTILIZING OUR LOGIC

$\llbracket \bigvee_{x \in \mathbb{A}} \langle x \rangle . [\nu X(z). \bigvee_{y \in \mathbb{A}} z \neq y \wedge \langle y \rangle X(y)](x) \rrbracket_{\xi}$

$\rightarrow \bigcup_{a \in \mathbb{A}} \llbracket \langle a \rangle . [\nu X(z). \bigvee_{y \in \mathbb{A}} z \neq y \wedge \langle y \rangle X(y)](a) \rrbracket_{\xi}$

- For all names a, examine configurations with an a transition

UTILIZING OUR LOGIC

$$\llbracket \bigvee_{x \in \mathbb{A}} \langle x \rangle . [\nu X(z). \bigvee_{y \in \mathbb{A}} z \neq y \wedge \langle y \rangle X(y)](x) \rrbracket_{\xi}$$

$$\to \bigcup_{a \in \mathbb{A}} \llbracket \langle a \rangle . [\nu X(z). \bigvee_{y \in \mathbb{A}} z \neq y \wedge \langle y \rangle X(y)](a) \rrbracket_{\xi}$$

- For all names a, examine configurations with an a transition

$$\xrightarrow{a} GFP(\lambda f . \lambda c \llbracket [\bigvee_{y \in \mathbb{A}} c \neq y \wedge \langle y \rangle X(y)] \rrbracket_{\xi[X \mapsto f]})(a)$$

UTILIZING OUR LOGIC

$$\llbracket \bigvee_{x\in\mathbb{A}} \langle x \rangle . [\nu X(z). \bigvee_{y\in\mathbb{A}} z \neq y \wedge \langle y \rangle X(y)](x) \rrbracket_{\xi}$$

$\rightarrow \bigcup_{a\in\mathbb{A}} \llbracket \langle a \rangle . [\nu X(z). \bigvee_{y\in\mathbb{A}} z \neq y \wedge \langle y \rangle X(y)](a) \rrbracket_{\xi}$

- For all names a, examine configurations with an a transition

$\xrightarrow{a} GFP(\lambda f . \lambda c \, \llbracket [\bigvee_{y\in\mathbb{A}} c \neq y \wedge \langle y \rangle X(y)] \rrbracket_{\xi[X\mapsto f]})(a)$

$\rightarrow \bigcup_{b\in\mathbb{A}} \llbracket a \neq b \wedge \langle b \rangle X(b) \rrbracket_{\xi[X\mapsto f]}$

- For all names b that are different to a, examine configurations that follow from the above with a b transition

UTILIZING OUR LOGIC

$$\llbracket \bigvee_{x\in\mathbb{A}} \langle x\rangle . [\nu X(z). \bigvee_{y\in\mathbb{A}} z \neq y \wedge \langle y\rangle X(y)](x) \rrbracket_{\xi}$$

$$\rightarrow \bigcup_{a\in\mathbb{A}} \llbracket \langle a\rangle . [\nu X(z). \bigvee_{y\in\mathbb{A}} z \neq y \wedge \langle y\rangle X(y)](a) \rrbracket_{\xi}$$

- For all names a, examine configurations with an a transition

$$\xrightarrow{a} GFP(\lambda f. \lambda c \, \llbracket [\bigvee_{y\in\mathbb{A}} c \neq y \wedge \langle y\rangle X(y)] \rrbracket_{\xi[X\mapsto f]})(a)$$

$$\rightarrow \bigcup_{b\in\mathbb{A}} \llbracket a \neq b \wedge \langle b\rangle X(b) \rrbracket_{\xi[X\mapsto f]}$$

- For all names b that are different to a, examine configurations that follow from the above with a b transition

$$\xrightarrow{b} \bigcup_{c\in\mathbb{A}} \llbracket b \neq c \wedge \langle c\rangle X(c) \rrbracket_{\xi[X\mapsto f]}$$

- For all names c that are different to b, examine configurations that follow from the above with a c transition

UTILIZING OUR LOGIC

$[\![\bigvee_{x\in\mathbb{A}}\langle x\rangle.[\nu X(z).\bigvee_{y\in\mathbb{A}} z \neq y \wedge \langle y\rangle X(y)](x)\ ]\!]_{\xi}$

$\rightarrow\ \bigcup_{a\in\mathbb{A}}[\![\langle a\rangle.[\nu X(z).\bigvee_{y\in\mathbb{A}} z \neq y \wedge \langle y\rangle X(y)](a)]\!]_{\xi}$

- For all names a, examine configurations with an a transition

$\xrightarrow{a} GFP(\lambda f.\lambda c\ [\![[\bigvee_{y\in\mathbb{A}} c \neq y \wedge \langle y\rangle X(y)]]\!]_{\xi[X\mapsto f]})(a)$

$\rightarrow \bigcup_{b\in\mathbb{A}}[\![a \neq b \wedge \langle b\rangle X(b)]\!]_{\xi[X\mapsto f]}$

- For all names b that are different to a, examine configurations that follow from the above with a b transition

$\xrightarrow{b} \bigcup_{c\in\mathbb{A}}[\![b \neq c \wedge \langle c\rangle X(c)]\!]_{\xi[X\mapsto f]}$

- For all names c that are different to b, examine configurations that follow from the above with a c transition

$\xrightarrow{c} \ldots$

FINITE SEMANTICS

Require a finite representation for model checking

Given a finite set $S \subseteq \mathbb{A}$, some $n \in \omega$, we define:

$$\mathcal{U} \upharpoonright S = \{x \in \mathcal{U} \mid supp(x) \subseteq S\}$$
$$S_n = S^n \to (\mathcal{U} \upharpoonright S)$$

FINITE SEMANTICS

$$\mathcal{U} \restriction S = \{x \in \mathcal{U} \mid supp(x) \subseteq S\}$$
$$S_n = S^n \to (\mathcal{U} \restriction S)$$

Require a finite representation for model checking

Let $S \subseteq \mathbb{A}$ be a finite set, large enough to cater all bound names (*and one fresh one!*)

the restricted definition uses same rules as before, except:

$$\begin{aligned}
[\![a = a]\!]^S_\xi &= \mathcal{U} \restriction S \\
[\![\bigvee_{x \in \mathbb{A}} \phi]\!]^S_\xi &= \bigcup_{a \in S} [\![\phi\{a/x\}]\!]^S_\xi \\
[\![\neg\phi]\!]^S_\xi &= (\mathcal{U} \restriction S) \setminus [\![\phi]\!]^S_\xi \\
[\![(\mu X(\vec{x}).\phi)(\vec{a})]\!]^S_\xi &= (\mathrm{lfp}(\lambda f^{S_n}.\lambda \vec{b}^{S^n}.[\![\phi\{\vec{b}/\vec{x}\}]\!]^S_{\xi[X \mapsto f]}))(\vec{a})
\end{aligned}$$

And where $\xi : VAR \rightharpoonup \bigcup_{n \in \mathbb{N}} S_n$

PROOF OF MODEL CHECKING

Proposition 21. *Let* $\mathcal{L} = \langle \mathcal{S}, \mathcal{L}, \rightarrow \rangle$ *be a orbit-finite normal-nominal LTS with with* $L = \{\tau\} \cup (\Sigma \times \mathbb{A})$*, let* ϕ *be a formula. There is a finite set* S *(depending linearly on* ϕ*) such that for each* $S \subseteq \mathbb{A}$ *with* $|\mathrm{supp}(\phi)| + \|\phi\| < |S|$ *and* $\mathrm{supp}(\phi) \subseteq S$*,* $[\![\phi]\!]_\xi = \mathcal{O}([\![\phi]\!]^S_\xi)$.

**orbit-finite means the state-space is finite up to name permutations*

HISTORY-DEPENDENT EXTENSION

As with FRAs, the previous definition is extended to account for global freshness

Adding the following construct to our definition:

$$Formulae \ni \phi ::= \cdots \mid \#u$$

i.e., adds semantics that a name will be *fresh* in the current state

SEMANTICS (EXTENDED WITH HISTORIES)

$$
\begin{aligned}
[\![\#a]\!]_\xi &= \{(s,H) \mid a \notin H\} \\
[\![a=b]\!]_\xi &= \emptyset \\
[\![a=a]\!]_\xi &= \mathscr{U} \\
[\![\phi_1 \vee \phi_2]\!]_\xi &= [\![\phi_1]\!]_\xi \cup [\![\phi_2]\!]_\xi \\
[\![\neg\phi]\!]_\xi &= \mathscr{U} \setminus [\![\phi]\!]_\xi \\
[\![\bigvee_{x\in\mathbb{A}} \phi]\!]_\xi &= \bigcup_{a\in\mathbb{A}} [\![\phi\{a/x\}]\!]_\xi \\
[\![\langle\ell\rangle\phi]\!]_\xi &= \{(s,H) \mid \exists (s,H) \xrightarrow{\ell} (s',H').\,(s',H') \in [\![\phi]\!]_\xi\} \\
[\![(\mu X(\vec{x}).\phi)(\vec{a})]\!]_\xi &= (\mathrm{lfp}(\lambda f.\lambda\vec{b}.[\![\phi\{\vec{b}/\vec{x}\}]\!]_{\xi[X\mapsto f]}))(\vec{a}) \\
[\![X(\vec{a})]\!]_\xi &= \xi(X)(\vec{a})
\end{aligned}
$$

PREVIOUS MOTIVATING EXAMPLE:

Suppose we wanted to check if the following property holds for an FRA:

$$\text{At each state, there is an infinite path } a_0, a_1, \ldots a_n \text{ such that } \forall a_i.\, a_i \notin \{a_0, \ldots, a_{i-1}\}$$

PREVIOUS MOTIVATING EXAMPLE:

Suppose we wanted to check if the following property holds for an FRA:

$$\text{At each state, there is an infinite path } a_0, a_1, \dots a_n \text{ such that } \forall a_i.\, a_i \notin \{a_0, \dots, a_{i-1}\}$$

$$\text{Does the FRA satisfy the following formula:}$$

$$\nu X. \bigvee_{x \in \mathbb{A}} \# x \wedge \langle x \rangle X$$

PREVIOUS MOTIVATING EXAMPLE:

$[\![\nu X. \bigvee_{x\in\mathbb{A}} \#x \wedge \langle x\rangle X]\!]$

$\rightarrow GFP(\lambda f.\lambda\_.[\![\bigvee_{x\in\mathbb{A}} \#x \wedge \langle x\rangle X]\!])$

PREVIOUS MOTIVATING EXAMPLE:

$[\![\nu X. \bigvee_{x \in \mathbb{A}} \#x \wedge \langle x \rangle X]\!]$

$\rightarrow GFP(\lambda f. \lambda\_. [\![\bigvee_{x \in \mathbb{A}} \#x \wedge \langle x \rangle X]\!])$

$\rightarrow \bigcup_{a \in \mathbb{A}} [\![\#a \wedge \langle a \rangle X]\!]$

- [all configurations where some name a is not in the history and has an a transition]

PREVIOUS MOTIVATING EXAMPLE:

$[\![\nu X. \bigvee_{x \in \mathbb{A}} \#x \wedge \langle x \rangle X]\!]$

$\to GFP(\lambda f. \lambda\_. [\![\bigvee_{x \in \mathbb{A}} \#x \wedge \langle x \rangle X]\!])$

$\to \bigcup_{a \in \mathbb{A}} [\![\#a \wedge \langle a \rangle X]\!]$

- [all configurations where some name a is not in the history and has an a transition]

$\xrightarrow{a} \bigcup_{b \in \mathbb{A}} [\![\#b \wedge \langle b \rangle X]\!]$

- [all configurations where some name b is not in the history and has a b transition]
  - The name a would be in the history at this point!

SUMMARY

Explored fresh-register automata and infinite alphabets

Looked at Nominal modal μ-calculus, a logic that can represent register automata

Detailed how to expand the logic to a history dependent setting

Examined some representable properties

THANK YOU

Mohamed Hamza Bandukara

m.h.bandukara@qmul.ac.uk