Harnessing LTL With Freeze Quantification

Daniel Hausmann, Stefan Milius and Lutz Schröder
University Erlangen-Nürnberg, Germany

Oberseminar

4 August 2020
Model Checking for Data Languages

- **Linear-time** (e.g. LTL) vs. branching-time (CTL, $\mu$-calculus)

**Basic linear-time model checking principle:**

Transform $\varphi$ to automaton $A(\varphi)$, check inclusion of model in $A(\varphi)$

Inclusion checking for “data automata” (infinite alphabet $\rightsquigarrow$ data):

- Register Automata (RA) (Kaminski et al. 1994) undecidable
- Nondeterministic Orbit-finite Automata (NOFA) (Neven et al. 2004, Boyańczyk et al. 2014) undecidable
- Variable Automata (Grumberg et al. 2010) undecidable
Logics with Freeze Quantification

Freeze LTL (Demri, Lazić, 2007):

- paths: data words \((P_1, d_1), (P_2, d_2), \ldots\)
- operators \(\downarrow_r \varphi: "d_i \rightarrow r; \varphi"\), \(\uparrow_r: "d_i = r?"\)

Flat Freeze LTL (Bollig et al. 2019):

- for all subformulae \(\phi_1 \mathbf{U} \phi_2\), no freeze operator in \(\phi_1\)

Model Checking for Freeze LTL:

- Freeze LTL over RA (Demri, Lazić, 2007) \text{undecidable}
- Flat Freeze LTL over OCA (Bollig et al. 2019) \text{NExpTime}

One-Counter Automata
(Schröder et al. 2017): Regular bar expressions and Regular Nondeterministic Nominal Automata (RNNA), using nominal sets

- RNNA inclusion checking is in para-PSpace

Our aim here: Linear-time fixpoint logic for RNNA

- Introduce alternating nominal automata (ANA)
- Transform formulae to equivalent ANA
- Generalize RNNA inclusion checking to ANA inclusion checking to obtain decidable model checking
**G-sets**

A **G-set** for group $G$: $(X, \cdot : G \times X \to X)$ such that

$$
\pi \cdot (\rho \cdot x) = (\pi \rho) \cdot x \\
1 \cdot x = x
$$

For $x \in X$, $Y \subseteq X$, put

$$
\text{fix } x = \{ \pi \in G \mid \pi \cdot x = x \} \\
\text{Fix } Y = \bigcap_{x \in Y} \text{fix } x
$$

$x \in X$ has finite **support** if there is finite set $Y \subseteq X$ such that

$$
\text{Fix}(Y) \subseteq \text{fix}(x)
$$

Then let $\text{supp}(x)$ denote least supporting set.
Nominal Sets

$G$-sets

$G$-set for group $G$: $(X, \cdot : G \times X \to X)$ such that

$$\pi \cdot (\rho \cdot x) = (\pi \rho) \cdot x \quad 1 \cdot x = x$$

For $x \in X$, $Y \subseteq X$, put

$$\text{fix } x = \{\pi \in G | \pi \cdot x = x\} \quad \text{Fix } Y = \bigcap_{x \in Y} \text{fix } x$$

$x \in X$ has finite support if there is finite set $Y \subseteq X$ such that

$$\text{Fix}(Y) \subseteq \text{fix}(x)$$

Then let $\text{supp}(x)$ denote least supporting set

Names, permutations

Fix countable set $A$ of names, $G$: group of fin. permutations on $A$

Then $(A, \cdot : G \times A \to A)$ with $\pi \cdot a = \pi(a)$ is a $G$-set
Nominal sets

Nominal set \( X \): \( G \)-set \((X, \cdot)\) s.t. all \( x \in X \) have finite support

Abstraction set: \([A]X = (A \times X)/\sim\) where

\[(a, x) \sim (b, y) \text{ if and only if } (ac) \cdot x = (bc) \cdot y \text{ for any fresh } c\]

\( \langle a \rangle x \): \( \sim \)-equivalence class of \((a, x)\)
Nominal Sets, ctd.

Nominal sets

Nominal set $X$: $G$-set $(X, \cdot)$ s.t. all $x \in X$ have finite support

Abstraction set: $[A]X = (A \times X)/\sim$ where 

$$(a, x) \sim (b, y) \text{ if and only if } (ac) \cdot x = (bc) \cdot y \text{ for any fresh } c$$

$\langle a \rangle x$: $\sim$-equivalence class of $(a, x)$

Bar strings

Set of finite bar strings: $\mathbb{B} = \mathbb{B}^*$ where $\mathbb{B} = A \cup \{ |a | a \in A \}$

$\equiv_\alpha$ on bar strings: equivalence generated by

$$w | av \equiv_\alpha w | bu \text{ iff } \langle a \rangle v = \langle b \rangle u \text{ in } [A]\mathbb{B}$$
$s()$ accepts e.g. $|a|b$ and $|b|a$ but does not accept $|a|a$
## Syntax

\[ \varphi, \psi ::= T \mid \epsilon \mid \neg \varphi \mid \varphi \land \psi \mid \Diamond_a \varphi \mid \Diamond_{\mid a} \varphi \mid X \mid \mu X. \varphi \]

\[ (a \in A, X \in V) \]

requiring positivity of fixpoint variables

Define \( \equiv_\alpha \) on formulae, e.g.

\[ \Diamond_{\mid a}(\Diamond_a T \lor \Box_b T) \equiv_\alpha \Diamond_{\mid c}(\Diamond_c T \lor \Box_b T) \]
A Linear-time Logic for RNNA

**Syntax**

\[
\phi, \psi ::= T \mid \epsilon \mid \neg \phi \mid \phi \land \psi \mid \lozenge_a \phi \mid \lozenge_{|a|} \phi \mid X \mid \mu X. \phi
\]

\( (a \in A, X \in V) \)

requiring positivity of fixpoint variables

Define \( \equiv_{\alpha} \) on formulae, e.g.

\[
\lozenge_{|a|} (\lozenge_a T \lor \Box_b T) \equiv_{\alpha} \lozenge_c (\lozenge_c T \lor \Box_b T)
\]

**Semantics (attempt)**

Interpret formulae over bar strings using \( \sigma : V \rightarrow \mathcal{P}(B) \):

\[
\left[ \epsilon \right]_\sigma = \{ \epsilon \}
\]

\[
\left[ \lozenge_a \phi \right]_\sigma = \{ w \in B \mid w = av, v \in \left[ \phi \right]_\sigma \}
\]

\[
\left[ \lozenge_{|a|} \phi \right]_\sigma = \{ w \in B \mid w = |bv, \exists \psi. \lozenge_{|a|} \phi \equiv_{\alpha} \lozenge_{|b|} \psi, v \in \left[ \psi \right]_\sigma \} \]
Recall:

\[
[\Diamond_a \varphi]_\sigma = \{ w \in B \mid w = av, v \in [\varphi]_\sigma \}
\]

\[
[\Diamond |a\varphi]_\sigma = \{ w \in B \mid w = bv, \exists \psi. \Diamond |a\varphi \equiv \alpha \Diamond |b\psi, v \in [\psi]_\sigma \}
\]
Recall: $\left\lbrack \Diamond a \varphi \right\rbrack_\sigma = \{ w \in B \mid w = av, v \in \left\lbrack \varphi \right\rbrack_\sigma \}$

$\left\lbrack \Diamond \lvert a \varphi \right\rbrack_\sigma = \{ w \in B \mid w = \lvert bv, \exists \psi. \Diamond \lvert a \varphi \equiv \alpha \Diamond \lvert b \psi, v \in \left\lbrack \psi \right\rbrack_\sigma \}$

Let $\varphi = \mu X. \psi$, $\psi = \Diamond \lvert a (\Diamond a \top \lor \Box b X)$ so that $\text{FN}(\varphi) = b$
Recall: \[
[\Diamond_a \varphi]_\sigma = \{ w \in B \mid w = av, v \in [\varphi]_\sigma \}
\]
\[
[\Diamond|a\varphi]_\sigma = \{ w \in B \mid w = |bv, \exists \psi. \Diamond|a\varphi \equiv \alpha \Diamond|b\psi, v \in [\psi]_\sigma \}
\]

Let \( \varphi = \mu X. \psi \), \( \psi = \Diamond|a(\Diamond_a \top \lor \Box_b X) \) so that \( \text{FN}(\varphi) = b \)

We have e.g.

1. \( |cc \in [\varphi] \) since \( \psi \equiv_\alpha \Diamond|c(\Diamond_c \top \lor \Box_b X) \) and \( c \in [\Diamond_c \top \lor \Box_b X] \)

2. \( |c|dc \not\in [\varphi] \) since \( |dc \not\in [\Diamond_c \top \lor \Box_b X] \) since \( c \not\in [\psi] \)

3. \( |cb|db|ee \in [\varphi] \) since \( |ee \in [\psi] \) so that \( b|ee \in [\Box_b \psi] \),
   \( |db|ee \in [\Diamond|a(\Diamond_a \top \lor \Box_b \psi)] \), \( b|db|ee \in [\Box_b(\Diamond|a(\Diamond_a \top \lor \Box_b \psi))] \)
Ensuring Monotonicity

Modal operators are not monotone (yet)!

e.g. $|bb \in [\lozenge_a(\lozenge_a \top \lor \bot)]$ since $b \in [\lozenge_b \top \lor \bot]$ but
$|bb \notin [\lozenge_a(\lozenge_a \top \lor \lozenge_b \top)]$ since $\forall \chi. \lozenge_a(\lozenge_a \top \lor \lozenge_b \top) \not\equiv \alpha \lozenge_b \chi$
Ensuring Monotonicity

Modal operators are not monotone (yet)!

\[ bb \in \models_{a}(\Diamond_{a}T \lor \bot) \] since \( b \in \models_{b}T \lor \bot \) but

\[ bb \notin \models_{a}(\Diamond_{a}T \lor \Diamond_{b}T) \] since \( \forall \chi. \models_{a}(\Diamond_{a}T \lor \Diamond_{b}T) \not\equiv_{\alpha} \Diamond_{b}\chi \)

Stepping stone: alternative semantics \( [-]_{\sigma}' \), closed under \( \equiv_{\alpha} \)

\[ [\Diamond_{a}\phi]_{\sigma}' = \{ w \in B \mid w \equiv_{\alpha} bv, \exists \psi. \models_{a}\phi \equiv_{\alpha} \Diamond_{b}\phi, v \in [\psi]_{\sigma}' \} \]
Ensuring Monotonicity

Modal operators are not monotone (yet)!

e.g. \( |bb \in [\lozenge_a(\lozenge_a \top \lor \bot)] \) since \( b \in [\lozenge_b \top \lor \bot] \) but
\( |bb \notin [\lozenge_a(\lozenge_a \top \lor \lozenge_b \top)] \) since \( \forall \chi. \lozenge_a(\lozenge_a \top \lor \lozenge_b \top) \not\equiv_{\alpha} \lozenge_b \chi \)

Stepping stone: alternative semantics \([\neg]\)'\( _\sigma \), closed under \( \equiv_{\alpha} \)

\[
[\lozenge_a \varphi]'_\sigma = \{ w \in B \mid w \equiv_{\alpha} |bv, \exists \psi. \lozenge_a \varphi \equiv_{\alpha} \lozenge_b \varphi, v \in [\psi]'_\sigma \}
\]

Then \( |bb \in [\lozenge_a(\lozenge_a \top \lor \bot)]' \) since \( b \in [\lozenge_b \top \lor \bot]' \) and
\( |bb \in [\lozenge_a(\lozenge_a \top \lor \lozenge_b \top)]' \) since \( |bb \equiv_{\alpha} |cc, c \in [\lozenge_c \top \lor \lozenge_b \top]' \)
Ensuring Monotonicity

Modal operators are not monotone (yet)!

e.g. \( |bb \in [\Diamond_a(\Diamond_a \top \lor \bot)] \) since \( b \in [\Diamond_b \top \lor \bot] \) but
\( |bb \notin [\Diamond_a(\Diamond_a \top \lor \Diamond_b \top)] \) since \( \forall \chi. \Diamond_a(\Diamond_a \top \lor \Diamond_b \top) \not\equiv_\alpha \Diamond_b \chi \)

Stepping stone: alternative semantics \([-\] \)', closed under \( \equiv_\alpha \)

\[
[\Diamond_a \varphi]_\sigma' = \{ w \in B \mid w \equiv_\alpha |bv, \exists \psi. \Diamond_a \varphi \equiv_\alpha \Diamond_b \varphi, v \in [\psi]'_\sigma \}
\]

Then \( |bb \in [\Diamond_a(\Diamond_a \top \lor \bot)]' \) since \( b \in [\Diamond_b \top \lor \bot]' \) and
\( |bb \in [\Diamond_a(\Diamond_a \top \lor \Diamond_b \top)]' \) since \( |bb \equiv_\alpha |cc, c \in [\Diamond_c \top \lor \Diamond_b \top]' \)

\( \rightsquigarrow \) Semantics well-defined but does not match RNNA (yet)!
Idea: incorporate explicit name “losing” (“forgetting”) in formulae

Abbreviate

\[
\text{choice}(S, a, \psi) = \text{nd}(S \setminus \{a\}, \psi) \lor \text{nd}(S \cup \{a\}, \psi)
\]

Put \( \text{nd}(\varphi) = \bigvee_{S \subseteq \text{FN}(\varphi)} \text{nd}(S, \varphi) \) where \( \text{nd}(S, \varphi) \) is defined by

\[
\text{nd}(S, \diamond a \psi) = \begin{cases} 
\diamond_a(\text{choice}(S, a, \psi)) & a \in S \\
\bot & a \notin S
\end{cases}
\]

\[
\text{nd}(S, \diamond |a \psi) = \diamond |a(\text{choice}(S, a, \psi))
\]

plus commutation with non-modal operators
Name dropping formulae, Example

Let $\varphi = \Diamond |a \psi$, $\psi = \Diamond |b \chi$, $\chi = \Diamond a \top \lor \Diamond b \top$

| $a|bb \in \llbracket \phi \rrbracket$ |
| $a|aa \notin \llbracket \phi \rrbracket$ |

| $a|bb \in \llbracket \text{nd}(\phi) \rrbracket$ |
| $a|aa \in \llbracket \text{nd}(\phi) \rrbracket$ |
Lemma
For all formulae $\varphi$, we have $[\varphi]' = \text{nd}(\varphi)' = \text{nd}(\varphi)$.

Define degree $\text{deg}(\varphi) = \max\{|\text{FN}(\psi)| \mid \psi \text{ is subformula of } \varphi\}$, closure $\text{cl}(\varphi)$ (can be seen as syntax graph of $\varphi$)

Lemma
For all formulae $\varphi$ such that $\text{deg}(\varphi) = k$, $|\text{cl}(\varphi)| = n$, we have

$$|\text{cl}(\text{nd}(\varphi))| \leq 2^{k+1}n.$$
For nominal set $X$, the orbit of $x \in X$ is $\{\pi \cdot x \mid \pi \in G\}$ and $S \subseteq X$ is equivariant if $\pi \cdot x \in S$ for all $\pi \in G$, $x \in S$.
For nominal set $X$, the orbit of $x \in X$ is $\{\pi \cdot x \mid \pi \in G\}$ and $S \subseteq X$ is equivariant if $\pi \cdot x \in S$ for all $\pi \in G$, $x \in S$.

**Definition (Alternating nominal automaton (ANA))**

$A = (Q_{\exists}, Q_{\forall}, \rightarrow, s, F)$ with
- orbit-finite nominal set $Q = Q_{\exists} \cup Q_{\forall}$ of states
- equivariant transition relation $\rightarrow \subseteq Q \times (B \cup \epsilon) \times Q$
- equivariant set $F$ of accepting states

such that $q \xrightarrow{a} q'$ and $\langle a \rangle q' = \langle b \rangle q''$ imply $q \xrightarrow{b} q''$ ($\alpha$-invariance)

and such that $\{(a, q') \mid q \xrightarrow{a} q'\}$ and $\{\langle a \rangle q' \mid q \xrightarrow{a} q'\}$ are finite.
Alternating Nominal Automata, acceptance

Runs of ANA $A = (Q_{\exists}, Q_{\forall}, \rightarrow, s, F)$ are trees labelled with states, not sequences of states

**Definition (Accepting run trees)**

A run tree for $w \in \mathbb{B}$ is accepting if its branching follows $\rightarrow$ along $w$ and adheres $Q_{\exists}$ and $Q_{\forall}$, and all its leaves have labels from $F$.

**Definition (Accepted language)**

Literal acceptance:

\[ L_0(A) = \{ w \in \mathbb{B} \mid \text{there is an accepting run tree of } A \text{ for } w \} \]

Accepted bar language:

\[ L_\alpha(A) = L_0/\equiv_\alpha \]
Given $\varphi$, define **formula automaton** $A(\varphi) = (Q_\exists, Q_\forall, \rightarrow, s, F)$:

$\triangleright$ $Q = \{ \pi \psi \mid \psi \in \text{cl}(\varphi), \pi \in G \}$ with obvious kind ($Q_\exists$ or $Q_\forall$)

$\triangleright$ $s = \varphi$, $F = \{ \top, \neg \epsilon \}$

\[
\begin{align*}
\phi \land \psi & \xrightarrow{\epsilon} \phi \\
\phi \lor \psi & \xrightarrow{\epsilon} \phi \\
\mu X. \phi & \xrightarrow{\epsilon} \phi[\mu X. \phi/X] \\
\Diamond a \phi & \xrightarrow{a} \phi \\
\Diamond |a \phi & \xrightarrow{|b} \chi \\
\langle a \rangle \phi & = \langle b \rangle \chi \\
\Box |a \phi & \xrightarrow{|b} \chi \\
\langle a \rangle \phi & = \langle b \rangle \chi
\end{align*}
\]

**Lemma**

For monotone $\varphi$, we have $L_\alpha(A(\varphi)) = \llbracket \varphi \rrbracket$. 

\[15 / 20\]
Let $\varphi = \Diamond |a\psi$, $\psi = \mu X. \chi$, $\chi = \Box |b\theta$, $\theta = \Diamond a \top \lor \Box b X$
Model Checking

bar NFA: NFA $M$ with alphabet $B$; $L_\alpha(M) = L_0(M)/\equiv_\alpha$

**Definition (satisfaction over bar NFA)**

For monotone $\varphi$,

$$M \models \varphi \text{ if and only if } L_\alpha(M) \subseteq \llbracket \varphi \rrbracket'$$

Model checking: Given $M$ and $\varphi$, check whether

$$L_\alpha(M) \subseteq \llbracket \varphi \rrbracket' = \llbracket \text{nd}(\varphi) \rrbracket = L_\alpha(A(\text{nd}(\varphi)))$$

name dropping construction  formulae are ANA
Nondeterministic Algorithm (check whether $L_{\alpha}(M) \not\subseteq L_{\alpha}(A)$)

1. Initialize $q = q_0$, $\Phi = \{\{q_0\}\}$.

2. If $\emptyset \in \Phi$, abort. If $q$ is accepting, guess whether word ends now. If it ends, terminate positively if all $\Gamma \in \Phi$ contain non-accepting state.

3. Guess $\alpha$ and $q'$ s.t. $q \xrightarrow{\alpha} q'$ in $M$. Put $\Phi := \bigcup_{\Gamma \in \Phi} (\text{succ}(\Gamma, \alpha))$,\n
$$\text{succ}(\psi, \alpha) = \begin{cases} \{\{\chi \mid \psi \xrightarrow{\alpha} \chi \text{ in } A\}\} & \psi \in Q_{\forall} \\ \{\{\chi\} \mid \psi \xrightarrow{\alpha} \chi \text{ in } A\} & \psi \in Q_{\exists} \end{cases}.$$ Goto 2.
Language Inclusion Checking

Given: bar NFA $M$, ANA $A$

Nondeterministic Algorithm (check whether $L_\alpha(M) \not\subseteq L_\alpha(A)$)

1. Initialize $q = q_0$, $\Phi = \{ \{ q_0' \} \}$.
2. If $\emptyset \in \Phi$, abort. If $q$ is accepting, guess whether word ends now. If it ends, terminate positively if all $\Gamma \in \Phi$ contain non-accepting state.
3. Guess $\alpha$ and $q'$ s.t. $q \xrightarrow{\alpha} q'$ in $M$. Put $\Phi := \bigcup_{\Gamma \in \Phi} (\text{succ}(\Gamma, \alpha))$,

\[
\text{succ}(\psi, \alpha) = \begin{cases} 
\{ \{ \chi \mid \psi \xrightarrow{\alpha} \chi \text{ in } A \} \} & \psi \in Q_\forall \\
\{ \{ \chi \} \mid \psi \xrightarrow{\alpha} \chi \text{ in } A \} & \psi \in Q_\exists
\end{cases}
\]

Goto 2.

Lemma

The inclusion problem is in $\text{ExpSpace}$.

(complement and nondeterminism do not affect space complexity)
Model Checking and Satisfiability Checking

Ingredients:

▶ Model checking: Given bar NFA $M$ and $\varphi$, check whether

$$L_\alpha(M) \subseteq L_\alpha(A(\text{nd}(\varphi)))$$

▶ Validity checking: Given universal RNNA $M_\top$ and $\varphi$, check

$$L_\alpha(M_\top) \subseteq L_\alpha(A(\text{nd}(\varphi)))$$

▶ We have $|\text{cl}(\text{nd}(\varphi))| \leq 2^{k+1}n$

▶ Inclusion problem is in $\text{ExpSpace} \left( \mathcal{O}(|M| + 2^{\text{cl}(\text{nd}(\varphi))}) \right)$

Corollary

The model checking and validity problems are in $2\text{ExpSpace}$ (and in para-$\text{ExpSpace}$ with $k$ as parameter).
Results so far:

- Linear-time logic for finite bar strings
  - name-dropping construction on formulae, blow-up: $2^{k+1}n$
- Alternating nominal automata (ANA), generalizing RNNA
- Name-dropping formulae are ANA
- Model / satisfiability checking over bar NFA is elementary!
  (in $2\text{ExpSpace}$ and para-$\text{ExpSpace}$)

Future work:

- How about infinite bar strings? (nominal Büchi automata)
- Conjecture: Inclusion checking between ANA is elementary