Runtime Monitoring for Hennessy-Milner Logic with Recursion over Systems with Data

Ongoing Work

Léo Exibard

Tuesday, December 6th, 2022

Séminaire du LIGM

Context: Formal Verification

Goal

Ensure that a system behaves correctly

System

- Non-terminating
- Produces execution traces

Correctness

Simple properties $\rightsquigarrow$ regular

(or almost)

Model-checking

Given a model of the system S and a property φ,

Check that S complies with φ

Context: Formal Verification

Goal

Ensure that a system behaves correctly

System

- Non-terminating
- Produces execution traces

Correctness

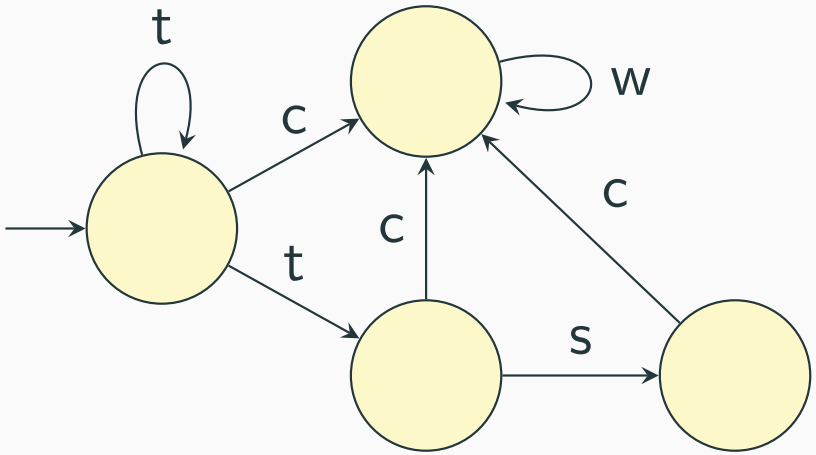
Simple properties $\rightsquigarrow$ regular (or almost)

Model-checking

Given a model of the system S and a property φ,

Check that S complies with φ

Example: Server producing tokens



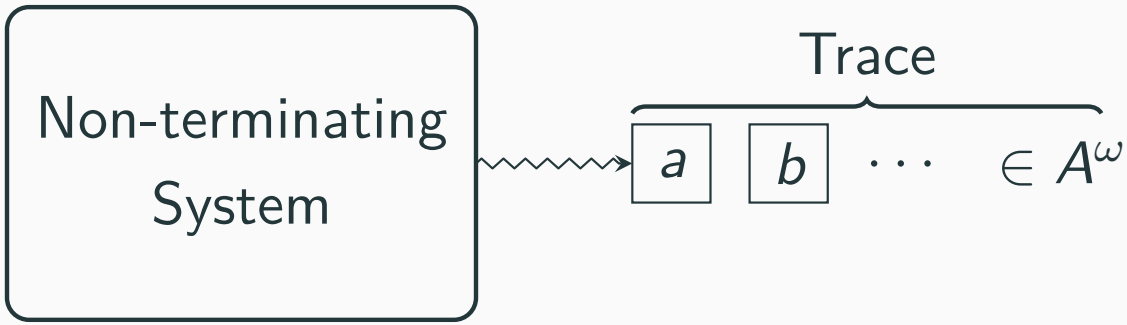
The server does not produce the secret token (**s**) before closing (**c**).

$\rightsquigarrow$ In LTL: $(\neg s)\mathbf{U}c$.

Runtime Verification

- We do not have a model of the system

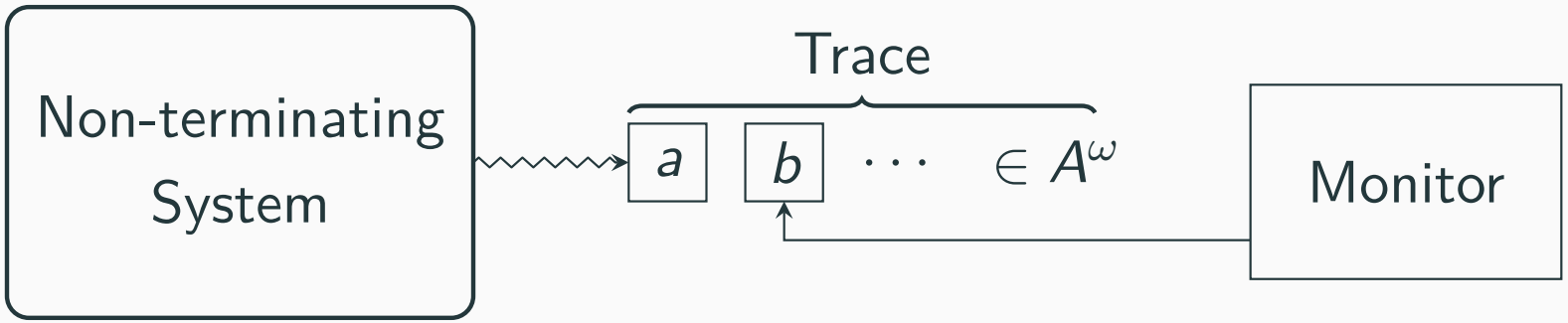
➔ Check whether the property holds *along the execution*



Runtime Verification

- We do not have a model of the system

➔ Check whether the property holds *along the execution*



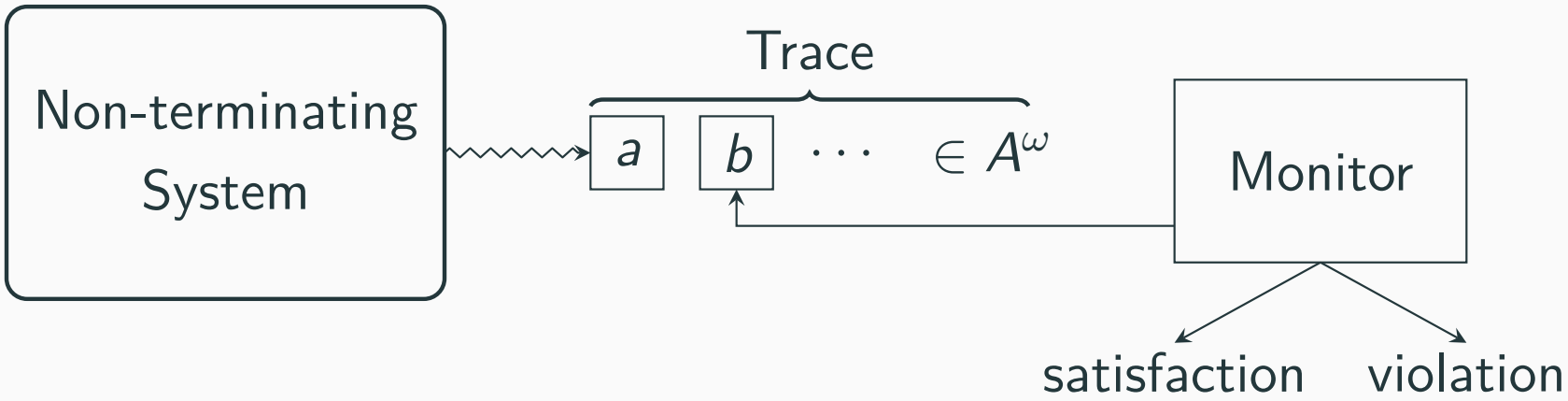
Monitor

Processes trace to raise a *verdict*:

Runtime Verification

- We do not have a model of the system

➔ Check whether the property holds *along the execution*



Monitor

Processes trace to raise a *verdict*:

- Satisfaction `yes`
- Violation `no`

Runtime Monitoring

Irrevocability

Once produced, a verdict cannot *change*

Runtime Monitoring vs Model-checking

- Model-checking: given *a model of the system* S and a property φ, check whether S satisfies $\varphi \rightarrow$ emptiness of $S \cap \overline{A_\varphi}$
- Runtime monitoring: given *a trace* t *of the system* and a property φ, check if t satisfies $\varphi \rightarrow$ membership of t in A_φ

Questions

- What properties are monitorable (for a given monitor model)?
- How to synthesise monitors from formulas?

Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$

Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

  $$\min X.\left(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a \neq s} [a]X\right)$$

Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\Big(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a\neq s}[a]X\Big)$$

Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\left(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a \neq s} [a]X\right)$$

Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\left(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a \neq s} [a]X\right)$$

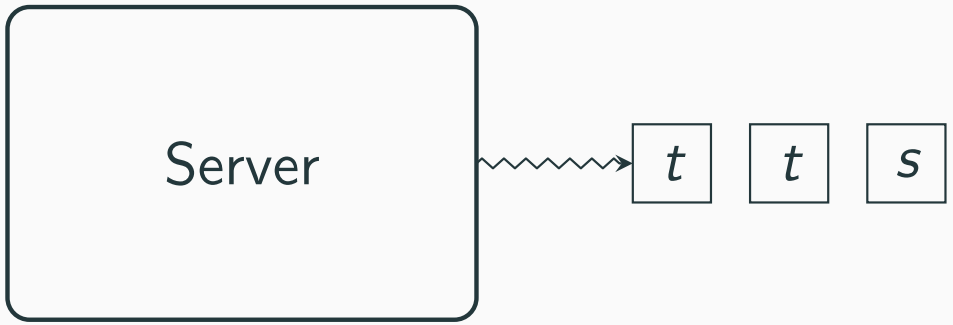
Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\Big(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a\neq s}[a]X\Big)$$



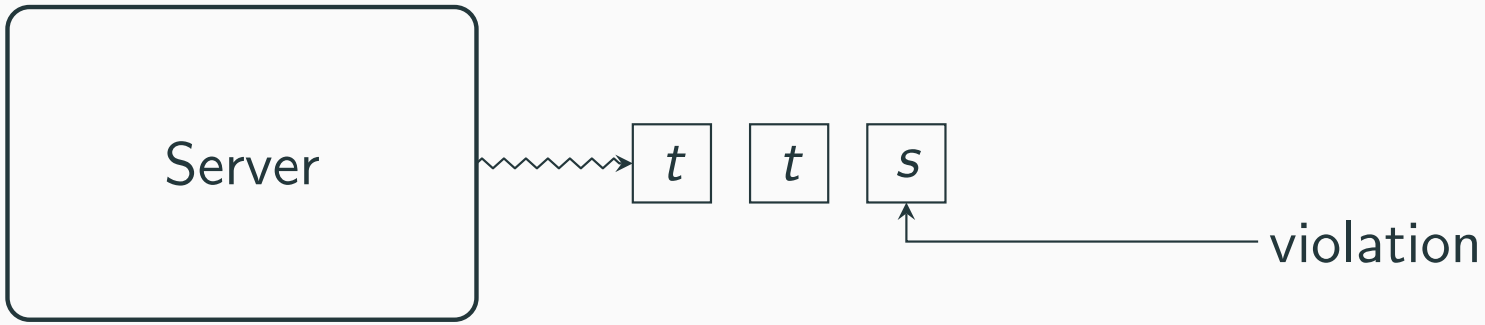
Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\left(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a \neq s} [a]X\right)$$



Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\Big(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a\neq s}[a]X\Big)$$

Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\left(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a \neq s} [a]X\right)$$

Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\left(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a \neq s} [a]X\right)$$

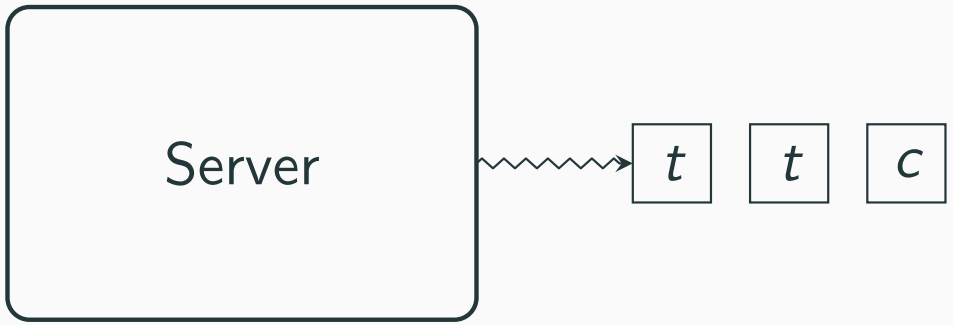
Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\left(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a \neq s} [a]X\right)$$



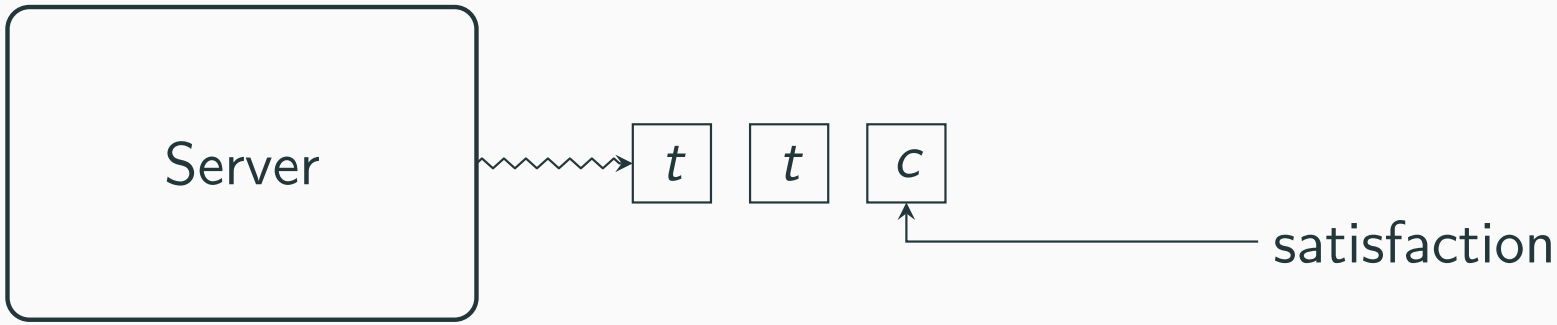
Example : A Server Producing Tokens

The secret token is not leaked before closing

- Linear-time Temporal Logic (LTL): $(\neg s)\mathbf{U}c$
- Linear-time modal μ-calculus:

  aka recHML

$$\min X.\left(\langle c\rangle \mathtt{tt} \vee ([s]\mathtt{ff} \wedge \bigwedge_{a \neq s} [a]X\right)$$



Hennessy-Milner Logic with Recursion

Syntax

$$\varphi, \psi \in \mathit{recHML} ::= \mathtt{tt} \mid \langle a \rangle \varphi \mid \varphi \vee \psi \mid \mathit{min}\ X.\varphi \mid X$$
$$\mid \mathtt{ff} \mid [a]\varphi \mid \varphi \wedge \psi \mid \mathit{max}\ X.\varphi$$

Recursion: Fixpoints

- min: least fixpoint
- max: greatest fixpoint

Linear Time

Models = traces

- $\langle a \rangle \varphi$: $a \underbrace{\quad t}_{\varphi}$
- $[a]\varphi$: $\overline{a}t$ $\quad a \underbrace{\quad t}_{\varphi}$

Monitoring recHML

Monitors

$$m, n \in \mathit{Mon} ::= v \in \mathit{Vrd} \mid a.m \mid m + n \mid \mathit{rec}\ X.m \mid X$$

$$v \in \mathit{Vrd} ::= \mathrm{end} \mid \mathrm{no} \mid \mathrm{yes}$$

Monitorable Fragments

$$\varphi, \psi \in \mathit{sHML} ::= \mathtt{tt} \mid \mathtt{ff} \mid [a]\varphi \mid \varphi \wedge \psi \mid \mathit{max}\ X.\varphi \mid X \text{ (violation)}$$

$$\varphi, \psi \in \mathit{cHML} ::= \mathtt{tt} \mid \mathtt{ff} \mid \langle a \rangle \varphi \mid \varphi \vee \psi \mid \mathit{min}\ X.\varphi \mid X \text{ (satisfaction)}$$

Monitor Synthesis

- Finite-state monitors suffice
- Compositional (Francalanza, Aceto, and Ingólfsdóttir 2015)
- Monitors can be determinised (Aceto et al. 2020)

Systems with Data

Data = elements from an infinite alphabet with some structure.

Data Domains

Infinite set with decidable theory: $(\mathbb{N}, =)$, $(\mathbb{Q}, <)$, $(\Sigma^*, <_{\text{prec}})$

Server providing tokens

- Tokens are now integers
- We only need equality checks

Various Logics

- Freeze LTL
- $FO^2[<, \sim]$
- etc (see Demri and Quaas 2021)
- Here: recHML with quantifiers (inspired from Groote and Mateescu 1998)

$$\begin{aligned}\varphi, \psi \in \mathit{recHML} ::= \ & \mathtt{tt} \mid \langle b(\star)\rangle\varphi \mid \varphi \vee \psi \mid \mathit{min}\ X.\varphi \mid X(\vec{v}) \mid \exists x.\varphi \\ & \mid \mathtt{ff} \mid [b(\star)]\varphi \mid \varphi \wedge \psi \mid \mathit{max}\ X.\varphi \mid \forall x.\varphi\end{aligned}$$

$b(\star)$: quantifier-free FO formula

Examples

The first token does not appear again

$$\varphi_{\mathsf{first}\neq} = \forall x\, [\star = x]\, \max X. \Big([\star = x]\mathtt{ff} \wedge [\star \neq x] X(x) \Big)$$

No two consecutive tokens are the same

$$\varphi_{\mathsf{succ}\neq} = \forall x. [\star = x]\, \max X. \Big(\forall y. [\star = y = x]\mathtt{ff} \wedge [\star = y \neq x] X(y) \Big)$$

No two tokens are the same

$$\varphi_{\forall\neq} = \max X. \Big(\forall x. [\star = x] \big(\max Y. [\star = x]\mathtt{ff} \wedge [\star \neq x] Y \big) \wedge [\star \neq x] X \Big)$$

Monitoring with Data

Monitors

$$m, n \in \mathit{Mon} ::= v \in \mathit{Vrd} \mid b(\star).m \mid \mathit{guess}\ x.m \mid m + n \mid \mathit{rec}\ X(\vec{v}).m \mid X(\vec{v})$$

$$v \in \mathit{Vrd} ::= \mathrm{end} \mid \mathrm{no} \mid \mathrm{yes}$$

Evaluating non-determinism

Run monitors in parallel

The 'guess' construct (Apt and Plotkin 1986)

- Guess the satisfaction (resp. violation) witness
  - Equality: accumulate forbidden values
  - Richer theories: accumulate constraints

Deterministic Monitors

$$m, n \in \mathit{DMon} ::= v \in \mathit{Vrd} \mid \bigotimes_{i \in I} b_i(\star) m_i \mid \mathit{rec}\ X(\vec{v}).m \mid X(\vec{v})$$

where for all $i \neq j \in I$, b_i and b_j are mutually exclusive.

Examples

The first token does not appear again

$$\varphi_{\mathsf{first}\neq} = \forall x\, [\star = x]\, \mathsf{max}\, X.\Big([\star = x]\mathtt{ff} \wedge [\star \neq x]X(x)\Big)$$

$$(\star = x)\mathit{rec}\ X.(\star = x)\mathrm{no} \otimes (\star \neq x)X$$

No two tokens are the same

$$\varphi_{\forall\neq} = \mathsf{max}\, X.\Big(\forall x.[\star = x]\big(\,\mathsf{max}\, Y.[\star = x]\mathtt{ff} \wedge [\star \neq x]\, Y\big) \wedge [\star \neq x]X\Big)$$

$$\mathit{rec}\ X.(\star = x)\mathit{rec}\ Y.\Big((\star = x)\mathrm{no} \wedge (\star \neq x)\, Y\Big) \wedge (\star \neq x)X$$

Monitorable Fragments

$$\varphi, \psi \in \mathit{cHML} ::= \mathtt{tt} \mid \mathtt{ff} \mid \langle b(\star) \rangle \varphi \mid \varphi \vee \psi \mid \mathit{min}\ X.\varphi \mid X(\vec{v}) \mid \exists x.\varphi$$

$$\varphi, \psi \in \mathit{sHML} ::= \mathtt{ff} \mid \mathtt{tt} \mid [b(\star)] \varphi \mid \varphi \wedge \psi \mid \mathit{max}\ X.\varphi \mid X(\vec{v}) \mid \forall x.\varphi$$

$$\varphi, \psi \in \mathit{detsHML} ::= \mathtt{ff} \mid \mathtt{tt} \mid \mathit{max}\, X.\varphi \mid X(\vec{v}) \mid \bigwedge_{i \in I} [\forall x. x = \star \wedge b_i(\star)] \varphi_i$$

Register Automata

Data-free setting: recHML $\equiv \omega$-regular

Register Automata (Kaminski and Francez 1994)

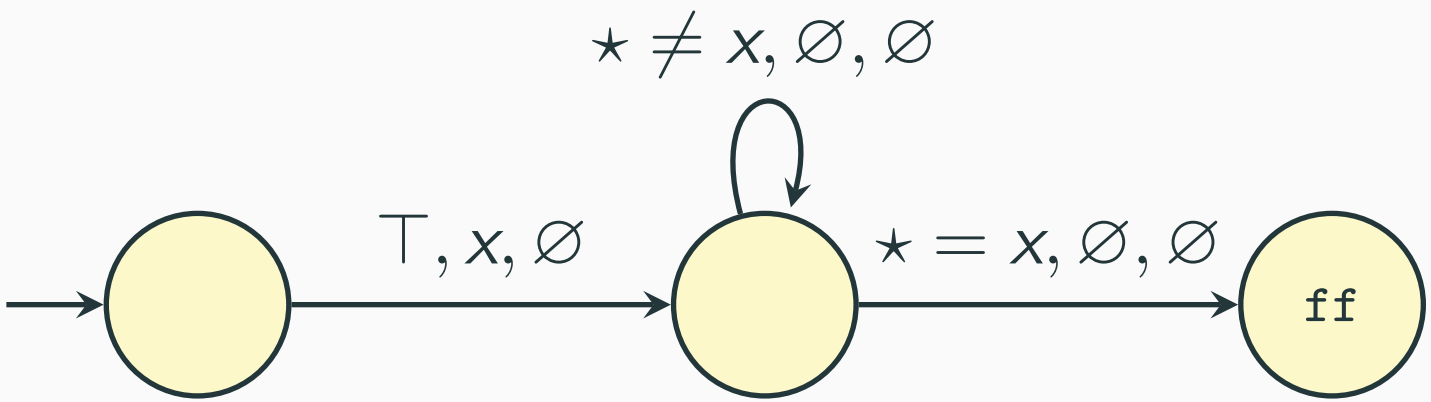
Finite automata with a finite set **R** of registers

- Store data
- Test register content

Transitions $q \xrightarrow{\varphi, A, G} q'$

- $\varphi \in \mathrm{QF}(R, \star)$: test
- $A \subseteq R$: assignment
- $G \subseteq R$: guessing

$$\forall x\, [\star = x] \max X. \Big([\star = x]\mathtt{ff} \wedge [\star \neq x] X(x) \Big)$$



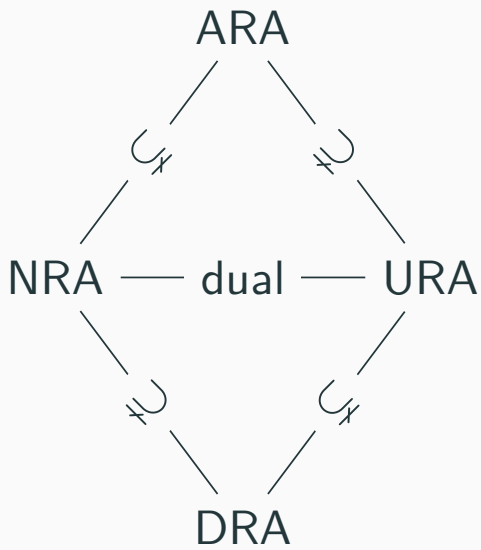
recHML (with data) and Register Automata

Theorem (work in progress)

recHML and register automata are equi-expressive. More precisely:

- recHML $\equiv$ alternating RA
- sHML $\equiv$ non-deterministic reachability RA
- cHML $\equiv$ universal safety RA
- $\text{sHML}^{\text{nf}} \equiv$ deterministic RA (in particular, no guessing)

Fact



Consequences

Strict hierarchy between fragments

➔ In particular, sHMLnf is not a normal form

➔ Monitors do not determinise (need for parallelism)

Undecidability Results

- (semantic) membership to a fragment is undecidable
- Monitorability is undecidable

Non-maximality

The data values in the first block are pairwise distinct:

$$\{d_0 \dots d_n \# \cdots \mid \forall 0 \leq i < j \leq n, d_i \neq d_j\}$$

➔ Violations can be detected by a NRA, not by a URA.

Conclusion

- Monitor synthesis extends to systems with data
- However, the logic is not as well-behaved...
- Leverage correspondence with register automata

Future Work

- Maximal fragments
- Fragments with efficient monitors
- First-order formulas in modalities
- How to evaluate accumulated constraints?

Other Interests

- Reactive synthesis
- Richer models of computation
- Games on graphs
- Multi-agent systems