Facing infinity in model checking highly expressive specification languages

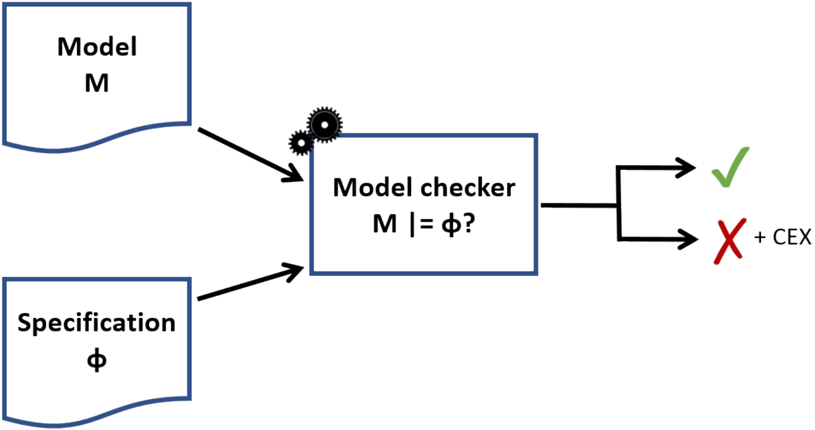
PhD candidate: Enrico Magnago[1][2]

advisor: Alessandro Cimatti[2]
co-advisor: Alberto Griggio[2]

[1]University of Trento, [2]Fondazione Bruno Kessler

Introduction

Context: Symbolic Model Checking



Model checking settings

Different modelling and specification formalisms

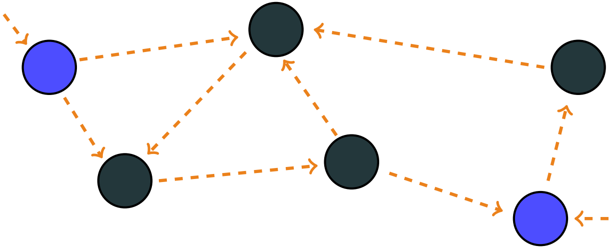
	invariants	**LTL**	**MTL**
Finite State TS	InvFTS	LTLFTS	MTLFTS
Infinite State TS	InvITS	LTLITS	MTLITS
Timed Automata	InvTA	LTLTA	MTLTA
Timed TS	InvTTS	LTLTTS	MTLTTS
Hybrid Systems	InvHS	LTLHS	MTLHS

Modelling languages: Transition Systems (TS)

$M \doteq \langle V, I, T \rangle$

- V : set of symbols/variables.
- I : formula over V representing initial states.
- T : formula over $V \cup V'$ representing the transitions.

Finite (FTS) if every $v \in V$ has a finite domain,
infinite (ITS) otherwise.

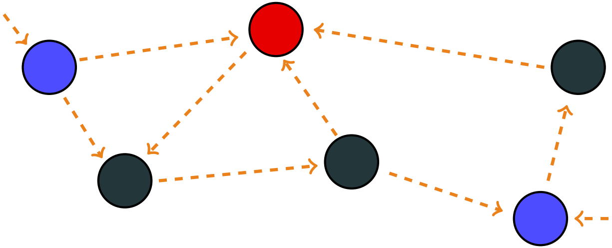


Modelling languages: Fair Transition Systems (TS)

$M \doteq \langle V, I, T, F \rangle$

- V : set of symbols/variables.
- I : formula over V representing initial states.
- T : formula over $V \cup V'$ representing the transitions.
- F : formula over V representing the fair states.

Finite (FTS) if every $v \in V$ has a finite domain,
infinite (ITS) otherwise.



Modelling languages: Timed Automata (TA)

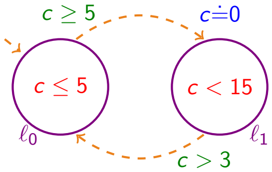
Timed Automata

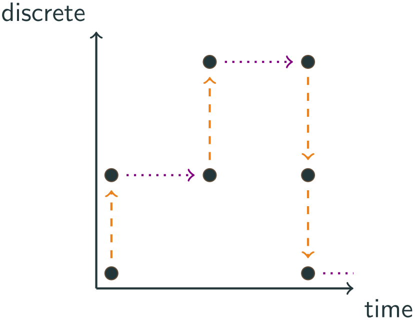
Finite set of locations: ℓ_0, ℓ_1;

location invariants;

guards and resets on

instantaneous (discrete) transitions.





Timed Transition System

$$\text{TTS} \doteq \text{ITS} + \text{time}.$$

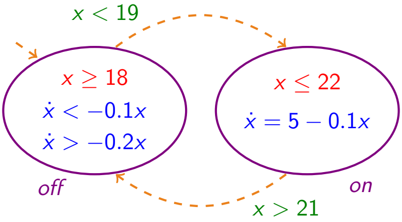
Modelling languages: Hybrid Systems (HS)

Hybrid Systems

Extend TAs with a more complex continuous behaviour: ODE.

Continuous component allows the modelling of physical phenomena.

Thermostat



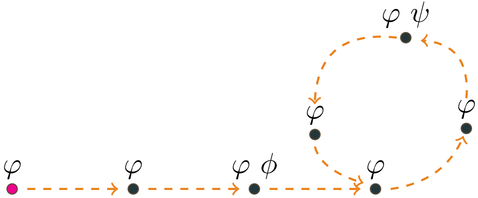
Specifications: Linear Temporal Logic (LTL)

LTL: relative ordering of events in a sequence of states

$\boldsymbol{G}\varphi$: φ always holds in the future;

$\boldsymbol{F}\phi$: ϕ will eventually hold;

$\boldsymbol{GF}\psi$: ψ happens infinitely often (*fairness*).

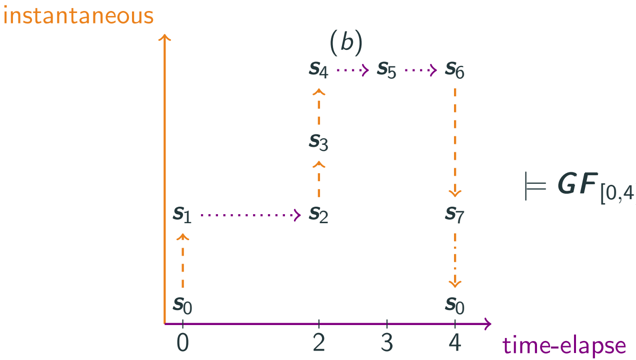


ϕ holds for M iff every execution π of M satisfies the property: $\pi \models \phi$.

Specifications: Metric Temporal Logic (MTL)

MTL: quantify "time" distance between events

Bounds on temporal operators specify quantitative time constraints.



$\models \boldsymbol{GF}_{[0,4)}\, b$

Dependencies between MC settings

Specification languages: Invariant, LTL, MTL.

Modelling languages: FTS, ITS, TTS, HS.

Overview of expressiveness dependencies between the different combinations of model and specification languages.

$$
\begin{tikzcd}
\textsc{Inv}\mathrm{FTS} \arrow[r] \arrow[d] & \mathrm{LTLFTS} \arrow[r] \arrow[d] & \mathrm{MTLFTS} \arrow[d] \\
\textsc{Inv}\mathrm{ITS} \arrow[r] \arrow[d] & \mathrm{LTLITS} \arrow[r] \arrow[d] & \mathrm{MTLITS} \arrow[d] \\
\textsc{Inv}\mathrm{TTS} \arrow[r] \arrow[d] & \mathrm{LTLTTS} \arrow[r] \arrow[d] & \mathrm{MTLTTS} \arrow[d] \\
\textsc{Inv}\mathrm{HS} \arrow[r] & \mathrm{LTLHS} \arrow[r] & \mathrm{MTLHS}
\end{tikzcd}
$$

Dependencies between MC settings

Specification languages: Invariant, LTL, MTL.

Modelling languages: FTS, ITS, TTS, HS.

Overview of expressiveness dependencies between the different combinations of model and specification languages.

$$
\begin{tikzcd}
\textsc{Inv}\mathrm{FTS} \arrow[r] \arrow[d] & \mathrm{LTLFTS} \arrow[r] \arrow[d] & \mathrm{MTLFTS} \arrow[d] \\
\textsc{Inv}\mathrm{ITS} \arrow[r] \arrow[d] & \mathrm{LTLITS} \arrow[r] \arrow[d] & \mathrm{MTLITS} \arrow[d] \\
\textsc{Inv}\mathrm{TTS} \arrow[r] \arrow[d] & \mathrm{LTLTTS} \arrow[r] \arrow[d] & \mathrm{MTLTTS} \arrow[d] \\
\textsc{Inv}\mathrm{HS} \arrow[r] & \mathrm{LTLHS} \arrow[r] & \mathrm{MTLHS} \arrow[uul, dashed, "\text{reduction}"']
\end{tikzcd}
$$

Automata-Theoretic LTL Model Checking

Let M be a fair transition system and φ a LTL property.

$$
\begin{gathered}
M \models \varphi \\
\Updownarrow \\
\mathcal{L}(M) \subseteq \mathcal{L}(\varphi) \\
\Updownarrow \\
\mathcal{L}(M) \cap \mathcal{L}(\neg\varphi) = \emptyset \\
\Updownarrow \\
\mathcal{L}(M) \cap \mathcal{L}(M_{\neg\varphi}) = \emptyset \\
\Updownarrow \\
\mathcal{L}(M \times M_{\neg\varphi}) = \emptyset
\end{gathered}
$$

Contributions

CAV 2019	Extending NUXMV with timed transition systems and timed temporal properties.
INFOCOMP 2020	SMT-Based Satisfiability of First-Order LTL with Event Freezing Functions and Metric Operators.
VMCAI 2021	Proving the existence of fair paths in infinite state systems.
ATVA 2021	Automatic discovery of fair paths in infinite-state transition systems.
INFOCOMP 2022	LTL falsification in infinite-state systems.

Outline

1. Temporal logics:
   a. $\mathrm{MTL}_{0,\infty}$ syntax and semantics;
   b. reduction from $\mathrm{MTL}_{0,\infty}$ to LTL with discrete time.
2. Falsification techniques:
   a. segmentation;
   b. decomposition.
3. Experimental results.
4. Conclusions.

MTL

Overview

Problem statement

Correctness of system dependent on timing of events.

Model of time depends on context and abstraction level.

Motivation

Need specification language to express *time* properties directly.

Approach

Reduce to "untimed" (ITS).

Employ techniques developed for "untimed" systems.

Time models

Time model $\langle \mathbb{T}, <, \mathbf{0}, v \rangle$**:**

- $\mathbb{T}$: temporal domain;
- $<\subseteq \mathbb{T} \times \mathbb{T}$: total order over $\mathbb{T}$;
- $\mathbf{0}$: minimal element of $\mathbb{T}$ w.r.t. $<$;
- $v : \mathbb{T} \mapsto \mathbb{R}_0^+$: map timepoint to time value or "real time".

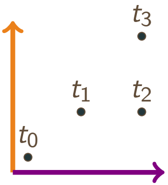
Selected time models

		real-time monotonicity	
		strict	weak
time-points	discrete	**discrete** $\mathbb{T} \doteq \mathbb{N}$	**super-discrete** $\mathbb{T} \doteq \mathbb{N} \times \mathbb{N}$
	dense	**dense** $\mathbb{T} \doteq \mathbb{R}_0^+$	**super-dense** $\mathbb{T} \doteq \mathbb{N} \times \mathbb{R}_0^+$

Time points sequence: $t_0 < t_1 < t_2 < t_3$.

Strictly monotonic

Weakly monotonic



MTL Syntax

$$
\begin{aligned}
\varphi ::\ & p(u,\ldots,u) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi \rightarrow \varphi \mid \varphi \leftrightarrow \varphi \mid \\
& \boldsymbol{X}\varphi \mid \tilde{\boldsymbol{X}}\varphi \mid \boldsymbol{Y}\varphi \mid \tilde{\boldsymbol{Y}}\varphi \mid \\
& \boldsymbol{G}_I\varphi \mid \boldsymbol{F}_I\varphi \mid \varphi\boldsymbol{U}_I\varphi \mid \boldsymbol{H}_I\varphi \mid \boldsymbol{P}_I\varphi \mid \varphi\boldsymbol{S}_I\varphi \mid \overrightarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \mid \overleftarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \\
u ::\ & c \mid x \mid f(u,\ldots,u) \mid u@\tilde{\boldsymbol{F}}(\varphi) \mid u@\tilde{\boldsymbol{P}}(\varphi) \mid \boldsymbol{Ite}(\varphi,u,u) \\
I ::\ & [cu,cu] \mid [cu,cu) \mid (cu,cu] \mid (cu,cu) \mid [cu,\infty) \mid (cu,\infty) \\
cu ::\ & c \mid f(cu,\ldots,cu)
\end{aligned}
$$

$\mathrm{MTL}_{0,\infty}$ restricts I: one of two boundaries must be 0 or ∞.

MTL Syntax

$$
\begin{aligned}
\varphi ::\ & p(u,\ldots,u) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi \to \\
& \boldsymbol{X}\varphi \mid \tilde{\boldsymbol{X}}\varphi \mid \boldsymbol{Y}\varphi \mid \tilde{\boldsymbol{Y}}\varphi \mid \\
& \boldsymbol{G}_I\varphi \mid \boldsymbol{F}_I\varphi \mid \varphi\boldsymbol{U}_I\varphi \mid \boldsymbol{H}_I\varphi \mid \boldsymbol{P}_I\varphi \mid \varphi\boldsymbol{S}_I\varphi \mid \overrightarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \mid \overleftarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \\
u ::\ & c \mid x \mid f(u,\ldots,u) \mid u@\tilde{\boldsymbol{F}}(\varphi) \mid u@\tilde{\boldsymbol{P}}(\varphi) \mid \boldsymbol{Ite}(\varphi,u,u) \\
I ::\ & [cu,cu] \mid [cu,cu) \mid (cu,cu] \mid (cu,cu) \mid [cu,\infty) \mid (cu,\infty) \\
cu ::\ & c \mid f(cu,\ldots,cu)
\end{aligned}
$$

counting operators

$\mathrm{MTL}_{0,\infty}$ restricts I: one of two boundaries must be 0 or ∞.

MTL Syntax

$$
\begin{aligned}
\varphi ::\; & p(u,\dots,u) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi \to \varphi \mid \varphi \leftrightarrow \varphi \mid \\
& \boldsymbol{X}\varphi \mid \tilde{\boldsymbol{X}}\varphi \mid \boldsymbol{Y}\varphi \mid \tilde{\boldsymbol{Y}}\varphi \mid \\
& \boldsymbol{G}_I\varphi \mid \boldsymbol{F}_I\varphi \mid \varphi\boldsymbol{U}_I\varphi \mid \boldsymbol{H}_I\varphi \mid \boldsymbol{P}_I\varphi \mid \varphi\boldsymbol{S}_I\varphi \mid \overrightarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \mid \overleftarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \\
u ::\; & c \mid x \mid f(u,\dots,u) \mid u@\tilde{\boldsymbol{F}}(\varphi) \mid u@\tilde{\boldsymbol{P}}(\varphi) \mid \boldsymbol{Ite}(\varphi,u,u) \\
I ::\; & [cu,cu] \mid [cu,cu) \mid (cu,cu] \mid (cu,cu) \mid [cu,\infty) \mid (cu,\infty) \\
cu ::\; & c \mid f(cu,\dots,cu)
\end{aligned}
$$

at next / last expr

$\mathrm{MTL}_{0,\infty}$ restricts I: one of two boundaries must be 0 or ∞.

MTL Syntax

discrete next / prev

$$
\begin{aligned}
\varphi ::\;& p(u,\ldots,u) \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \neg\varphi \mid \varphi\rightarrow\varphi \mid \varphi\leftrightarrow\varphi \mid \\
& \boldsymbol{X}\varphi \mid \tilde{\boldsymbol{X}}\varphi \mid \boldsymbol{Y}\varphi \mid \tilde{\boldsymbol{Y}}\varphi \mid \\
& \boldsymbol{G}_I\varphi \mid \boldsymbol{F}_I\varphi \mid \varphi\boldsymbol{U}_I\varphi \mid \boldsymbol{H}_I\varphi \mid \boldsymbol{P}_I\varphi \mid \varphi\boldsymbol{S}_I\varphi \mid \overrightarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \mid \overleftarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \\
u ::\;& c \mid x \mid f(u,\ldots,u) \mid u@\tilde{\boldsymbol{F}}(\varphi) \mid u@\tilde{\boldsymbol{P}}(\varphi) \mid \boldsymbol{Ite}(\varphi,u,u) \\
I ::\;& [cu,cu] \mid [cu,cu) \mid (cu,cu] \mid (cu,cu) \mid [cu,\infty) \mid (cu,\infty) \\
cu ::\;& c \mid f(cu,\ldots,cu)
\end{aligned}
$$

$\mathrm{MTL}_{0,\infty}$ restricts I: one of two boundaries must be 0 or ∞.

MTL Syntax

timed next / prev

$$\begin{aligned}
\varphi &:: \; p(u,\ldots,u) \mid \varphi\wedge\varphi \mid \varphi\vee\varphi \mid \neg\varphi \mid \varphi\rightarrow\varphi \mid \varphi\leftrightarrow\varphi \mid \\
&\quad \boldsymbol{X}\varphi \mid \tilde{\boldsymbol{X}}\varphi \mid \boldsymbol{Y}\varphi \mid \tilde{\boldsymbol{Y}}\varphi \mid \\
&\quad \boldsymbol{G}_I\varphi \mid \boldsymbol{F}_I\varphi \mid \varphi\boldsymbol{U}_I\varphi \mid \boldsymbol{H}_I\varphi \mid \boldsymbol{P}_I\varphi \mid \varphi\boldsymbol{S}_I\varphi \mid \overrightarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \mid \overleftarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \\
u &:: \; c \mid x \mid f(u,\ldots,u) \mid u@\tilde{\boldsymbol{F}}(\varphi) \mid u@\tilde{\boldsymbol{P}}(\varphi) \mid \boldsymbol{Ite}(\varphi,u,u) \\
I &:: \; [cu,cu] \mid [cu,cu) \mid (cu,cu] \mid (cu,cu) \mid [cu,\infty) \mid (cu,\infty) \\
cu &:: \; c \mid f(cu,\ldots,cu)
\end{aligned}$$

$\mathrm{MTL}_{0,\infty}$ restricts I: one of two boundaries must be 0 or ∞.

$\mathbf{MTL}_{0,\infty}$ Semantics: Next State

Next state operators and time models

	Strictly monotonic		Weakly monotonic	
	Discrete	Dense	Super-Discrete	Super-Dense
$X\ \top$	✓	✗	Instantaneous transitions	
$Y\ \top$	✓	✗	Instantaneous transitions	
$\tilde{X}\ \top$	✗	✓	Time-elapse transitions	
$\tilde{Y}\ \top$	✗	✓	Time-elapse transitions	

$\mathbf{MTL}_{0,\infty}$ Semantics: Next Occurrence

Next occurrence

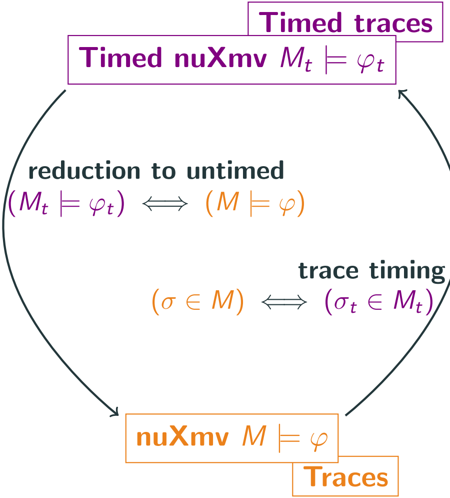
In discrete time next occurrence of φ described as:

$$\neg\varphi \boldsymbol{U} \varphi$$

In [super-]dense time there could be no such time point.
Consider last point before left-open interval:

$$\neg\varphi \boldsymbol{U}(\varphi \vee (\neg\varphi \wedge \tilde{\boldsymbol{X}}\varphi))$$

Implementation in nuXmv v2.



Falsification

Overview

Problem statement

Falsification techniques consider only restricted class of CEX.

Motivation

Inconclusive analyses hinder development of formal models.

Coarse spuriousness checks hinder convergence of CEGAR.

Approach

CEX as ITS underapproximating the model.

Language of CEX non-empty and contains only witnesses.

Search by segmentation (funnels) and composition (E-comps).

Fair paths representation

Assume we want to prove the existence of an infinite run for the code below in which $c = 0$ infinitely often.

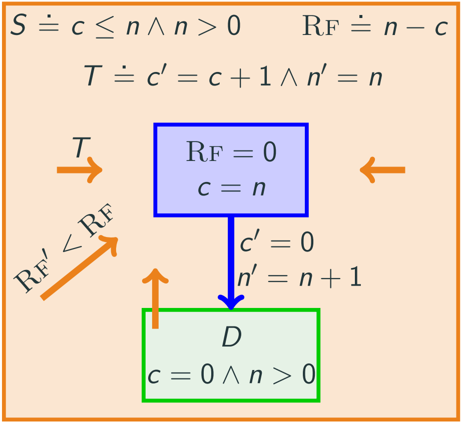
```
int c, n
while ⊤ do
    while c < n do
        c ← c + 1
    end while
    c ← nondet()
    n ← n + 1
end while
```

```
VAR c: integer; n: integer;

TRANS
  (c < n ∧ next(c)=c + 1 ∧ next(n)=n) ∨
  (c ≥ n ∧ next(n)=n + 1);

FAIRNESS c = 0;
```

Funnel-loop: reachable, non-empty underapproximation with only fair paths.



Fair paths representation

Assume we want to prove the existence of an infinite run for the code below in which $c = 0$ infinitely often.

```
int c, n
while ⊤ do
    while c < n do
        c ← c + 1
    end while
    c ← nondet()
    n ← n + 1
end while
```

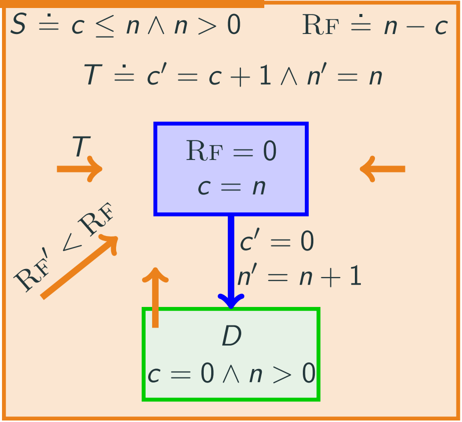
```
VAR c: integer; n: integer;

TRANS
  (c < n ∧ next(c)=c + 1 ∧ next(n)=n) ∨
  (c ≥ n ∧ next(n)=n + 1);

FAIRNESS c = 0;
```

F… non-empty … th only fair paths.

Source region with ranking function $\mathrm{R_F}$, ensures termination of inner loop.



Fair paths representation

Assume we want to prove the existence of an infinite run for the code below in which $c = 0$ infinitely often.

```
int c, n
while ⊤ do
    while c < n do
        c ← c + 1
    end while
    c ← nondet()
    n ← 
end whil
```

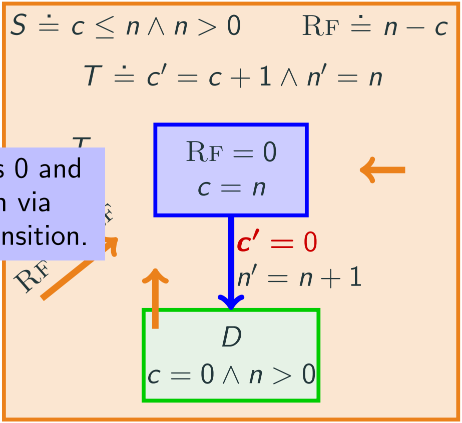
Eventually $\mathrm{R{\scriptstyle F}}$ becomes 0 and reach destination region via underapproximating transition.

```
VAR c: integer; n: integer;

TRANS
  (c < n ∧ next(c)=c + 1 ∧ next(n)=n) ∨
  (c ≥ n ∧ next(n)=n + 1);

FAIRNESS c = 0;
```

Funnel-loop: reachable, non-empty underapproximation with only fair paths.



Fair paths representation

Assume we want to prove the existence of an infinite run for the code below in which $c = 0$ infinitely often.

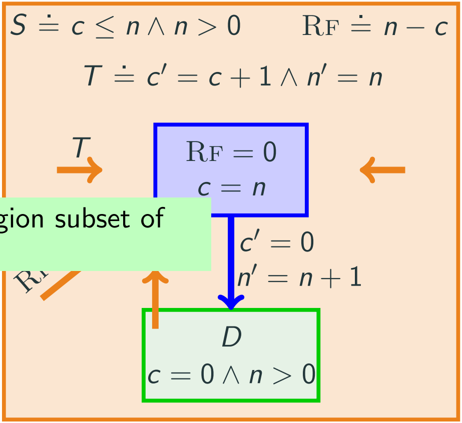
```
int c, n
while ⊤ do
    while c < n do
        c ← c + 1
    end while
    c ← nondet()
    n ← n + 1
end while
```

```
VAR c: integer; n: integer;

TRANS
  (c < n ∧ next(c)=c + 1 ∧ next(n)=n) ∨
  (c ≥ n ∧ next(n)=n + 1);

FAIRNESS c = 0;
```

Funnel-loop: reachable, non-empty underapproximation with only fair paths.



Destination region subset of the fair states.

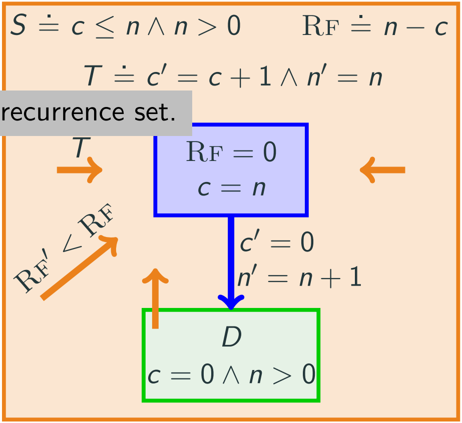
Fair paths representation

Assume we want to prove the existence of an infinite run for the code below in which $c = 0$ infinitely often.

```
int c, n
while ⊤ do
    while c < n do
        c ← c + 1
    end while
    c ←
    n ← n + 1
end while
```

Source region is a closed recurrence set.

Funnel-loop: reachable, non-empty underapproximation with only fair paths.




```
VAR c: integer; n: integer;

TRANS
  (c < n ∧ next(c)=c + 1 ∧ next(n)=n) ∨
  (c ≥ n ∧ next(n)=n + 1);

FAIRNESS c = 0;
```

Funnels (a.k.a. Inductive Reachability Witnesses[1])

Funnel $\langle V, S, T, D, \mathrm{R{\small F}} \rangle$:

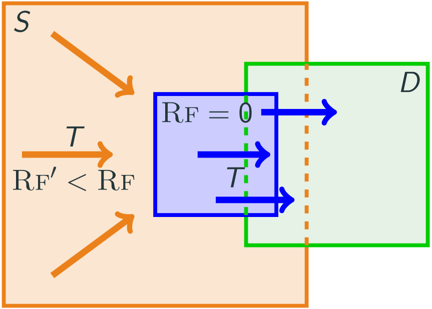
V : set of symbols;

S : source region;

T : left-total transition relation;

D : destination region;

$\mathrm{R{\small F}}$: ranking function.



From S eventually reach D via finite number of T-transitions.

Note: number of T-transitions may not have an upper bound.

[1] Ali Asadi, Krishnendu Chatterjee, Hongfei Fu, Amir Kafshdar Goharshady, Mohammad Mahdavi: Polynomial reachability witnesses via Stellensätze. PLDI 2021

Funnels (a.k.a. Inductive Reachability Witnesses)

Funnel $\langle V, S, T, D, \mathrm{R{\scriptstyle F}} \rangle$**:**

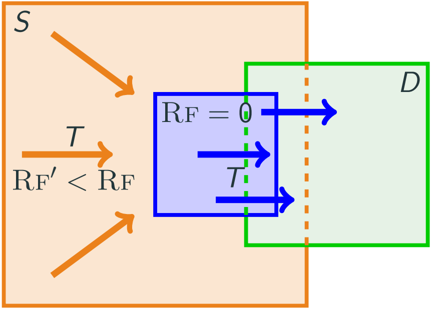
V : set of symbols;

S : source region;

T : left-total transition relation;

D : destination region;

$\mathrm{R{\scriptstyle F}}$: ranking function.



F.1 $\forall V \exists V' : S \rightarrow T;$

F.2 $\forall V, V' : (S \land \mathbf{0} < \mathrm{R{\scriptstyle F}} \land T) \rightarrow S';$

F.3 $\forall V, V' : (S \land \mathbf{0} < \mathrm{R{\scriptstyle F}} \land T) \rightarrow \mathrm{R{\scriptstyle F}}' < \mathrm{R{\scriptstyle F}};$

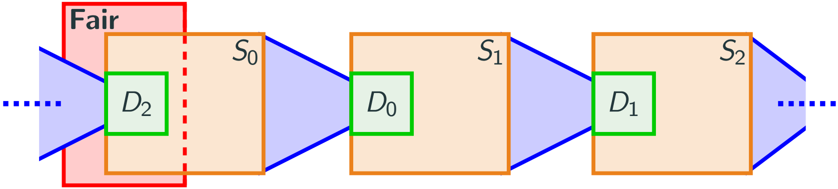
F.4 $\forall V, V' : (S \land \mathrm{R{\scriptstyle F}} = \mathbf{0} \land T) \rightarrow D'.$

Funnel-loops

Funnel-loop $[fnl_i]_{i=0}^{n-1}$**:**

Loop of funnels: $\models D_i \rightarrow S_{i+_n 1}$, for all i.

Ensure fairness: $\models D_i \rightarrow F$ for some i.



Funnel-loop: Sufficient Conditions

Search problem

Given $M \doteq \langle V, I^M, T^M, F^M \rangle$, find funnel-loop $floop \doteq [fnl_i]_{i=0}^{n-1}$ s.t.:

$M \rightsquigarrow \bigvee_{i=0}^{n-1} S_i$, *floop* is reachable in M;

$\models D_{n-1} \rightarrow F^M$, *floop* describes fair loops;

$\models S_i \wedge T_i \rightarrow T^M$, for all i, *floop* underapproximates M.

Funnel-loop: Soundness and Completeness

Search problem

Given $M \doteq \langle V, I^M, T^M, F^M \rangle$, find funnel-loop $floop \doteq [fnl_i]_{i=0}^{n-1}$ s.t.:

$$M \rightsquigarrow \bigvee_{i=0}^{n-1} S_i,$$
$$\models D_{n-1} \rightarrow F^M,$$
$$\models \bigwedge_{i=0}^{n-1} (S_i \wedge T_i \rightarrow T^M).$$

Soundness

If M admits a funnel-loop, then $\mathcal{L}(M)$ is non-empty.

Completeness

If $\mathcal{L}(M)$ is not empty, then M admits a funnel-loop of length one. The funnel-loop may not have a finite representation in the logic of choice.

Segmentation

Segmentation

Model evolves by going through sequence of phases.

Infinite fair path composed of finite segments.

Represent each segment as a funnel: look for funnel-loops of length greater than 1.

Search

Trade-off complexity of formulae with length of the funnel-loop.

Segmentation Example

Consider the ITS $M \doteq \langle V, I, T, F\rangle$, such that:

$$
\begin{aligned}
V &\doteq \{\delta, p\}, \text{ both with domain } \mathbb{R};\\
I(V) &\doteq \delta > 0 \wedge p = 0;\\
T(V, V') &\doteq (|p| < 10 \wedge \delta' = \delta \wedge p' = p + \delta) \vee\\
&\qquad (|p| \geq 10 \wedge \delta' = -\tfrac{\delta}{2} \wedge p' = \delta' + (\delta > 0 \text{ ? } 10 : -10));\\
F(V) &\doteq |p| \geq 10.
\end{aligned}
$$

$\mathcal{L}(M)$**:**

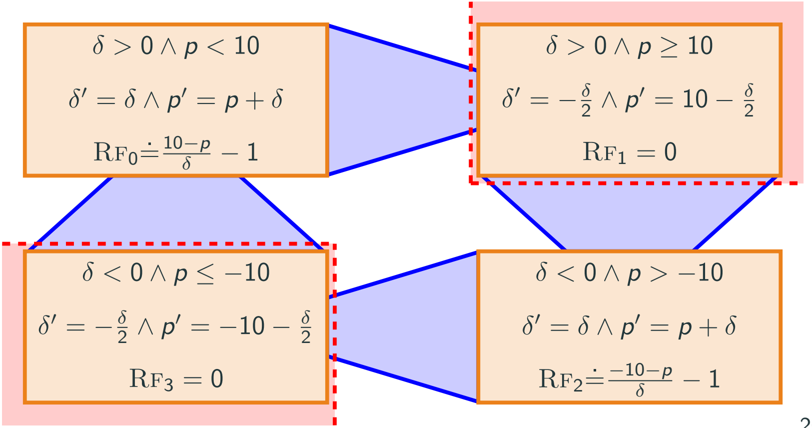
v ranges in $(-10 - |\delta|, 10 + |\delta|)$ via δ increases;

every time p reaches the boundary, $\delta' = -\frac{\delta}{2}$.

Segmentation Example

Consider the ITS $M \doteq \langle V, I, T, F \rangle$, such that:

$$\begin{aligned} T(V, V') &\doteq (|p| < 10 \wedge \delta' = \delta \wedge p' = p + \delta) \vee \\ &\quad (|p| \geq 10 \wedge \delta' = -\tfrac{\delta}{2} \wedge p' = \delta' + (\delta > 0 \ ? \ 10 : -10)); \\ F(V) &\doteq |p| \geq 10. \end{aligned}$$



Decomposition

Analyse components separately, identify CEX by composition.

Decomposition

Analyse components separately, identify CEX by composition.

Steps

1. Partition set of symbols.

Decomposition

Analyse components separately, identify CEX by composition.

Steps

1. Partition set of symbols.
2. Define E-comps for each partition:
   describe possible infinite paths projected over those symbols.

Decomposition

Analyse components separately, identify CEX by composition.

Steps

1. Partition set of symbols.
2. Define E-comps for each partition:
   describe possible infinite paths projected over those symbols.
3. Select one E-comp for each partition.

Decomposition

Analyse components separately, identify CEX by composition.

Steps

1. Partition set of symbols.
2. Define E-comps for each partition:
   describe possible infinite paths projected over those symbols.
3. Select one E-comp for each partition.
4. Compose E-comps while preserving existence of infinite paths.

Decomposition

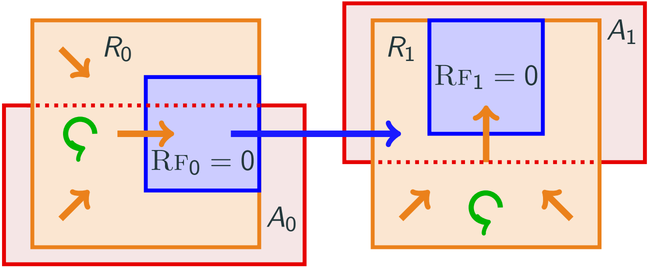
Analyse components separately, identify CEX by composition.

Steps

1. Partition set of symbols.
2. Define E-comps for each partition:
   describe possible infinite paths projected over those symbols.
3. Select one E-comp for each partition.
4. Compose E-comps while preserving existence of infinite paths.
5. Find composition in which one of such paths is fair.

E-comps

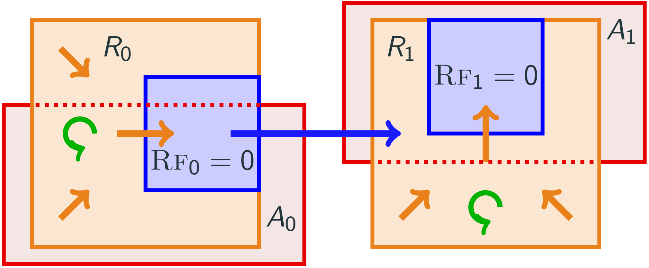
Structure with composition operator preserving infinite paths.



E-comps

Structure with composition operator preserving infinite paths.

Defines behaviour of subset of symbols provided the other meet some assumptions (A_0, A_1).

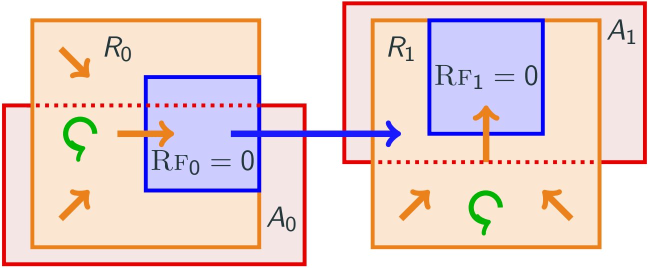


E-comps

Structure with composition operator preserving infinite paths.

Defines behaviour of subset of symbols provided the other meet some assumptions (A_0, A_1).

States grouped into regions (R_0, R_1).



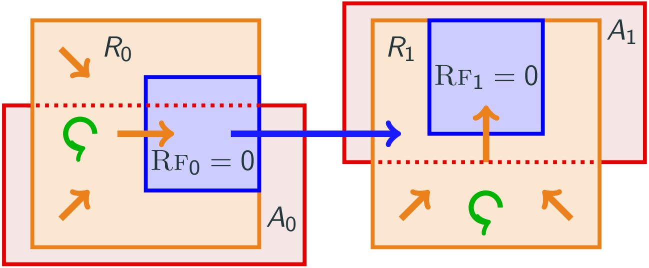
E-comps

Structure with composition operator preserving infinite paths.

Defines behaviour of subset of symbols provided the other meet some assumptions (A_0, A_1).

States grouped into regions (R_0, R_1).

Transitions between regions grouped into 3 categories: *stutter*, *ranked*, *progress*.



E-comp Composition and Projection

Composition

Product of E-comps: responsible for union of symbols.

Transition between regions iff respective assumptions are met.

Projection

Obtain smaller E-comp by considering subset of regions.

Restricts language of E-comp.

E-comp Example

Consider again the ITS $M \doteq \langle V, I, T, F \rangle$, such that:

$$
\begin{aligned}
V &\doteq \{\delta, p\}, \text{ both with domain } \mathbb{R}; \\
I(V) &\doteq \delta > 0 \land p = 0; \\
T(V, V') &\doteq (|p| < 10 \land \delta' = \delta \land p' = p + \delta) \lor \\
&\quad\ (|p| \geq 10 \land \delta' = -\tfrac{\delta}{2} \land p' = \delta' + (\delta > 0 \text{ ? } 10 : -10)); \\
F(V) &\doteq |p| \geq 10.
\end{aligned}
$$

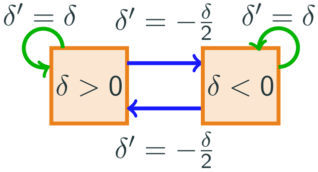
$\mathcal{L}(M)$:

p ranges in $(-10 - \delta, 10 + \delta)$ via δ increases;

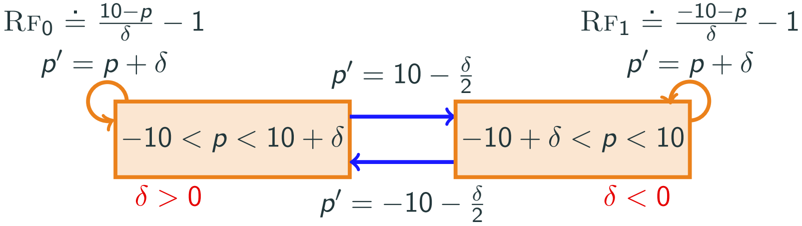
every time p reaches the boundary, $\delta' = -\frac{\delta}{2}$.

E-comp Example Components

E-comp responsible for δ

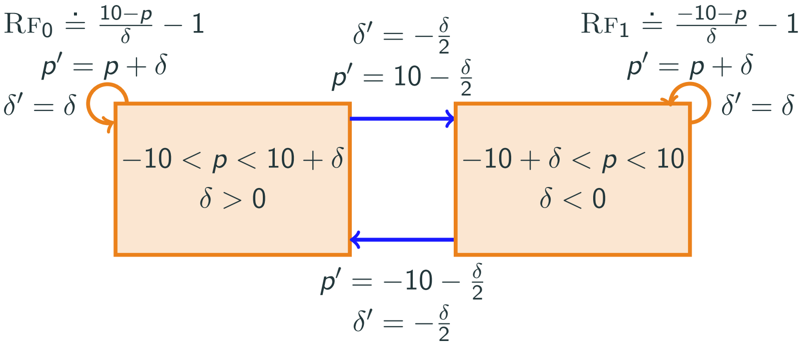


E-comp responsible for p



E-comp Example Composition

E-comp responsible for $\{p, \delta\}$



From E-comp to Funnel-loop and Back

Funnel-loop to E-comp

> Funnel-loop with $T_i(V, \widehat{V}')$ for $\widehat{V} \subseteq V$ corresponds to E-comp responsible for $\widehat{V}$.

E-comp to funnel-loop

> E-comp responsible for all symbols in V corresponds to a funnel-loop.

These results allow us to employ composition and segmentation together in the search procedure.

E-comp Synthesis

Strategy

Fragments of the system can be known problems.

Employ specific analysis on the fragment to generate E-comp.

Rely on automated procedure to find composition.

Examples

Lyapunov Stability.

Ultimately Diverging Symbols.

Ultimately Diverging Symbols

Inspired by region abstraction for timed automata.

Goal

Find region where truth value of model predicates is constant.

Define E-comp representing such regions.

Idea

Consider $x' \geq f(x)$;

let $\alpha : \frac{\partial^i (f-x)}{\partial x^i}(\alpha) \geq 0$ for all $i \in \mathbb{N}$, then $\forall x \geq \alpha : f(x) \geq x$;

$x \geq \alpha$ is closed with respect to $x' \geq f(x)$.

Ultimately Diverging Symbols

Inspired by region abstraction for timed automata.

Goal

Find region where truth value of model predicates is constant.

Define E-comp repres Absolute positiveness bound

Idea

Consider $x' \geq f(x)$;

let $\alpha : \frac{\partial^i (f - x)}{\partial x^i}(\alpha) \geq 0$ for all $i \in \mathbb{N}$, then $\forall x \geq \alpha : f(x) \geq x$;

$x \geq \alpha$ is closed with respect to $x' \geq f(x)$.

Inspired by region abstraction for timed automata.

Goal

Find region where truth value of model predicates is constant.

Idea

Many formulae over-approximate absolute positiveness bound for uni/multi-variate polynomials.

e.g. $\mathit{Cauchy\_bound}(x^n + \sum_{i=0}^{n-1} a_i x^i) \doteq 1 + \max_{0 \leq i < n}(|a_i|)$.

let $\alpha : \frac{\partial^i (f - x)}{\partial x^i}(\alpha) \geq 0$ for all $i \in \mathbb{N}$, then $\forall x \geq \alpha : f(x) \geq x$;

$x \geq \alpha$ is closed with respect to $x' \geq f(x)$.

Ultimately Diverging Symbols

Find absolute positiveness bound $bound_x$ based on the predicates in which x and x' appear.

Example

$$T \doteq \ \ldots\ x \geq 3 \wedge x' \leq x^3 - 5x^2 \wedge x' \geq x^2\ \ldots$$

$$\begin{aligned} bound_x &\doteq max(abspos_3,\ abspos_{x^3-5x^2},\ abspos_{x^2},\ abspos_{x^3-6x^2}) \\ &\leq\ max(3,\ 1+|-5|,\ 1+1,\ 1+|-6|)\ =\ 7 \end{aligned}$$

$$x^2 \leq x' \leq x^3 - 5x^2$$

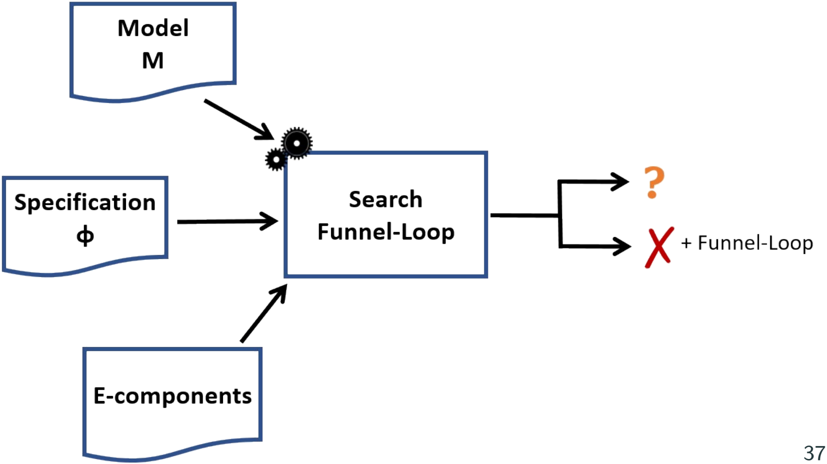
$$\boxed{x \geq 7}$$

Search Procedure

Funnel-loop search procedure

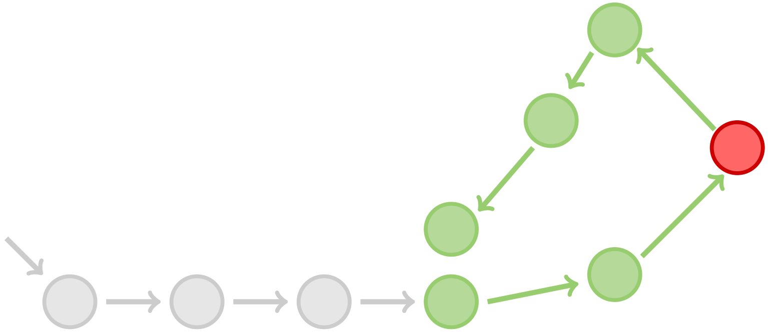
Via reduction to E-CHCs.

Ad-hoc implementation in Find-Fair-Funnel (F3).



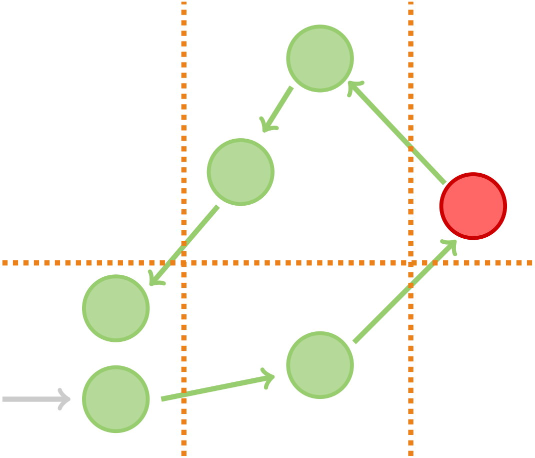
Find-Fair-Funnel Overview [1/3]

Enumerate paths that could correspond to an iteration over some funnel-loop.

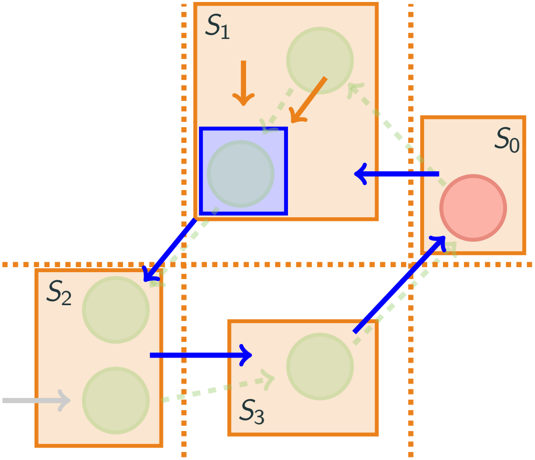


Find-Fair-Funnel Overview [2/3]

Consider sequence of states that correspond to a fair loop in some abstract space.



Define funnel-loop as strengthening of regions and transitions.



Experimental Evaluation

Benchmarks

LS 52 non-terminating linear software programs.

NS 30 non-terminating nonlinear software programs.

ITS 70 infinite-state transition systems;
scaling of 2 mutual exclusion protocol + other 12 instances.

TA 174 timed automata;
critical, *csma*, *fddi*, *fischer*, *lynch* and *train-gate* protocols.

TTS 120 timed transition systems;
Extended *csma*, *fischer*, *lynch* and *token-ring* protocols.

HS 9 hybrid systems;
adaptive cruise control, *train track permission controller*, *tank-pipe network*, 2 *synchronisation protocol on ethernet network*, 4 variations of *bouncing ball*.

Timed Automata Experimental Results

Spec	Result	Num	ATMOC	CTAV	DiVinE3	F3 (no hints)	LTSmin	nuXmv-BMC	nuXmv-IC3	Uppaal
INV	**True**	174	95_6	44	43	–	33[*a]	–	$\mathbf{144}_{38}$	82_1
INV	**False**	174	74	137	42	–	88	**142**	137	99
LTL	**True**	174	–	14	48	–	$\mathbf{61}_{16}$	–	39_{19}	6[*b]
LTL	**False**	174	151	148	71	140	**163**	156	90	116[*c]
MTL	**True**	174	–	13_1	–	–	–	–	$\mathbf{57}_{45}$	–
MTL	**False**	174	148	$\mathbf{152}_{26}$	–	–	–	147	121	–

Entries marked with "–" denote that the tool cannot handle the given benchmarks.

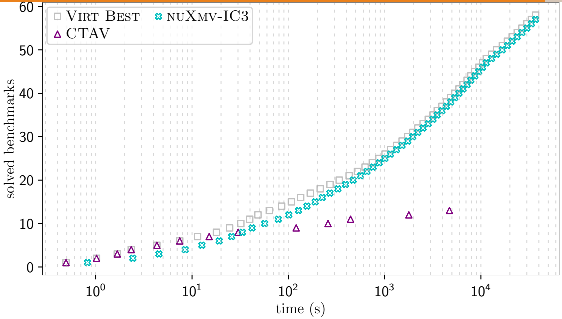
*a LTSmin supports only 145 true invariant specifications.

*b Uppaal supports only 29 true LTL specifications.

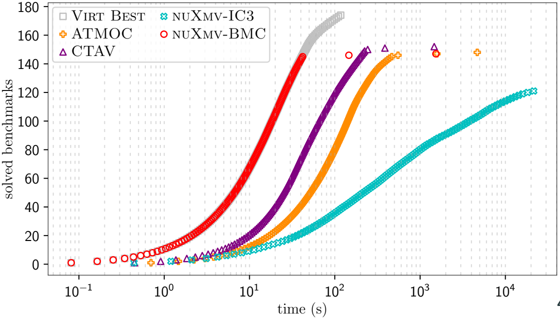
*c Uppaal supports only 116 false LTL specifications.

Timed Automata Experimental Results: MTL

True MTL:



False MTL:



Falsification Experimental Results

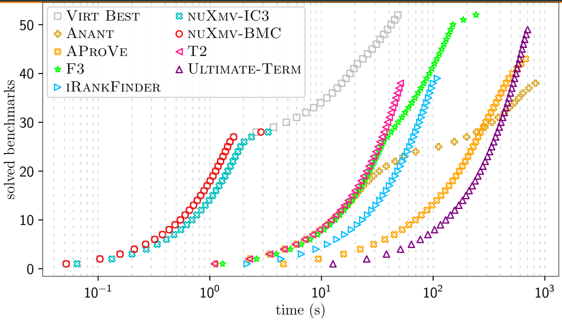
Model	Num	Anant	AProVe	ATMOC	CTAV	DiVinE3	F3 (no hints)	iRankFinder	LTSmin	nuXmv-BMC	nuXmv-IC3	T2	Ultimate	Uppaal
LS	52	38	43	–	–	–	**52**	39	–	28	28	38	49	–
NS	30	26	5	–	–	–	**30**$_2$	6	–	14	14	2	–	–
ITS	70	–	–	–	–	–	**65**$_{57}$	–	–	4	4	–	8	–
TA	174	–	–	151	148	71	140	–	**164**	155	89	–	–	116*a
TTS	120	–	–	–	–	–	**74**$_{72}$	–	–	29$_{27}$	10	–	–	–
HS	9	–	–	–	–	–	**3**$_3$	–	–	0	0	–	–	–
Total	455	64	48	151	148	71	**364**	45	164	230	145	40	56	116

Entries marked with "–" denote that the tool cannot handle the given benchmarks.

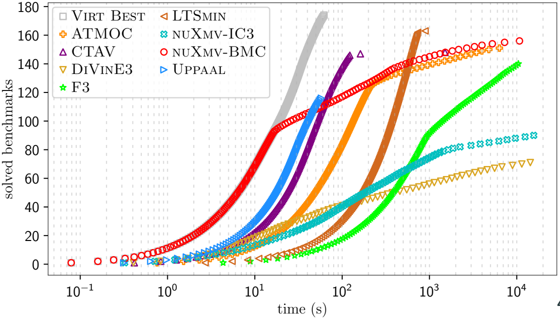
*a UPPAAL supports only 116 false LTL specifications.

Falsification Experimental Results: LS, TA

LS:



TA:



Conclusions and Future Work

Conclusions

Timed Systems

Support a broader class of verification problem than TA.

Experiments shows competitiveness on the common fragment.

Falsification

Effectiveness across very different contexts: highest number of solved instances.

Solves many instances no other tool successfully addressed.

Future Work

Timed Systems

Support dynamical systems "directly" (e.g. ODE).

Optimise encoding, in particular MTL to LTL rewriting.

Falsification

Beyond arithmetic theories, e.g. algebraic data-types.

Proof generation: constructive and non-constructive.

Apply to other verification contexts (e.g. CTL^*).

Heuristics to drive candidates and funnel-loop templates generation.

Integration with techniques to prove language is empty.

Contributions Highlights

Timed Systems

CAV 2019, **INFOCOMP 2020**.

Employ techniques developed in ITS context for HS and TTS verification.

Falsification technique tailored for TTS.

Falsification

VMCAI 2021, **ATVA 2021**, **INFOCOMP 2022**.

Identify infinite paths via segmentation and composition.

Employ specialised techniques to analyse different parts of the model.

Backup slides

Experimental evaluation

Tools

Model	Spec	Result	Num	Anant	AProVe	ATMOC	CTAV	DiVinE3	F3	iRankFinder	LTSmin	nuXmv-BMC	nuXmv-IC3	T2	Ultimate	Uppaal
LS	Term	False	52	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	✗
NS	Term	False	30	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓	✓	✗	✗
ITS	LTL	False	70	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✓	✗
TA	INV	False	174	✗	✗	✓	✓	✓	✗	✗	✓	✓	✓	✗	✗	✓
TA	INV	True	174	✗	✗	✓	✓	✓	✗	✗	✓*a	✗	✓	✗	✗	✓
TA	LTL	False	174	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	✓*b
TA	LTL	True	174	✗	✗	✗	✓	✓	✗	✗	✓	✗	✓	✗	✗	✓*c
TA	MTL	False	174	✗	✗	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗
TA	MTL	True	174	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
TTS	LTL	False	120	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗	✗
HS	LTL	False	9	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗	✗

*a LTSMIN does not handle the invariant specification of the *csma* protocol.

*b UPPAAL supports only the false LTL specifications of *fischer* and *lynch*.

*c UPPAAL supports only the true LTL specification of the *csma* protocol.

Benchmarks

LS 52 non-terminating linear software programs.

NS 30 non-terminating nonlinear software programs.

ITS 70 infinite-state transition systems;
scaling of 2 mutual exclusion protocol + other 12 instances.

TA 174 timed automata;
critical, *csma*, *fddi*, *fischer*, *lynch* and *train-gate* protocols.

TTS 120 timed transition systems;
Extended *csma*, *fischer*, *lynch* and *token-ring* protocols.

HS 9 hybrid systems;
adaptive cruise control, *train track permission controller*, *tank-pipe network*, 2 *synchronisation protocol on ethernet network*, 4 variations of *bouncing ball*.

Timed Automata Experimental Results

Spec	Result	Num	ATMOC	CTAV	DiVinE3	F3 (no hints)	LTSmin	nuXmv-BMC	nuXmv-IC3	Uppaal
INV	True	174	95_{6}	44	43	–	33*a	–	**144**$_{38}$	82_{1}
INV	False	174	74	137	42	–	88	**142**	137	99
LTL	True	174	–	14	48	–	**61**$_{16}$	–	39_{19}	6*b
LTL	False	174	151	148	71	140	**163**	156	90	116*c
MTL	True	174	–	13_{1}	–	–	–	–	**57**$_{45}$	–
MTL	False	174	148	**152**$_{26}$	–	–	–	147	121	–

Entries marked with "–" denote that the tool cannot handle the given benchmarks.

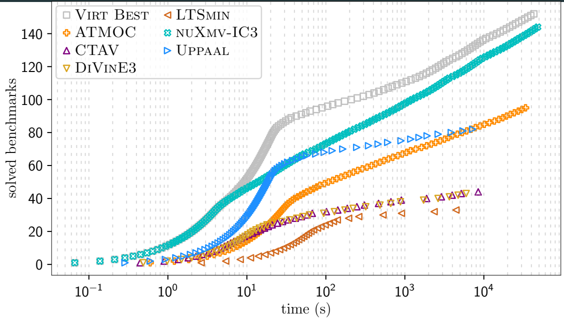
*a LTSMIN supports only 145 true invariant specifications.

*b UPPAAL supports only 29 true LTL specifications.

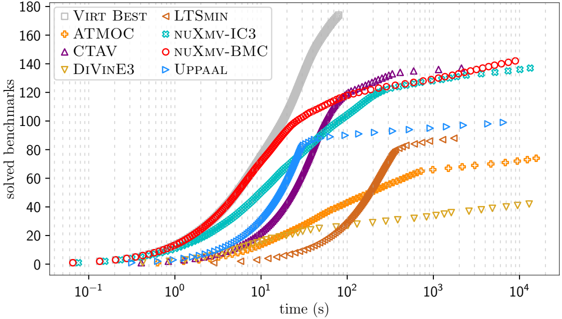
*c UPPAAL supports only 116 false LTL specifications.

Timed Automata Experimental Results: Invariants

True INV:

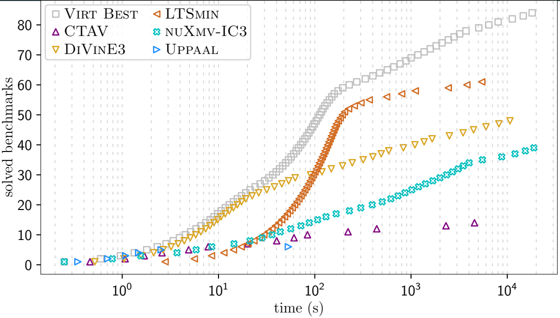


False INV:

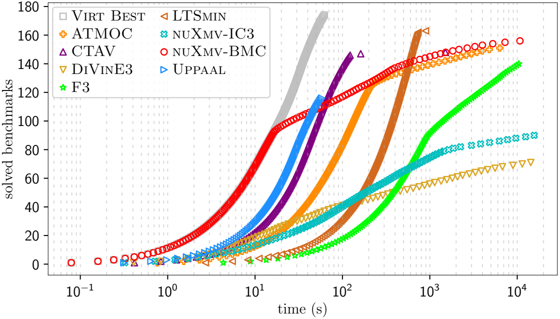


Timed Automata Experimental Results: LTL

True LTL:

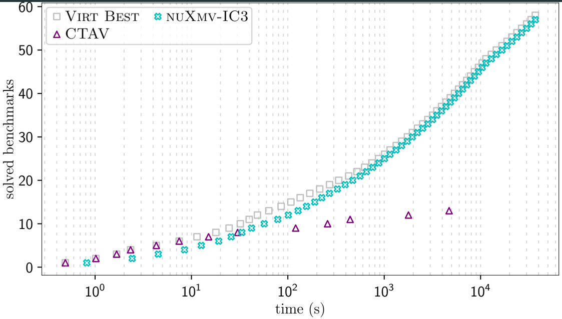


False LTL:

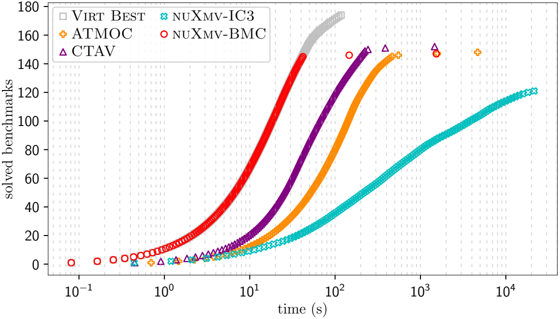


Timed Automata Experimental Results: MTL

True MTL:



False MTL:



Falsification Experimental Results

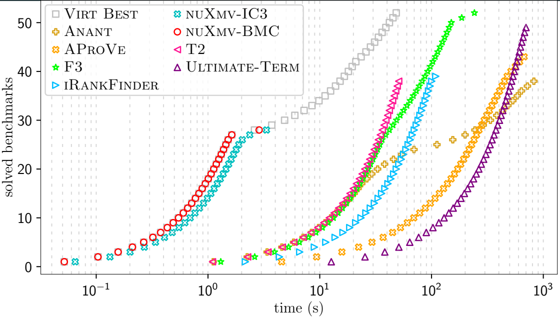
Model	Num	Anant	AProVe	ATMOC	CTAV	DiVinE3	F3 (no hints)	iRankFinder	LTSmin	nuXmv-BMC	nuXmv-IC3	T2	Ultimate	Uppaal
LS	52	38	43	–	–	–	**52**	39	–	28	28	38	49	–
NS	30	26	5	–	–	–	$\mathbf{30}_{2}$	6	–	14	14	2	–	–
ITS	70	–	–	–	–	–	$\mathbf{65}_{57}$	–	–	4	4	–	8	–
TA	174	–	–	151	148	71	140	–	**164**	155	89	–	–	116[*a]
TTS	120	–	–	–	–	–	$\mathbf{74}_{72}$	–	–	29_{27}	10	–	–	–
HS	9	–	–	–	–	–	$\mathbf{3}_{3}$	–	–	0	0	–	–	–
Total	455	64	48	151	148	71	**364**	45	164	230	145	40	56	116

Entries marked with "–" denote that the tool cannot handle the given benchmarks.

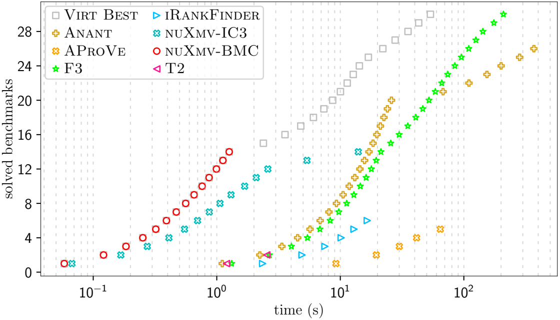
*a UPPAAL supports only 116 false LTL specifications.

Falsification Experimental Results: Software Non-Termination

LS:

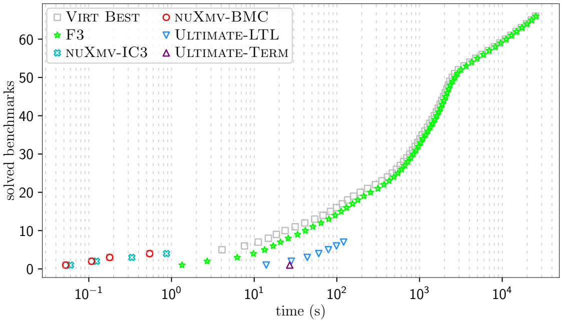


NS:



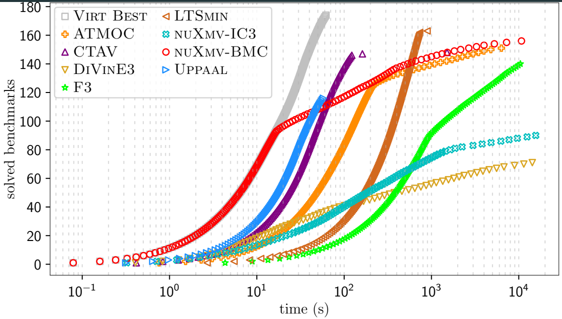
Falsification Experimental Results: ITS

ITS:

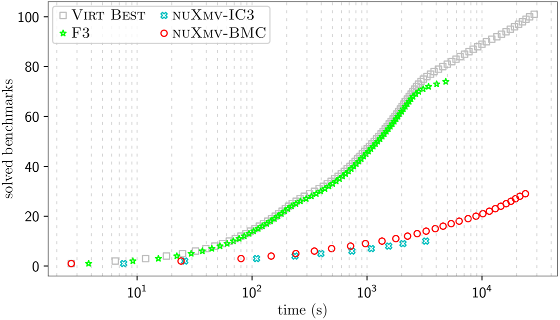


Falsification Experimental Results: TA, TTS

TA:



TTS:



LTL-EF Syntax and Semantics

LTL-EF Syntax

$$\begin{aligned} \varphi ::\; & p(u,\ldots,u) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi \rightarrow \varphi \mid \varphi \leftrightarrow \varphi \mid \\ & \boldsymbol{X}\varphi \mid \boldsymbol{G}\varphi \mid \boldsymbol{F}\varphi \mid \varphi\boldsymbol{U}\varphi \mid \boldsymbol{Y}\varphi \mid \boldsymbol{H}\varphi \mid \boldsymbol{P}\varphi \mid \varphi\boldsymbol{S}\varphi \mid \tilde{\boldsymbol{X}}\varphi \mid \tilde{\boldsymbol{Y}}\varphi \\ u ::\; & c \mid x \mid f(u,\ldots,u) \mid u\boldsymbol{@}\tilde{\boldsymbol{F}}(\varphi) \mid u\boldsymbol{@}\tilde{\boldsymbol{P}}(\varphi) \mid \boldsymbol{Ite}(\varphi,u,u) \end{aligned}$$

where:

- x is a variable in V,
- p is a predicate,
- f is a function,
- c is a constant symbol in Σ.

LTL-EF Semantics: Ordered Time Points

For three time points $t, t', t'' \in \tau$,

$$ordered_\tau(t, t'', t') \doteq \begin{cases} 0 \leq t < t'' < t' & \text{if } \tau \in \{discrete, dense\}; \\ \exists i, r, i', r', r'' : 0 \leq i \leq i' \wedge 0 \leq r < r'' \leq r' \wedge \\ \quad t = \langle i, r \rangle \wedge t' = \langle i', r' \rangle \wedge t'' = \langle i, r'' \rangle & \text{otherwise.} \end{cases}$$

The formula $\forall t' > t \ \exists t'' : ordered_\tau(t, t'', t')$, requires the existence of a intermediate time point t'' for any successor t' of t.

LTL-EF Semantics

$\sigma, t \models \tilde{\boldsymbol{X}}\varphi$ iff $\forall t' \in \tau : t' > t$, $\exists t'' \in \tau$ such that $\mathit{ordered}_\tau(t, t'', t')$ holds and $\sigma, t'' \models \varphi$;

$\sigma, t \models \tilde{\boldsymbol{Y}}\varphi$ iff $t > 0$ and $\forall t' \in \tau : t' < t$, $\exists t'' \in \tau$ such that $\mathit{ordered}_\tau(t', t'', t)$ holds and $\sigma, t'' \models \varphi$;

$$\sigma(t)(u@\tilde{\boldsymbol{F}}(\varphi)) = \begin{cases} \sigma(t)(u) & \text{if } \sigma, t \models \tilde{\boldsymbol{X}}\varphi; \\ \sigma(t')(u) & \text{if } \exists t' > t : \sigma, t' \models \varphi \vee \tilde{\boldsymbol{X}}\varphi \text{ and} \\ & \forall t'' \in \tau : t < t'' < t' \rightarrow \sigma, t'' \not\models \varphi; \\ \sigma(t)(\mathit{def}_{u@\tilde{\boldsymbol{F}}(\varphi)}) & \text{otherwise;} \end{cases}$$

$$\sigma(t)(u@\tilde{\boldsymbol{P}}(\varphi)) = \begin{cases} \sigma(t)(u) & \text{if } \sigma, t \models \tilde{\boldsymbol{Y}}\varphi; \\ \sigma(t')(u) & \text{if } \exists t' < t : \sigma, t' \models \varphi \vee \tilde{\boldsymbol{Y}}\varphi \text{ and} \\ & \forall t'' \in \tau : t' < t'' < t \rightarrow \sigma, t'' \not\models \varphi; \\ \sigma(t)(\mathit{def}_{u@\tilde{\boldsymbol{P}}(\varphi)}) & \text{otherwise;} \end{cases}$$

where $\mathit{def}_{u@\tilde{\boldsymbol{F}}(\varphi)}$ and $\mathit{def}_{u@\tilde{\boldsymbol{P}}(\varphi)}$ are fresh variables.

MTL Syntax and Semantics

MTL Syntax

$$
\begin{aligned}
\varphi ::&\ p(u,\dots,u) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi \rightarrow \varphi \mid \varphi \leftrightarrow \varphi \mid \\
&\ \boldsymbol{X}\varphi \mid \tilde{\boldsymbol{X}}\varphi \mid \boldsymbol{Y}\varphi \mid \tilde{\boldsymbol{Y}}\varphi \mid \\
&\ \boldsymbol{G}_I\varphi \mid \boldsymbol{F}_I\varphi \mid \varphi\boldsymbol{U}_I\varphi \mid \boldsymbol{H}_I\varphi \mid \boldsymbol{P}_I\varphi \mid \varphi\boldsymbol{S}_I\varphi \mid \overrightarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \mid \overleftarrow{\boldsymbol{C}}^{k}_{[0,cu)}\varphi \\
u ::&\ c \mid x \mid f(u,\dots,u) \mid u@\tilde{\boldsymbol{F}}(\varphi) \mid u@\tilde{\boldsymbol{P}}(\varphi) \mid \boldsymbol{Ite}(\varphi,u,u) \\
I ::&\ [cu,cu] \mid [cu,cu) \mid (cu,cu] \mid (cu,cu) \mid [cu,\infty) \mid (cu,\infty) \\
cu ::&\ c \mid f(cu,\dots,cu)
\end{aligned}
$$

where:

- x is a variable in V,
- p is a predicate symbol,
- f is a function symbol,
- c is a constant in Σ.

MTL Semantics

$\sigma, t \models \varphi_0 \boldsymbol{U}_I \varphi_1$ iff $\exists t' \in \tau : t' \geq t \wedge v(t') - v(t) \in M(I)$ such that $\sigma, t' \models \varphi_1$ and $\forall t'' \in \tau : t \leq t'' < t'$ then $\sigma, t'' \models \varphi_0$;

$\sigma, t \models \varphi_0 \boldsymbol{S}_I \varphi_1$ iff $\exists t' \in \tau : t' \leq t \wedge v(t) - v(t') \in M(I)$ such that $\sigma, t' \models \varphi_1$ and $\forall t'' \in \tau : t' < t'' \leq t$ then $\sigma, t'' \models \varphi_0$;

$\sigma, t \models \overrightarrow{\boldsymbol{C}}^k_{<cu} \varphi$ iff $\exists t_1, \ldots t_k \in \tau : t < t_1 < \ldots < t_k \wedge v(t_k) - v(t) < M(cu)$ such that $\forall i \in \{j\}_{j=1}^k : \sigma, t_i \models \varphi$;

$\sigma, t \models \overleftarrow{\boldsymbol{C}}^k_{<cu} \varphi$ iff $\exists t_1, \ldots t_k \in \tau : 0 \leq t_k < \ldots < t_1 < t \wedge v(t) - v(t_k) < M(cu)$ such that $\forall i \in \{j\}_{j=1}^k : \sigma, t_i \models \varphi$.

Reduction to discrete LTL model checking

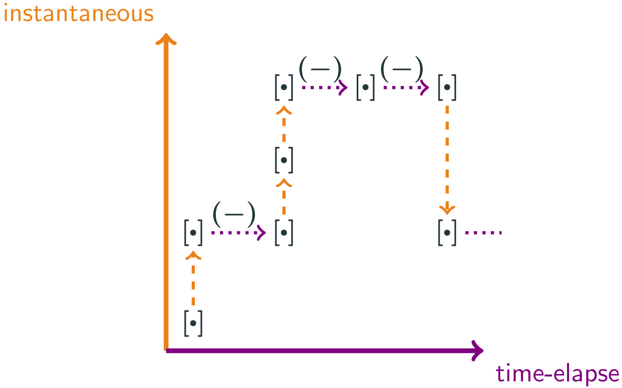
Reduction steps: timed to “untimed” model

Timed to “untimed” TS

clocks symbols reduced to *real* / *integer* symbols.

δ symbol prescribes amount of time elapse for every transition.

ι: alternation of singular $[\bullet]$ and open $(-)$ time intervals.



Reduction steps: $\mathbf{MTL}_{0,\infty}$ to discrete time LTL

Properties rewriting

$\mathbf{MTL}_{0,\infty}$

$$\boxed{F_{[0,5]}\ p}$$

$\downarrow$ rewrite

XLTL-EF

$$((\neg p U p) \land \mathit{time\_until}(p) \leq 5) \lor$$
$$((\neg p U \tilde{X} p) \land \mathit{time\_until}(p) < 5)$$

$\downarrow$ to discrete time

LTL-EF

$$((\neg p U p) \land (\mathit{time}@\tilde{F} p) - \mathit{time} \leq 5) \lor$$
$$((\neg p U((\neg\iota \land p) \lor X(\neg\iota \land p))) \land (\mathit{time}@\tilde{F} p) - \mathit{time} < 5)$$

$\downarrow$ remove event-freezing

LTL

$\mathbf{MTL}_{0,\infty}$ to XLTL-EF Reduction

Remove bounds on temporal operators [1/3]

$$\overrightarrow{C}^{k}_{[0,p)}(\varphi) \equiv_G time@\tilde{F}^{k}(\varphi) - time < p \wedge \tilde{F}^{k}(\varphi);$$

$$\overleftarrow{C}^{k}_{[0,p)}(\varphi) \equiv_G time - time@\tilde{P}^{k}(\varphi) < p \wedge \tilde{P}^{k}(\varphi);$$

$$\varphi_1 U_{(0,p)} \varphi_2 \equiv_G \varphi_1 U(\tilde{X} \varphi_1 U \varphi_2 \wedge time@\tilde{F}(\varphi_2) - time < p);$$

$$\varphi_1 S_{(0,p)} \varphi_2 \equiv_G \varphi_1 S(\tilde{Y} \varphi_1 S \varphi_2 \wedge time - time@\tilde{P}(\varphi_2) < p);$$

$$\varphi_1 U_{[0,p)} \varphi_2 \equiv_G \varphi_1 U \varphi_2 \wedge time@F(\varphi_2) - time < p;$$

$$\varphi_1 S_{[0,p)} \varphi_2 \equiv_G \varphi_1 S \varphi_2 \wedge time - time@P(\varphi_2) < p;$$

Remove bounds on temporal operators [2/3]

$$
\begin{aligned}
\varphi_1 \boldsymbol{U}_{(0,p]} \varphi_2 \;\equiv_G\; & \varphi_1 \boldsymbol{U}((\tilde{\boldsymbol{X}} \varphi_1 \boldsymbol{U} \varphi_2) \wedge \\
& (((\tilde{\boldsymbol{X}} \neg \varphi_2 \boldsymbol{U} \varphi_2) \wedge (time@\tilde{\boldsymbol{F}}(\varphi_2) - time \leq p)) \vee \\
& ((\tilde{\boldsymbol{X}} \neg \varphi_2 \boldsymbol{U} \tilde{\boldsymbol{X}} \varphi_2) \wedge (time@\tilde{\boldsymbol{F}}(\varphi_2) - time < p)))); \\
\varphi_1 \boldsymbol{S}_{(0,p]} \varphi_2 \;\equiv_G\; & \varphi_1 \boldsymbol{S}((\tilde{\boldsymbol{Y}} \varphi_1 \boldsymbol{S} \varphi_2) \wedge \\
& (((\tilde{\boldsymbol{Y}} \neg \varphi_2 \boldsymbol{S} \varphi_2) \wedge (time - time@\tilde{\boldsymbol{P}}(\varphi_2) \leq p)) \vee \\
& ((\tilde{\boldsymbol{Y}} \neg \varphi_2 \boldsymbol{S} \tilde{\boldsymbol{Y}} \varphi_2) \wedge (time - time@\tilde{\boldsymbol{P}}(\varphi_2) < p))));
\end{aligned}
$$

Remove bounds on temporal operators [3/3]

$$
\begin{aligned}
\varphi_1 \boldsymbol{U}_{[0,p]} \varphi_2 \equiv_G\ & \varphi_1 \boldsymbol{U} \varphi_2 \wedge \\
& (\neg\varphi_2 \boldsymbol{U} \varphi_2 \wedge \mathit{time}@\boldsymbol{F}(\varphi_2) - \mathit{time} \leq p \vee \\
& \neg\varphi_2 \boldsymbol{U} \tilde{\boldsymbol{X}} \varphi_2 \wedge \mathit{time}@\boldsymbol{F}(\varphi_2) - \mathit{time} < p); \\
\varphi_1 \boldsymbol{S}_{[0,p]} \varphi_2 \equiv_G\ & \varphi_1 \boldsymbol{S} \varphi_2 \wedge \\
& (\neg\varphi_2 \boldsymbol{S} \varphi_2 \wedge \mathit{time} - \mathit{time}@\boldsymbol{P}(\varphi_2) \leq p \vee \\
& \neg\varphi_2 \boldsymbol{S} \tilde{\boldsymbol{Y}} \varphi_2 \wedge \mathit{time} - \mathit{time}@\boldsymbol{P}(\varphi_2) < p).
\end{aligned}
$$

Discretization

$$
\begin{aligned}
\mathcal{D}(p(u_0, \ldots, u_n)) &\doteq p(u_0, \ldots, u_n);\\
\mathcal{D}(\neg\varphi) &\doteq \neg\mathcal{D}(\varphi);\\
\mathcal{D}(\varphi_1 \wedge \varphi_2) &\doteq \mathcal{D}(\varphi_1) \wedge \mathcal{D}(\varphi_2);\\
\mathcal{D}(\varphi_1 \boldsymbol{U} \varphi_2) &\doteq \mathcal{D}(\varphi_2) \vee \mathcal{D}(\varphi_1)\boldsymbol{U}(\mathcal{D}(\varphi_2) \wedge (\mathcal{D}(\varphi_1) \vee \mathit{closed}));\\
\mathcal{D}(\varphi_1 \boldsymbol{S} \varphi_2) &\doteq \mathcal{D}(\varphi_2) \vee \mathcal{D}(\varphi_1)\boldsymbol{S}(\mathcal{D}(\varphi_2) \wedge (\mathcal{D}(\varphi_1) \vee \mathit{closed}));\\
\mathcal{D}(\boldsymbol{X}\varphi) &\doteq \mathit{closed} \wedge \boldsymbol{X}(\neg \mathit{open} \wedge \mathcal{D}(\varphi));\\
\mathcal{D}(\tilde{\boldsymbol{X}}\varphi) &\doteq (\mathit{open} \wedge \mathcal{D}(\varphi)) \vee \boldsymbol{X}(\mathit{open} \wedge \mathcal{D}(\varphi));\\
\mathcal{D}(\boldsymbol{Y}\varphi) &\doteq \mathit{closed} \wedge \boldsymbol{Y}(\neg \mathit{open} \wedge \mathcal{D}(\varphi));\\
\mathcal{D}(\tilde{\boldsymbol{Y}}\varphi) &\doteq (\mathit{open} \wedge \mathcal{D}(\varphi)) \vee \boldsymbol{Y}(\mathit{open} \wedge \mathcal{D}(\varphi)).
\end{aligned}
$$

Funnels and Funnel-loops

Funnel Definition

$$fnl \doteq \langle V, S(V), T(V, V'), D(V), \mathrm{R\small{F}}(V) \rangle$$

V is a set of symbols,

$\mathrm{R\small{F}}$ is a ranking function with minimal element 0,

S, D and T are formulae representing respectively the source region, destination region and transition relation.

F.1 $\forall V \exists V' : S \rightarrow T$;

F.2 $\forall V, V' : (S \wedge 0 < \mathrm{R\small{F}} \wedge T) \rightarrow S'$;

F.3 $\forall V, V' : (S \wedge 0 < \mathrm{R\small{F}} \wedge T) \rightarrow \mathrm{R\small{F}}' < \mathrm{R\small{F}}$;

F.4 $\forall V, V' : (S \wedge \mathrm{R\small{F}} = 0 \wedge T) \rightarrow D'$.

Funnel-loops Definition

$$floop \doteq [fnl_i]_{i=0}^{n-1}$$

for $n \geq 1$ over symbols V such that:

FL.1 $\forall i \in \{h\}_{h=0}^{n-1}, V : D_i \rightarrow S_{i+_n 1}$.

Note: does not ensure fairness of the paths.

Funnel-loop Sufficient Conditions

Let $M \doteq \langle V, I^M, T^M, F^M \rangle$ be a fair transition system. Let *floop* be a funnel-loop of length n over the symbols V and funnels $[fnl_i]_{i=0}^{n-1}$ that satisfy the following hypotheses.

FF.1 $M \rightsquigarrow \bigvee_{i=0}^{n-1} S_i$,

FF.2 $\forall V \ : \ D_{n-1} \rightarrow F^M$,

FF.3 $\forall V, V' \ : \ S_i \wedge T_i \rightarrow T^M$, for every $0 \leq i < n$.

Then M admits at least one fair path.

Funnel-loop Search as E-CHC

$$
\begin{aligned}
\top \;&\rightarrow\; \exists c, V : R(c, V) \wedge I^M(V) && (1)\\
T(V, V') \;&\rightarrow\; \exists c : R(c, V') && (2)\\
R(c, V) \wedge T(V, V') \;&\rightarrow\; T^M(V, V') && (3)\\
R(c, V) \;&\rightarrow\; \exists V' : T(V, V') && (4)\\
c \wedge R(c, V) \;&\rightarrow\; F^M(V) && (5)\\
\neg c \wedge R(c, V) \wedge T(V, V') \;&\rightarrow\; \mathit{Rank}(V, V') && (6)\\
\mathit{wf}(\mathit{Rank}) & && (7)
\end{aligned}
$$

Let $\mathit{Cex} \doteq \langle V, \exists c : R(c, V), T(V, V'), \top \rangle$

E-comps

E-comp Definition [1/2]

Given a set of symbols V such that $\{V^i\}_{i=0}^n$ is a partitioning of V for some $n \in \mathbb{N}$. An E-comp H^i of length $m^i \in \mathbb{N}$ and responsible for V^i is a transition system $\langle V, I^i(V), T^i(V, V') \rangle$ associated with:

1. a set of regions $\mathcal{R}^i \doteq \{R^i_j(V) \mid 0 \leq j < m^i\}$;
2. a set of assumptions $\mathcal{A}^i \doteq \{A^i_j(V^{\neq i}) \mid 0 \leq j < m^i\}$, where $V^{\neq i} \doteq \bigcup_{0 \leq k < n, k \neq i} V^k$ and $A^i_j(V^{\neq i}) \doteq \bigwedge_{0 \leq k < n, k \neq i} A^{i,k}_j(V^k)$;
3. a set of functions $\mathcal{W}^i \doteq \{\mathrm{RF}^i_j(V) \mid 0 \leq j < m^i\}$ such that each RF^i_j is a ranking function with respect to a well-founded relation $<^i_j$ and minimal element 0^i_j;

E-comp Definition [2/2]

EC.1 $H^i \models \bigvee_{j=0}^{m^i-1} R^i_j \wedge A^i_j$;

EC.2 $\forall j : 0 \leq j < m^i \rightarrow$

$$\exists V, V' : rankedT^i_j(V, V') \quad \models \quad \forall V \exists V^{i\prime} \forall V^{\neq i\prime} :$$
$$R^i_j \wedge A^i_j \wedge 0^i_j <^i_j \mathrm{RF}^i_j \wedge A^{i\prime}_j \rightarrow R^{i\prime}_j \wedge T^i \wedge \mathrm{RF}^{i\prime}_j <^i_j \mathrm{RF}^i_j;$$

EC.3 $\forall j : 0 \leq j < m^i \rightarrow$

$$\exists V, V' : stutterT^i_j(V, V') \quad \models \quad \forall V \exists V^{i\prime} \forall V^{\neq i\prime} :$$
$$R^i_j \wedge A^i_j \wedge A^{i\prime}_j \rightarrow R^{i\prime}_j \wedge T^i \wedge \mathrm{RF}^{i\prime}_j = \mathrm{RF}^i_j;$$

EC.4 $\forall j, j' : 0 \leq j < m^i \wedge 0 \leq j' < m^i \rightarrow$

$$\exists V, V' : progressT^i_{j,j'}(V, V') \quad \models \quad \forall V \exists V^{i\prime} \forall V^{\neq i\prime} :$$
$$R^i_j \wedge A^i_j \wedge \mathrm{RF}^i_j = 0^i_j \wedge {A^i_{j'}}' \rightarrow {R^i_{j'}}' \wedge T.$$

E-comp Projection

Given an E-comp $H \doteq \langle V, I, T \rangle$ over m regions $\mathcal{R}$, assumptions $\mathcal{A}$ and ranking functions $\mathcal{W}$, we define its projection to a sequence of k indexes $idxs \doteq \langle j_0^{\downarrow}, \ldots, j_{k-1}^{\downarrow} \rangle \subseteq \{j\}_{j=0}^{m-1}$ as the E-comp $H^{\downarrow} \doteq \langle V, I^{\downarrow}, T^{\downarrow} \rangle$ associated with regions $\mathcal{R}^{\downarrow}$, assumptions $\mathcal{A}^{\downarrow}$ and ranking functions $\mathcal{W}^{\downarrow}$ such that:

1. $I^{\downarrow} \doteq I \wedge \bigvee_{j \in idxs} (R_j \wedge A_j)$;
2. $T^{\downarrow} \doteq T \wedge \bigwedge_{h=0}^{k-1} R_{j_h^{\downarrow}} \rightarrow ((R'_{j_h^{\downarrow}} \wedge \mathrm{RF}'_{j_h^{\downarrow}} < \mathrm{RF}_{j_h^{\downarrow}}) \vee (\mathrm{RF}_{j_h^{\downarrow}} = 0 \wedge R'_{j_{h+_k 1}^{\downarrow}}))$;
3. $\mathcal{R}^{\downarrow} \doteq \{R_j \mid j \in idxs \wedge R_j \in \mathcal{R}\}$;
4. $\mathcal{A}^{\downarrow} \doteq \{A_j \mid j \in idxs \wedge A_j \in \mathcal{A}\}$;
5. $\mathcal{W}^{\downarrow} \doteq \{\mathrm{RF}_j \mid j \in idxs \wedge \mathrm{RF}_j \in \mathcal{W}\}$.

E-comp Compatible Transitions

$$compatible_{\{H^i\}_{i=0}^n}(\widehat{V},\widehat{V}') \doteq \forall V, V' : \bigwedge_{\underbrace{0 \leq j_0 < m^0, 0 \leq j_0' < m^0, \ldots, 0 \leq j_n < m^n, 0 \leq j_n' < m^n}_{\text{all possible pair of indexes for the } E\text{-comps } \{H^i\}_{i=0}^n}}$$

$$(\bigwedge_{i=0}^{n} \underbrace{R^i_{j_i}(\widehat{V}) \wedge A^i_{j_i}(\widehat{V}^{\neq i}) \wedge R^i_{j_i'}(\widehat{V}') \wedge A^i_{j_i'}(\widehat{V}^{\neq i\prime})}_{j_i, j_i' \text{ containing both } \widehat{V} \text{ and } \widehat{V}'} \wedge$$

$$\underbrace{R^i_{j_i}(V) \wedge A^i_{j_i}(V^{\neq i}) \wedge R^i_{j_i'}(V') \wedge A^i_{j_i'}(V^{\neq \{h\}_{h=0}^n \prime}) \wedge T^i(V, V')}_{\substack{\text{for all } V \text{ in } j_i,\ V' \text{ in } j_i' \text{ such that } V, V' \models T^i \text{ and } V' \text{ meets all} \\ \text{assumptions of } H^i \text{ at } j_i' \text{ on symbols of } E\text{-comps not in } \{H^i\}_{i=0}^n}} \wedge$$

$$\underbrace{(\mathrm{RF}^i_{j_i'}(\widehat{V}') < \mathrm{RF}^i_{j_i}(\widehat{V}) \leftrightarrow \mathrm{RF}^i_{j_i'}(V') < \mathrm{RF}^i_{j_i}(V))}_{\text{transition } V, V' \text{ of the same type of transition} \widehat{V}, \widehat{V}'} \wedge$$

$$(0 < \mathrm{RF}^i_{j_i}(\widehat{V}) \leftrightarrow 0 < \mathrm{RF}^i_{j_i}(V)) \wedge (0 < \mathrm{RF}^i_{j_i'}(\widehat{V}') \leftrightarrow 0 < \mathrm{RF}^i_{j_i'}(V'))) \rightarrow$$

$$\bigwedge_{i=0}^{n} \underbrace{\bigwedge_{h=0, h\neq i}^{n} A^{i,h}_{j_i'}(V^{h\prime})}_{\text{all assumptions of } H^i \text{ on the } \{V^h\}_{h=0}^n \text{ are met}}.$$

E-comp Independent Ranks

$$
\begin{aligned}
&\mathit{indepRank}_{\{H^i\}_{i=0}^n}(\widehat{V},\widehat{V}') \doteq \underbrace{\bigwedge_{0\le j_0<m^0,\ldots,0\le j_n<m^n}}_{\text{all possible indexes for the } E\text{-comps } \{H^i\}_{i=0}^n} \\
&\Big(\Big(\underbrace{(\sum_{i=0}^n \mathrm{RF}^i_{j_i})(\widehat{V}') < (\sum_{i=0}^n \mathrm{RF}^i_{j_i})(\widehat{V})}_{\text{some ranking function decreases, all others remain constant}} \wedge \\
&\underbrace{\bigwedge_{i=0}^n R^i_{j_i}(\widehat{V}) \wedge A^i_{j_i}(\widehat{V}^{\neq i}) \wedge R^i_{j_i}(\widehat{V}') \wedge A^i_{j_i}(\widehat{V}^{\neq i\prime})}_{\widehat{V},\widehat{V}' \text{are in restricted regions } j_i, j_i'}\Big) \rightarrow \\
&\bigwedge_{i=0}^n \underbrace{(\forall V : (\bigwedge_{h=0}^n R^h_{j_h}(V) \wedge A^h_{j_h}(V^{\neq h})) \rightarrow \mathrm{RF}^i_{j_i}(V) = 0)}_{\text{current ranking function } \mathrm{RF}^i_{j_i} \text{ is always } 0} \vee \\
&\exists V, V' : \underbrace{(\bigwedge_{h=0}^n R^h_{j_h}(V) \wedge A^h_{j_h}(V^{\neq h}) \wedge T^h(V,V') \wedge R^h_{j_h}(V') \wedge A^h_{j_h}(V^{\neq h\prime}))}_{V,V' \text{ in same restricted regions of } \widehat{V},\widehat{V}'} \wedge \\
&\underbrace{\mathrm{RF}^i_{j_i}(V') < \mathrm{RF}^i_{j_i}(V) \wedge (\bigwedge_{h=0,h\neq i}^n \mathrm{RF}^h_{j_h}(V') = \mathrm{RF}^h_{j_h}(V))}_{\text{current ranking function decreases, all others remain constant}} \wedge \mathit{compatible}_{\{H^k\}_{k=0}^n}(V,V')\Big)
\end{aligned}
$$

E-comp Composition

$H^c \doteq \bigotimes_{i=0}^n H^i = \langle V, I^c, T^c \rangle$, for pairwise disjoint $\{V^i\}_{i=0}^n$.
H^c responsible for $V^c \doteq \bigcup_{i=0}^n V^i$, associated with regions $\mathcal{R}^c$, assumptions $\mathcal{A}^c$ and ranking functions $\mathcal{W}^c$; where:

1. $V^c \doteq \bigcup_{i=0}^n V^i$;
2. $\mathcal{R}^c \doteq \{\bigwedge_{i=0}^n R^i_{j_i} \wedge \bigwedge_{h=0,h\neq i}^n A^{i,h}_{j_i} \mid j_i \in \{k\}_{k=0}^{m^i-1} \wedge R^i_{j_i} \in \mathcal{R}^i \wedge A^i_{j_i} \in \mathcal{A}^i \wedge A^{i,h}_{j_i} \in A^i_{j_i}\}$;
3. $\mathcal{A}^c \doteq \{\bigwedge_{i=0}^n \bigwedge_{h\notin\{k\}_{k=0}^n} A^{i,h}_{j_i} \mid j_i \in \{k\}_{k=0}^{m^i-1} \wedge A^i_{j_i} \in \mathcal{A}^i \wedge A^{i,h}_{j_i} \in A^i_{j_i}\}$;
4. $\mathcal{W}^c \doteq \{\sum_{i=0}^n \mathrm{RF}^i_{j_i} \mid j_i \in \{k\}_{k=0}^{m^i-1} \wedge \mathrm{RF}^i_{j_i} \in \mathcal{W}^i\}$;
5. $I^c \doteq \bigwedge_{i=0}^n I^i$;
6. $T^c \doteq \mathit{compatible}_{\{H^i\}_{i=0}^n} \wedge \mathit{indepRank}_{\{H^i\}_{i=0}^n} \wedge \bigwedge_{i=0}^n T^i$.

F3

Funnel-loop search: overall procedure

SEARCH-FUNNEL-LOOP(M, $\mathcal{H}$)

```
▷ Iterate over candidate loops of increasing length.
for all ⟨prefix, loop_r, loop_t, H⟩ ∈ GENERATE-CANDIDATE-LOOPS(M, 𝓗) do
    v0 ← prefix[len(prefix) − 1]                    ▷ Witness for reachability.
    ▷ Iterate over funnel-loop templates for current candidate loop.
    for all template ∈ GENERATE-TEMPLATES(v0, loop_r, loop_t, H) do
        ef_constrs ← template.ef_constraints()              ▷ Get ∃∀ problem.
        ⟨found, model⟩ ← SEACH-PARAMETER-ASSIGNMENT(ef_constrs)
        if found == ⊤ then              ▷ Replace parameters with assignment.
            fnl_loop ← template.instantiate(model)
            return ⟨prefix, fnl_loop⟩     ▷ Reachability witness and funnel-loop.
        end if
    end for
end for
return unknown
```

Funnel-loop search: overall procedure

SEARCH-FUNNEL-LOOP(M, $\mathcal{H}$)

Given a fair transition system M and set of E-comps $\mathcal{H}$.

▷ Iterate over candidate loops of in

for all $\langle \mathit{prefix}, \mathit{loop\_r}, \mathit{loop\_t}, H \rangle \in$ GENERATE-CANDIDATE-LOOPS$(M, \mathcal{H})$ **do**

  $v_0 \leftarrow \mathit{prefix}[\mathbf{len}(\mathit{prefix}) - 1]$ ▷ Witness for reachability.

  ▷ Iterate over funnel-loop templates for current candidate loop.

  for all $\mathit{template} \in$ GENERATE-TEMPLATES$(v_0, \mathit{loop_r}, \mathit{loop_t}, H)$ **do**

    $\mathit{ef\_constrs} \leftarrow \mathit{template}.\mathit{ef\_constraints}()$ ▷ Get $\exists\forall$ problem.

    $\langle \mathit{found}, \mathit{model} \rangle \leftarrow$ SEACH-PARAMETER-ASSIGNMENT$(\mathit{ef_constrs})$

    if $\mathit{found} == \top$ **then** ▷ Replace parameters with assignment.

      $\mathit{fnl\_loop} \leftarrow \mathit{template}.\mathit{instantiate}(\mathit{model})$

      return $\langle \mathit{prefix}, \mathit{fnl\_loop} \rangle$ ▷ Reachability witness and funnel-loop.

    end if

  end for

end for

return *unknown*

Funnel-loop search: overall procedure

SEARCH-FUNNEL-LOOP(M, $\mathcal{H}$)

▷ Iterate over candidate loops of increasing length.
for all $\langle \textit{prefix}, \textit{loop\_r}, \textit{loop\_t}, H \rangle \in$ GENERATE-CANDIDATE-LOOPS($M, \mathcal{H}$) **do**
  $v_0 \leftarrow \textit{prefix}[\textbf{len}(\textit{prefix}) - 1]$ ▷ Witness for reachability.
  ▷ Iterate over funnel-loop templates for c[illegible]
  for all *template* $\in$ GENERATE-TEMPLATES[illegible]
    $\textit{ef\_constrs} \leftarrow \textit{template}.\textit{ef\_constraints}$[illegible]
    $\langle \textit{found}, \textit{model} \rangle \leftarrow$ SEACH-PARAMETER[illegible]
    if $\textit{found} == \top$ **then** ▷ [illegible]
      $\textit{fnl\_loop} \leftarrow \textit{template}.\textit{instantiate}(\textit{model})$
      return $\langle \textit{prefix}, \textit{fnl\_loop} \rangle$ ▷ Reachability witness and funnel-loop.
    end if
  end for
end for
return *unknown*

Enumerate underapproximations of M that represent fair loops over some predicates.

Funnel-loop search: overall procedure

SEARCH-FUNNEL-LOOP(M, $\mathcal{H}$)

▷ Iterate over candidate loops of increasing length.

for all $\langle$*prefix*, *loop_r*, *loop_t*, $H\rangle \in$ GENERATE-CANDIDATE-LOOPS($M, \mathcal{H}$) **do**

$v_0 \leftarrow$ *prefix*[**len**(*prefix*) − 1] ▷ Witness for reachability.

loop_r is a sequence of regions,
loop_t is a sequence of transitions,
H is a E-comp obtained from $\mathcal{H}$,
prefix is a path from an initial state to the first region.

…rrent candidate loop.

…v_0, *loop_r*, *loop_t*, H) **do**

… ▷ Get ∃∀ problem.

…ASSIGNMENT(*ef_constrs*)

…Replace parameters with assignment.

…*del*)

…Reachability witness and funnel-loop.

end if

end for

end for

return *unknown*

Funnel-loop search: overall procedure

SEARCH-FUNNEL-LOOP(M, $\mathcal{H}$)

```
▷ Iterate over candidate loops of increasing length.
for all ⟨prefix, loop_r, loop_t, H⟩ ∈ GENERATE-CANDIDATE-LOOPS(M, 𝓗) do
    v_0 ← prefix[len(prefix) − 1]                          ▷ Witness for reachability.
    ▷ Iterate over funnel-loop templates for current candidate loop.
    for all template ∈ GENERATE-TEMPLATES(v_0, loop_r, loop_t, H) do
        ef_constrs ← template.ef_constrai
        ⟨found, model⟩ ← SEACH-PARAMET
        if found == ⊤ then
            fnl_loop ← template.instantiate
            return ⟨prefix, fnl_loop⟩
        end if
    end for
end for
return unknown
```

Generate funnel-loop templates by strengthening *loop_r* and *loop_t* with parametric predicates.

Funnel-loop search: overall procedure

SEARCH-FUNNEL-LOOP(M, $\mathcal{H}$)

▷ Iterate over candidate loops of increasing length.

for all $\langle$*prefix*, *loop_r*, *loop_t*, $H\rangle \in$ GENERATE-CANDIDATE-LOOPS($M, \mathcal{H}$) **do**

  $v_0 \leftarrow$ *prefix*[**len**(*prefix*) $- 1$] ▷ Witness for reachability.

  ▷ Iterate over funnel-loop templates for current candidate loop.

  for all *template* $\in$ GENERATE-TEMPLATES(v_0, *loop_r*, *loop_t*, H) **do**

    ef_constrs $\leftarrow$ *template.ef_constraints*() ▷ Get $\exists\forall$ problem.

    $\langle$*found*, *model*$\rangle \leftarrow$ SEACH-PARAMETER-ASSIGNMENT(*ef_constrs*)

    if *found* $== \top$ **then**

      fnl_loop $\leftarrow$ *template.ins*

      return $\langle$*prefix*, *fnl_loop*$\rangle$

    end if

  end for

end for

return *unknown*

Get $\exists\forall$ quantified formula: exists assignment to parameters such that template corresponds to funnel-loop.

Funnel-loop search: overall procedure

SEARCH-FUNNEL-LOOP(M, $\mathcal{H}$)

▷ Iterate over candidate loops of increasing length.

for all $\langle prefix, loop\_r, loop\_t, H\rangle \in$ GENERATE-CANDIDATE-LOOPS($M, \mathcal{H}$) **do**

  $v_0 \leftarrow prefix[\mathbf{len}(prefix) - 1]$ ▷ Witness for reachability.

  ▷ Iterate over funnel-loop templates for current candidate loop.

  for all $template \in$ GENERATE-TEMPLATES($v_0, loop\_r, loop\_t, H$) **do**

    $ef\_constrs \leftarrow template.ef\_constraints()$ ▷ Get ∃∀ problem.

    $\langle found, model\rangle \leftarrow$ SEACH-PARAMETER-ASSIGNMENT($ef\_constrs$)

    if $found == \top$ **then** ▷ Replace parameters with assignment.

    ty witness and funnel-loop.

Solve ∃∀ problem: find assignment for parameters.

  end for

end for

return *unknown*

Funnel-loop search: overall procedure

SEARCH-FUNNEL-LOOP(M, $\mathcal{H}$)

```
    ▷ Iterate over candidate loops of increasing length.
for all ⟨prefix, loop_r, loop_t, H⟩ ∈ GENERATE-CANDIDATE-LOOPS(M, ℋ) do
    v_0 ← prefix[len(prefix) − 1]                                  ▷ Witness for reachability.
    ▷ Iterate over funnel-loop templates for current candidate loop.
    for all template ∈ GENERATE-TEMPLATES(v_0, loop_r, loop_t, H) do
        ef_constrs ← template.ef_constraints()                      ▷ Get ∃∀ problem.
        ⟨found, model⟩ ← SEACH-PARAMETER-ASSIGNMENT(ef_constrs)
        if found == ⊤ then                           ▷ Replace parameters with assignment.
            fnl_loop ← template.instantiate(model)
            return ⟨prefix, fnl_loop⟩                ▷ Reachability witness and funnel-loop.
        end if
    end for
end for
return unknown
```

If successful return funnel-loop, otherwise analyse next template or candidate loop

Funnel-loop search: enumerate candidate loops

GENERATE-CANDIDATE-LOOPS(M, $\mathcal{H}$)

▷ L2S encoding into reachability problem and E-comp selection.

$\langle V, I, T, bad\rangle \leftarrow$ ENCODE-L2S-FAIR-ABSTRACT-LOOP(M, $\mathcal{H}$)
for $k \in [0, 1, 2, \ldots]$ **do** ▷ BMC unrolling: k steps.
    $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k)$ ▷ BMC reachability.
    $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) ▷ Find first path of length k.
    $refs \leftarrow []$ ▷ Keep track of visited paths of length k.
    while sat **do** ▷ Generate all candidates from paths of same length.
        $H \leftarrow$ GET-CANDIDATE-COMPOSITION($model$) ▷ Path selects hints.
        $\langle conflict\rangle \leftarrow$ GET-COMP-ERROR(H)
        if $conflict \neq \bot$ **then** ▷ Learn incompatible transitions.
            $\langle V, I, T, bad\rangle \leftarrow$ REMOVE-CONFLICT($V, I, T, bad, conflict$)
        else ▷ H is valid E-comp.
            $\langle loop\_r, loop\_t\rangle \leftarrow$ UNDERAPPROXIMATE($model, query, H$)
            $\langle is\_ranked, rf\rangle \leftarrow$ RANK-LOOP($loop\_r, loop\_t, H$)
            if is_ranked **then** ▷ Learn ranking function.
                $\langle V, I, T, bad\rangle \leftarrow$ REMOVE-RANKED-LOOPS(V, I, T, bad, rf)
            else ▷ Unable to find ranking function, could be nonterminating.
                $prefix \leftarrow$ GET-PREFIX($model$) ▷ Get stem of abstract lasso.
                yield $\langle prefix, loop\_r, loop\_t, H\rangle$ ▷ Coroutine returns triples.
                $refs.append(\neg(\bigwedge_{r \in loop\_r} r \wedge \bigwedge_{t \in loop\_t} t))$ ▷ Mark visited.
            end if
        end if
        $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k) \wedge \bigwedge_{ref \in refs} ref$
        $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) ▷ Find next path of length k.
    end while
end for

Funnel-loop search: enumerate candidate loops

GENERATE-CANDIDATE-LOOPS(M, $\mathcal{H}$)

$\triangleright$ L2S encoding into reachability problem and E-comp selection.

$\langle V, I, T, bad \rangle \leftarrow$ ENCODE-L2S-FAIR-ABSTRACT-LOOP(M, $\mathcal{H}$)
for $k \in [0, 1, 2, \ldots]$ **do** $\triangleright$ BMC unrolling: k steps.
  $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k)$ $\triangleright$ BMC reachability.
  $\langle sat,$
  $refs \leftarrow$
  while
    H
    $\langle c$
    if

L2S encoding for the search of fair loops and compositions. The loop-back is identified in the abstract space defined by a set of predicates.

      $\langle V, I, T, bad \rangle \leftarrow$ REMOVE-CONFLICT($V, I, T, bad, conflict$)
    else $\triangleright$ H is valid E-comp.
      $\langle loop\_r, loop\_t \rangle \leftarrow$ UNDERAPPROXIMATE($model, query, H$)
      $\langle is\_ranked, rf \rangle \leftarrow$ RANK-LOOP($loop\_r, loop\_t, H$)
      if is_ranked **then** $\triangleright$ Learn ranking function.
        $\langle V, I, T, bad \rangle \leftarrow$ REMOVE-RANKED-LOOPS(V, I, T, bad, rf)
      else $\triangleright$ Unable to find ranking function, could be nonterminating.
        $prefix \leftarrow$ GET-PREFIX($model$) $\triangleright$ Get stem of abstract lasso.
        yield $\langle prefix, loop\_r, loop\_t, H \rangle$ $\triangleright$ Coroutine returns triples.
        $refs.append(\neg(\bigwedge_{r \in loop\_r} r \wedge \bigwedge_{t \in loop\_t} t))$ $\triangleright$ Mark visited.
      end if
    end if
    $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k) \wedge \bigwedge_{ref \in refs} ref$
    $\langle sat, model \rangle \leftarrow$ SMT-SOLVE($query$) $\triangleright$ Find next path of length k.
  end while
end for

Funnel-loop search: enumerate candidate loops

GENERATE-CANDIDATE-LOOPS(M, $\mathcal{H}$)

$\triangleright$ L2S encoding into reachability problem and E-comp selection.

$\langle V, I, T, bad\rangle \leftarrow$ ENCODE-L2S-FAIR-ABSTRACT-LOOP(M, $\mathcal{H}$)

for $k \in [0, 1, 2, \ldots]$ **do** $\triangleright$ BMC unrolling: k steps.

  $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k)$ $\triangleright$ BMC reachability.

  $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) $\triangleright$ Find first path of length k.

  $refs \leftarrow []$ $\triangleright$ Keep track of visited paths of length k.

  while sat **do** $\triangleright$ Generate a … ame length.

    $H \leftarrow$ GET-CANDIDATE-COMPOSITION($model$) … elects hints.

    $\langle conflict\rangle \leftarrow$ GET-COMP-ERROR(H)

    if $conflict \neq \perp$ **then** $\triangleright$ Learn incompatible transitions.

      $\langle V, I, T, bad\rangle \leftarrow$ REMOVE-CONFLICT($V, I, T, bad, conflict$)

    else $\triangleright$ H is valid E-comp.

      $\langle loop\_r, loop\_t\rangle \leftarrow$ UNDERAPPROXIMATE($model, query, H$)

      $\langle is\_ranked, rf\rangle \leftarrow$ RANK-LOOP($loop\_r, loop\_t, H$)

      if is_ranked **then** $\triangleright$ Learn ranking function.

        $\langle V, I, T, bad\rangle \leftarrow$ REMOVE-RANKED-LOOPS(V, I, T, bad, rf)

      else $\triangleright$ Unable to find ranking function, could be nonterminating.

        $prefix \leftarrow$ GET-PREFIX($model$) $\triangleright$ Get stem of abstract lasso.

        yield $\langle prefix, loop\_r, loop\_t, H\rangle$ $\triangleright$ Coroutine returns triples.

        $refs.append(\neg(\bigwedge_{r \in loop\_r} r \wedge \bigwedge_{t \in loop\_t} t))$ $\triangleright$ Mark visited.

      end if

    end if

    $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k) \wedge \bigwedge_{ref \in refs} ref$

    $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) $\triangleright$ Find next path of length k.

  end while

end for

BMC unrolling.

Funnel-loop search: enumerate candidate loops

GENERATE-CANDIDATE-LOOPS(M, $\mathcal{H}$)

▷ L2S encoding into reachability problem and E-comp selection.

$\langle V, I, T, bad\rangle \leftarrow$ ENCODE-L2S-FAIR-ABSTRACT-LOOP(M, $\mathcal{H}$)
for $k \in [0, 1, 2, \ldots]$ **do** ▷ BMC unrolling: k steps.
  $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k)$ ▷ BMC reachability.
  $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) ▷ Find first path of length k.
  $refs \leftarrow []$ ▷ Keep track of visited paths of length k.
  while sat **do**
    $H \leftarrow$ GET-CANDIDATE-COMPOSI
    $\langle conflict\rangle \leftarrow$ GET-COMP-ERROR(H)
    if $conflict \neq \bot$ **then** ▷ Learn incompatible transitions.
      $\langle V, I, T, bad\rangle \leftarrow$ REMOVE-CONFLICT($V, I, T, bad, conflict$)
    else ▷ H is valid E-comp.
      $\langle loop\_r, loop\_t\rangle \leftarrow$ UNDERAPPROXIMATE($model, query, H$)
      $\langle is\_ranked, rf\rangle \leftarrow$ RANK-LOOP($loop\_r, loop\_t, H$)
      if is_ranked **then** ▷ Learn ranking function.
        $\langle V, I, T, bad\rangle \leftarrow$ REMOVE-RANKED-LOOPS(V, I, T, bad, rf)
      else ▷ Unable to find ranking function, could be nonterminating.
        $prefix \leftarrow$ GET-PREFIX($model$) ▷ Get stem of abstract lasso.
        yield $\langle prefix, loop\_r, loop\_t, H\rangle$ ▷ Coroutine returns triples.
        $refs.append(\neg(\bigwedge_{r \in loop\_r} r \wedge \bigwedge_{t \in loop\_t} t))$ ▷ Mark visited.
      end if
    end if
    $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k) \wedge \bigwedge_{ref \in refs} ref$
    $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) ▷ Find next path of length k.
  end while
end for

Iterate over candidate loops of length k.

Funnel-loop search: enumerate candidate loops

GENERATE-CANDIDATE-LOOPS(M, $\mathcal{H}$)

$\triangleright$ L2S encoding into reachability problem and E-comp selection.
$\langle V, I, T, bad\rangle \leftarrow$ ENCODE-L2S-FAIR-ABSTRACT-LOOP(M, $\mathcal{H}$)
for $k \in [0, 1, 2, \ldots]$ **do** $\triangleright$ BMC unrolling: k steps.
  $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k)$ $\triangleright$ BMC reachability.
  $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) … th k.
  $refs \leftarrow []$ … th k.
  while sat **do** $\triangleright$ Generate all candidates from paths of same length.
    $H \leftarrow$ GET-CANDIDATE-COMPOSITION($model$) $\triangleright$ Path selects hints.
    $\langle conflict\rangle \leftarrow$ GET-COMP-ERROR(H)
    if $conflict \neq \perp$ **then** $\triangleright$ Learn incompatible transitions.
      $\langle V, I, T, bad\rangle \leftarrow$ REMOVE-CONFLICT($V, I, T, bad, conflict$)
    else $\triangleright$ H is valid E-comp.
      if is_ranked **then** $\triangleright$ Learn ranking function.
        $\langle V, I, T, bad\rangle \leftarrow$ REMOVE-RANKED-LOOPS(V, I, T, bad, rf)
      else $\triangleright$ Unable to find ranking function, could be nonterminating.
        $prefix \leftarrow$ GET-PREFIX($model$) $\triangleright$ Get stem of abstract lasso.
        yield $\langle prefix, loop\_r, loop\_t, H\rangle$ $\triangleright$ Coroutine returns triples.
        $refs.append(\neg(\bigwedge_{r \in loop\_r} r \wedge \bigwedge_{t \in loop\_t} t))$ $\triangleright$ Mark visited.
      end if
    end if
    $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k) \wedge \bigwedge_{ref \in refs} ref$
    $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) $\triangleright$ Find next path of length k.
  end while
end for

Get guessed composition of E-comps.

Check its correctness, learn from error if it is not.

Funnel-loop search: enumerate candidate loops

GENERATE-CANDIDATE-LOOPS(M, $\mathcal{H}$)

$\triangleright$ L2S encoding into reachability problem and E-comp selection.

$\langle V, I, T, bad \rangle \leftarrow$ ENCODE-L2S-FAIR-ABSTRACT-LOOP(M, $\mathcal{H}$)
for $k \in [0, 1, 2, \ldots]$ **do** $\triangleright$ BMC unrolling: k steps.
    $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k)$ $\triangleright$ BMC reachability.
    $\langle sat, model \rangle \leftarrow$ SMT-SOLVE($query$) $\triangleright$ Find first path of length k.
    $refs \leftarrow []$ $\triangleright$ Keep track of visited paths of length k.
    while sat **do** $\triangleright$ Generate all candidates from paths of same length.
        $H \leftarrow$ GET-CANDIDATE-COMP
        $\langle conflict \rangle \leftarrow$ GET-COMP-ERR
        if $conflict \neq \bot$ **then**
            $\langle V, I, T, bad \rangle \leftarrow$ REMO
        else $\triangleright$ H is valid E-comp.
            $\langle loop\_r, loop\_t \rangle \leftarrow$ UNDERAPPROXIMATE($model$, $query$, H)
            $\langle is\_ranked, rf \rangle \leftarrow$ RANK-LOOP($loop\_r$, $loop\_t$, H)
            if is_ranked **then** $\triangleright$ Learn ranking function.
                $\langle V, I, T, bad \rangle \leftarrow$ REMOVE-RANKED-LOOPS(V, I, T, bad, rf)
            else $\triangleright$ Unable to find ranking function, could be nonterminating.
                $prefix \leftarrow$ GET-PREFIX($model$) $\triangleright$ Get stem of abstract lasso.
                yield $\langle prefix, loop\_r, loop\_t, H \rangle$ $\triangleright$ Coroutine returns triples.
                $refs.append(\neg(\bigwedge_{r \in loop\_r} r \wedge \bigwedge_{t \in loop\_t} t))$ $\triangleright$ Mark visited.
            end if
        end if
        $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k) \wedge \bigwedge_{ref \in refs} ref$
        $\langle sat, model \rangle \leftarrow$ SMT-SOLVE($query$) $\triangleright$ Find next path of length k.
    end while
end for

Compute underapproximation of M corresponding to the loop described by $model$.

Funnel-loop search: enumerate candidate loops

GENERATE-CANDIDATE-LOOPS(M, $\mathcal{H}$)

▷ L2S encoding into reachability problem and E-comp selection.

$\langle V, I, T, bad \rangle \leftarrow$ ENCODE-L2S-FAIR-ABSTRACT-LOOP(M, $\mathcal{H}$)
for $k \in [0, 1, 2, \ldots]$ **do** ▷ BMC unrolling: k steps.
  $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k)$ ▷ BMC reachability.
  $\langle sat, model \rangle \leftarrow$ SMT-SOLVE($query$) ▷ Find first path of length k.
  $refs \leftarrow []$ ▷ Keep track of visited paths of length k.
  while sat **do** ▷ Generate all candidates from paths of same length.
    $H \leftarrow$ GET-CANDIDATE-COMPOSITION($model$) ▷ Path selects hints.
    $\langle conflict \rangle \leftarrow$ GET-COMP-ERROR(H)
    if $conflict \neq \bot$ **then**
      $\langle V, I, T, bad \rangle \leftarrow$ REM
    else
      $\langle loop\_r, loop\_t \rangle \leftarrow$ UNDERAPPROXIMATE($model$, $query$, H)
      $\langle is\_ranked, rf \rangle \leftarrow$ RANK-LOOP($loop\_r$, $loop\_t$, H)
      if is_ranked **then** ▷ Learn ranking function.
        $\langle V, I, T, bad \rangle \leftarrow$ REMOVE-RANKED-LOOPS(V, I, T, bad, rf)
      else ▷ Unable to find ranking function, could be nonterminating.
        of abstract lasso.
        tine returns triples.
        ▷ Mark visited.
      end
    end if
    $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k) \wedge \bigwedge_{ref \in refs} ref$
    $\langle sat, model \rangle \leftarrow$ SMT-SOLVE($query$) ▷ Find next path of length k.
  end while
end for

Try synthesise ranking function for candidate.

In case of success remove all loops ranked by the function.

Funnel-loop search: enumerate candidate loops

GENERATE-CANDIDATE-LOOPS(M, $\mathcal{H}$)

▷ L2S encoding into reachability problem and E-comp selection.

$\langle V, I, T, bad\rangle \leftarrow$ ENCODE-L2S-FAIR-ABSTRACT-LOOP(M, $\mathcal{H}$)

for $k \in [0, 1, 2, \ldots]$ **do** ▷ BMC unrolling: k steps.

  $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k)$ ▷ BMC reachability.

  $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) ▷ Find first path of length k.

  $refs \leftarrow []$ ▷ Keep track of visited paths of length k.

  while sat **do** ▷ Generate all candidates from paths of same length.

    $H \leftarrow$ GET-CANDIDATE-COMPOSITION($model$) ▷ Path selects hints.

    $\langle conflict\rangle \leftarrow$ GET-COMP-ERROR(H)

    if $conflict \neq \bot$ **then** ▷ Learn incompatible transitions.

      $\langle V, I, T, bad\rangle \leftarrow$ REMOVE-CONFLICT($V, I, T, bad, conflict$)

    else ▷ H is valid E-comp.

      $\langle loop\_r, loop\_t\rangle \leftarrow$ UNDERAPPROXIMATE($model, query, H$)

      $\langle is\_ranked, rf\rangle \leftarrow$ RANK-

      if is_ranked **then**

        $\langle V, I, T, bad\rangle \leftarrow$ RE

      else ▷ Unable to find ranking function, could be nonterminating.

        $prefix \leftarrow$ GET-PREFIX($model$) ▷ Get stem of abstract lasso.

        yield $\langle prefix, loop\_r, loop\_t, H\rangle$ ▷ Coroutine returns triples.

        $refs.append(\neg(\bigwedge_{r \in loop\_r} r \wedge \bigwedge_{t \in loop\_t} t))$ ▷ Mark visited.

      end if

    end if

    $query \leftarrow I(V_0) \wedge \bigwedge_{i=0}^{k-1} T(V_i, V_{i+1}) \wedge bad(V_k) \wedge \bigwedge_{ref \in refs} ref$

    $\langle sat, model\rangle \leftarrow$ SMT-SOLVE($query$) ▷ Find next path of length k.

  end while

end for

Get reachability witness from *model* and return current candidate.

Funnel-loop templates

GENERATE-TEMPLATES($\boldsymbol{v}_0$, *loop_r*, *loop_t*, H)

$ineqs \leftarrow$ HEURISTIC-PICK-NUM-INEQS(*loop_r*, *loop_t*, H)
$\langle V^H, I^H, T^H, \mathcal{R}, \mathcal{A}, \textsc{Rf}\rangle \leftarrow H$ ▷ Get components defining H.
for $ineq \in ineqs$ **do** ▷ Use *ineq* parametric inequalities in regions.
$n \leftarrow$ **len**(*loop_r*) ▷ Length of template + 1: loop-back region.
$funnels \leftarrow []$ ▷ List of funnels for funnel-loop template.
for $i \in [0..n-2]$ **do** ▷ Create i^{th} funnel: $\langle V, src, t, rf, dst\rangle$.
$src \leftarrow loop\_r[i] \wedge \mathcal{R}[i] \wedge \mathcal{A}[i] \wedge \bigwedge_{j=0}^{ineq-1}$ NEW-PARAM-EXPR$(V) \geq 0$
if $\exists V : 0 < \textsc{Rf}[i](V)$ **then**
$rf \leftarrow \textsc{Rf}[i]$ ▷ H defines ranking function.
else
$rf \leftarrow$ NEW-PARAM-EXPR(V) ▷ Parametric ranking function.
end if
$t \leftarrow \mathcal{R}[i] \wedge \mathcal{A}[i]$ ▷ Transition of H in i^{th} region.
$t \leftarrow t \wedge T^H \wedge ((0 < rf \wedge \mathcal{R}[i]' \wedge rf' \leq rf) \vee (rf = 0 \wedge \mathcal{R}[i+1]'))$
for $v_{i+1} \in V_{i+1} \setminus V_{i+1}^H$ **do** ▷ Add functional assign for v_{i+1} in t
if $v_{i+1} = f(V_i) \in loop\_t[i]$ for some function f **then**
$t \leftarrow t \wedge v_{i+1} = f(V_i)$ ▷ Functional assignment in candidate.
else
$t \leftarrow t \wedge v_{i+1} =$ NEW-PARAM-EXPR(V_i) ▷ Create new expr.
end if
end for
$P \leftarrow$ COLLECT-PARAMETERS(src, rf, t) ▷ Params in current funnel.
$dst(V', P) \leftarrow \exists V : src(V, P) \wedge rf(V, P) = 0 \wedge t(V, V', P)$
$funnels.append(\textsc{Funnel}(src, t, rf, dst))$
end for
yield FUNNEL-LOOP(*funnels*, $\boldsymbol{v}_0$) ▷ Coroutine returns templates.
end for

Funnel-loop templates

GENERATE-TEMPLATES($\boldsymbol{v}_0$, $loop\_r$, $loop\_t$, H)

$ineqs \leftarrow$ HEURISTIC-PICK-NUM-INEQS($loop\_r$, $loop\_t$, H)
$\langle V^H, I^H, T^H, \mathcal{R}, \mathcal{A}, \text{RF} \rangle \leftarrow H$ ▷ Get components defining H.
for $ineq \in ineqs$ **do**
  $n \leftarrow$ **len**($loop\_r$)
  $funnels \leftarrow []$
  for $i \in [0..n-2]$ **do**
    $src \leftarrow loop\_r[i] \wedge \mathcal{R}[i] \wedge \mathcal{A}[i] \wedge \bigwedge_{j=0}^{ineq-1}$ NEW-PARAM-EXPR(V) ≥ 0
    if $\exists V : 0 < \text{RF}[i](V)$ **then**
      $rf \leftarrow \text{RF}[i]$ ▷ H defines ranking function.
    else
      $rf \leftarrow$ NEW-PARAM-EXPR(V) ▷ Parametric ranking function.
    end if
    $t \leftarrow \mathcal{R}[i] \wedge \mathcal{A}[i]$ ▷ Transition of H in i^{th} region.
    $t \leftarrow t \wedge T^H \wedge ((0 < rf \wedge \mathcal{R}[i]' \wedge rf' \leq rf) \vee (rf = 0 \wedge \mathcal{R}[i+1]'))$
    for $v_{i+1} \in V_{i+1} \setminus V_{i+1}^H$ **do** ▷ Add functional assign for v_{i+1} in t
      if $v_{i+1} = f(V_i) \in loop\_t[i]$ for some function f **then**
        $t \leftarrow t \wedge v_{i+1} = f(V_i)$ ▷ Functional assignment in candidate.
      else
        $t \leftarrow t \wedge v_{i+1} =$ NEW-PARAM-EXPR(V_i) ▷ Create new expr.
      end if
    end for
    $P \leftarrow$ COLLECT-PARAMETERS(src, rf, t) ▷ Params in current funnel.
    $dst(V', P) \leftarrow \exists V : src(V, P) \wedge rf(V, P) = 0 \wedge t(V, V', P)$
    $funnels.append$(FUNNEL(src, t, rf, dst))
  end for
  yield FUNNEL-LOOP($funnels$, $\boldsymbol{v}_0$) ▷ Coroutine returns templates.
end for

Create a funnel template for every region in the candidate loop.

Funnel-loop templates

GENERATE-TEMPLATES($\boldsymbol{v}_0$, *loop_r*, *loop_t*, H)

$ineqs \leftarrow$ HEURISTIC-PICK-NUM-INEQS($loop\_r$, $loop\_t$, H)
$\langle V^H, I^H, T^H, \mathcal{R}, \mathcal{A}, \mathrm{RF} \rangle \leftarrow H$ ▷ Get components defining H.
for $ineq \in ineqs$ **do** ▷ Use *ineq* parametric inequalities in regions.
  $n \leftarrow$ **len**($loop\_r$) ▷ Length of template + 1: loop-back region.
  $funnels \leftarrow []$ ▷ List of funnels for funnel-loop template.
  for $i \in [0..n-2]$ **do** ▷ Create i^{th} funnel: $\langle V, src, t, rf, dst \rangle$.
    $src \leftarrow loop\_r[i] \wedge \mathcal{R}[i] \wedge \mathcal{A}[i] \wedge \bigwedge_{j=0}^{ineq-1}$ NEW-PARAM-EXPR$(V) \geq 0$
    if $\exists V : 0 < \mathrm{RF}[i](V)$ **then**
      $rf \leftarrow \mathrm{RF}[i]$ ▷ H defines ranking function.
    else
      ...king function.
    end
    $t \leftarrow$ ... H in i^{th} region.
    $t \leftarrow t \wedge$ ...
    for $v_{i+1} \in V_{i+1} \setminus V_{i+1}^H$ **do** ▷ Add functional assign for v_{i+1} in t
      if $v_{i+1} = f(V_i) \in loop\_t[i]$ for some function f **then**
        $t \leftarrow t \wedge v_{i+1} = f(V_i)$ ▷ Functional assignment in candidate.
      else
        $t \leftarrow t \wedge v_{i+1} =$ NEW-PARAM-EXPR(V_i) ▷ Create new expr.
      end if
    end for
    $P \leftarrow$ COLLECT-PARAMETERS(src, rf, t) ▷ Params in current funnel.
    $dst(V', P) \leftarrow \exists V \;:\; src(V, P) \wedge rf(V, P) = 0 \wedge t(V, V', P)$
    $funnels.append$(FUNNEL(src, t, rf, dst))
  end for
  yield FUNNEL-LOOP($funnels$, $\boldsymbol{v}_0$) ▷ Coroutine returns templates.
end for

Strengthen region with *ineq* new parametric predicates, e.g. linear combinations $\sum_{v \in V_i} \lambda_v v$.

Funnel-loop templates

GENERATE-TEMPLATES($\boldsymbol{v}_0$, $loop\_r$, $loop\_t$, H)

$ineqs \leftarrow$ HEURISTIC-PICK-NUM-INEQS($loop\_r$, $loop\_t$, H)
$\langle V^H, I^H, T^H, \mathcal{R}, \mathcal{A}, \mathrm{RF} \rangle \leftarrow H$ ▷ Get components defining H.
for $ineq \in ineqs$ **do** ▷ Use $ineq$ parametric inequalities in regions.
  $n \leftarrow$ **len**($loop\_r$) ▷ Length of template + 1: loop-back region.
  $funnels \leftarrow []$ ▷ List of funnels for funnel-loop template.
  for $i \in [0..n-2]$ **do** ▷ Create i^{th} funnel: $\langle V, src, t, rf, dst \rangle$.
    $src \leftarrow loop\_r[i] \wedge \mathcal{R}[i] \wedge \mathcal{A}[i] \wedge \bigwedge_{j=0}^{ineq-1}$ NEW-PARAM-EXPR$(V) \geq 0$
    if $\exists V : 0 < \mathrm{RF}[i](V)$ **then**
      $rf \leftarrow \mathrm{RF}[i]$ ▷ H defines ranking function.
    else
      $rf \leftarrow$ NEW-PARAM-EXPR(V) ▷ Parametric ranking function.
    end if
    $t \leftarrow \mathcal{R}[i] \wedge \mathcal{A}[i]$ ▷ Transition of H in i^{th} region.
    $t \leftarrow t \wedge T^H \wedge ((0 < rf \wedge \mathcal{R}[i]' \wedge rf' \leq rf) \vee (rf = 0 \wedge \mathcal{R}[i+1]'))$
    for $v_{i+1} \in V_{i+1} \setminus V_{i+1}^H$ **do** ▷ Add functional assign for v_{i+1} in t
      if $v_{i+1} = f(V_i) \in loop\_t[i]$ for some function f **then**
        $t \leftarrow t \wedge v_{i+1} = f(V_i)$ ▷ Functional assignment in candidate.
      else
        $t \leftarrow t \wedge v_{i+1} =$ NEW-PARAM-EXPR(V_i) ▷ Create new expr.
      end if
    end for
    $P \leftarrow$ COLLECT-PARAMETERS(src, rf, t) ▷ Params in current funnel.
    $dst(V', P) \leftarrow \exists V : src(V, P) \wedge rf(V, P) = 0 \wedge t(V, V', P)$
    $funnels.append$(FUNNEL(src, t, rf, dst))
  end for
  yield FUNNEL-LOOP($funnels$, $\boldsymbol{v}_0$) ▷ Coroutine returns templates.
end for

Transition relation of composed E-comps.

Funnel-loop templates

GENERATE-TEMPLATES($\boldsymbol{v}_0$, *loop_r*, *loop_t*, H)

$ineqs \leftarrow$ HEURISTIC-PICK-NUM-INEQS(*loop_r*, *loop_t*, H)
$\langle V^H, I^H, T^H, \mathcal{R}, \mathcal{A}, \mathrm{RF} \rangle \leftarrow H$ ▷ Get components defining H.
for $ineq \in ineqs$ **do** ▷ Use *ineq* parametric inequalities in regions.
  $n \leftarrow$ **len**(*loop_r*) ▷ Length of template + 1: loop-back region.
  $funnels \leftarrow []$ ▷ List of funnels for funnel-loop template.
  for $i \in [0..n-2]$ **do** ▷ Create i^{th} funnel: $\langle V, src, t, rf, dst \rangle$.
    $src \leftarrow loop\_r[i] \wedge \mathcal{R}[i] \wedge \mathcal{A}[i] \wedge \bigwedge^{ineq-1}$ NEW-PARAM-EXPR$(V) \geq 0$
    if $\exists V : 0 < \mathrm{RF}[i](V)$ **then**
      $rf \leftarrow \mathrm{RF}[i]$
    else
      $rf \leftarrow$ NEW-PARAM-EXPR(V)
    end if
    $t \leftarrow \mathcal{R}[i] \wedge \mathcal{A}[i]$ ▷ Transition of H in i^{th} region.
    $t \leftarrow t \wedge T^H \wedge ((0 < rf \wedge \mathcal{R}[i]' \wedge rf' \leq rf) \vee (rf = 0 \wedge \mathcal{R}[i+1]'))$
    for $v_{i+1} \in V_{i+1} \setminus V_{i+1}^H$ **do** ▷ Add functional assign for v_{i+1} in t
      if $v_{i+1} = f(V_i) \in loop\_t[i]$ for some function f **then**
        $t \leftarrow t \wedge v_{i+1} = f(V_i)$ ▷ Functional assignment in candidate.
      else
        $t \leftarrow t \wedge v_{i+1} =$ NEW-PARAM-EXPR(V_i) ▷ Create new expr.
      end if
    end for
    $P \leftarrow$ COLLECT-PARAMETERS(src, rf, t) ▷ Params in current funnel.
    $dst(V', P) \leftarrow \exists V : src(V, P) \wedge rf(V, P) = 0 \wedge t(V, V', P)$
    $funnels.append(\text{FUNNEL}(src, t, rf, dst))$
  end for
  yield FUNNEL-LOOP$(funnels, \boldsymbol{v}_0)$ ▷ Coroutine returns templates.
end for

Next assignments are function of current assignments: transition relation left-total by construction.

Funnel-loop templates

GENERATE-TEMPLATES($\boldsymbol{v}_0$, $loop\_r$, $loop\_t$, H)

$ineqs \leftarrow$ HEURISTIC-PICK-NUM-INEQS($loop\_r$, $loop\_t$, H)
$\langle V^H, I^H, T^H, \mathcal{R}, \mathcal{A}, \mathrm{RF} \rangle \leftarrow H$ ▷ Get components defining H.
for $ineq \in ineqs$ **do** ▷ Use $ineq$ parametric inequalities in regions.
    $n \leftarrow$ **len**($loop\_r$) ▷ Length of template + 1: loop-back region.
    $funnels \leftarrow []$ ▷ List of funnels for funnel-loop template.
    for $i \in [0..n-2]$ **do** ▷ Create i^{th} funnel: $\langle V, src, t, rf, dst \rangle$.
        $src \leftarrow loop\_r[i] \wedge \mathcal{R}[i] \wedge \mathcal{A}[i] \wedge \bigwedge_{j=0}^{ineq-1}$ NEW-PARAM-EXPR$(V) \geq 0$
        if $\exists V : 0 < \mathrm{RF}[i](V)$ **then**
            $rf \leftarrow \mathrm{RF}[i]$ ▷ H defines ranking function.
        else
            $rf \leftarrow$ NEW-PARAM-EXPR(V) ▷ Parametric ranking function.
        end if
        $t \leftarrow \mathcal{R}[i] \wedge \mathcal{A}[i]$ ▷ Transition of H in i^{th} region.
        $t \leftarrow t \wedge T^H \wedge ((0 < rf \wedge \mathcal{R}[i]' \wedge rf' \leq rf) \vee (rf = 0 \wedge \mathcal{R}[i+1]'))$
        for $v_{i+1} \in V_{i+1} \setminus V_{i+1}^H$ **do** ▷ Add functional assign for v_{i+1} in t
            if $v_{i+1} = f(V_i) \in loop\_t[i]$ for some function f **then**
                $t \leftarrow t \wedge v_{i+1} = f(V_i)$ ▷ Functional assignment in candidate.
            else
                $t \leftarrow t \wedge v_{i+1} =$ NEW-PARAM-EXPR(V_i) ▷ Create new expr.
            end if
        end for
        $P \leftarrow$ COLLECT-PARAMETERS(src, rf, t) ▷ Params in current funnel.
        $dst(V', P) \leftarrow \exists V : src(V, P) \wedge rf(V, P) = 0 \wedge t(V, V', P)$
        $funnels.append$(FUNNEL(src, t, rf, dst))
    end for
    yield FUNNEL-LOOP($funnels$, $\boldsymbol{v}_0$) ▷ Coroutine returns templates.
end for

Destination region implicitly defined.

Funnel-loop template example

Assume $loop\_r \doteq [\{k > 0\}, \{k < 0\}]$ and $loop\_t \doteq [\{k' = k - n\}, \{k' = k + n\}]$.

For *ineq* equal to 1 we generate a funnel-loop described by the following components:

$$
\begin{aligned}
S_0 &\doteq k > 0 \wedge \lambda_0 k + \lambda_1 n + \lambda_2 \geq 0;\\
t_0 &\doteq k' = k - n \wedge n' = \lambda_3 k + \lambda_4 n + \lambda_5;\\
\mathrm{RF}_0 &\doteq \lambda_6 n + \lambda_7 k + \lambda_8;
\end{aligned}
$$

$$
\begin{aligned}
S_1 &\doteq k < 0 \wedge \lambda_9 k + \lambda_{10} n + \lambda_{11} \geq 0;\\
t_1 &\doteq k' = k + n \wedge n' = \lambda_{12} k + \lambda_{13} n + \lambda_{14};\\
\mathrm{RF}_1 &\doteq \lambda_{15} n + \lambda_{16} k + \lambda_{17}.
\end{aligned}
$$

Objective: find assignment to the $\{\lambda_i\}_{i=0}^{17}$.

Funnel-loop template example

Assume $loop\_r \doteq [\{k > 0\}, \{k < 0\}]$ and $loop\_t \doteq [\{k' = k - n\}, \{k' = k + n\}]$.

For *ineq* equal to 1 we generate a funnel-loop described by the following components:

$$
\begin{aligned}
S_0 &\doteq k > 0 \wedge \lambda_0 k + \lambda_1 \ldots \\
t_0 &\doteq k' = k - n \wedge n' = \ldots \\
\mathrm{RF}_0 &\doteq \lambda_6 n + \lambda_7 k + \lambda_8; \\
S_1 &\doteq k < 0 \wedge \lambda_9 k + \lambda_{10} \ldots \\
t_1 &\doteq k' = k + n \wedge n' = \ldots \\
\mathrm{RF}_1 &\doteq \lambda_{15} n + \lambda_{16} k + \lambda_{17} \ldots
\end{aligned}
$$

Solution:

$$
\begin{aligned}
S_0 &\doteq k > 0 \wedge n \geq k + 1; \\
t_0 &\doteq k' = k - n \wedge n' = n + 1; \\
\mathrm{RF}_0 &\doteq 0 \\
S_1 &\doteq k < 0 \wedge n \geq -k + 1; \\
t_1 &\doteq k' = k + n \wedge n' = n + 1; \\
\mathrm{RF}_1 &\doteq 0.
\end{aligned}
$$

Objective: find assignment to the $\{\lambda_i\}_{i=0}^{17}$.

Funnel-loop synthesis problem

For a funnel-loop template of length n, search for an assignment to the parameters P such that the following hold:

$\boldsymbol{v}_0$ is in the first region (i.e. funnel-loop is reachable):

$$\exists P \;:\; S_0(\boldsymbol{v}_0, P)$$

Remain in the same region as long as the ranking function is positive:

$$\exists P \forall V, V' : S_i(V,P) \land \mathrm{RF}_i(V,P) > 0 \land T_i(V,V',P) \rightarrow \\ S_i(V',P) \land \mathrm{RF}_i(V',P) < \mathrm{RF}_i(V,P)$$

Reach next region when ranking function is 0:

$$\exists P \forall V, V' : S_i(V,P) \land \mathrm{RF}_i(V,P) = 0 \land T_i(V,V',P) \rightarrow S_{i+_n 1}(V',P)$$

Every step underapproximates the transition relation of M:

$$\exists P \forall V, V' : S(V,P) \land T(V,V',P) \rightarrow T_M(V,V')$$