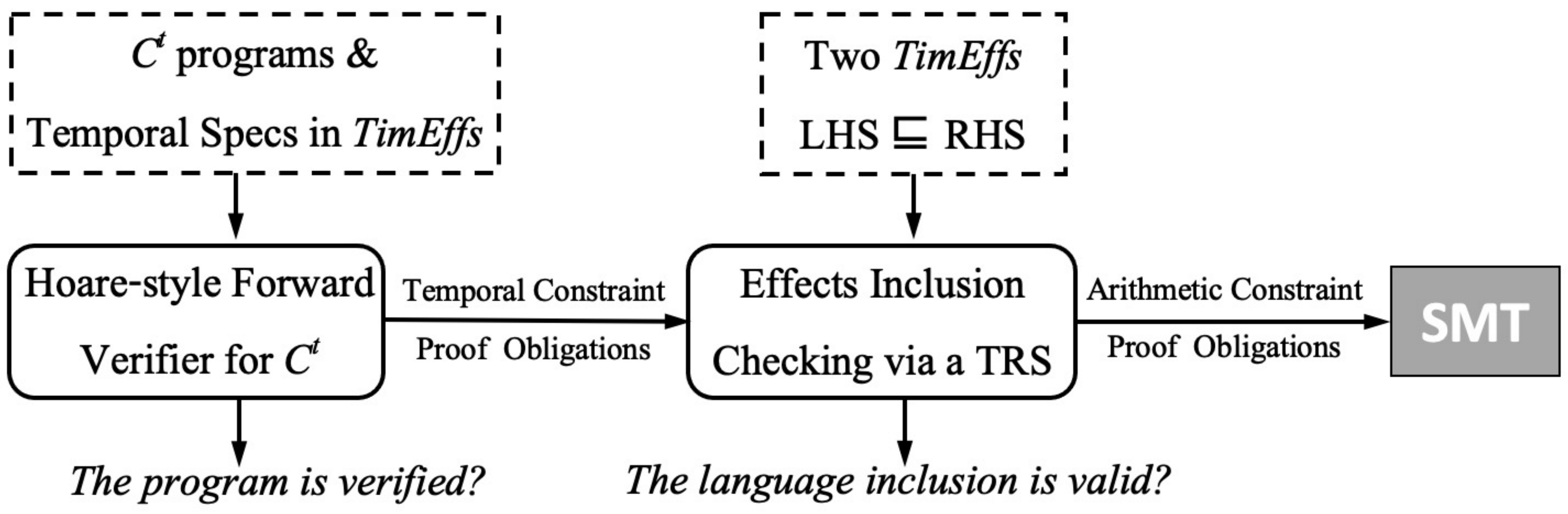
Automated Temporal Verification for Real-Time Systems via Implicit Clocks and an Extended Antimirov Algorithm

Yahui Song @ SPLASH-SRC 2022 ✉ yahuis@comp.nus.edu.sg

National University of Singapore

Overview

To go beyond the existing *Timed Automata (TA)* based techniques, we propose a novel solution that integrates a modular Hoare-style forward verifier with a term rewriting system (TRS) on Timed Effects (*TimEffs*). The purposes are to: increase expressiveness, dynamically manipulate clocks, and efficiently solve clock constraints. The main contributions are:



- Language Abstraction, C^t: generalizes the real-time systems with mutable variables and timed behavioural patterns.
- Novel Specification, *TimEffs*: extends regular expressions with dependent values and arithmetic constraints.
- Efficient Term Rewriting System, *TRS*: solves inclusions between *TimEffs*, by iterated checking of their *derivatives*.

TimEffs (Symbolic Timed Automata)

```
void addOneSugar()
/* req: true ∧ _*
   ens: t>1 ∧ ϵ # t */
{ timeout ((), 1); }

void addNSugar (int n)
/* req: true ∧ _*
   ens: t≥n ∧ EndSugar#t */
{ if (n == 0)
    event["EndSugar"];
  else {
    addOneSugar();
    addNSugar (n-1);}}

void makeCoffee (int n)
/* req: n≥0 ∧ _*· CupReady
   ens: n≤t≤5 ∧ t'≤4 ∧ */
     (EndSugar # t) · (Coffee # t') */
{ deadline (addNSugar(n), 5);
  deadline (event["Coffee"],4)}

int main ()
/* req: true ∧ ϵ
   ens: t≤9 ∧ ((!Done)* # t) · Done*/
{ event["CupReady"];
  makeCoffee (3);
  event["Done"];}
```

$$(Timed\ Effects)\quad \Phi ::= \pi \wedge \theta \mid \Phi_1 \vee \Phi_2$$

$$(Event\ Sequences)\quad \theta ::= \bot \mid \epsilon \mid ev \mid \theta_1 \cdot \theta_2 \mid \theta_1 \vee \theta_2 \mid \theta_1 \| \theta_2 \mid \pi?\theta \mid \theta\#t \mid \theta^\star$$

$$(Events)\quad ev ::= \mathtt{A}(v, \alpha^*) \mid \tau(\pi) \mid \bar{\mathtt{A}} \mid \_$$

$$(Pure)\quad \pi ::= True \mid False \mid bop(t_1, t_2) \mid \pi_1 \wedge \pi_2 \mid \pi_1 \vee \pi_2 \mid \neg\pi \mid \pi_1 \Rightarrow \pi_2$$

$$(Real\text{-}Time\ Terms)\quad t ::= c \mid x \mid t_1 + t_2 \mid t_1 - t_2$$

$c \in \mathbb{Z}$ $\quad x \in \mathbf{var}$ $\quad (Real\ Time\ Bound)\ \#$ $\quad (Kleene\ Star)\ \star$

Our proposal overcomes the following existing limitations:

1) TAs cannot be used to specify/verify incompletely specified systems, i.e., whose timing constants have yet to be known, and hence cannot be used in early design phases;
2) verifying a system with a set of timing constants usually requires enumerating all of them if they are integer-valued;
3) TAs cannot be used to verify systems with timing constants to be taken in a real-valued dense interval.

Language Inclusion – the Antimirov Algorithm

Our TRS is an extension of Antimirov and Mosses' algorithm, which can be deployed to decide the inclusions of two regular expressions (REs) through an iterated process of checking the inclusions of their partial derivatives.

Definition 1 (Derivatives). *Given any formal language S over an alphabet Σ and any string u∈ Σ*, the derivatives of S w.r.t u is defined as:* $u^{-1}S = \{w \in \Sigma^* \mid uw \in S\}$.

Definition 2 (Regular Expression Inclusion). *For REs r and s,*

$$r \preceq s \Leftrightarrow \forall A \in \Sigma.\ A^{-1}(r) \preceq A^{-1}(s).$$

Definition 3 (*TimEffs* Inclusion). *For TimEffs* Φ_1 *and* Φ_2,

$$\Phi_1 \sqsubseteq \Phi_2 \Leftrightarrow \forall A \in \Sigma.\ \forall t \geq 0.\ (A\#t)^{-1}\Phi_1 \sqsubseteq (A\#t)^{-1}\Phi_2.$$

➢ Antimirov V M, Mosses P D. Rewriting extended regular expressions[J]. Theoretical Computer Science, 1995, 143(1): 51-72.

Expressiveness of *TimEffs*

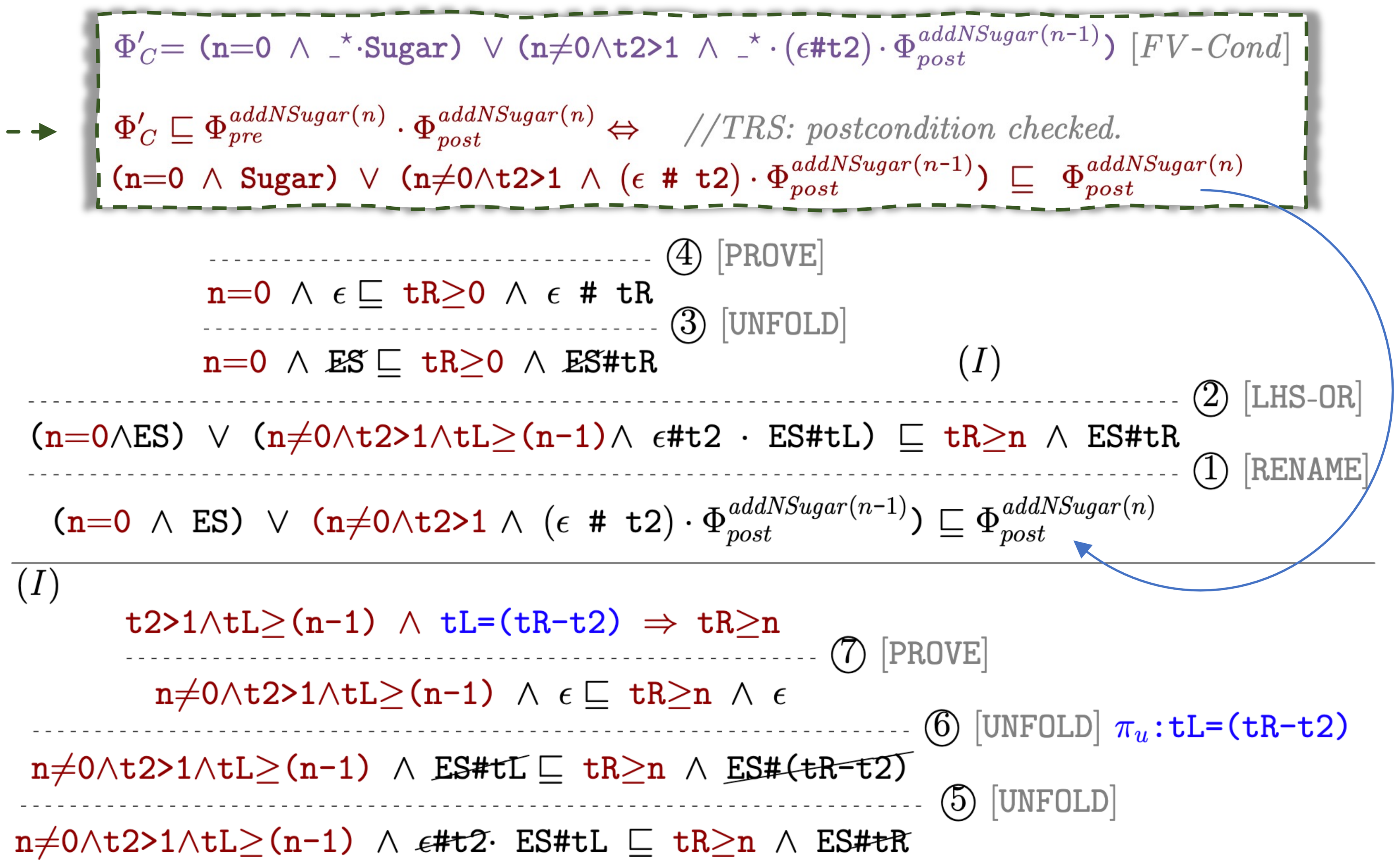
TimEffs draw similarities to Metric Temporal Logic (MTL), derived from LTL, where a set of non-negative real numbers is added to temporal modal operators. Basic operators are:

Φ_{post}	$\Box_I A \equiv (A^\star)\#t$	$\Diamond_I A \equiv (\_^\star \cdot A)\#t$	$\bigcirc_I A \equiv (\_)\#t \cdot A$	$A\mathcal{U}_I B \equiv (A^\star)\#t \cdot B$
Φ_{pre}	$\overleftarrow{\Box}_I A \equiv (A^\star)\#t$	$\overleftarrow{\Diamond}_I A \equiv (A \cdot \_^\star)\#t$	$\ominus_I A \equiv A \cdot ((\_)\#t)$	$A\mathcal{S}_I B \equiv B \cdot ((A^\star)\#t)$

□ - "globally"; ◇ - "finally; ○ - "next"; U – "until", and their past time-reversed versions: $\overleftarrow{\Box}$; $\overleftarrow{\Diamond}$; and ⊖ for "previous"; S for "since".

TimEffs in the precondition, encode past-time temporal specifications. I in MTL is the time interval with concrete upper/lower bounds, whereas in *TimEffs* they can be symbolic bounds, dependent on program inputs.

A Demonstration of the Automated TRS



Limitation of Our TRS:

Our TRS is incomplete, meaning there exist valid inclusions which will be disproved in our system. That is mainly because of insufficient unification in favor of achieving automation.

Scan Me For the Project Repository