Translating Timed I/O Automata to PVS

Hongping Lim, Dilsun Kaynar, Nancy Lynch and Sayan Mitra

Theory of Distributed Systems Group (TDS)

Computer Science and AI Lab, MIT

FORMATS 2005, Uppsala, Sweden

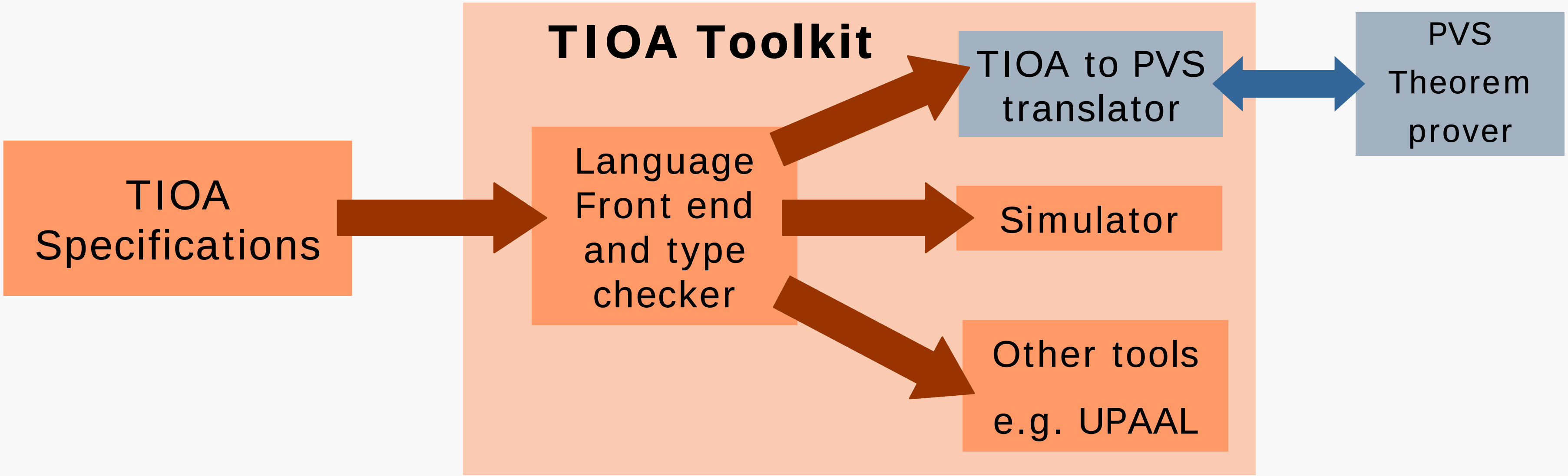
Project Goals

- Develop formal framework for modeling and reasoning about complex, interacting systems
  - Timing-dependent behavior: schedules, deadlines
  - Hybrid behavior: continuous interactions
  - Probabilistic behavior
- Build language for specifying formal models
  - Extend of IOA language
- Build Tool support based on the specification language
  - Interface to Theorem Provers
  - Simulator
  - Model checking

Flavors of I/O automaton models

- Infinite state automata with external interface, abstraction, composition
- Basic IOA (synchronous distributed algorithms)
  - Sequential order of actions, no timing information
  - Interaction through shared actions
- TIOA (timing based systems, hybrid systems || environment)
  - Actions and trajectories
  - Trajectories may describe complex continuous dynamics
  - No continuous interaction between components
- HIOA (embedded systems, software + physical processes)
  - Continuous interactions through shared variables
- PIOA, PTIOA, PHIOA (security protocols, stochastic hybrid systems)
  - Probabilistic transitions, trajectories, …

TIOA Toolkit



Outline

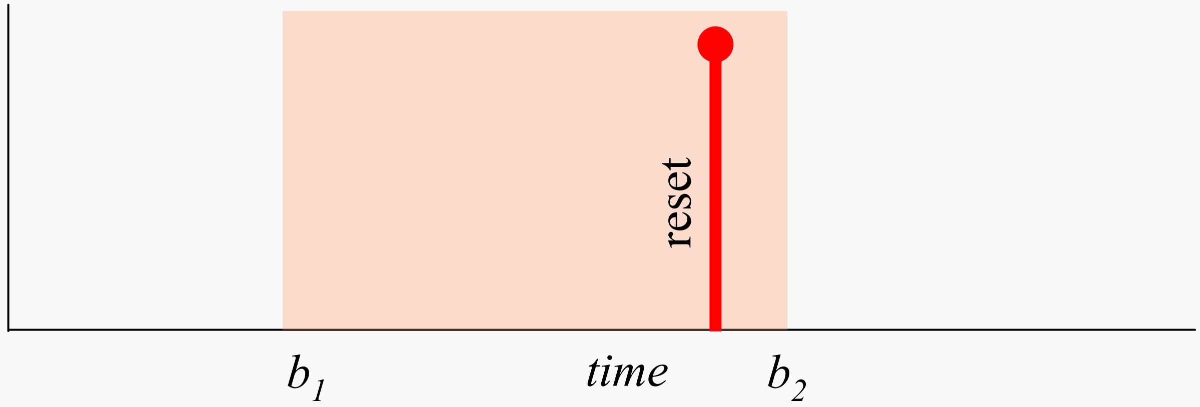
- [x] Introduction
- [ ] TIOA model and language
- [ ] Translation to PVS
- [ ] Examples

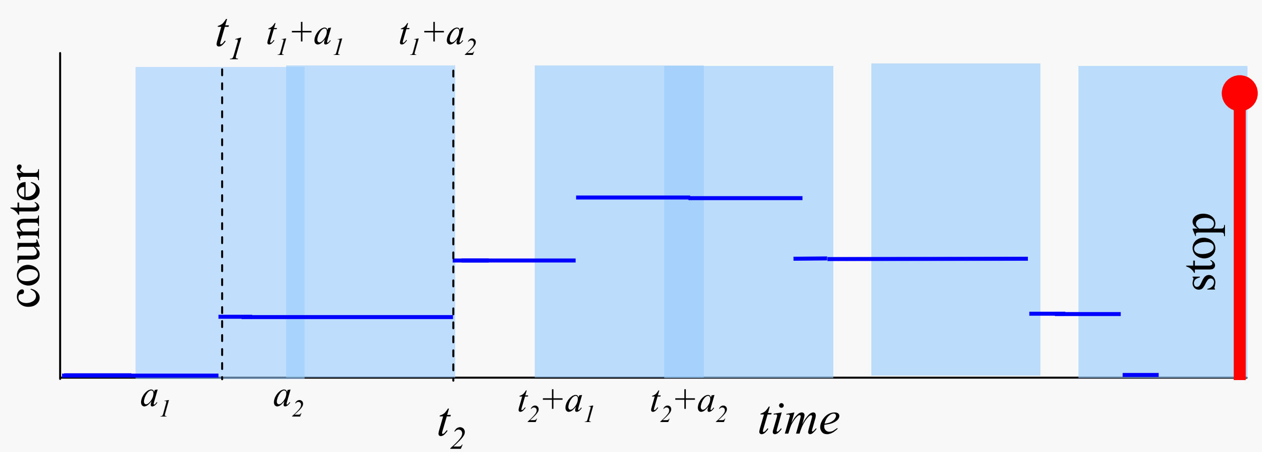
TIOA model

- *Timed I/O Automaton* [Kaynar,Lynch,Segala,Vaandrager]
  - State *variables X* (+ *input/output variables* = HIOA)
  - *Start states Θ*
  - *Actions A*, partitioned into *input, output,* and *internal* subsets
  - *Discrete transitions D,* (**x**,*a*,**x'**)
  - *Trajectories T, τ* maps interval of time to variable values
- Executions $\alpha = \tau_0 a_1 \tau_1 a_2 \tau_2 \ldots a_n \tau_n$
- Invariant properties, proof by induction
- Traces
- Simulation relations: sufficient conditions for $traces(A) \subseteq traces(B)$
- Composition, $A||B$

$$traces(A) \subseteq traces(B) \Rightarrow traces(A \,||\, C) \subseteq traces(B \,||\, C)$$

Two task example





TIOA Specification

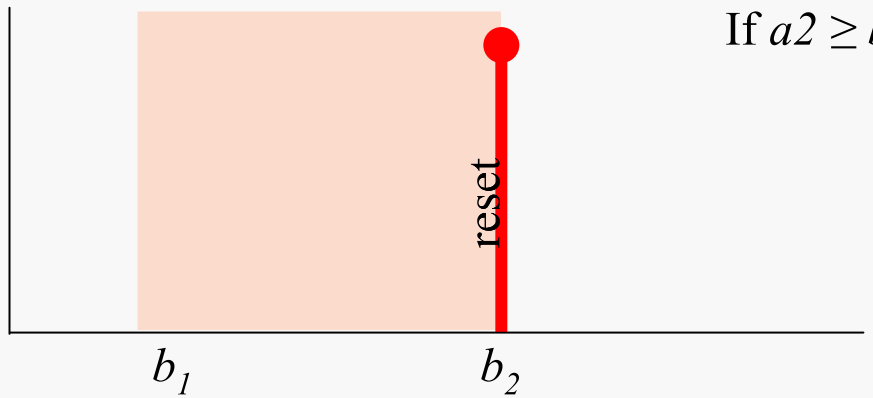
- *Variables*
  - *count, flag,t*
  - *u_reset, l_reset*
  - *u_count,l_count*
- + action
  - Precondition:
    - not *flag* and $t \geq l\_count$
  - Effect:
    - *count++*
    - $l\_count = t + a_1$
    - $u\_count = t + a_2$
- *reset* action
  - Precondition:
    - not *flag* and $t \geq l\_reset$
  - Effect:
    - *flag = true*
- *trajdef*
  - Evolve: $d(t) = 1$
  - Stop when: $t = u\_count$ *or* $t = u\_reset$

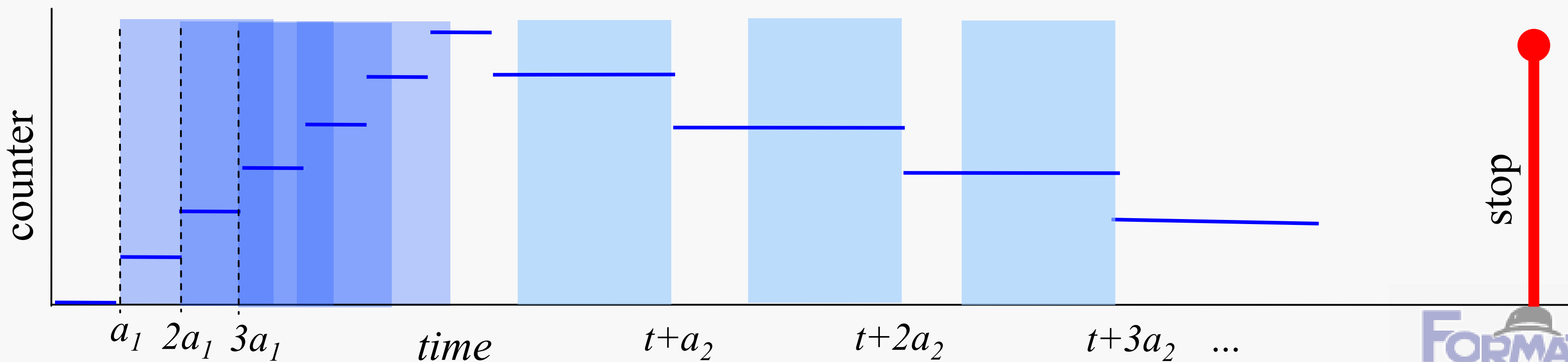
Upper bound for stop

How late can it *stop* ? $b_2 + a_2 + \frac{b_2 a_2}{a_1}$

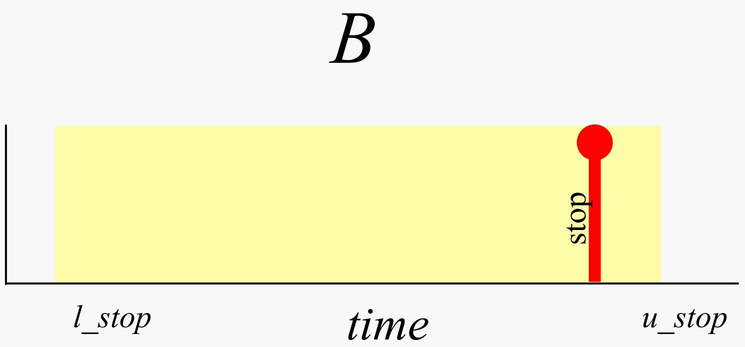
How early ?

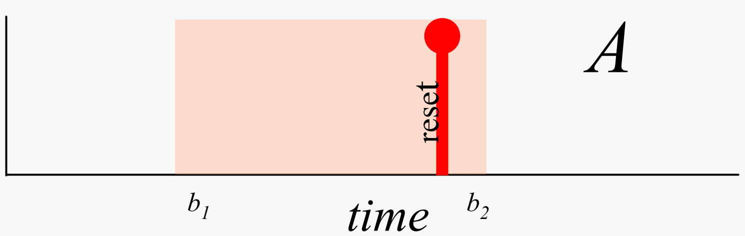
If $a2 \geq b1$ then $a1$ else $\min(b_1, a_1) + \frac{(b_1 - a_1)a_1}{a_2}$

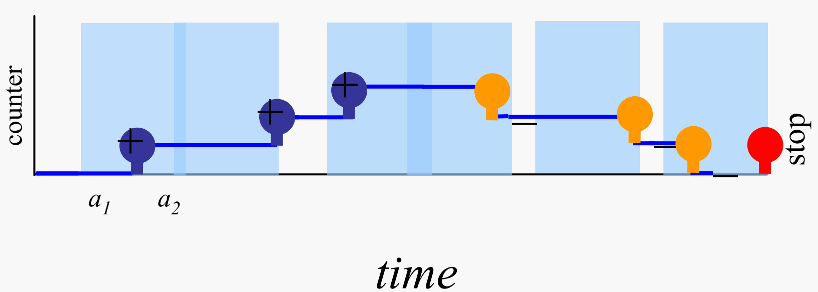




Case studies: Two task example







- Abstract automaton B with one action *stop*
  - $u\_stop = b_2 + a_2 + \frac{b_2 a_2}{a_1}$
  - $l\_stop = \text{if } a2 \geq b1 \text{ then } a1 \text{ else}$
    $$\min(b_1, a_1) + \frac{(b_1 - a_1)a_1}{a_2}$$
- Prove trace inclusion (time bounds for *stop*) with simulation relations
- Prove forward simulation $R \subseteq Q_A \times Q_B$

Simulation Relation

$\sim flag \wedge l\_count \leq u\_reset \rightarrow u\_stop \geq u\_reset + a_2(count + 2 + (u\_reset - l\_count)/a_1)$

$flag \vee l\_count > u\_reset \rightarrow u\_stop \geq u\_count + a2\ count$

$\sim flag \wedge u\_count < l\_reset \rightarrow l\_stop \leq min(l\_reset, l\_count) + a_1(count + (l\_reset - u\_count)/a_2)$

$flag \vee u\_count \geq l\_reset \rightarrow l\_stop \leq l\_count + a_1\ count$

- Requires creativity/insight to come up with the right *R*
- Proof by induction
  1. There are related start states
  2. Every action/trajectory of *A* can be emulated by an execution fragment of *B* with the same trace.
- Use PVS prover [SRI] for proving interactively
- Strategies set up the induction and case analysis automatically
- Nonlinear real inequalities handled by *Field* and *Manip* strategy packages[Muñoz, deVito]

Translation to PVS

- Stylized proofs lead to partial automation [Archer, Mitra 05]
- Proof management (E.g., ABD implementation [Chockler,Lynch,Mitra, Tauber,DISC'05])
- Rechecking proofs after making changes to spec
- Generation of human readable proofs

- Rewrite TIOA specs for the theorem prover! Different language and style.

- Why not specify automata directly in PVS ?
  - TIOA provides structures for natural description of automata
    - Programs for effects as opposed to functions or relations
    - Differential equations and stopping conditions for trajectories
  - Other TIOA tools

States and Actions

TIOA	PVS
States variables →	tuple of variables
Initial state →	predicate on state variables
Actions →	new datatype called *actions*
Preconditions →	predicates on state variables, action parameters, automaton parameters
Effects →	…
Trajdefs →	…

Translating action effects

- TIOA effects are nondeteministic, e.g.,

  plus_something(): **effect** x := x + **choose** [1,5];

- TIOA effects are programs with operational semantics:

  x := x + 5; y := 2.x; …

- We want the PVS effects to be functions:
  - *plus*(k ϵ [1,5]): x' := x + k
  - Substitution:

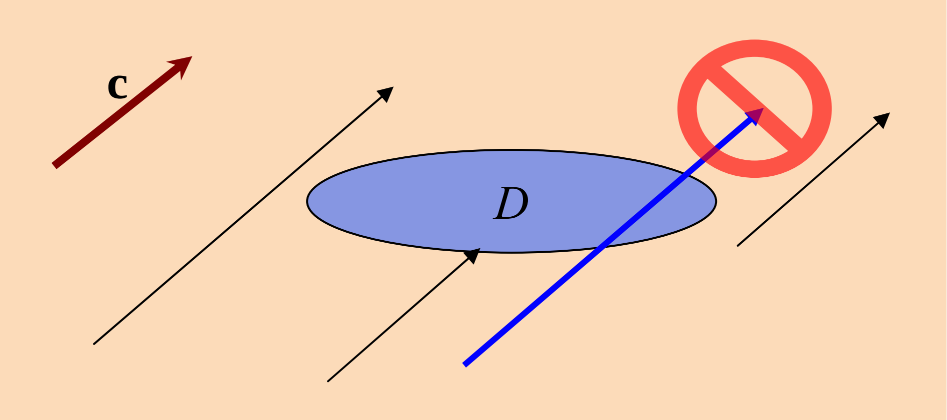
    x' = x + 5; y' = 2 (x + 5)
  - PVS assignments:

    s' = LET s := s WITH [x := x(s) + 5] in

    LET s := s WITH [y := 2 × x(s)] in s

Trajdef → *time_elapse action*

- Trajdef
- **Evolve:** $d(\mathbf{x}) = \mathbf{c}$
- **Stop when:** $\mathbf{x} \in D$



- PVS action
- time_elapse($t{:}R^{\geq 0}$, $\tau{:}\ [0,t] \rightarrow R^2$)
- Enabled if
  - for all $t_1 \in [0,t]$,
    - If $\tau(t_1) \in D$ then $t_1 = t$
    - $\tau(t_1) = \mathbf{x} + \mathbf{c}\, t_1$
- Effect
  - $s' = \tau(t)$

Translating trajectories

Works for general *trajdefs*

- Trajdef
- **Evolve:** $d(\mathbf{x}) = c\mathbf{x}$
- **Stop when:** $\mathbf{x} \in D$

- PVS action
- time_elapse($dt{:}R^{\geq 0}$, $\tau{:}\ [0,t] \rightarrow R^2$)
- Enabled at s If
  - for all $t_1 \in [0,t]$,
    - If $\tau(t_1) \in D$ then $t_1 = t$
    - $\tau(t_1) = \mathbf{x}\, e^{ct1}$
- Effect
  - $s' = \tau(t)$

Case Studies

Fischer's Mutual Exclusion Algorithm

- *N* processes, each go through *try, test, etc.*, to get to *critical*
- Transitions determined by deadlines
- Single TIOA written as the composition of *N* automata
- Safety property: no two processes are *critical* simultaneously

- Automata and properties translated to PVS
- Invariant proved using induction

Ongoing: SATS

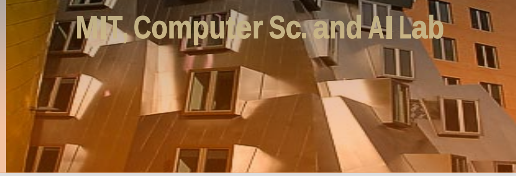
- Small Aircraft Transportation System (SATS) [Muñoz, NASA]
  - Discrete model: airport space partitioned into several logical zones
  - Each zones represented by a queue
  - Transitions represent aircrafts moving from one zone to another
  - Various constraints on transitions
- Safety property: upper bound on the number of aircrafts in each zone

- TIOA description of system
  - Special operators declared in TIOA and defined in PVS, e.g. recursive functions
- Translated to PVS
- Properties written directly in PVS and proved

Ongoing: ABD

- ABD atomic register implementation
  - Read and write quorums for 2 phase reads and writes
  - Partial functions & graphs
- Proof of correctness using an abstract Partial Order automaton
- Simulation proof, ABD implements PO-automaton
- Directly coded in PVS

Conclusion



- Translator is part of TIOA toolkit, implemented in Java
  Available from: http://web.mit.edu/hongping/www/tioa/tioa2pvs-translator/
- Future directions:
  - Extend translator
    - systems with linear dynamics
    - composed TIOAs
  - Develop new PVS strategies
  - New case studies
    - Linear hybrid systems
    - Atomic registers implementations
    - Continuous SATS