Conformance testing of vdta

Hervé Marchand[1], **Omer Nguena Timo**[2] and Antoine Rollet[2]

[1]INRIA Rennes - Bretagne Atlantique, France
[2]LaBRI, Université de Bordeaux - CNRS, France

MTVV 2012, November 12, Palaiseau

Outline

Motivation

- Context: ANR TESTEC (LURPA, I3S, LABRI, INRIA, Dassault Systems, EDF)
- A case study

Specification of a two button machine

- The machine starts (S:=1) as soon as the two buttons are pressed. The delay between the pressure times of the two buttons should be not greater than 1 t.u
- The starting process is initialised if more than 1 t.u had elapsed after a button have being pressed.
- The machine stops (S=0) as soon as one of the two buttons is released.

- Modelling open real-time systems
- The size of the event-based models are often huge
- Theoretical supports and tools (urgency, variable driven models)

Outline

Variable Driven Timed Automata

- Extension of the Djikstra model with dense time clock variables
- Modelling open real-time systems
- Transitions are guarded with constraints on **input variables, output variables** and **clocks variables**
- Transitions are fired as soon as their guards are satisfied

Variable Driven Timed Automata

Definition (VDTA [NTR10])

$$\mathcal{A} = \langle \mathbf{L}, \ell^{\mathbf{0}}, \mathbf{I}, \mathbf{O}, \mathbf{X}, \mathbf{Aff}, \mathbf{\Delta}_{\mathcal{A}} \rangle$$

$$\mathbf{\Delta}_{\mathcal{A}} \subseteq L \times \mathcal{G}(I) \times \mathcal{G}(O) \times \mathcal{G}(X) \times Aff(O) \times 2^X \times L$$

Settings

1. The disjoint sets of **input, output** and **clock** variables I, O and X
2. Operations on variables
   - Aff: assignments of constant to input/output variables.
   - Reset of the clocks
3. $\mathcal{G}(I)$, $\mathcal{G}(O)$ and $\mathcal{G}(X)$
   $G ::= i \bowtie c \mid G \wedge G \mid G \vee G$
   where $c \in Dom(i)$

Semantics

- A **state** is a tuple (ℓ, i, o, x)
- A transition $\ell \xrightarrow{g_I, g_O, g_X, a, \mathcal{X}} \ell'$ is passed as soon as g_I, g_O and g_X are satisfied.
- Three kinds of **moves** from states:
  - **output update**: by passing transitions
  - **input update**, when no transition can be passed
  - **Time elapsing**, when no transition can be passed

Variable Driven Timed Automata

Definition (VDTA [NTR10])

$$\mathcal{A} = \langle \mathbf{L}, \ell^{\mathbf{0}}, \mathbf{I}, \mathbf{O}, \mathbf{X}, \mathbf{Aff}, \mathbf{\Delta}_{\mathcal{A}} \rangle$$

$$\mathbf{\Delta}_{\mathcal{A}} \subseteq L \times \mathcal{G}(I) \times \mathcal{G}(O) \times \mathcal{G}(X) \times Aff(O) \times 2^X \times L$$

Settings

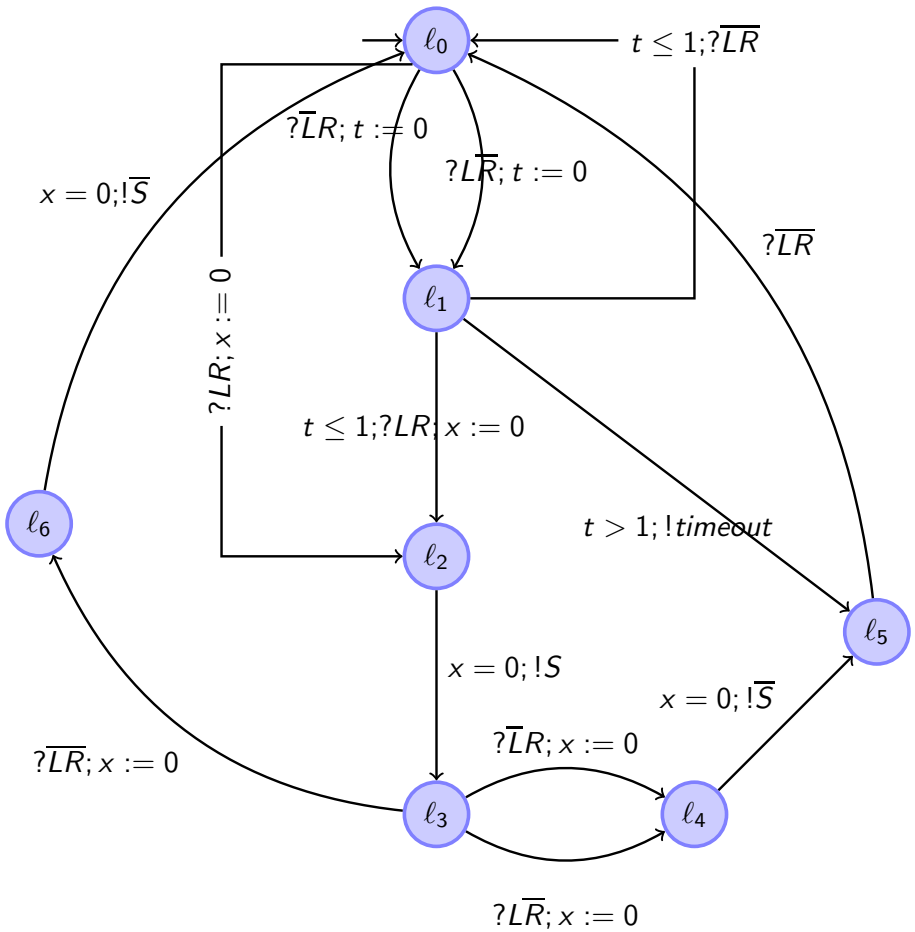
1. The disjoint sets of **input, output** and **clock** variables I, O and X
2. Operations on variables
   - Aff: assignments of constant to input/output variables.
   - Reset of the clocks
3. $\mathcal{G}(I)$, $\mathcal{G}(O)$ and $\mathcal{G}(X)$
   $G ::= i \bowtie c \mid G \wedge G \mid G \vee G$
   where $c \in Dom(i)$

Semantics

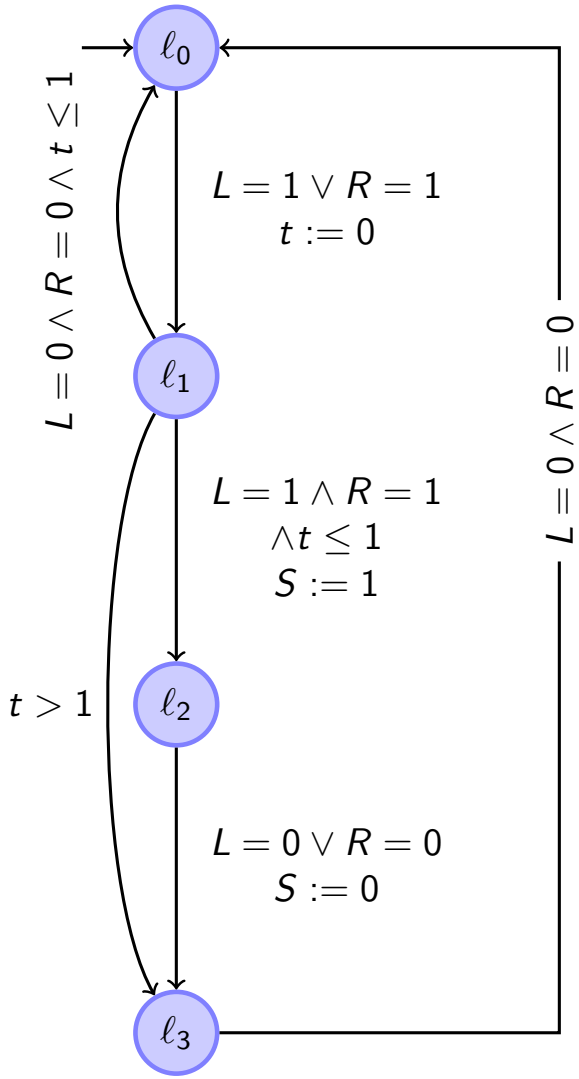
- A **state** is a tuple (ℓ, i, o, x)
- A transition $\ell \xrightarrow{g_I, g_O, g_X, a, \mathcal{X}} \ell'$ is passed as soon as g_I, g_O and g_X are satisfied.
- Three kinds of **moves** from states:
  - **output update**: by passing transitions
  - **input update**, when no transition can be passed
  - **Time elapsing**, when no transition can be passed

Modelling the two buttons machine

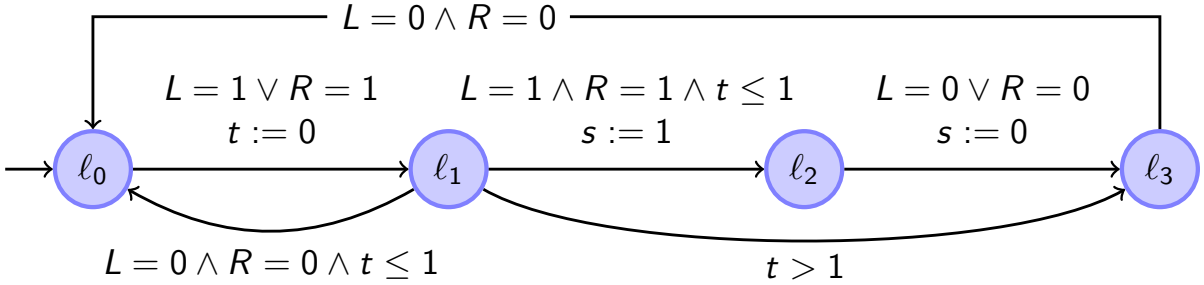
Event-based approach:



VDTA model:

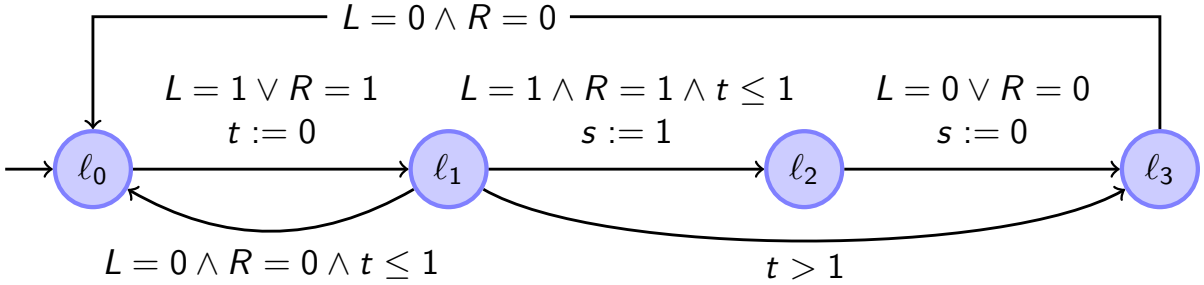


Run, Stable run, Operating trace



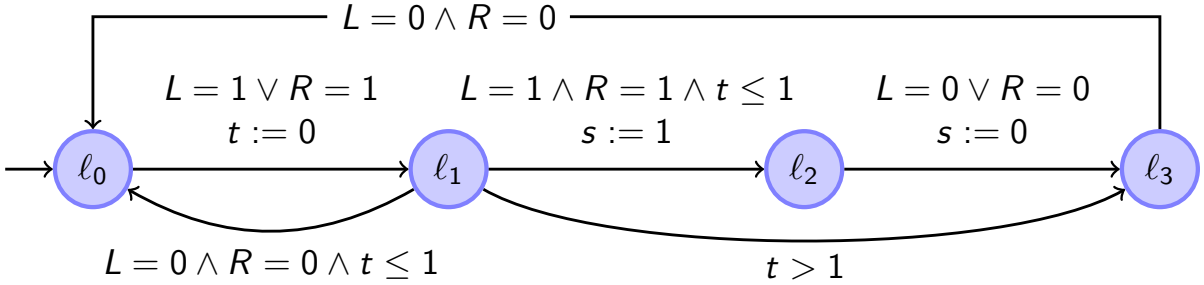
- A **run** is a sequence of moves.
- A **stable state** is such that no transition can be passed from it.
  e.g: $(\ell_0, (0,0), 0, 0)) \xrightarrow{L:=1} (\ell_0, (1,0), 0, 0)) \xrightarrow{Id_O} (\ell_1, (1,0), 0, 0)) \xrightarrow{0.3}$
  $(\ell_1, (1,0), 0, 0.3)) \xrightarrow{R:=1} (\ell_1, (1,1), 0, 0.3)) \xrightarrow{s:=1} (\ell_2, (1,1,1,0.3))$
- A **maximal run** is **stable** if it ends in a **stable state**
  e.g: $(\ell_0, (0,0,0,0)) \xrightarrow{L:=1;R:=1} (\ell_0, (1,1,0,0))$ is not stable.
  $(\ell_0, (0,0,0,0)) \xrightarrow{L:=1;R:=1} (\ell_0, (1,1,0,0)) \xrightarrow{Id_O} (\ell_1, (1,1,0,0))$ is not stable.
- A **Trace** is a sequence of action. e.g: $L := 1; 0.3; R := 1; S := 1$

Run, Stable run, Operating trace



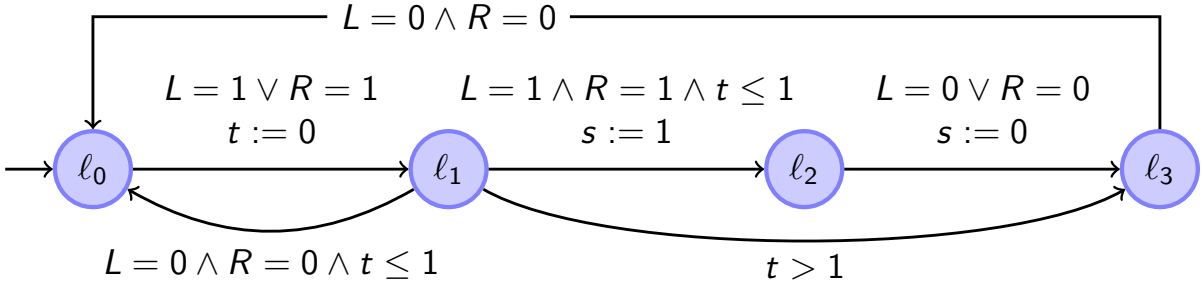
- A **run** is a sequence of moves.
- A **stable state** is such that no transition can be passed from it.
  e.g: $(\ell_0,(0,0),0,0)) \xrightarrow{L:=1} (\ell_0,(1,0),0,0)) \xrightarrow{Id_O} (\ell_1,(1,0),0,0)) \xrightarrow{0.3}$
  $(\ell_1,(1,0),0,0.3)) \xrightarrow{R:=1} (\ell_1,(1,1),0,0.3)) \xrightarrow{s:=1} (\ell_2,(1,1,1,0.3))$
- A **maximal run** is **stable** if it ends in a **stable state**
  e.g: $(\ell_0,(0,0,0,0)) \xrightarrow{L:=1;R:=1} (\ell_0,(1,1,0,0))$ is not stable.
  $(\ell_0,(0,0,0,0)) \xrightarrow{L:=1;R:=1} (\ell_0,(1,1,0,0)) \xrightarrow{Id_O} (\ell_1,(1,1,0,0))$ is not stable.
- A **Trace** is a sequence of action. e.g: $L := 1; 0.3; R := 1; S := 1$

Run, Stable run, Operating trace



- A **run** is a sequence of moves.
- A **stable state** is such that no transition can be passed from it.
  e.g: $(\ell_0,(0,0),0,0)) \xrightarrow{L:=1} (\ell_0,(1,0),0,0)) \xrightarrow{Id_O} (\ell_1,(1,0),0,0)) \xrightarrow{0.3}$
  $(\ell_1,(1,0),0,0.3)) \xrightarrow{R:=1} (\ell_1,(1,1),0,0.3)) \xrightarrow{s:=1} (\ell_2,(1,1,1,0.3))$
- A **maximal run** is **stable** if it ends in a **stable state**
  e.g: $(\ell_0,(0,0,0,0)) \xrightarrow{L:=1;R:=1} (\ell_0,(1,1,0,0))$ is not stable.
  $(\ell_0,(0,0,0,0)) \xrightarrow{L:=1;R:=1} (\ell_0,(1,1,0,0)) \xrightarrow{Id_O} (\ell_1,(1,1,0,0))$ is not stable.
- A **Trace** is a sequence of action. e.g: $L := 1; 0.3; R := 1; S := 1$

Run, Stable run, Operating trace



- A **run** is a sequence of moves.
- A **stable state** is such that no transition can be passed from it.
  e.g: $(\ell_0,(0,0),0,0)) \xrightarrow{L:=1} (\ell_0,(1,0),0,0)) \xrightarrow{Id_O} (\ell_1,(1,0),0,0)) \xrightarrow{0.3}$
  $(\ell_1,(1,0),0,0.3)) \xrightarrow{R:=1} (\ell_1,(1,1),0,0.3)) \xrightarrow{s:=1} (\ell_2,(1,1,1,0.3))$
- A **maximal run** is **stable** if it ends in a **stable state**
  e.g: $(\ell_0,(0,0,0,0)) \xrightarrow{L:=1;R:=1} (\ell_0,(1,1,0,0))$ is not stable.
  $(\ell_0,(0,0,0,0)) \xrightarrow{L:=1;R:=1} (\ell_0,(1,1,0,0)) \xrightarrow{Id_O} (\ell_1,(1,1,0,0))$ is not stable.
- A **Trace** is a sequence of action. e.g: $L := 1; 0.3; R := 1; S := 1$

Deterministic VDTA

Deterministic VDTA

$\mathcal{A}$ is *deterministic* if G_0 is satisfied by at most one valuation (i^0, o^0); and if $\ell \xrightarrow{G_1, A_1, \mathcal{X}_1} \ell_1$ and $\ell \xrightarrow{G_2, A_2, \mathcal{X}_2} \ell_2$ then $G_1 \cap G_2$ is not satisfiable.

Outline

Conformance Testing relation

Test

- Input: An implementation *Imp* and a Specification $\mathcal{A}$
- Problem: Does *Imp* conform to $\mathcal{A}$?

Imp conforms to $\mathcal{A}$ provided that *Imp* produces valid and expected outputs.

We assume that *Imp* and $\mathcal{A}$ are deterministic.

Definition (Conformance relation [NTR10, NTMR10])

$$(Imp \textbf{ tvco } \mathcal{A}) \text{ iff } \mathcal{S}_{Imp} \text{ tvco } \mathcal{S}_{\mathcal{A}}$$

and

$$\mathcal{S}_{Imp} \text{ tvco } \mathcal{S}_{\mathcal{A}} \text{ iff } \forall \sigma \in \text{ObsTr}(s_0^{\mathcal{A}}), \text{out}(s_0^{Imp} \text{ Safter } \sigma) \subseteq \text{out}(s_0^{\mathcal{A}} \text{ Safter } \sigma)$$

Stimuli, and Observation graph, Observed traces

- **Stimuli**: actions performed by the tester: input updates or delays. e.g: $L := 1,\ 0.3$
- After a stimuli the implementation runs until it reaches a stable state.
  e.g: $(\ell_1,(1,0),0,0.3)) \stackrel{R:=1}{\Longrightarrow} (\ell_2,(1,1),1,0.3))$ and $(\ell_1,(1,0),0,0.3)) \stackrel{0.9}{\Longrightarrow} (\ell_3,(1,0),0,1.2))$
- **Observation graph:** $Obs(\mathcal{A}) = (S, s^0, A(I) \cup \mathbb{R}_+, \Longrightarrow)$ is inductively generated from the semantics of $\mathcal{A}$ by starting from s^0 (that is supposed to be stable) and by using $\Longrightarrow$.
  e.g: $(\ell_0,(0,0),0,0)) \xrightarrow{L:=1} (\ell_0,(1,0),0,0)) \xrightarrow{Id_O} (\ell_1,(1,0),0,0)) \xrightarrow{0.3} (\ell_1,(1,0),0,0.3)) \xrightarrow{R:=1} (\ell_1,(1,1),0,0.3)) \xrightarrow{s:=1} (\ell_2,(1,1),1,0.3))$
  $(\ell_0,(0,0),0,0)) \stackrel{L:=1}{\Longrightarrow} (\ell_1,(1,0),0,0)) \stackrel{0.3}{\Longrightarrow} (\ell_1,(1,0),0,0.3)) \stackrel{R:=1}{\Longrightarrow} (\ell_2,(1,1),1,0.3))$
- $ObsRun(\mathcal{A}) = Run(Obs(\mathcal{A}))$, $ObsTr(\mathcal{A}) = Tr(Obs(\mathcal{A}))$
  e.g: $\alpha = L := 1; 0.3; R := 1$
- s Safter $\alpha = \{s' \mid s \stackrel{\alpha}{\Longrightarrow} s'\}$ and $\mathcal{A}$ *Safter* $\alpha = s^0$ *Safter* α.
- The outputs are fetched only when the implementation is in a stable state. e.g: $out(\ell_1,(1,0),0,0.3)) = 0$ and $out(\ell_2,(1,1),1,0.3)) = 1$

Stimuli, and Observation graph, Observed traces

- **Stimuli**: actions performed by the tester: input updates or delays.
  e.g: $L := 1,\ 0.3$
- After a stimuli the implementation runs until it reaches a stable state.
  e.g: $(\ell_1, (1,0), 0, 0.3)) \stackrel{R:=1}{\Longrightarrow} (\ell_2, (1,1), 1, 0.3))$ and
  $(\ell_1, (1,0), 0, 0.3)) \stackrel{0.9}{\Longrightarrow} (\ell_3, (1,0), 0, 1.2))$
- **Observation graph:** $Obs(\mathcal{A}) = (S, s^0, A(I) \cup \mathbb{R}_+, \Longrightarrow)$ is inductively generated from the semantics of $\mathcal{A}$ by starting from s^0 (that is supposed to be stable) and by using $\Longrightarrow$.
  e.g: $(\ell_0, (0,0), 0, 0)) \xrightarrow{L:=1} (\ell_0, (1,0), 0, 0)) \xrightarrow{Id_O} (\ell_1, (1,0), 0, 0)) \xrightarrow{0.3}$
  $(\ell_1, (1,0), 0, 0.3)) \xrightarrow{R:=1} (\ell_1, (1,1), 0, 0.3)) \xrightarrow{s:=1} (\ell_2, (1,1), 1, 0.3))$
  $(\ell_0, (0,0), 0, 0)) \stackrel{L:=1}{\Longrightarrow} (\ell_1, (1,0), 0, 0)) \stackrel{0.3}{\Longrightarrow} (\ell_1, (1,0), 0, 0.3)) \stackrel{R:=1}{\Longrightarrow}$
  $(\ell_2, (1,1), 1, 0.3))$
- $ObsRun(\mathcal{A}) = Run(Obs(\mathcal{A}))$, $ObsTr(\mathcal{A}) = Tr(Obs(\mathcal{A}))$
  e.g: $\alpha = L := 1; 0.3; R := 1$
- s Safter $\alpha = \{s' \mid s \stackrel{\alpha}{\Longrightarrow} s'\}$ and $\mathcal{A}$ *Safter* $\alpha = s^0$ *Safter* α.
- The outputs are fetched only when the implementation is in a stable state. e.g: $out(\ell_1, (1,0), 0, 0.3)) = 0$ and $out(\ell_2, (1,1), 1, 0.3)) = 1$

Stimuli, and Observation graph, Observed traces

- **Stimuli**: actions performed by the tester: input updates or delays.
  e.g: $L := 1,\ 0.3$
- After a stimuli the implementation runs until it reaches a stable state.
  e.g: $(\ell_1, (1,0), 0, 0.3)) \stackrel{R:=1}{\Longrightarrow} (\ell_2, (1,1), 1, 0.3))$ and
  $(\ell_1, (1,0), 0, 0.3)) \stackrel{0.9}{\Longrightarrow} (\ell_3, (1,0), 0, 1.2))$
- **Observation graph:** $Obs(\mathcal{A}) = (S, s^0, A(I) \cup \mathbb{R}_+, \Longrightarrow)$ is inductively generated from the semantics of $\mathcal{A}$ by starting from s^0 (that is supposed to be stable) and by using $\Longrightarrow$.
  e.g: $(\ell_0, (0,0), 0, 0)) \xrightarrow{L:=1} (\ell_0, (1,0), 0, 0)) \xrightarrow{Id_O} (\ell_1, (1,0), 0, 0)) \xrightarrow{0.3}$
  $(\ell_1, (1,0), 0, 0.3)) \xrightarrow{R:=1} (\ell_1, (1,1), 0, 0.3)) \xrightarrow{s:=1} (\ell_2, (1,1), 1, 0.3))$
  $(\ell_0, (0,0), 0, 0)) \stackrel{L:=1}{\Longrightarrow} (\ell_1, (1,0), 0, 0)) \stackrel{0.3}{\Longrightarrow} (\ell_1, (1,0), 0, 0.3)) \stackrel{R:=1}{\Longrightarrow}$
  $(\ell_2, (1,1), 1, 0.3))$
- $ObsRun(\mathcal{A}) = Run(Obs(\mathcal{A}))$, $ObsTr(\mathcal{A}) = Tr(Obs(\mathcal{A}))$
  e.g: $\alpha = L := 1; 0.3; R := 1$
- s Safter $\alpha = \{s' \mid s \stackrel{\alpha}{\Longrightarrow} s'\}$ and $\mathcal{A}$ *Safter* $\alpha = s^0$ *Safter* α.
- The outputs are fetched only when the implementation is in a stable state. e.g: $out(\ell_1, (1,0), 0, 0.3)) = 0$ and $out(\ell_2, (1,1), 1, 0.3)) = 1$

Stimuli, and Observation graph, Observed traces

- **Stimuli**: actions performed by the tester: input updates or delays.
  e.g: $L := 1,\ 0.3$
- After a stimuli the implementation runs until it reaches a stable state.
  e.g: $(\ell_1, (1,0), 0, 0.3)) \stackrel{R:=1}{\Longrightarrow} (\ell_2, (1,1), 1, 0.3))$ and
  $(\ell_1, (1,0), 0, 0.3)) \stackrel{0.9}{\Longrightarrow} (\ell_3, (1,0), 0, 1.2))$
- **Observation graph:** $Obs(\mathcal{A}) = (S, s^0, A(I) \cup \mathbb{R}_+, \Longrightarrow)$ is inductively generated from the semantics of $\mathcal{A}$ by starting from s^0 (that is supposed to be stable) and by using $\Longrightarrow$.
  e.g: $(\ell_0, (0,0), 0, 0)) \xrightarrow{L:=1} (\ell_0, (1,0), 0, 0)) \xrightarrow{Id_O} (\ell_1, (1,0), 0, 0)) \xrightarrow{0.3}$
  $(\ell_1, (1,0), 0, 0.3)) \xrightarrow{R:=1} (\ell_1, (1,1), 0, 0.3)) \xrightarrow{s:=1} (\ell_2, (1,1), 1, 0.3))$
  $(\ell_0, (0,0), 0, 0)) \stackrel{L:=1}{\Longrightarrow} (\ell_1, (1,0), 0, 0)) \stackrel{0.3}{\Longrightarrow} (\ell_1, (1,0), 0, 0.3)) \stackrel{R:=1}{\Longrightarrow}$
  $(\ell_2, (1,1), 1, 0.3))$
- $ObsRun(\mathcal{A}) = Run(Obs(\mathcal{A}))$, $ObsTr(\mathcal{A}) = Tr(Obs(\mathcal{A}))$
  e.g: $\alpha = L := 1; 0.3; R := 1$
- s Safter $\alpha = \{s' \mid s \stackrel{\alpha}{\Longrightarrow} s'\}$ and $\mathcal{A}$ *Safter* $\alpha = s^0$ *Safter* α.
- The outputs are fetched only when the implementation is in a stable state. e.g: $\mathrm{out}(\ell_1, (1,0), 0, 0.3)) = 0$ and $\mathrm{out}(\ell_2, (1,1), 1, 0.3)) = 1$

Stimuli, and Observation graph, Observed traces

- **Stimuli**: actions performed by the tester: input updates or delays.
  e.g: $L := 1,\ 0.3$
- After a stimuli the implementation runs until it reaches a stable state.
  e.g: $(\ell_1, (1,0), 0, 0.3)) \stackrel{R:=1}{\Longrightarrow} (\ell_2, (1,1), 1, 0.3))$ and
  $(\ell_1, (1,0), 0, 0.3)) \stackrel{0.9}{\Longrightarrow} (\ell_3, (1,0), 0, 1.2))$
- **Observation graph:** $Obs(\mathcal{A}) = (S, s^0, A(I) \cup \mathbb{R}_+, \Longrightarrow)$ is inductively generated from the semantics of $\mathcal{A}$ by starting from s^0 (that is supposed to be stable) and by using $\Longrightarrow$.
  e.g: $(\ell_0, (0,0), 0, 0)) \xrightarrow{L:=1} (\ell_0, (1,0), 0, 0)) \xrightarrow{Id_O} (\ell_1, (1,0), 0, 0)) \xrightarrow{0.3}$
  $(\ell_1, (1,0), 0, 0.3)) \xrightarrow{R:=1} (\ell_1, (1,1), 0, 0.3)) \xrightarrow{s:=1} (\ell_2, (1,1), 1, 0.3))$
  $(\ell_0, (0,0), 0, 0)) \stackrel{L:=1}{\Longrightarrow} (\ell_1, (1,0), 0, 0)) \stackrel{0.3}{\Longrightarrow} (\ell_1, (1,0), 0, 0.3)) \stackrel{R:=1}{\Longrightarrow}$
  $(\ell_2, (1,1), 1, 0.3))$
- $ObsRun(\mathcal{A}) = Run(Obs(\mathcal{A}))$, $ObsTr(\mathcal{A}) = Tr(Obs(\mathcal{A}))$
  e.g: $\alpha = L := 1; 0.3; R := 1$
- s Safter $\alpha = \{s' \mid s \stackrel{\alpha}{\Longrightarrow} s'\}$ and $\mathcal{A}$ *Safter* $\alpha = s^0$ *Safter* α.
- The outputs are fetched only when the implementation is in a stable state. e.g: $\mathrm{out}(\ell_1, (1,0), 0, 0.3)) = 0$ and $\mathrm{out}(\ell_2, (1,1), 1, 0.3)) = 1$

Stimuli, and Observation graph, Observed traces

- **Stimuli**: actions performed by the tester: input updates or delays.
  e.g: $L := 1,\ 0.3$
- After a stimuli the implementation runs until it reaches a stable state.
  e.g: $(\ell_1,(1,0),0,0.3)) \overset{R:=1}{\Longrightarrow} (\ell_2,(1,1),1,0.3))$ and
  $(\ell_1,(1,0),0,0.3)) \overset{0.9}{\Longrightarrow} (\ell_3,(1,0),0,1.2))$
- **Observation graph:** $Obs(\mathcal{A}) = (S, s^0, A(I) \cup \mathbb{R}_+, \Longrightarrow)$ is inductively generated from the semantics of $\mathcal{A}$ by starting from s^0 (that is supposed to be stable) and by using $\Longrightarrow$.
  e.g: $(\ell_0,(0,0),0,0)) \xrightarrow{L:=1} (\ell_0,(1,0),0,0)) \xrightarrow{Id_O} (\ell_1,(1,0),0,0)) \xrightarrow{0.3}$
  $(\ell_1,(1,0),0,0.3)) \xrightarrow{R:=1} (\ell_1,(1,1),0,0.3)) \xrightarrow{s:=1} (\ell_2,(1,1),1,0.3))$
  $(\ell_0,(0,0),0,0)) \overset{L:=1}{\Longrightarrow} (\ell_1,(1,0),0,0)) \overset{0.3}{\Longrightarrow} (\ell_1,(1,0),0,0.3)) \overset{R:=1}{\Longrightarrow}$
  $(\ell_2,(1,1),1,0.3))$
- $ObsRun(\mathcal{A}) = Run(Obs(\mathcal{A}))$, $ObsTr(\mathcal{A}) = Tr(Obs(\mathcal{A}))$
  e.g: $\alpha = L := 1; 0.3; R := 1$
- s Safter $\alpha = \{s' \mid s \overset{\alpha}{\Longrightarrow} s'\}$ and $\mathcal{A}$ *Safter* $\alpha = s^0$ *Safter* α.
- The outputs are fetched only when the implementation is in a stable state. e.g: $\mathrm{out}(\ell_1,(1,0),0,0.3)) = 0$ and $\mathrm{out}(\ell_2,(1,1),1,0.3)) = 1$

Comformance testing relation

Definition (Conformance relation [NTR10, NTMR10])

$$(\mathit{Imp} \textbf{ tvco } \mathcal{A}) \text{ iff } \mathcal{S}_{\mathit{Imp}} \text{ tvco } \mathcal{S}_{\mathcal{A}}$$

and

$$\mathcal{S}_{\mathit{Imp}} \text{ tvco } \mathcal{S}_{\mathcal{A}} \text{ iff } \forall \sigma \in \text{ObsTr}(s_0^{\mathcal{A}}), \text{out}(s_0^{\mathit{Imp}} \text{ Safter } \sigma) \subseteq \text{out}(s_0^{\mathcal{A}} \text{ Safter } \sigma)$$

Test case and test suite

- A **test case** can be represented with a finite and deterministic labelled transition system with two special *terminal* states *pass* and *fail*.
- A **test suite** is a set of test cases.

Definition

An implementation **passes** a test case if and only if the execution of every test leads to a final state of the test case labelled with the verdict *pass*.

Testing strategies

Testing Strategies:

- Without test purposes: full examination of the implementation w.r.t the specification
- With test purposes
  Specify the behaviours/properties we want to test
  Select test cases that allow to test the behaviours/properties

Testing strategies

Testing Strategies:

- Without test purposes: full examination of the implementation w.r.t the specification
- With test purposes
  Specify the behaviours/properties we want to test
  Select test cases that allow to test the behaviours/properties

Outline

Algorithm Overview

An on-the-fly testing experiment is a game between the tester and the implementation. The purpose of the game is returning a verdict PASS or FAIL.

Require: $\mathcal{A} = \langle \mathbf{L}, \ell^{\mathbf{0}}, \mathbf{I}, \mathbf{O}, \mathbf{X}, \mathbf{Aff}, \mathbf{\Delta}_{\mathcal{A}} \rangle$
Require: IUT

```
1: Z = {(l^0, i^0, o^0, x^0)}
2: while TRUE do
3:    operation = switch-randomly (assignment, delay, pass)
4:    if operation == assignment then
5:       Do 1...
6:    else if operation == delay then
7:       Do 2...
8:    else if operation == pass then
9:       RETURN pass
10:   end if
11: end while
```

Algorithm I

Require: $\mathcal{A} = \langle \mathbf{L}, \ell^{\mathbf{0}}, \mathbf{I}, \mathbf{O}, \mathbf{X}, \mathbf{Aff}, \mathbf{\Delta}_{\mathcal{A}} \rangle$
Require: IUT
1: $Z = \{(l^0, i^0, o^0, x^0)\}$
2: **while** TRUE **do**
3: operation = switch-randomly (assignment, delay, pass)
4: **if** operation == assignment **then**
5: Randomly choose an assignment $a \in Aff(l)$
6: $Z \leftarrow Z \,\mathrm{SAfter}\, a$
7: execute a and let o be the observed output.
8: **if** $o \notin Out(Z)$ **then**
9: RETURN fail
10: **end if**
11: **else if** operation == delay **then**
12: Randomly-choose a delay d
13: Sleep for d T.U and wake up on an observation of an access to output variables

Algorithm II

Let o be the observed output.
if o occurs at d' with $d' \leq d$ **then**
$Z \leftarrow Z \,\mathrm{Safter}\, d'$
if $o \notin out(Z)$ **then**
RETURN fail
end if
else
$Z \leftarrow Z \; After \; d$
end if
else if operation == pass **then**
RETURN pass
end if
end while

Completeness - Definition

Definition

Let $\mathcal{A}$ be a specification and T be a test suite; then for the relation **tvco**:

- T is **complete** means $\forall Imp : \mathrm{IUT}$, Imp **tvco** $\mathcal{A}$ iff Imp **passes** T.
- T is **sound** means $\forall Imp : \mathrm{IUT}$, Imp **tvco** $\mathcal{A}$ implies Imp **passes** T.
- T is **exhaustive** means $\forall Imp : \mathrm{IUT}$, Imp **tvco** $\mathcal{A}$ if Imp **passes** T.

Completeness - Results

Proposition

The testing algorithm is complete for $\mathcal{A}$ with respect to all possible test cases and the relation **tvco**.

Completeness - sketch of proof

- **Soundness:**
  - Assume that *Imp* **tvco** $\mathcal{A}$ (h1) and *Imp* **not passes** T (h2)
  - (h2) implies the algorithm returns the verdict fail which in turn implies the existence of an unexpected output. That is in contradiction with (h1)
- **Exhaustivity:**
  - Assume that *Imp* passes T (h1) and **not** *Imp* **tvco** $\mathcal{A}$ (h2)
  - From (h2), there exits a sequence σ such that $out(Imp\ After\ \sigma) \not\subseteq out(\mathcal{A}\ After\ \sigma)$.
  - We can construct a test case that contains the state *fail*, getting a contradiction with (h1).

Outline

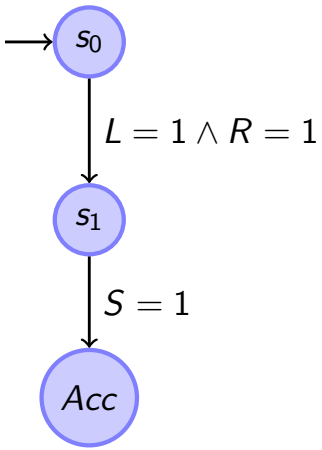
Test Purpose

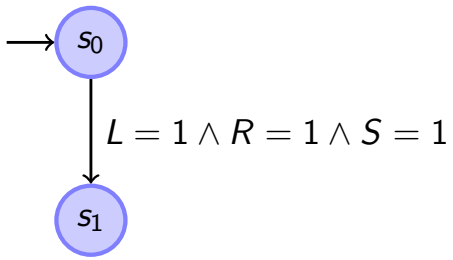
Definition (Test purposes [NTMR10])

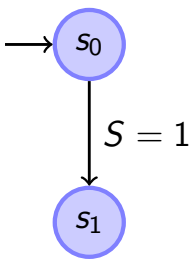
A *test purpose* TP of a specification $\mathcal{A} = \langle L, X, I, O, l^0, G^0, \Delta_{\mathcal{A}} \rangle$ is a deterministic VDTA $\langle S, X \cup X', I, O, s^0, G^0, \Delta_{TP} \rangle$ such that:

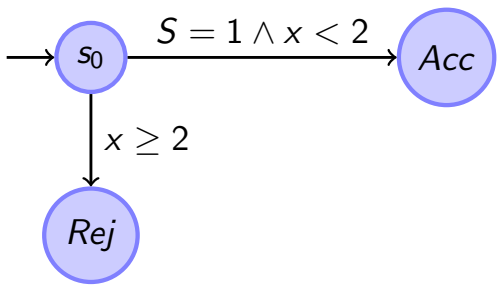
- TP has a special trap location $Acc_{TP} \in S$ (and eventually a location Rej_{TP}),
- I, O and X are respectively the input, output and clock variables of the specification;
- the initial condition $G^0 \in \mathcal{G}(I, O)$ is the one of $\mathcal{A}$;
- TP has private clocks in X' with h $X' \cap X = \emptyset$
- TP should not change input/output variables of the specification.
- TP observes all the variables of the specification.

Test Purpose - Examples

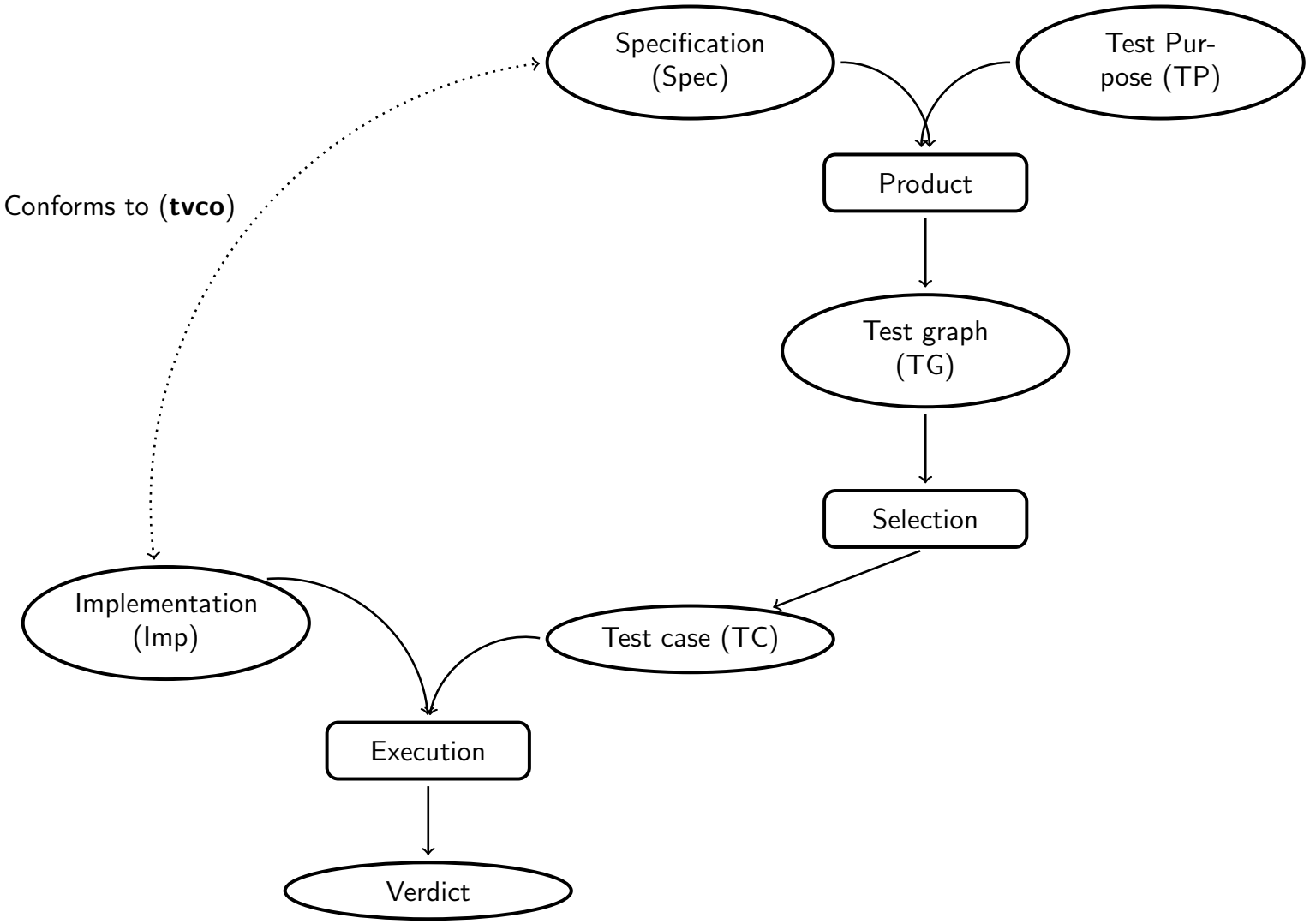








Test selection: An overview



Test case selection

Selection

- Input: a test graph with *Accept* and eventually *Reject*.
- Result: A set of test cases that allow to **reach** *Accept*.

The selection algorithm computes the co-reachability analysis of *Accept*

1. We use abstract constructs (region [NTMR10] or zones [NTR11]) for building a finite representation of the test graph.
   States of the test graph: $((\ell, s), G_I, G_O, G_X)$
2. We compute the least fixpoint of a predecessor operator.
   Take care of the "as soon as possible" semantics for transitions.
   Transitions of the reachability Graph:
   $((\ell, s), G_I, G_O, G_X) \rightarrow ((\ell', s'), G'_I, G'_O, G'_X)$ asserts that from (ℓ', s') is reachable from (ℓ, s) provided that G_I, G_O, G_X are satisfied.

Test case selection

Selection

- Input: a test graph with *Accept* and eventually *Reject*.
- Result: A set of test cases that allow to **reach** *Accept*.

The selection algorithm computes the co-reachability analysis of *Accept*

1. We use abstract constructs (region [NTMR10] or zones [NTR11]) for building a finite representation of the test graph.
   States of the test graph: $((\ell, s), G_I, G_O, G_X)$
2. We compute the least fixpoint of a predecessor operator.
   Take care of the "as soon as possible" semantics for transitions.
   Transitions of the reachability Graph:
   $((\ell, s), G_I, G_O, G_X) \rightarrow ((\ell', s'), G'_I, G'_O, G'_X)$ asserts that from (ℓ', s') is reachable from (ℓ, s) provided that G_I, G_O, G_X are satisfied.

Test graph

1. Method 1: **product** of a test purpose with the specification [NTMR10]
2. Method 2: **observation product** [NTR11]

Test Graph - Product

Definition

$$\mathcal{A} \times TP = \langle L \times S, (\ell^0, s^0), I, O \cup O', X \cup X', Assign(O \cup O'), \Delta \rangle$$

where $\Delta_{\mathcal{A} \times TP}$ is defined:

- $(\ell, s) \xrightarrow{G \wedge G_s, A, \mathcal{X}} (\ell', s)$ with $G_s = \bigwedge_{G' \in G_{TP}(s)} \neg G'$ iff there is $I \xrightarrow{G, A, \mathcal{X}} \ell'$
- $(\ell, s) \xrightarrow{G_I \wedge G', Id_O, \mathcal{X}'} (\ell, s')$ with $G_\ell = \bigwedge_{G' \in G_{\mathcal{A}}(\ell)} \neg G'$ iff there is $s \xrightarrow{G, Id_O, \mathcal{X}'} s'$
- $(\ell, s) \xrightarrow{G \wedge G', Id_O, \mathcal{X} \cup \mathcal{X}'} (\ell', s')$ iff there are $\ell \xrightarrow{G, A, \mathcal{X}} \ell'$ and $s \xrightarrow{G', Id_O, \mathcal{X}'} s'$

We denote $Accept = \{(\ell, Acc_{TP})\}$

Test Graph - Product

Definition

$$\mathcal{A} \times TP = \langle L \times S, (\ell^0, s^0), I, O \cup O', X \cup X', Assign(O \cup O'), \Delta \rangle$$

where $\Delta_{\mathcal{A} \times TP}$ is defined:

- $(\ell, s) \xrightarrow{G \wedge G_s, A, \mathcal{X}} (\ell', s)$ with $G_s = \bigwedge_{G' \in G_{TP}(s)} \neg G'$ iff there is $I \xrightarrow{G, A, \mathcal{X}} \ell'$
- $(\ell, s) \xrightarrow{G_I \wedge G', Id_O, \mathcal{X}'} (\ell, s')$ with $G_\ell = \bigwedge_{G' \in G_{\mathcal{A}}(\ell)} \neg G'$ iff there is $s \xrightarrow{G, Id_O, \mathcal{X}'} s'$
- $(\ell, s) \xrightarrow{G \wedge G', Id_O, \mathcal{X} \cup \mathcal{X}'} (\ell', s')$ iff there are $\ell \xrightarrow{G, A, \mathcal{X}} \ell'$ and $s \xrightarrow{G', Id_O, \mathcal{X}'} s'$

We denote $Accept = \{(\ell, Acc_{TP})\}$

Test Graph - Product

Definition

$$\mathcal{A} \times TP = \langle L \times S, (\ell^0, s^0), I, O \cup O', X \cup X', Assign(O \cup O'), \Delta \rangle$$

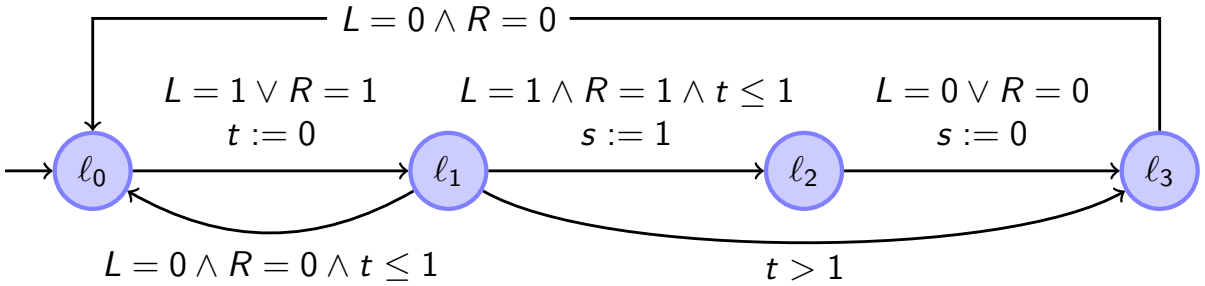
where $\Delta_{\mathcal{A} \times TP}$ is defined:

- $(\ell, s) \xrightarrow{G \wedge G_s, A, \mathcal{X}} (\ell', s)$ with $G_s = \bigwedge_{G' \in G_{TP}(s)} \neg G'$ iff there is $I \xrightarrow{G, A, \mathcal{X}} \ell'$
- $(\ell, s) \xrightarrow{G_I \wedge G', Id_O, \mathcal{X}'} (\ell, s')$ with $G_\ell = \bigwedge_{G' \in G_{\mathcal{A}}(\ell)} \neg G'$ iff there is $s \xrightarrow{G, Id_O, \mathcal{X}'} s'$
- $(\ell, s) \xrightarrow{G \wedge G', Id_O, \mathcal{X} \cup \mathcal{X}'} (\ell', s')$ iff there are $\ell \xrightarrow{G, A, \mathcal{X}} \ell'$ and $s \xrightarrow{G', Id_O, \mathcal{X}'} s'$

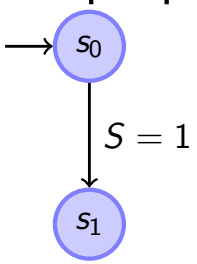
We denote $Accept = \{(\ell, Acc_{TP})\}$

Test Graph- Example

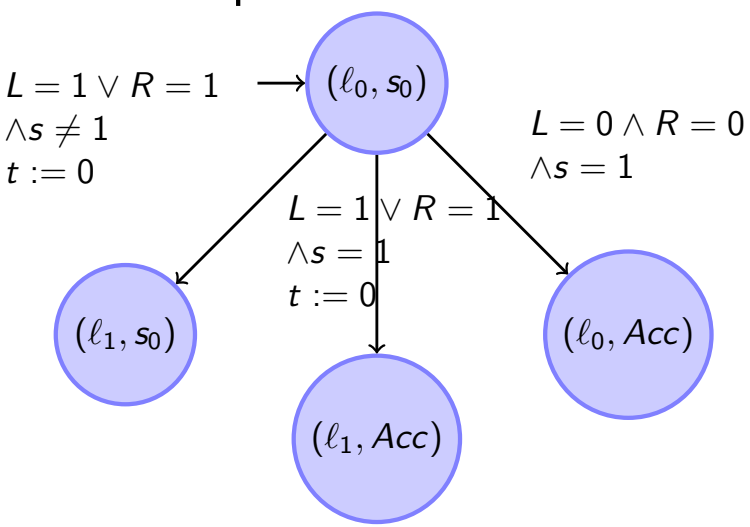
Specification:



Test purpose:



Part of the product :



Test Graph - Observation product

Definition

$$\mathcal{A} \otimes TP = \langle L \times S, (l^0, s^0), I, O \cup O', X \cup X', Assign(O \cup O'), \Delta_{\mathcal{A} \otimes TP} \rangle$$

where $\Delta_{\mathcal{A} \otimes TP}$ is defined:

$\mathcal{R}_1$: $(\ell, s) \xrightarrow{G_\ell, A, \mathcal{X}} (\ell', s)$ iff there is $\ell \xrightarrow{G_\ell, A, \mathcal{X}} \ell'$

$\mathcal{R}_2$: $(\ell, s) \xrightarrow{G \wedge G_s, Id_O, \mathcal{X}'} (\ell, s')$ with $G = \bigwedge_{G' \in G_{\mathcal{A}}(\ell)} \neg G'$ iff there is $s \xrightarrow{G_s, Id_O, \mathcal{X}'} s'$

We denote $Accept = \{(I, Acc_{TP})\}$

Test Graph - Observation product

Definition

$$\mathcal{A} \otimes TP = \langle L \times S, (l^0, s^0), I, O \cup O', X \cup X', Assign(O \cup O'), \Delta_{\mathcal{A} \otimes TP} \rangle$$

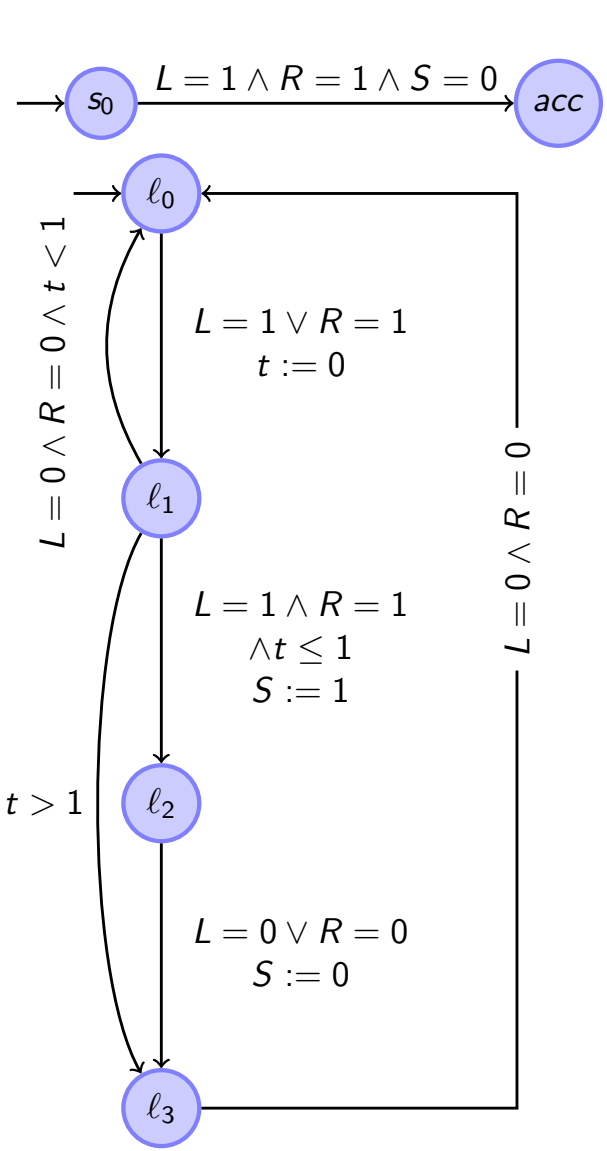
where $\Delta_{\mathcal{A} \otimes TP}$ is defined:

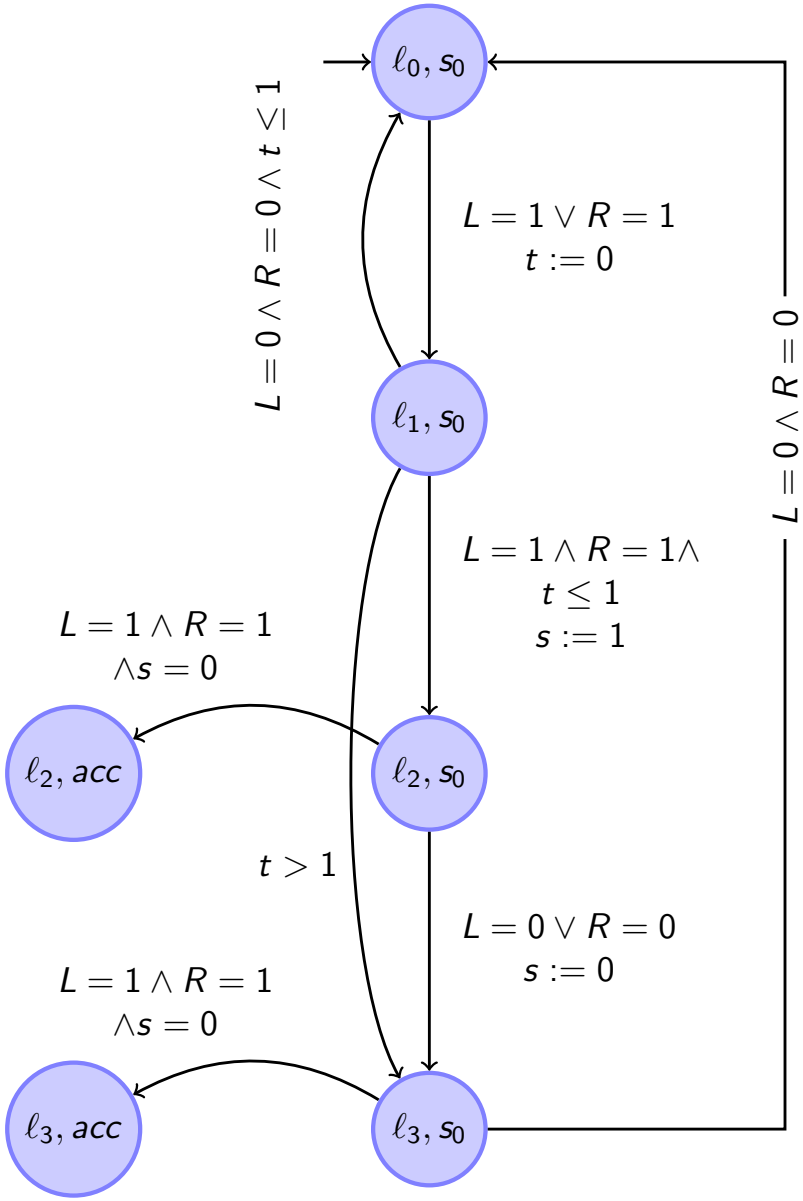
$\mathcal{R}_1$: $(\ell, s) \xrightarrow{G_\ell, A, \mathcal{X}} (\ell', s)$ iff there is $\ell \xrightarrow{G_\ell, A, \mathcal{X}} \ell'$

$\mathcal{R}_2$: $(\ell, s) \xrightarrow{G \wedge G_s, Id_O, \mathcal{X}'} (\ell, s')$ with $G = \bigwedge_{G' \in G_{\mathcal{A}}(\ell)} \neg G'$ iff there is $s \xrightarrow{G_s, Id_O, \mathcal{X}'} s'$

We denote $Accept = \{(I, Acc_{TP})\}$

Observation product - Example





Outline

Conclusion

1. Contributions:
   - A formal model for open real-time systems driven by variables.
   - A theoretical testing framework (testing algorithm and test selection)
2. Perspectives
   - Designing robust test processes that considers the communication delays, and imperfection of the hardware (e.g clocks drifts).


Omer Nguena T., Hervé Marchand, and Antoine Rollet.
Automatic test generation for data-flow reactive systems with time constraints.
In *22nd IFIP International Conference on Testing Software and Systems (ICTSS'10), Natal, Brazil*. CRIM-Canada, Nov 2010.


Omer Nguena T. and Antoine Rollet.
Conformance testing of variable driven automata.
In *8th IEEE WFCS 2010, Nancy, France*, May 2010.


Omer Nguena T. and Antoine Rollet.
A zone-based reachability analysis of variable driven timed automata.

In *3rd International Conference on Advances in System Testing and Validation Lifecycle (VALID'11)*. Xpert Publishing Services, Nov 2011.

Conformance testing of vdta

Hervé Marchand[1], **Omer Nguena Timo**[2] and Antoine Rollet[2]

[1]INRIA Rennes - Bretagne Atlantique, France
[2]LaBRI, Université de Bordeaux - CNRS, France

MTVV 2012, November 12, Palaiseau