Automatic Test Generation for Data-Flow Reactive Systems with time constraints*

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

[1] LaBRI (University of Bordeaux - CNRS), Talence, France
[2] INRIA, Centre Rennes-Bretagne Atlantique

Abstract. In this paper, we handle the problem of conformance testing for data-flow critical systems with time constraints. We present a formal model (Variable Driven Timed Automata) adapted for such systems inspired from timed automata using variables as inputs and outputs, and clocks. In this model, we consider urgency and the possibility to fire several transitions instantaneously. We present a conformance relation for this model and we propose a test generation method using a test purpose approach, based on a region graph transformation of the specification.

1 Conformance testing with VDTA

We define the conformance testing relation **tvco** for Data-flow reactive systems. such systems are characterized by the fact that they interact with their environment in a continuous way by means of continuous input and output set of events (taking their values in (possibly) infinite domains), while obeying some timing constraints. In this framework, continuous means that the values of the inputs events can be updated at anytime, while the value of the outputs events can always be observed. We choose to model our sytems by Variable Driven Timed Automata (VDTA) [7] that is a variant of timed automata [2] with only urgent transitions (i.e fired as soon as constraints are true). VDTA rather uses variables communication mechanism than synchronising actions. We first introduce the VDTA model and then our conformance relation **tvco**.

1.1 Variable driven timed automata (VDTA)

Given a set of variables V, each variable $V_i \in V$ ranges over a (infinite) domain $Dom(V_i)$ in $\mathbb{N}$, $\mathbb{Q}_+$ or $\mathbb{R}_+$. We denote $\mathcal{G}(V)$ the set of variable constraints defined as boolean combinations of constraints of the form $V_i \bowtie c$ with $V_i \in V$, $c \in Dom(V_i)$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. For a set V, variable assignment for V is a tuple $\Pi_{i\in[1..n]}(\{V_i\} \times (Dom(V_i) \cup \{\perp\}))$ and we denote by $A(V)$ the set of variable assignments for V. Given $G \in \mathcal{G}(V)$ and a valuation $v \in Dom(V)$, we write $v \models G$ when $G(v) \equiv true$. Given a valuation $v = (v_1, \cdots, v_n)$ of V and $A \in A(V)$, we define the valuations $v[A]$ as $v[A](V_i) = c$ if $(V_i, c) \in A$ and $c \neq \perp$, and $v[A](V_i) = v_i$ otherwise. Intuitively, (V_i, c) of A requires to assign c to V_i if c is a constant from $Dom(V_i)$; otherwise c is equal to $\perp$ and *no access* to V_i should be done. The identity $Id_V \in A(V)$ lets unchanged all the variables of V. We define the constraint $[A]G = \{v \mid v[A] \models G\}$. $Proj_{V_i}(G)$ denotes the constraint such that $(v_1, \cdots, v_{i-1}, v_{i+1}, \cdots, v_n) \models Proj_{V_i}(G)$ iff $\exists v_i \in Dom(V_i)$ such that $(v_1, \cdots, v_n) \models G$. We extend it to a subset V' of V and we denote it $Proj_{V'}(G)$.

* This work was supported by French National Agency ANR Testec Project.

Definition 1 (VDTA). *A* Variable Driven Timed Automaton *(VDTA) is a tuple* $\mathcal{A} = \langle L, X, I, O, l^0, G^0, \Delta_\mathcal{A} \rangle$*, where* L *is a finite set of* locations*,* $l^0 \in L$ *is the* initial location*,* $G^0 \in \mathcal{G}(I, O)$ *is the* initial condition*, a constraint with variables in* $I \cup O$ *and transitions in the transition relation* $\Delta_\mathcal{A} \subseteq L \times \mathcal{G}(I, O, X) \times A(O) \times 2^X \times L$ *:* $\langle l, G, A, \mathcal{X}, l' \rangle \in \Delta_\mathcal{A}$ *is a* transition such *that*

- $G \in \mathcal{G}(I, O, X)$ *is a boolean combination of elements of* $\mathcal{G}(I)$*,* $\mathcal{G}(O)$ *and* $\mathcal{G}(X)$ *whereas* $A \in A(O)$ *is an assignement on output variables.*
- $\mathcal{X} \in 2^X$ *is a set of clocks that are reset when triggering the transition.*

$\mathcal{A}$ *is* deterministic *if* G_0 *is satisfied by at most one valuation* (i^0, o^0)*; and for all* $l \in L$*, for all* $G, G' \in G_\mathcal{A}(l)$ *s.t.* $G \neq G'$*,* $G \cap G'$ *is unsatisfiable.*

In the sequel, we write $l \xrightarrow{G,A,\mathcal{X}} l'$ when $\langle l, G, A, \mathcal{X}, l' \rangle \in \Delta_\mathcal{A}$. A *state* of a VDTA as above defined is of the form (ℓ, i, o, x) where i, o and x are *valuations* of input, output and clock variables. A valuation $v \in Val(V)$ is simply a function that returns the values of the variables in V. If $A \in A(I)$ is an assignement on input variables, the valuation $i[A]$ change the value of input variables according to the assignement. If x is clock valuation, $\mathcal{X}$ is a subset of clocks, and $\delta \in \mathbb{R}_+$ a delay, the valuation $x + \delta$ add δ to each clock value and the valuation $x[\mathcal{X}' \leftarrow 0]$ resets from x all clocks in $\mathcal{X}$. These notions are standard in the literature.

Definition 2. *The semantics of a VDTA* $\mathcal{A} = \langle L, X, I, O, l^0, G^0, \Delta_\mathcal{A} \rangle$*, is a TTS defined by the tuple* $[\![\mathcal{A}]\!] = \langle S, s^0, \Sigma, \rightarrow \rangle$ *where* $S = L \times Dom(I) \times Dom(O) \times \mathbb{R}_+^X$ *is the (infinite) set of states,* $s^0 = (l^0, i^0, o^0, x^0)$ *is the initial configuration where* x^0 *is the clock valuation that maps every clock to* 0 *and* (i^0, o^0) *is the only solution of* G^0*,* $\Sigma = A(I) \cup A(O) \cup \mathbb{R}_+^X$ *is the (infinite) set of actions, and* $\rightarrow$ *is the transition relation with the following three types of transitions:*

T1 $(l, i, o, x) \xrightarrow{A} (l', i, o[A], x[\mathcal{X} \leftarrow 0])$ *if there exists* $(l, G, A, \mathcal{X}, l') \in \Delta_\mathcal{A}$ *such that* $(i, o, x) \models G$*,*

T2 $(l, i, o, x) \xrightarrow{A} (l, i[A], o, x)$ *with* $A \in A(I)$ *if* $\forall (l, G, A', \mathcal{X}, l') \in \Delta_\mathcal{A}$*,* $(i, o, x) \not\models G$*.*

T3 $(l, i, o, x) \xrightarrow{\delta} (l, i, o, x + \delta)$ *with* $\delta > 0$ *if for every* $\delta' < \delta$*, for every symbolic transition* $(l, G, \mathcal{X}', l') \in \Delta_\mathcal{A}$*, we have* $(i, o, x + \delta') \not\models G$*.*

The environment (e.g. the tester) of a system modeled by a VDTA observes all the variables; it can assign a value by to an input in I by performing a transition of type **T2**. Only the system can assign values to outputs in O by performing a transition of type **T1**. Transitions in $\mathcal{A}$ are urgent and *output-update transitions (T1)* are taken prior to *input-update* (T2) and *time-elapsing* (T3) transitions.

Notations. We denote $\rightarrow_{T_i}$ for transitions of type $T_i, i = 1...3$. $\text{Out}(s) = o$ gives access to the output value of $[\![\mathcal{A}]\!]$ in state $s = (l, i, o, x) \in S$. We write $s \xrightarrow{A}$ when there exists s' such that $s \xrightarrow{A} s'$ otherwise we write $s \not\xrightarrow{A}$. The latter notation is standard and it extends to sequences in Σ^*. A *run* is a sequence of alternating states and actions $s = s_0 a_1 s_1 \cdots a_n s_n$ in $S.(\Sigma.S)^*$ such that $\forall i \geq 0, s_i \xrightarrow{a_{i+1}} s_{i+1}$. $Run(s, [\![\mathcal{A}]\!])$ denotes the set of runs that can be executed in $[\![\mathcal{A}]\!]$ starting in state s and we let $Run([\![\mathcal{A}]\!]) = Run(s^0, [\![\mathcal{A}]\!])$. The *trace* $\rho(r)$ of a run $r = s_0 a_1 s_1 \cdots a_n s_n$ is given by the sequence $\rho(r) = Proj_S(r) = a_1 \cdots a_n \in \Sigma^*$.

$Tr([\![\mathcal{A}]\!]) = \{\rho(r) | r \in Run([\![\mathcal{A}]\!])\}$ is the set of traces generated by $\mathcal{A}$. We note $CoReach(S)$ the set of states from which S can be reached in $[\![\mathcal{A}]\!]$.

Stable VDTA. Thanks to their priority, several output-update transitions can be triggered in null delay. A *stable state* is a state from which no output-update transition can be fired. Formally a state s of $[\![\mathcal{A}]\!]$ is *stable* whenever for every $A \in A(O)$, $s \overset{A}{\not\longrightarrow}$. To leave this state, either the input values need to be updated or we need to let the time elapse. A *stable run* is a run that ends in a stable state. A VDTA $\mathcal{A}$ is *stable* if there is no loop of unstable states in $[\![\mathcal{A}]\!]$. In the sequel, we shall only consider stable VDTA.

Relation with other models. Timed automata [2] and most of its extensions (with delayable, eager [3], lazy transitions and variables [1] or UPPAAL model) use synchronising actions even for passing variable communication values. But the use of synchronising actions is not adequate for variable-based communication systems as the models should describe all synchronisations with the environment and they become rather unclear and big. To our knowledge urgency and variable passing mechanism are not accurately studied in timed systems for model-based testing. Results in [3] show that timed automata and urgent timed automata (with only clock variables) are equivalent in a language point of view.

1.2 Conformance testing relation

Conformance testing consists in checking that an implementation exhibits an observable behavior consistent with its specification. The main idea of conformance relation is that all observable behaviours of the implementation have to be allowed by the specification. Especially:

1. An implementation is not allowed to update a variable in a time (too late or too early) when it is not allowed by the specification.
2. An implementation is not allowed to omit to change a memory-variable at the time it is required by the specification.

A general conformance testing activity consists in executing sequences of input and delays and in checking whether the output of the implementation coincide with those of the specification. The **tioco** relation [5] (a timed extension of **ioco** [9]) requires to observe all synchronising output actions on sequences (even of duration equals to zero) the conformance relation for VDTA considers the values of outputs in states many assignements on outputs (state changing) can occur in zero time unit and each of these assignements on outputs can or not be observed depending on the granularities (frequency) of clocks of the tester and the implementation. We assume the same granularity and thus the outputs are observed only when the implementation is in *stable* states. Given a stable state s, we will thus be interested in the next stable states (or a singleton if the VDTA is deterministic) the implementation can reach from s after the execution of an input-update $A_i \in A(I)$. Given two stable states $s, s' \in S$, we write:

- $s \overset{A_i}{\Longrightarrow} s'$ if there exists a sequence $\sigma \in A(O)^*$ s.t $s \overset{A_i}{\longrightarrow} s$" $\overset{\sigma}{\rightarrow} s'$, i.e. s' is the unique stable state that can be reached from s after updating the input variables with A_i, only triggering urgent transitions in zero time unit.

– $s \stackrel{\delta}{\Longrightarrow} s'$ if there is a sequence $\sigma = \sigma_1 \cdots \sigma_n \in (\{Id_O\} \cup \mathbb{R}_+)^*$ s.t $s \stackrel{\sigma}{\rightarrow} s'$ and $\delta = \sum_{\delta_i \in Proj_O(\sigma)} \delta_i$, i.e. s' is the stable state that can be reached by letting the time elapse during δ units of time with no observable output update.

The timed transtion system $Obs(\mathcal{A}) = (S, s^0, A(I) \cup \mathbb{R}_+, \Longrightarrow)$ is inductively generated from $[\![\mathcal{A}]\!]$ by starting from s^0 (that is supposed to be stable) and by using the two previous rules. We let $ObsRun(\mathcal{A}) = Run(Obs(\mathcal{A}))$ and $ObsTr(\mathcal{A}) = Tr(Obs(\mathcal{A}))$ denote the set of observed runs and observed traces of $\mathcal{A}$. Finally, we define s *Safter* $\alpha = \{s' \mid s \stackrel{\alpha}{\Longrightarrow} s'\}$ and $\mathcal{A}$ *Safter* $\alpha = s^0$ *Safter* α.

We assume that the implementation Imp and the specification $\mathcal{A}$ are both modeled by compatible VDTA (i.e. they share the same input and output variables).

Definition 3. *Imp conforms to* $\mathcal{A}$ (Imp *tvco* $\mathcal{A}$) *whenever*

$$\forall \sigma \in ObsTr(\mathcal{A}), Out(Imp \text{ } Safter \text{ } \sigma) \subseteq Out(\mathcal{A} \text{ } Safter \text{ } \sigma)$$

This relation intends to check if the values of output variables of the implementation are correct after any sequence made of assignments of input variables or time elapsing. The tester can observe all output variables of the implementation. The tester can only update the input variables of the implementation or let the time elapse. Note that the blocking aspect as in [9] is not considered so far.

2 Reachability analysis

When testing, it is sometimes useful to target some particular states of the systems. To do so, we shall use a backward reachability analysis algorithm on the so-called *timed abstract graph* based on the known *region* abstraction [2]. We let $Reg(X)$ be the finite set of clock regions with respect to K, the maximal constant clocks in $\mathcal{A}$ are compared with. $[x]$ denotes the clock region that contains x, and $[\![r]\!]$ denotes the set of clock valuations whose clock region is equal to r. $I\_Succ(r)$ denotes *the immediate successor* of r. A region can be represented by a *diagonal clock constraint* that involves comparisons of two clocks. If r is a region, then G_r denotes the unique atomic (smallest) clock constraint such that $r \subseteq [\![G_r]\!]$. Given a location ℓ and a region r, $Gds_{\mathcal{A}}(\ell, r) = \{G \mid \ell \xrightarrow{G,A,\mathcal{X}} \ell' \text{ and } [\![r]\!] \subseteq [\![G]\!]\}$ is the set of constraints whose timing part is satisfied by r.

Definition 4 (Time-abstract graph). *The* time-abstract graph (TAG) *of a VDTA* $\mathcal{A}$ *as above, is the VDTA* $RG(\mathcal{A}) = \langle Reg(\mathcal{A}), X, I, O, (\ell^0, r^0), G^0, \Delta_{RG} \rangle$ *where* $Reg(\mathcal{A}) = L \times Reg(X)$ *is the set of locations of* $RG(\mathcal{A})$*, The transition relation,* $\Delta_{RG} \subseteq Reg(\mathcal{A}) \times \mathcal{G}(I, O, X) \times A(O) \times Reg(\mathcal{A})$ *is such that:*

U1 $(\ell, r) \xrightarrow{G_I \wedge G_O \wedge G_r, A, \mathcal{X}} (\ell', r')$ *iff* $\exists \ell \xrightarrow{G,A,\mathcal{X}} \ell'$ *s.t.* $[\![r]\!] \subseteq [\![G_X]\!]$, $r' = r[\mathcal{X} \leftarrow 0]$

U2 $(\ell, r) \xrightarrow{G' \wedge G_{r'}, Id_O, \emptyset} (\ell, r')$ *with* $G' = \neg(\bigvee_{G \in Gds_{\mathcal{A}}(\ell,r)} G_I \wedge G_O)$, $r' = I\_Succ(r)$

Our backward reachability algorithm for deterministic VDTA works on $RG(\mathcal{A})$ instead of $[\![\mathcal{A}]\!]$. It computes *predecessors* from which we can reach the target *configuration* of $RG(\mathcal{A})$. A configuration of $RG(\mathcal{A})$ is of the form (q, G) where q is a state of $RG(\mathcal{A})$ and G is a constraint of $\mathcal{G}(I, O)$. We consider *urgent*

abstract predecessors ($aPred_u$), *time-elapsing abstract predecessor* ($aPred_\delta$) and *input-update abstract predecessors* ($aPred_e$) defined over as follows:

$$aPred_u(q,G) = \{(q', Proj_X(G') \wedge Proj_{Var(a)}(G) \mid q' \xrightarrow{G',a,Y}_{U_1} q \\ \wedge Proj_{I\cup X}(G')[a] \subseteq Proj_I(G)\}$$

$$aPred_\delta(q,G) = (q,G) \cup \{(q', Proj_X(G') \wedge G \mid q' \xrightarrow{G',Id_O,\emptyset}_{U_2} q\}$$

$$aPred_e(q,G) = (q, (\neg \bigvee_{G'\in Gds_{\mathcal{A}}(q)} Proj_X(G')) \wedge Proj_I(G))$$

We consider $aPred(Q,G) = aPred_u(Q,G) \cup aPred_\delta(Q,G) \cup aPred_e(Q,G)$ where $aPred_\theta(Q,G) = \bigcup_{q\in Q} aPred_\theta(q,G)$ for $\theta \in \{u,i,\delta\}$ and a set of configurations Q. Finally, $CoReach_a(Q,G) = \mu X.(Q,G) \cup aPred(X)$.

Proposition 1. *Given $q = (l,[x])$ and $G \in \mathcal{G}(I,O)$, $CoReach_a(q,G)$ is effectively computable.*

$\mathcal{Q} = L \times Reg(X) \times \mathcal{C}_M(I,O)$ where $\mathcal{C}_M(I,O)$ denotes the set of constraints on input/outputs the maximal constant occuring in them is lower or equal to M, is finite and $aPred : 2^{\mathcal{Q}} \to 2^{\mathcal{Q}}$ is monotonic. Using fixpoint computation results in [8], we get the termination of the computation of $CoReach_a$. Proposition 2 allows to use $CoReach_a$ in $RG(\mathcal{A})$ instead of $CoReach$ in $[\![\mathcal{A}]\!]$.

Proposition 2. *Let $G' \in \mathcal{G}(I,O)$ be a constraint. It holds that*
$(l,i,o,x) \in Coreach(l', G' \wedge [x'])$ *iff* $((l,[x]),i,o) \in Coreach_a((l',[x']),G')$

Note that *Zones* [4] can also be used for the analysis of VDTA provided a sophisticate construction to handle the urgency in the model. For practical issues zones are more suitable but regions are adequate for a theoretical issues.

3 Automatic test generation

In the sequel, specifications are deterministic. The test selection is based on *Test Purposes* (TP) that allow to select behaviors to be tested from the specification.

Definition 5. *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: S is a finite set of locations with a special trap location $Accept_{TP} \in S$, s^0 is the initial location; 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 $\mathcal{A}$; X' is the set of clocks with $X' \cap X = \emptyset$, the transition relation*[1] *is $\Delta_{TP} \subseteq S \times \mathcal{G}(I,O,X,X') \times Id_O \times 2^{X'} \times S$.*

TP is non intrusive with respect to the specification: it does not reset clocks of the specification S and does not assign new values to the output variables. Test purposes are complete, meaning that whatever is the observation of variables or clocks either a transition is taken or the current location does not change. The derivation of test cases proceeds in two steps:

Step 1. Product of the specification $\mathcal{A}$ and the test purpose TP

[1] $G \in \mathcal{G}(I,O,X,X')$ if G is a combination of elements of $\mathcal{G}(I)$, $\mathcal{G}(O)$, $\mathcal{G}(X)$ and $\mathcal{G}(X')$

Definition 6 (Synchronous product). *Given* $\mathcal{A} = \langle L, X, I, O, l^0, G^0, \Delta_{\mathcal{A}} \rangle$ *a specification and a test purpose* $TP = \langle S, X' \cup X, I, O, s^0, G^0, \Delta_{TP} \rangle$*, the synchronous product of* $\mathcal{A}$ *and* TP *is the VDTA* $\mathcal{A} \times TP = \langle L \times S, X \cup X', I, O, (l^0, s^0), G^0, \Delta_{\mathcal{A} \times TP} \rangle$ *where* $\Delta_{\mathcal{A} \times TP}$ *is defined by the following rules (*$\mathcal{R}_1$*,* $\mathcal{R}_2$*,* $\mathcal{R}_3$*):*

$\mathcal{R}_1$*:* $(l, s) \xrightarrow{G \wedge G_s, A, \mathcal{X}} (l', s)$ *with* $G_s = \bigwedge_{G' \in G_{TP}(s)} \neg G'$ *iff there is* $l \xrightarrow{G, A, \mathcal{X}} l'$

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

$\mathcal{R}_3$*:* $(l, s) \xrightarrow{G \wedge G', Id_O, \mathcal{X} \cup \mathcal{X}'} (l, s')$ *iff there are* $l \xrightarrow{G, A, \mathcal{X}} l'$ *and* $s \xrightarrow{G', Id_O, \mathcal{X}'} s'$

An output-update transition can be performed either exclusively in the specification ($\mathcal{R}_1$), or exclusively in the test purpose ($\mathcal{R}_2$), or simultaneously in both systems ($\mathcal{R}_3$). We can show that $Tr(\mathcal{A}) = Tr(\mathcal{A} \times TP)$ and $ObsTr(\mathcal{A}) = ObsTr(\mathcal{A} \times TP)$. The set of locations of the form $(l, Accept_{TP})$ is denoted Acc.

Step 2. Symbolic test case selection and execution. Let $Pass$ be the set of locations of the form (Acc, r) in $RG(\mathcal{A} \times TP)$. Locations of $RG(\mathcal{A}_{TP})$ are tagged with adequate constraints on the input/output variables when computing $CoReach_a(Pass)$. A *symbolic test case* is a path from the initial location of $RG(\mathcal{A} \times TP)$ to a location in $Pass$. The execution of a test case consists, in each location, to select an input or a delay that satifies the tagged input constraint or allows to change the region. An implementation Imp **passes** a test case if and only if any test run leads to a *pass* verdict. Similar to [7], we can show that our algorithm for a specification $\mathcal{A}$ and for all test purposes TP is complete w.r.t. all possible test cases and the relation *tvco*.

4 Concluding Remarks

We described an adapted conformance relation and a model-based testing framework with test purposes for reactive systems modeled by VDTA. A long version of this paper is available in [6].

References

1. R. Alur, T. Dang, and F. Ivancic. Reachability analysis of hybrid systems via predicate abstraction. In *Hybrid Systems: Computation and Control, LNCS 2289*, pages 35–48, 2002.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. Roberto Barbuti and Luca Tesei. Timed automata with urgent transitions. *Acta Inf.*, 40(5):317–347, 2004.
4. Johan Bengtsson and Wang Yi. Timed automata: Semantics, algorithm and tools. In *Lectures on Concurrency and Petri Nets*, LNCS, pages 87–124, 2004.
5. M. Krichen and S. Tripakis. Conformance testing for real-time systems. *Formal Methods in System Design*, 34(3):238 – 304, 2009.
6. O. Nguena Timo, H. Marchand, and A. Rollet. Automatic test generation for dataflow reactive systems with time constraints. Technical report, Labri, 2010.
7. O. Nguena Timo and A. Rollet. Conformance testing of variable driven automata. In *8th IEEE WFCS 2010, Nancy, France*, May 2010.
8. Alfred Tarski. A lattice theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.
9. J. Tretmans. Test generation with inputs, outputs, and repetitive quiescence. *Software-Concepts and Tools*, 17:103–120, 1996.