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 systems 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 \(\text{Dom}(V_i)\) 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 \(V_i \in V\), \(c \in \text{Dom}(V_i)\) and \(\bowtie \in \{<,\leq,=,\geq,>\}\). For a set \(V\), variable assignment for \(V\) is a tuple \(\Pi_{i\in[1..n]}(\{V_i\} \times (\text{Dom}(V_i) \cup \{\perp\}))\) and we denote by \(A(V)\) the set of variable assignments for \(V\). Given \(G \in \mathcal{G}(V)\) and a valuation \(v \in \text{Dom}(V)\), we write \(v \models G\) when \(G(v) \equiv \text{true}\). Given a valuation \(v = (v_1,\ldots,v_n)\) 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 \(\text{Dom}(V_i)\); 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 \([A]G = \{v \mid v[A] \models G\}\). \(\text{Proj}_{V_i}(G)\) denotes the constraint such that \((v_1,\ldots,v_{i-1},v_{i+1},\ldots,v_n) \models \text{Proj}_{V_i}(G)\) iff \(\exists v_i \in \text{Dom}(V_i)\) such that \((v_1,\ldots,v_n) \models G\). We extend it to a subset \(V'\) of \(V\) and we denote it \(\text{Proj}_{V'}(G)\).

\* This work was supported by French National Agency ANR Testec Project.
Definition 1 (VDTA). A Variable Driven Timed Automaton (VDTA) is a tuple $A = \langle L, X, I, O, I^0, G^0, \Delta_A \rangle$, where $L$ is a finite set of locations, $I^0 \in L$ 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 $\Delta_A \subseteq L \times \mathcal{G}(I, O, X) \times A(O) \times 2^X \times L : \langle l, G, A, X, l' \rangle \in \Delta_A$ 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 assignment on output variables.
- $X \subseteq 2^X$ is a set of clocks that are reset when triggering the transition. $A$ is deterministic if $G_0$ is satisfied by at most one valuation $(i^0, o^0)$; and for all $l \in L$, for all $G, G' \in G_A(l)$ s.t. $G \neq G'$, $G \cap G'$ is unsatisfiable.

In the sequel, we write $l \xrightarrow{G,A,X} l'$ when $\langle l, G, A, X, l' \rangle \in \Delta_A$. 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 assignment on input variables, the valuation $i[A]$ change the value of input variables according to the assignment. If $x$ is clock valuation, $X$ is a subset of clocks, and $\delta \in \mathbb{R}_+$ a delay, the valuation $x + \delta$ add $\delta$ to each clock value and the valuation $x[X' \leftarrow 0]$ resets from $x$ all clocks in $X$. These notions are standard in the literature.

Definition 2. The semantics of a VDTA $A = \langle L, X, I, O, I^0, G^0, \Delta_A \rangle$, is a TTS defined by the tuple $[A] = \langle S, s^0, \Sigma, \rightarrow \rangle$ where $S = L \times \text{Dom}(I) \times \text{Dom}(O) \times \mathbb{R}_+^X$ 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** $(l, i, o, x) \xrightarrow{A} (l', i, o[A], x[X \leftarrow 0])$ if there exists $(l, G, A, X, l') \in \Delta_A$ such that $(i, o, x) \not\models G$.
- **T2** $(l, i, o, x) \xrightarrow{A} (l, i[A], o, x)$ with $A \in A(I)$ if$\forall (l, G, A', X, l') \in \Delta_A$, $(i, o, x) \not\models G$.
- **T3** $(l, i, o, x) \xrightarrow{A} (l, i, o, x + \delta)$ with $\delta > 0$ if for every $\delta' < \delta$, for every symbolic transition $(l, G, X', l') \in \Delta_A$, 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 $A$ are urgent and output-update transitions (T1) are taken prior to input-update (T2) and time-elapsing (T3) transitions.

Notations. We denote $\rightarrow_T$ for transitions of type $T_i, i = 1...3$. Out$(s) = o$ gives access to the output value of $[A]$ 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 $s \xrightarrow{\not{A}}$. The latter notation is standard and it extends to sequences in $\Sigma^*$. A run is a sequence of alternating states and actions $s = s_0a_1s_1\cdots a_ns_n$ in $S.(\Sigma,S)^*$ such that $\forall i \geq 0, s_i \xrightarrow{a_{i+1}} s_{i+1}$. Run$(s,[A])$ denotes the set of runs that can be executed in $[A]$ starting in state $s$ and we let Run$([A]) = \text{Run}(s^0, [A])$. The trace $\rho(r)$ of a run $r = s_0a_1\cdots a_ns_n$ is given by the sequence $\rho(r) = \text{Proj}_S(r) = a_1\cdots a_n \in \Sigma^*$.
Tr([A]) = \{ \rho(r) | r \in Run([A]) \} is the set of traces generated by \( A \). We note CoReach(S) the set of states from which \( S \) can be reached in \([A]\).

**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 \([A]\) is *stable* whenever for every \( A \in A(O) \), \( s \not\rightarrow A \). 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 \( A \) is *stable* if there is no loop of unstable states in \([A]\). 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 \textit{tioco} relation [5] (a timed extension of \textit{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:

\[ s \xrightarrow{A_i} s' \] if there exists a sequence \( \sigma \in A(O)^* \) s.t. \( s \xrightarrow{A_i} s'' \xrightarrow{\sigma} s' \), 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.
The timed transition system $\text{Obs}(A) = (S, s^0, A(I) \cup \mathbb{R}_+, \Rightarrow)$ is inductively generated from $[A]$ by starting from $s^0$ (that is supposed to be stable) and by using the two previous rules. We let $\text{ObsRun}(A) = \text{Run}(\text{Obs}(A))$ and $\text{ObsTr}(A) = \text{Tr}(\text{Obs}(A))$ denote the set of observed runs and observed traces of $A$. Finally, we define $s \xrightarrow{\text{Safter}} \alpha = \{s' \mid s \xrightarrow{\alpha} s'\}$ and $A \xrightarrow{\text{Safter}} \alpha = s^0 \xrightarrow{\text{Safter}} \alpha$.

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

**Definition 3.** $\text{Imp}$ conforms to $A$ ($\text{Imp}$ timed $A$) whenever

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

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 $\text{Reg}(X)$ be the finite set of clock regions with respect to $K$, the maximal constant clocks in $A$ are compared with. $[x]$ denotes the clock region that contains $x$, and $\llbracket r \rrbracket$ denotes the set of clock valuations whose clock region is equal to $r$. $\text{I.Succ}(r)$ 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 $r \subseteq \llbracket G_r \rrbracket$. Given a location $l$ and a region $r$, $\text{Gds}_A(l, r) = \{G \mid l, G.A.X, l' \text{ and } \llbracket r \rrbracket \subseteq \llbracket G \rrbracket\}$ 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 $A$ as above, is the VDTA $\text{RG}(A) = (\text{Reg}(A), X, I, O, (l^0, r^0), G^0, \Delta_{\text{RG}})$ where $\text{Reg}(A) = L \times \text{Reg}(X)$ is the set of locations of $\text{RG}(A)$. The transition relation, $\Delta_{\text{RG}} \subseteq \text{Reg}(A) \times G(I, O, X) \times A(O) \times \text{Reg}(A)$ is such that:

$\text{U1} \quad (l, r) \xrightarrow{G_{l^0} \land G_O \land G_{-A.X}} (l', r')$ iff $\exists l', G_{A.X}, l' \text{ s.t. } \llbracket r \rrbracket \subseteq \llbracket G_X \rrbracket, r' = r[X \leftarrow 0]$  

$\text{U2} \quad (l, r) \xrightarrow{G' \land G_{-l^0} \land G_{-O}} (l', r') \text{ with } G' = \neg(\bigvee_{G \in \text{Gds}_A(l, r)} G_{l^0} \land G_O), r' = \text{I.Succ}(r)$

Our backward reachability algorithm for deterministic VDTA works on $\text{RG}(A)$ instead of $[A]$. It computes predecessors from which we can reach the target configuration of $\text{RG}(A)$. A configuration of $\text{RG}(A)$ is of the form $(q, G)$ where $q$ is a state of $\text{RG}(A)$ and $G$ is a constraint of $G(I, O)$. We consider urgent
abstract predecessors \((a\text{Pred}_a)\), time-elapsing abstract predecessor \((a\text{Pred}_d)\) and input-update abstract predecessors \((a\text{Pred}_e)\) defined over as follows:

\[
a\text{Pred}_a(q,G) = \{(q', \text{Proj}_X(G') \cap \text{Proj}_{\text{Var}(a)}(G)) \mid \exists a' \in a\text{Pred}_a(q,G) \land q' \not\in \text{Proj}_{\text{Var}(a)}(G) \}
\]

\[
a\text{Pred}_d(q,G) = (q,G) \cup \{(q', \text{Proj}_X(G') \cap G) \mid q' \not\in \text{Proj}_{\text{Var}(a)}(G) \}
\]

\[
a\text{Pred}_e(q,G) = (q, (-\bigvee_{G' \in \text{Gd}_{a}(q)} \text{Proj}_X(G')) \land \text{Proj}_I(G))
\]

We consider \(a\text{Pred}(Q,G) = a\text{Pred}_a(Q,G) \cup a\text{Pred}_d(Q,G) \cup a\text{Pred}_e(Q,G)\) where \(a\text{Pred}_d(Q,G) = \bigcup_{q \in Q} a\text{Pred}_d(q,G)\) for \(\theta \in \{u,i,\delta\}\) and a set of configurations \(Q\). Finally, \(\text{CoReach}_a(Q,G) = \mu X. (Q,G) \cup a\text{Pred}(X)\).

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

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

**Proposition 2.** Let \(G' \in \mathcal{G}(I,O)\) be a constraint. It holds that

\((l, i, o, x) \in \text{Coreach}([l', G' \land [x']])\) iff \((l, [x]), i, o) \in \text{Coreach}_a((l', [x']), G')\)

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 \(A = \langle L, X, I, O, l^0, G^0, \Delta_A \rangle\) 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 \(A\); \(X'\) is the set of clocks with \(X' \cap X = \emptyset\), the transition relation\(^1\) is \(\Delta_{TP} \subseteq S \times \mathcal{G}(I,O,X,X') \times I_dO \times X'\).

\(TP\) 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 \(A\) and the test purpose \(TP\)**
Definition 6 (Synchronous product). Given $A = (L, X, I, O, l^0, G^0, \Delta_A)$ a specification and a test purpose $TP = (S, X' \cup X, I, O, s^0, G^0, \Delta_{TP})$, the synchronous product of $A$ and $TP$ is the VDTA $A \times TP = (L \times S, X \cup X', I, O, (l^0, s^0), G^0, \Delta_{A \times TP})$ where $\Delta_{A \times TP}$ is defined by the following rules ($R_1$, $R_2$, $R_3$):

$R_1$: $(l, s) \xrightarrow{G \land G', A, X}(l', s)$ with $G_s = \bigwedge_{G' \in G_A} G'_s$ if $l \xrightarrow{G, A, X} l'$

$R_2$: $(l, s) \xrightarrow{G \land G', I_d, O, X'} (l', s')$ with $G_i = \bigwedge_{G' \in G_A} G'_i$ if $s \xrightarrow{G, I_d, X'} s'$

$R_3$: $(l, s) \xrightarrow{G \land G', I_d, O, X \cup X'} (l', s')$ if there are $l \xrightarrow{G, A, X} l'$ and $s \xrightarrow{G, I_d, X'} s'$

An output-update transition can be performed either exclusively in the specification ($R_1$), or exclusively in the test purpose ($R_2$), or simultaneously in both systems ($R_3$). We can show that $Tr(A) = Tr(A \times TP)$ and $ObsTr(A) = ObsTr(A \times TP)$. The set of locations of the form $(l, \text{Accept}_{TP})$ is denoted $\text{Acc}$.

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

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

7. O. Nguena Timo and A. Rollet. Conformance testing of variable driven automata.
   In 8th IEEE WFCS 2010, Nancy, France, May 2010.