TimeSquare: a Multiform Time Simulation Environment

Charles André, Benoît Ferrero, Frédéric Mallet
Aoste Project I3S-INRIA, Université de Nice-Sophia Antipolis
INRIA Sophia Antipolis Méditerranée, FRANCE
{candre,bferreo,fmallet}@sophia.inria.fr

Abstract

TIMESQUARE *is a software environment for modeling and analysis of timed systems. It supports an implementation of the Time Model introduced in the* UML MARTE *(Modeling and Analysis of Real-Time Embedded systems) and its companion Clock Constraint Specification Language (*CCSL*).* TIMESQUARE *displays possible time evolutions—solutions to the clock constraint specification—as waveforms generated in the standard* VCD *format.*

In TIMESQUARE*, time is usually multiform that is, relying on several clocks either chronometric (classical time modeling) or logical. The latter are useful at the design level.*

This paper briefly introduces the MARTE *time model and the clock constraints. Then, the formal foundations of this model are given. The main* TIMESQUARE *functionalities are presented and illustrated on an unusual example.*

1 Time Model in MARTE

Time is almost absent in UML. The UML profile for Modeling and Analysis of Real-Time Embedded systems (MARTE) [1] addresses this issue and enriches UML with time-related concepts such as instants, durations, timed events, timed behavior. . . .

MARTE time model deals with both *discrete* and *dense* time. A *clock* gives access to a *time structure*. A clock can be either *chronometric* or *logical*. The former implicitly refers to “physical time”, whereas the latter does not. Logical clocks, which focus on the ordering of instants, *may* ignore the physical duration between instants, but this does not preclude quantitative information attached to (logical) clock instants. MARTE also allows *multiform* time modeling. This concept is inherited from synchronous languages [2]: time observations (through clocks) can rely on different referentials. Control systems often resort to multiform time, for instance in automotive applications, time can be measured as an angular position of the crankshaft in a 4-stroke engine [3]. The many clocks of a model are *a priori* independent. MARTE introduces a stereotype of UML Constraint that constrains clock instants of the model. These constraints can be expressed with the non normative Clock Constraint Specification Language (CCSL) annexed to MARTE specification [1, Annex C]. The formal semantics of this language, which is outside the OMG specification, is briefly described below.

TIMESQUARE mainly supports logical time, so, the presentation lays stress on this form of time.

2 Formal Foundations of the Time Model

2.1 Clock and Time Structure

A *Clock* is a 5-tuple $\langle \mathcal{I}, \prec, \mathcal{D}, \lambda, u \rangle$ where $\mathcal{I}$ is a set of instants, $\prec$ is a quasi-order relation on $\mathcal{I}$, named *strict precedence*, $\mathcal{D}$ is a set of labels, $\lambda : \mathcal{I} \rightarrow \mathcal{D}$ is a labeling function, u is a symbol, standing for a *unit*. For logical clocks, u is often called tick, it can be processorCycle (or a busCycle) as well or any other logical activation of a behavior. The *ordered set* $\langle \mathcal{I}, \prec \rangle$ is the temporal structure associated with the clock. $\prec$ is a total, irreflexive, and transitive binary relation on $\mathcal{I}$.

A *discrete-time clock* c is a clock with a discrete set of instants $\mathcal{I}$. Since $\mathcal{I}$ is discrete, it can be indexed by natural numbers in a fashion that respects the ordering on $\mathcal{I}$. $c[k]$ denotes the k^{th} instant. Moreover, in the discrete case, each instant, but the first one, has a unique direct predecessor.

A set of clocks constrained by *clock constraints* defines a *Time Structure*. More formally, a time structure is a pair $\langle C, \preccurlyeq \rangle$ where C is a set of clocks, $\preccurlyeq$ is a binary relation on $\bigcup_{c \in C} \mathcal{I}_c$, named *precedence*. $\preccurlyeq$ is reflexive and transitive. From $\preccurlyeq$ we derive four new instant relations: *Coincidence* ($\equiv \triangleq \preccurlyeq \cap \succcurlyeq$), *Strict precedence* ($\prec \triangleq \preccurlyeq \setminus \equiv$), *Independence* ($\parallel \triangleq \overline{\preccurlyeq \cup \succcurlyeq}$), and *Exclusion* ($\# \triangleq \prec \cup \succ$). The set of instants, quotiented by $\equiv$ is a poset $\left\langle \bigcup_{c \in C} \mathcal{I}_c / \equiv, \preccurlyeq \right\rangle$.

Instant relations are defined on pairs of instants. This is obviously not suitable for time structure specification. Instead we have defined constraints on clocks: a clock constraint imposes many—usually infinitely many—instant constraints.

2.2 Clock constraints

Clock constraints can be divided into four categories: *synchronous*, *asynchronous*, *mixed*, and *NFP chronometric* (NFP stands for Non Functional Properties).

Synchronous clock constraints rely on *coincidence*. *Subclocking* is such a constraint: each instant of the *subclock* must coincide with one instant of the *superclock*. Of course, the mapping must be order-preserving.

Asynchronous clock constraints are based on *precedence*. A (discrete) clock a is *faster than* clock b if for all natural number k, the k^{th} instant of a precedes the k^{th} instant of b ($\forall k \in \mathbb{N}, a[k] \prec b[k]$).

Mixed clock constraints combine both coincidence and precedence. For instance the *sampling* constraint: $c = a \ \mathsf{sampledOn} \ b$ imposes c to tick synchronously with b whenever a tick of a precedes a tick of b.

NFP chronometric constraints apply to chronometric clocks. They specify temporal properties such as *stability*, *offset*, *jitter*, etc. These constraints are used to characterize imperfect chronometric clocks.

Semantics of clock constraints

A *Time Structure* is considered as a dynamic system and its behavior is defined by an infinite sequence of *steps*. A step consists of simultaneous clock ticks. When a (discrete) clock ticks, its current instant changes for the next one (its current index is incremented by 1). We call *configuration* of a time structure $\langle C, \preccurlyeq \rangle$ a mapping $c : C \to \mathbb{N}$. For each discrete clock clk, $c(clk)$ is the current index of clock clk. This index denotes the *current instant* of clk.

For a set of clocks subject to a conjunction of clock constraints, the challenge is, "given a configuration, determine a step that meets all the constraints". There may be 0 (inconsistent constraints), 1 (deterministic) or several satisfying steps (non deterministic).

To address this challenge, we have endowed CCSL with a structural operational semantics. It is sufficient to define SOS rules for a *kernel* CCSL (less than 20 rules). For illustration purpose, consider the "faster than" relation $c_1 \boxed{\prec} c_2$.

$$\frac{\begin{array}{l} c_1, c \vdash B_1 \\ c_2, c \vdash B_2 \\ \beta_1 \triangleq (c\,(c_1) = c\,(c_2)) \\ \beta_2 \triangleq (c\,(c_1) > c\,(c_2)) \end{array}}{c_1 \boxed{\prec} c_2, c \vdash B_1 \wedge B_2 \wedge \mathsf{ite}\,(\beta_1, \neg\gamma_2, \beta_2)}$$

This rule reads that, for the given configuration c, constraint "c_1 faster than c_2" implies the Boolean expression on the right-hand side. In this rule, γ_k is a Boolean variable associated with clock c_k. $\gamma_k = \mathtt{true}$ means that c_k can tick. The Boolean expression refers to Boolean expressions (B_1, B_2) attached to the concerned clocks (c_1, c_2), and imposes additional logical constraints, specific to the faster-than relation: $\mathsf{ite}\,(\beta_1, \neg\gamma_2, \beta_2)$, where ite is the if–then–else ternary operator. An equivalent, but more verbose expression, is $((\beta_1 \wedge \neg\gamma_2) \vee (\neg\beta_1 \wedge \beta_2))$.

Thus, from a CCSL specification we derive a set of Boolean expressions. Let B be the conjunction of all these expressions. Starting with B, we determine the set of all possible (logical) solutions. From this set we deduce the set of *Enabled Clocks* (E). A subset F (*Fired Clocks*) of E characterizes the new *step*. Not all subsets of E are correct solutions because a step must contain all or none of the clocks that have coincident instants. To derive F from E, the user may choose among different policies: minimal solution, maximal solution, and user's defined policies. The default policy is a random selection of a minimal correct solution.

Applying *rewriting rules* of the form $c_1 \xrightarrow{c_1 \in F} c_1'$ for all fired clocks yields the new set of clock constraints.

3 TimeSquare Environment

3.1 Functionalities

TIMESQUARE has four main functionalities: 1) interactive clock-related specifications, 2) clock constraint checking, 3) generation of a solution, 4) displaying and exploring waveforms.

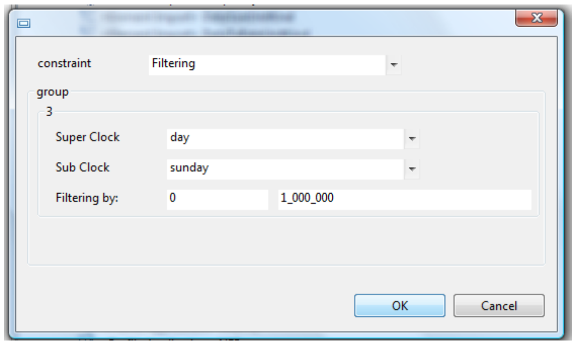
constraint: Filtering
group
3
Super Clock: day
Sub Clock: sunday
Filtering by: 0 1_000_000

Figure 1. Dialog box for clock constraints.

TIMESQUARE has been designed to be used with UML tools applying the MARTE profile. In this profile, clocks and clock constraints can be associated with many and various model elements. A wizard is included in TIMESQUARE. It facilitates clock definitions, clock constraint specifications, model element browsing, and parameter setting. Figure 1

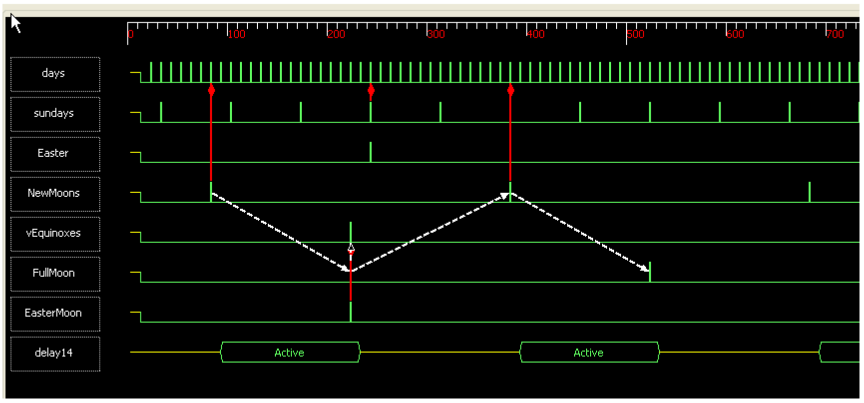

Figure 2. Waveforms for Easter.

shows an example of dialog box for specifying clock constraints.

The second functionality checks constraint sanity and is called when the above mentioned wizard is not used.

The third functionality relies on a *constraint solver* that yields a satisfying execution trace or issues an error message in case of inconsistency. The traces are given as waveforms written in VCD format. VCD (Value Change Dump) [4] is an IEEE standard textual format for dumpfiles used by EDA (Electronic Design Automation) logic simulation tools. The solver intensively uses Binary Decision Diagrams (BDD).

Waveforms can be displayed with any VCD viewer. TIMESQUARE has its own viewer enriched with interactive constraint highlighting and access facilities. For instance, the screen copy in Figure 2 shows precedence relations (white oblique dashed arrows) and coincidence relations (red vertical solid lines).

3.2 Implementation

TIMESQUARE is a plug-in developed with Ganymede Eclipse Modeling Tools (Eclipse packaging including EMF, GMF, MDT XSD/OCL/UML2, M2M, M2T, and EMFT). ANTLR for constraint parsing, and JavaBDD for the solver are also used. TIMESQUARE is integrated in the OpenEmbeDD platform.

4 Example

To illustrate clock constraints we deliberately choose a non technical system. The Easter Day is determined by a canonical rule and depends on a complex conjunction of vernal equinox, new moon, and Sundays. All these constraints are captured by CCSL. A full specification of this system, along with a more detailed presentation of the clock constraints are available in a research report [5].

Figure 2 shows the Time Structure for Easter 2008.

5 Perspectives

At present, TIMESQUARE simulates the behavior of a Time Structure specified in CCSL. Instead of a simple trace generation, we could analyze the set of reachable configurations for a given policy. This work is in progress and should be greatly facilitated by our underlying formal semantics.

References

[1] OMG. *UML Profile for MARTE, beta 2*. Object Management Group, June 2008. OMG document number: ptc/08-06-08.

[2] A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. Le Guernic, and R. de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.

[3] C. André, F. Mallet, and M-A Peraldi-Frati. A multiform time approach to real-time system modeling; application to an automotive system. In *Industrial Embedded Systems, 2007. SIES '07*, pages 234–241, Lisbon, July 2007. IEEE.

[4] IEEE Standards Association. *IEEE Standard for Verilog Hardware Description Language*. Design Automation Standards Committee, 2005. IEEE Std 1364TM-2005.

[5] C. André and F. Mallet. Clock constraints in UML MARTE CCSL. Research Report 6540, INRIA, 05 2008.