Model Checking for Timed Statecharts

Junyan Qian[1,2] and Baowen Xu[1]

[1] Department of Computer Science and Engineering,
Southeast University, Nanjing 210096, China
http://cse.seu.edu.cn/people/bwxu/
[2] Department of Computer Science and Technology,
Guilin University of Electronic Technology, Guilin 541004, China
qjy@gliet.edu.cn, bwxu@seu.edu.cn

Abstract. Timed Statecharts, which can efficiently specify explicit dense time, is an extension to the visual specification language Statecharts with real-time constructs. We give a definition of timed Statecharts that specifies explicit temporal behavior as timed automata does. It is very difficult to verify directly whether timed Statecharts satisfies the required properties. However, by compiling it into timed automata, timed Statecharts may be checked using UPPAAL tool. In the paper, the state of timed Statecharts is represented by inductive term, and a step semantics of timed Statecharts is briefly described. The translation rules are shown by a compositional approach for formalizing the timed Statecharts semantics directly on sequences of micro steps. Timed automata corresponding to timed Statecharts was also discussed.

1 Introduction

Statecharts [1] is a visual language for specifying the behavior of complex reactive system. The formalism extends traditional finite state machines with notions of hierarchy, concurrency, and priority. In short, one can say: Statecharts = state-diagrams + depth + orthogonality + broadcast-communication. Now there also exists many related specification formalisms such as Modecharts [2] and RSML [3]. Statecharts is the most important UML component specifying complex reactive system such as communication protocol and digital control unit.

Statecharts, a synchronous visual modeling language, adopts *fictitious clock model* that only requires the sequence of integer times to be non-decreasing. All components are driven by common global clocks, called *tick* clock. However, it is not sufficient to specify time-critical systems with fictitious clock. Statecharts has to face the problems that it can't specify the required temporal behavior as timed automata does. In order to efficiently specify explicit dense time, Statecharts is extended with real-time constructs, including clocks, timed guards and invariants. The advantages of modeling complex reactive behavior with Statecharts are combined with the advantages of specifying temporal behavior with timed automata, resulting in the real-time extension of Statecharts; we call it *timed Statecharts*.

Model checking [4] is an automatic technique for verifying finite state reactive systems. In order to verify whether a timed Statecharts model satisfies the required properties, we present a model checking algorithm for timed Statecharts. Just as verifying

F. Wang (Ed.): FORTE 2005, LNCS 3731, pp. 261 – 274, 2005.

Statecharts, we first flat timed Statecharts and then apply a model checking tool to verify the resulting model. The translation rules that compile timed Statecharts into an equivalence timed automata are discussed by a compositional approach which formalizes the timed Statecharts semantics directly on sequences of micro steps and describes parallel behavior by process algebra.

Relate work. In the past two decades, model checking, which was first introduced for ordinary finite-state machines in Clarke and Emerson [5], has emerged as a promising and powerful approach to fully automatic verification of systems. Given a state transition system and a property, model checking algorithms exhaustively explore the state space to determine whether the system satisfies the property. The result is either a claim that the property is true or else a counterexample failing to the property.

It was very successful for the Statecharts language to specify reactive systems by its intuitive syntax and semantics. Since the original formalism of Harel, the theory of Statecharts has been under an extensive research and many different semantic approaches evolved from the academic world [6][7][8][9][10][11]. But for timed Statecharts, only hierarchical timed automata with an operational semantic to analyze timed Statecharts was discussed in [18][19].

Extended Hierarchical Automata, as the structural basis of Statecharts semantics, were introduced in [12] for Statemate and in [13] for UML. It translates Statecharts into PROMELA that is the input language of the SPIN model checker to perform the verification. Gnesi [14] uses a formal operational semantics for building a labeled transition system which is then used as a model to be checked against correctness requirements expressed in the action based temporal logics ACTL. In their reference verification environment JACK, automata are represented in a standard format, which facilitates the use of different tools for automatic verification. Pap [15] describes methods and tools for automated safety analysis of UML Statecharts specifications. Chan [16] and Schmidt [17] also contribute to mode checking for Statecharts. David [18] gives a formal verification of UML Statecharts with real-time extensions using hierarchical timed automata, while our method is to translate directly timed Statecharts to flat timed automata that can be used in UPPAAL.

The remainder of this paper is organized as follows. The next section introduces timed automata and its operational semantics, and section 3 defines timed Statecharts and its terms. Section 4 formulates a step semantics. Section 5 formalizes our compositional semantics and gives our translation rules from timed Statecharts to timed automata. Finally, section 6 provides our conclusions.

2 Timed Automata

Timed automaton [20] is an extended automaton to model the behavior of real time system over time. We consider a variant of timed automata without accepting states. The next subsection gives the operational semantics of the automata.

DEFINITION 1. (Clock) A clock is a variable ranging over R^+, the set of non-negative real numbers.

Let C be a finite set of variables called clocks. A clock valuation is a function that assigns a non-negative real-value to every clock. The set of valuations of C, denoted

V_C, is the set $[C \to R^+]$ of total mappings from C to R^+. Let $v \in V_C$ and $t \in R^+$, the clock valuation $v+t$ denotes that every clock is increased by t with respect to valuation v. It is defined by $(v+t)x=v(x)+t$ for every clock $x \in C$.

Definition 2. (Clock constraints) For set C of clocks with $x, y \in C$, the set Ψ_C of clock constraints over C is defined by

$$\delta ::= x \prec c \mid x-y \prec c \mid \neg\delta \mid (\delta \wedge \delta)$$

where $c \in R^+$ and $\prec \in \{<, \leq\}$

Clock constraints are evaluated over clock valuations. For $x, y \in C$, $v \in V_C$ and let α, $\beta \in \Psi_C$ we have

- $v \models x \prec c$ iff $v(x) \prec c$
- $v \models x-y \prec c$ iff $v(x)-v(y) \prec c$
- $v \models \neg\alpha$ iff $v \not\models \alpha$
- $v \models \alpha \wedge \beta$ iff $v \models \alpha$ and $v \models \beta$

Definition 3. (Timed automaton) A timed automaton is a tuple TA = $(S, C, s_0, L, Inv, \to)$ where: S is a finite set of states, C a finite set of time clocks, $s_0 \in S$ an initial state, L a set of labels, Inv: $S \to \Psi_C$ a function that associates a timing constraint to each state, called state invariant, $\to \in S \times (L \times \Psi_C \times 2^C \times \{\textbf{true}, \textbf{false}\}) \times S$ a set of transitions, where a transition $t=(s, e, g, r, u, s')$ connects a source state s and a target state s' with label e, timing constraint guard g, clock resets r and urgency flag u.

The function Inv associates a time constraint to each state $s \in S$, i.e., the automaton can stay in the state only while the current time clock valuation satisfies $Inv(s)$. The state invariant forces the automaton to translate before it becomes **false**, so that it avoids the automaton to get stuck at the state s. when the time constraint g associated to the edge is satisfied by current values of time clocks, the automaton may perform a translation.

The transition system underlying timed automaton TA, denoted $\mathcal{M}$(TA), be defined as $(Q, q_0, \to)$ where:

- $Q=\{(s, v) \in S \times V_C \mid v=Inv(s)\}$;
- $q_0=(s_0, v_0)$ where $v_0(x)=0$ for $x \in C$;
- The transition relation of timed automaton $t \in Q \times (L \times \Psi_C \times 2^C \times \{\textbf{true}, \textbf{false}\}) \times Q$, which describes how to evolve from one state to another, is defined by the following rules:
  - $(s_i, v) \xrightarrow{*} (s_j, \textbf{reset } R \textbf{ in } v)$ if the following conditions hold:
    i. $t=<s_i, E, A, G, R, u, j>$;
    ii. E are satisfied;
    iii. $v \models G$;
    iv. $(\textbf{reset } R \textbf{ in } v) \models Inv(s_j)$;
  - $(s_i, v) \xrightarrow{d} (s_i, v+d)$, for positive real d, if the following condition holds:
    i. $\forall d' \leq d, v+d' \models Inv(s_i)$;
    ii. $\neg\textbf{urgent}(t)$, the urgency flag of transition $\hat{t}$ is **false**.

where clock valuation **reset** R **in** v, valuation v with clock x reset, is define by:

$$(\textbf{reset } x \textbf{ in } v)(y) = \begin{cases} v(y) & \text{if } x \neq y \\ 0 & \text{if } x = 0 \end{cases}$$

3 Timed Statecharts

In this section we firstly define the formal syntax of timed Statecharts and give a simple example of a timed Statecharts, and then represent timed Statecharts state not visually but by terms. A timed Statecharts is in fact a Statecharts equipped with a set of real-valued clocks. Clocks are used to precisely measure the elapse of time between events.

3.1 Timed Statecharts Definition

Definition 4. (Time Statecharts) A timed Statecharts is an eight tuple TS=($\mathcal{N}$, $\mathcal{N}_0$, σ, type, C, I, $\mathcal{L}$, T), where:

1. a finite set $\mathcal{N}$ of states.
2. a subset $\mathcal{N}_0 \subseteq \mathcal{N}$ of initial states.
3. $\sigma : \mathcal{N} \to 2^{\mathcal{N}}$, $\sigma(n)$ gives the sub-states of n which are called sons of n, σ defines a tree structure.
4. type: $\mathcal{N} \to$ {AND, OR, BASIC} is the type function.
5. a finite set C of clocks.
6. Inv: $\mathcal{N} \to \Psi_C$, a function that assigns to each state an invariant.
7. A set of transition labels $\mathcal{L}$, partitioned into two disjoint sets $\mathcal{L}=L_T \cup L_E$, where $L_T \subseteq \Psi_C \times 2^C \times \mathcal{U}$ represents a set of clock constraints label, where $\mathcal{U}$={**true**, **false**} is a set of urgency flag; and $L_E \subseteq Event \times Cond \times Action$ a set of unclock constraints label, where *Event* is a set of event, *Cond* a set of condition, *Action* a set of action..
8. $T \subseteq \mathcal{N} \times \mathcal{L} \times \mathcal{N}$ is a set of transition relation, where a transition $t=(n, e, c, a, g, r, u, n')$ connects two states n and n', and have a source state n, a target state n', a event e, a condition c, a action a, a guard g, an clock resets r and an urgency flag u.

Properties of σ which assure the well-formed tree structure are:

- disjoint super-states: if $n \neq n'$ then $\sigma(n) \cap \sigma(n') = \emptyset$;
- no recursion: if $n' \in \sigma^*(n)$ then $n \notin \sigma(n')$;
- *root* has no ancestor: $\forall n \in \mathcal{N}$, $root \notin \sigma(n)$;
- basic nodes are empty: type(n)=BASIC $\Leftrightarrow \sigma(n) = \emptyset$;
- sub-states of AND are not BASIC: type(n)=AND $\wedge\ n_1 \in \sigma(n) \Rightarrow \sigma(n_1) \neq \emptyset$;
- if type(n)=AND then there is no $n_1 \to n_2$ for all $n_1, n_2 \in \sigma(n)$;

A traditional Statecharts models the system as being in a number of states that describe its operations. A state can be considered a point in the computation. States are denoted by rectangles with rounded corners and transitions as arrows. A state can be BASIC, AND or OR. If a state is BASIC, it has no sub-states, called BASIC-state. An OR-state has sub-states and exactly one of them is active at a certain point of time.

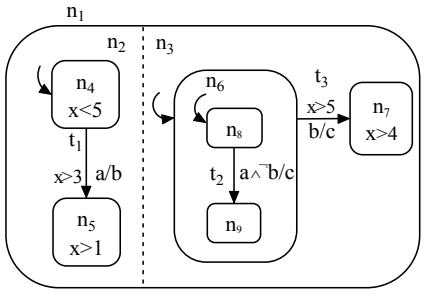

Fig. 1. A simple example of a timed Statecharts

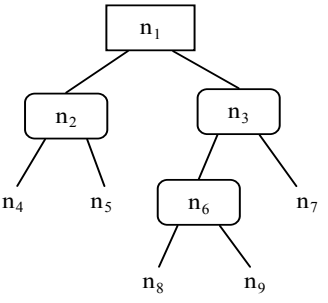

Fig. 2. A state hierarchical structure

An AND-state has OR sub-states, and all of them are active if the parent state is active.

Figure 1 shows a simple example of a timed Statecharts. The state labeled n_1 is split into two concurrent sub-states n_2 and n_3 by the dashed line through its middle. n_1 is called an AND-state because it has these orthogonal components. n_2 is decomposed into sub-states labeled n_4 and n_5 to indicate that the model can be in only one of those states at any time, so n_2 is an OR-state. When a state is not decomposed into AND or OR-states, it is called a BASIC state. The State n_4 has the time invariant $x<5$, invariant of the n_5 is $x>1$(x denotes clocks), and the two States are connected by transition t_1. In the simple case transitions are connected directly with a source and a target state. The transition t_1 is triggered by event a and timed constraint $x>3$.

If a state is entered, one direct sub-state is entered in the OR case and all direct sub-states are entered in the AND case. Exiting a state is analogous. AND, OR and BASIC states form a tree structure and this hierarchy allows for stepwise refinement of the behavior of complex systems. All states in the largest rounded corners rectangle come into being a hierarchical structure as a tree that is shown Fig. 2. State n_1 is an ancestor of State n_2 and n_3, while State n_2 and n_3 is an offspring of State n_1.

3.2 Statecharts Terms

For description convenience we assume that state and transition name of timed Statecharts are unique, clock invariants of OR-state and AND-state are always true, and also ignores interlevel transitions, i.e. transitions crossing borderlines of states. Timed Statecharts is represent by terms, as done in [6]. Formally, suppose $\mathcal{N}$ be a set of names for timed Statecharts states, $\mathcal{T}$ a set of names for timed Statecharts transitions, Π a countable set of timed Statecharts events, $\mathcal{G}$ a set of clocks constraints, $\mathcal{R}$ a set of clocks resets, an $\mathcal{U}=\{\textbf{true}, \textbf{false}\}$ represent a set of urgent flag. Inv is a set of invariant over timed Statecharts states. With every event $e\in\Pi$, we associate a negated counterpart $\bar{}e$ and $\bar{}\bar{}e=_{\text{def}} e$ as well as $\bar{}E=_{\text{def}} \{\bar{}e \mid e\in E\}$ for $E\subseteq\Pi\cup\{\bar{}e \mid e\in\Pi\}$. The set SC of timed Statecharts terms is then defined by the following inductive rules.

BASIC-state: If $n\in\mathcal{N}$, $\xi\in Inv$, then $s=[n,\xi]$ is a timed Statecharts term.

OR-state: Suppose $n \in \mathcal{N}$, and that $s_{1,\ldots,}s_k$ are timed Statecharts terms for k>0, with $\vec{s} =_{def} (s_{1,\ldots,}s_k)$. Also let $\rho =_{def} \{1,\ldots,k\}$ and $l \in \rho$, with $T \subseteq \mathcal{T} \times \rho \times 2^{\Pi \cup \neg \Pi} \times 2^{\Pi} \times \mathcal{G} \times \mathcal{R} \times \mathcal{U} \times \rho$. Then $s = [n: \vec{s}\ ; l; T]$ is a timed Statecharts term. Here $s_{1,\ldots,}s_k$ are the sub-states of s, set T contains the transitions connecting these states, s_1 is the default state of s, and s_l is the currently active sub-state of s.

AND-state: If $n \in \mathcal{N}$, if $s_{1,\ldots,}s_k$ are timed Statecharts terms for k > 0, and $\vec{s} =_{def} (s_{1,\ldots,}s_k)$, then $s = [n: \vec{s}\ ]$ is a timed Statecharts term, where $s_{1,\ldots,}s_k$ are the parallel sub-states of s.

$s_1 = [n_1: (s_1, s_2)]$
$s_2 = [n_2: (s_4, s_5); 1; \{t_1, 1, \{a\}, \{b\}, 2\}]$
$s_3 = [n_3: (s_6, s_7); 1; \{t_3, 1, \{b\}, \{c\}, 2\}]$
$s_6 = [n_6: (s_8, s_9); 1; \{t_2, 1, \{a \wedge \neg b\}, \{c\}, 2\}]$
$s_4 = [n_4, x<5]$ $s_5 = [n_5, x>1]$ $s_7 = [n_7, x>4]$
$s_8 = [n_8]$ $s_9 = [n_9]$

Fig. 3. Statecharts terms

Transitions of OR-states $[n: \vec{s}\ ; l; T]$ are those of the form $\hat{t} = <t, i, E, A, G, R, u, j>$, where (i) t is the name of $\hat{t}$, **name**($\hat{t}$) $=_{def} t$, (ii) **source**($\hat{t}$) $=_{def} s_i$ is the source state of $\hat{t}$, (iii) **ev**($\hat{t}$) $=_{def} E$ is the trigger of $\hat{t}$, (iv)**act**($\hat{t}$) $=_{def} A$ is the action of $\hat{t}$, (v)**guard**($\hat{t}$) $=_{def} G$ is the clock constraints of $\hat{t}$, (vi) **reset**($\hat{t}$) $=_{def} R$ is the clock resets of $\hat{t}$, (vii) **urgent**($\hat{t}$) $=_{def} u$ is the urgency flag of $\hat{t}$, and (viii) **target**($\hat{t}$) $=_{def} s_j$ is the target state of $\hat{t}$. The timed Statecharts term corresponding to the time Statecharts depicted in Fig. 1 is term s_1, which is defined in Fig. 3.

4 A Step Semantics of Timed Statecharts

The transition relation of timed Statecharts $\hat{t} \in \mathcal{T} \times \rho \times 2^{\Pi \cup \neg \Pi} \times 2^{\Pi} \times \mathcal{G} \times \mathcal{R} \times \mathcal{U} \times \rho$, which describes how to evolve from one state to another, is defined by the following rules.

- $(s_i, v) \xrightarrow{*} (s_j,$ **reset reset** $(\hat{t})$ **in** $v)$ if the following conditions hold:
  i. $\hat{t} = <t, i, E, A, G, R, u, j>$;
  ii. **ev**($\hat{t}$) are satisfied;
  iii. **guard**($\hat{t}$) $\models G$;
  iv. (**reset reset** ($\hat{t}$) **in** v) $\models Inv(s_j)$;
  v. **act**($\hat{t}$) are generated.
- $(s_i, v) \xrightarrow{d} (s_i, v+d)$, for positive real d, if the following condition holds:
  i. $\forall d' \le d$, $v+d' \models Inv(s_i)$;
  ii. $\neg$**urgent**($\hat{t}$), the urgency flag of transition $\hat{t}$ is **false**.

For BASIC-states of timed Statecharts, the transition relation is similar to the transition relation of timed automaton. However, in practice, we have to consider other property for timed Statecharts, such as hierarchy, concurrency and priority. Before defining

translation rules for timed Statecharts based on its operational semantics, we discuss classical Statecharts semantics as proposed by Pnueli and Shalev [7].

We sketch the semantics of timed Statecharts terms adopted in [8], which is a slight variant of the classical Statecharts operation semantics. A timed Statecharts s reacts to the arrival of some external events by triggering and clock constraints enabled micro steps in a chain-reaction manner. When this chain reaction comes to a halt, a complete macro step has been performed. More precisely, a macro step comprises a maximal set of micro steps, or transitions, that (i) are *relevant*, (ii) are mutually *consistent*, (iii) are triggered by events $E \subseteq \Pi$ offered by the environment or generated by other micro steps, (iv) satisfy clock constraints $G \subseteq \mathcal{G}$, (v) satisfy invariant of target state, (vi) are mutually *compatible*, and (vii) obey the principle of *causality*. Finally, we say that transition t is enabled in $s \rightarrow s'$ with respect to event set E, clock constraints G and transition set T, if $t \in \mathbf{En}(s, E, G, T, s')$, $s' \in \mathbf{target}(T)$, which is defined as follows.

$$\mathbf{En}(s, E, G, T) =_{\text{def}} \mathbf{relevant}(s) \cap \mathbf{consistent}(s, T) \cap (\bar{}\mathbf{invariant}(s) \vee (\mathbf{invariant}(s) \wedge \mathbf{urgent}(T))) \cap \mathbf{invariant}(s') \cap \mathbf{triggered}(s, (E \cup \bigcup_{t \in T} \mathbf{act}(t)) \cap G)$$

where:

- **relevant**(s) is the set of transitions whose source is in the set s;
- **consistent**(s, T) is the set of transitions that do not conflict with anything in T;
- **invariant**(s) represents that state s satisfy invariant;
- **urgent**(T) represents that the urgency flag of transition T is **true**;
- **triggered**(s,($E \cap G$) is the set of transitions whose triggers are satisfied by the event set E and clock constraint G. This is where global in consistency is eliminated;
- **act**(t) is the set of events generated by transition t.

Given a time Statecharts term s, a set E of events, and a set G of clock constraints, the non-deterministic *step-construction* function presented in Fig. 4 computes a set T^* of transitions. By executing the transitions in T^*, timed Statecharts term s may evolve in the single macro step $s \xrightarrow[A,R]{E,G} s'$ to timed Statecharts term s', producing the events $A = \bigcup_{t \in T^*} \mathbf{act}(t)$ and clock reset $R = \bigcup_{t \in T^*} \mathbf{reset}(t)$. term s' can be derived from s by updating the index l in every OR-state $[n: \vec{s}\ ; l; T]$ of s satisfying $t \in T^*$ for some $t \in T$.

```
function step_construction(s, E);
var T:=∅;
begin
  while T⊂En(s,E,G,T) do
     choose  t∈En(s,E,G,T)\T;
     T := T∪{t};
  od;
  return T
end
```

Fig. 4. A step-construction function

5 Model Checking for Timed Statecharts

A macro step of Timed Statecharts comprises a maximal set of micro steps. We directly define the semantics on sequences of micro steps, and use timed automaton as the semantics domain. Given a timed Statecharts TS, we translate TS to timed automata TA by a mapping Δ: TS$\rightarrow$TA, where TA-states model timed Statecharts terms, TA-labels describe unclock constraints labels L_E(i.e. event/action) of timed Statecharts, TA-clocks denote timed Statecharts clocks, TA-clock constraints express timed Statecharts clock constraints, TA-state invariants model timed Statecharts invariants, and TA-transitions is sequences of timed Statecharts micro steps.

5.1 Translation Rules for Time Statecharts Based on Operational Semantics

For convenience, we define $\vec{s}_{l\rightarrow s'} =_{\text{def}} (s_1,\ldots,s_{l-1},s',s_{l+1},\ldots, s_k)$ for all $1\le l\le$k and $s'\in$SC. Furthermore, we need function **default**: SC$\rightarrow$SC which sets the default state for given a Statecharts term s. **default**($[n,\xi]$)=$_{\text{def}}[n,\xi]$, **default**($[n\colon \vec{s}\ ;l;T]$) =$_{\text{def}}$ **default**(s_1), **default**($[n\colon \vec{s}\ ]$) =$_{\text{def}}$ $\bigcup_{1\le i\le k}$ **default**(s_i). Defining for function η: SC$\rightarrow\mathcal{N}$, γ: SC$\rightarrow Inv$, which sets the state and the invariant for given a Statecharts terms s. (i) $\eta([n,\xi])=\{\{n\}\}$, $\gamma([n,\xi])=\{\{\xi\}\}$; (ii) $\eta([n\colon \vec{s}\ ;l;T])= \bigcup_{1\le i\le k}\ \{\{n\}\cup q_i \mid q_i\in\eta(s_i)\}$, $\gamma([n\colon \vec{s}\ ;l;T])= \bigcup_{1\le i\le k}\ \{r_i \mid r_i\in\gamma(s_i)\}$; (iii) $\eta([n\colon \vec{s}\ ])=\{\{n\}\cup \bigcup_{1\le i\le k}\ q_i \mid q_i\in\eta(s_i)\}$, $\gamma([n\colon \vec{s}\ ])=\{\bigcap_{1\le i\le k} r_i \mid r_i\in\gamma(s_i)\}$.

However, it is practical and important to consider history states in OR-states. For recording a history state, we additionally define a flag of history state $\tau\in\{none, deep, shallow\}$. *None* means that history states are not considered. *Deep* means that the old active state of the or-state and the old active states of all its sub-states are restored. *Shallow* means that only the active state of the or-state is restored and that its sub-states are reinitialized as usual. The modification of function **default** that just has to replace function **default**(s) by function **default**(τ, s) is done by integrated a history mechanism. The terms **default**(*none*, s) and **default**(*deep*, s) are simply defined by **default**(s) and s, respectively. The definition of **default**(*shallow*, s) can be done along the structure of timed Statecharts terms as follows.

default(*shallow,* $[n,\xi]$) =$_{\text{def}}$ $[n,\xi]$
default(*shallow,* $[n\colon \vec{s}\ ;l;T]$) =$_{\text{def}}$ $[n\colon \vec{s}_{[l\rightarrow \text{default}(s_l)]}\ ;l;T]$
default(*shallow,* $[n\colon \vec{s}\ ]$) =$_{\text{def}}$ **default**(*shallow,* $\vec{s}\ $)

Transition relation $\rightarrow$ is defined by using SOS rules by Plotkin [21] as follows.

$$name\ \frac{premise}{conclusion}\ (side\ condition) \quad \text{as well as} \quad name\ \frac{premise}{conclusion}\ (side\ condition)$$

In this subsection, operational semantics of timed Statecharts transition in BASIC-states, AND-states and OR-states was defined. There are three rules about BASIC-states: **BAS-1** rule describes the execution from one BASIC-state to another, where **source**(t)=$[n,\xi]$, **target**(t)=$[n',\xi']$, if the event **ev**(t), the clock constraints **guard**(t) of

$t \in T$ and the invariants of the target state are satisfied, the transition be enabled, and the actions **act**(t) and the clock reset **reset**(t) are done.

$$\textbf{BAS-1}\ \frac{\mathrm{En}([n,\xi],E,G,T,[n',\xi'])}{[n,\xi]\xrightarrow[\mathrm{act}(t),\mathrm{reset}(t)]{\mathrm{ev}(t),\mathrm{guard}(t)}_{\mathrm{name}(t)}[n',\xi']}$$
$$\begin{pmatrix} t\in T,\mathrm{source}(t)=[n,\xi], \\ \mathrm{target}(t)=[n',\xi'], \\ (\textbf{reset}\ \mathrm{reset}(t)\ \textbf{in}\ v)\models\xi' \end{pmatrix}$$

BAS-2 rule describes the execution from one BASIC-state to one AND-state which is its brother. As noted above, for all super states (i.e. OR-state and AND-state), their state invariants are always true, but when an OR-state is entered, one direct sub-state is entered, and until a BASIC-state. So we need to consider state invariant which can get from function γ. **BAS-2** rule defines as follows.

$$\textbf{BAS-2}\ \frac{\mathrm{En}([n,\xi],E,G,T,[n':\vec{s};l;T])}{[n,\xi]\xrightarrow[\mathrm{act}(t),\mathrm{reset}(t)]{\mathrm{ev}(t),\mathrm{guard}(t)}_{\mathrm{name}(t)}[n:\vec{s}_{[l\to\mathrm{default}(\tau,s_l)]};l;T]}$$
$$\begin{pmatrix} t\in T,\mathrm{source}(t)=[n,\xi], \\ \mathrm{target}(t)=[n':\vec{s};l;T], \\ (\textbf{reset}\ \mathrm{reset}(t)\ \textbf{in}\ v)\models\gamma([n':\vec{s};l;T]) \end{pmatrix}$$

BAS-3 rule demonstrate the delay of BASIC-states, where $v+d$ stands for the current clock assignment plus the delay for all the clocks, we have

$$\textbf{BAS-3}\ \frac{(v+d)\models Inv([n,\xi])\wedge\neg\mathrm{urgent}(t)}{([n,\xi],v)\xrightarrow{d}([n,\xi],v+d)}$$
$$\left(\forall d'\le d, v+d'\models Inv([n,\xi])\right)$$

There are also three rules about OR-states: one rules describes the execution of a timed Statecharts transition $t \in T$ of an OR-state $[n:\vec{s}\ ;i;T]$. It defines that the OR-state with currently active sub-state s_i may change to OR-state $[n:\vec{s}_{[l\to\mathrm{default}(\tau,s_l)]};l;T]$ with currently active sub-state s_l as rule **OR-1**.

$$\textbf{OR-1}\ \frac{\mathrm{En}([n:\vec{s};i;T],E,G,T,[n:\vec{s};l;T])}{[n:\vec{s};i;T]\xrightarrow[\mathrm{act}(t),\mathrm{reset}(t)]{\mathrm{ev}(t),\mathrm{guard}(t)}_{\mathrm{name}(t)}[n:\vec{s}_{[l\to\mathrm{default}(\tau,s_l)]};l;T]}$$
$$\begin{pmatrix} t\in T,\mathrm{source}(t)=[n:\vec{s};i;T], \\ \mathrm{target}(t)=[n':\vec{s};l;T], \\ (\textbf{reset}\ \mathrm{reset}(t)\ \textbf{in}\ v)\models\gamma([n':\vec{s};l;T]) \end{pmatrix}$$

Other rule that describes from the OR-state $[n:\vec{s}\ ;l;T]$ to BASIC-state, is not discussed particularly due to similar to **BAS-1** rule. Another rule describes that the OR-state $[n:\vec{s}\ ;l;T]$ with currently active sub-state s_l may change with same label to the OR-state $[n:\vec{s}_{[l\to s_l']};l;T]$ with currently active sub-state s_l' as rule **OR-2**.

$$\textbf{OR-2}\ \frac{s_l \xrightarrow[A,R]{E,G}_L s_l'}{[n:\vec{s};l;T] \xrightarrow[A,R]{E,G}_L [n:\vec{s}_{[l\to s_l']};l;T]}$$

It is indispensable for transition rule of Statecharts AND-states to consider many enabled transitions to execute in parallel as rule **AND**. For AND-state's parallel description, we firstly introduce process algebra. Process algebra [22] is a powerful formal method for depicting algebra structure and analyzing parallel system. Basic process algebra (BPA) is a core in all process algebra theory. Basic process terms are built from atomic actions, alternative composition and sequential composition.

- An atomic action represents indivisible behavior, including event and action.
- The symbol · denotes sequential composition. The process term $p{\cdot}q$ executes p, and upon successful termination proceeds to execute q;
- The symbol + denotes alternative composition. The process term $p+q$ executes behavior of either p or q.

By appending merge ||, left merge ⌊ and communication merge |, BPA is extended to express process communication in parallel system. The merge || executes the two process terms in its arguments in parallel, the left merge ⌊ executes an initial transition of its first argument, and the communication merge | executes a communication between initial transitions of its arguments. The process term $p\|q$ executes p and q in parallel; analogously, $p\lfloor q$ executes restrictedly p in an initial transition; $p|q$ executes a communication p and q.

AND-state of Statecharts specifies the parallel behavior of reactive system. In Fig. 1, the AND-state n_1 comprises two concurrent sub-states n_2 and n_3. Suppose the state configuration is currently in n_4 and n_8, if the event a and b occurs, the transition t_1 and t_3 is enabled. Because transitions can be taken in the sub-states of an AND-state simultaneously, the transition t_1 and t_3 is executed in parallel, as written $t_1\|t_3$. Based on parallel axiom of process algebra, merge $t_1 \| t_3 = (t_1 \lfloor t_3 + t_3 \lfloor t_1) + t_1 \mid t_3$.

$$\textbf{AND}\ \frac{\begin{array}{c}(\forall m \in M : s_m \xrightarrow[A_m,R_m]{E_m,G_m}_{L_m} s_m') \\ \wedge(\forall i,j \in M : (\neg \mathrm{ev}(t_i)) \cap \mathrm{act}(t_j) = \varnothing)\end{array}}{[n:\vec{s}] \xrightarrow[\bigcup_{m\in M} A_m \cup \bigcup_{m\in M} R_m]{(\bigcup_{l\in H}(E_l \setminus \bigcup_{j=1}^{l-1} act(t_j))) \wedge \bigcup_{l\in H} G_l}_{\bigcup_{m\in M} L_m} [n:\vec{s}_1'] \| \ldots \| [n:\vec{s}_{|M|}']}$$

$$\left(\begin{array}{l} M \subseteq K = \{1,\ldots,k\},\ H = \{1,\ldots,|M|\}, \\ \mathrm{source}(L_m) = s_m, \mathrm{target}(L_m) = s_m' \end{array}\right)$$

When AND-state includes k OR sub-states, an execution of k transitions in parallel need be considered. As above-mentioned, we can define **AND** rule, which an execution of $|M|$ transitions in parallel in all sub-states s_m of AND-state $[n{:}\,\vec{s}\,]$ may be specified to $[n{:}\,\vec{s}_1'\,]\|\ldots\|[n{:}\,\vec{s}_{|M|}'\,]$ by merge of process algebra.

5.2 Macro Step

The above rules realize a compositional semantics of timed Statecharts on sequences of the micro steps. However, we consider even more the classical macro-step semantics of timed Statecharts. Let $s, s' \in \mathrm{SC}$, E, $A \subseteq \prod$, $G \subseteq \mathcal{G}$ and $R \subseteq \mathcal{R}$, we write $s \Rightarrow s'$ and say s may perform a macro step with input E, output A, clock constraints G and clock reset R to s', if $\exists s_1,\ldots,s_m \in \mathrm{SC}$, $\exists E_1,\ldots,E_\mathrm{m} \subseteq \prod \cup \neg \prod$, $\exists A_1,\ldots,A_\mathrm{m} \subseteq \prod$, $\exists G_1,\ldots,G_\mathrm{m} \subseteq \mathcal{G}$, $\exists R_1,\ldots,R_\mathrm{m} \subseteq \mathcal{R}$, such that (i) $s \xrightarrow[A_1,R_1]{E_1,G_1} s_1 \xrightarrow[A_2,R_2]{E_2,G_2} \ldots \xrightarrow[A_m,R_m]{E_m,G_m} s_m \to s'$, (ii) $\bigcup_{i=1}^{m} E_i \subseteq E$ and $\bigcup_{i=1}^{m} A_i \cap E = \varnothing$, (iii) $\mathrm{A} = \mathbf{act}(s_m) \cap \prod$, (iv) $\bigcup_{i=1}^{m} G_i \subseteq G$, (v) (**reset** R_i **in** v) $\models \xi_i$, $0 < i \le m$, (vi) $\nexists$ s_{m+1}, E_{m+1}, A_{m+1}, G_{m+1}, R_{m+1}, $s_m \xrightarrow[A_{m+1},R_{m+1}]{E_{m+1},G_{m+1}} s_{m+1}$, where $E_{m+1} \subseteq E$ and $A_{m+1} \cap E = \varnothing$. If timed Statecharts term s satisfies event E, clock constraints G, We may say, s may evolve in the single macro step $s \xrightarrow[A,R]{E,G} s'$ to timed Statecharts term s', generate action A and reset clock R.

5.3 Translate Time Statecharts into Timed Automata

Given timed Statecharts, it can be translated into timed automata by a mapping Δ: TS$\to$TA. To define the mapping function Δ, we firstly suppose a timed Statecharts p by terms,, and define the entities $S(p)$, $C(p)$, $L(p)$,$\to_p$ and $Inv(p)$, which mean respectively the states set, the clock set, the label set, the set of transition relation and the state invariant function of TA $\Delta(p)$, where:

- $S(p)$ is a set of state configurations of Statecharts term p. The definition of $S(p)$ can be done as follows.
  i. $S([n,\xi]) = \{\{\eta([n,\xi])\}\} = \{\{n\}\}$
  ii. $S([n{:}\,\vec{s}\,;l;T]) = \bigcup_{1 \le i \le k} \{\{\eta([n{:}\,\vec{s}\,;l;T])\} \cup q_i \mid q_i \in S(s_i)\}$
  iii. $S([n{:}\,\vec{s}\,]) = \{\{\eta([n{:}\,\vec{s}\,])\} \cup \bigcup_{1 \le i \le k} q_i \mid q_i \in S(s_i)\}$
- $C(p) = C$ a set of the timed Statecharts clocks;
- $L(p) = 2^{\Pi_p \cup \neg \Pi_p} \times 2^{\Pi_p}$ represents the set of timed Statecharts event and action, written event/action;
- $\to_p \subseteq S(p) \times L(p) \times \mathcal{G} \times \mathcal{R} \times S(p)$ that operation rules have already been discussed in the above, represent the sequence macro step of Statecharts. Assume a translation $e = (s, L, G, R, s')$ connects two states s and s', describes $s \xrightarrow[A_1,R_1]{E_1,G_1} s_1 \xrightarrow[A_2,R_2]{E_2,G_2} \ldots \xrightarrow[A_m,R_m]{E_m,G_m} s_m \to s'$, where $L = E_1 \wedge E_2 \wedge \ldots \wedge E_m / A_1 \vee A_2 \vee \ldots \vee A_m$, $G = G_1 \wedge G_2 \wedge \ldots \wedge G_m$, $R = R_1 \vee R_2 \vee \ldots \vee R_m$;
- $Inv(p)$: $S(p) \to \Psi_C$, a function that assigns to each state an invariant, where $Inv([n,\xi]) = \gamma([n,\xi])$, $Inv([n{:}\,\vec{s}\,;i;T]) = \gamma([n{:}\,\vec{s}\,;l;T])$, $Inv([n{:}\,\vec{s}\,]) = \gamma([n{:}\,\vec{s}\,])$
- $\varsigma(p)$ expresses the initial term set of timed Statecharts. We may define $\varsigma([n,\xi]) = \{n\}$, $\varsigma([n{:}\,\vec{s}\,;i;T]) = \{n\} \cup s_1$, $\varsigma([n{:}\,\vec{s}\,]) = \{n\} \cup \bigcup_{1 \le i \le k} s_i$.

Considering our example of timed Statecharts of Fig. 1, its translation TA of timed Statecharts is depicted in Figure 5.

Without loss of generality, we wish to consider interlevel transitions and clock invariants of OR-state and AND-state for timed Statecharts. Harel considers interlevel transitions as important concept of the language [1]: "…as our methods does not necessarily advocate layer-by-layer development; it is more flexible and encourages interlevel connections too, whenever appropriate." Hence we can not rule them out. This intricacy is mainly caused by interlevel transitions, but we wish to describe interlevel transitions but have simple operational semantics. It is feasible and practical to change from interlevel transitions to non-interlevel transitions. Our approach that is similar to [12] is lift interlevel transitions to the uppermost states that are exited and entered when transitions is taken. Let $sr(t)$ (called *source restriction*) is a set of states which were the original states of the transition t, and $td(t)$ (called *target determinator*) is a set of states that were entered originally.

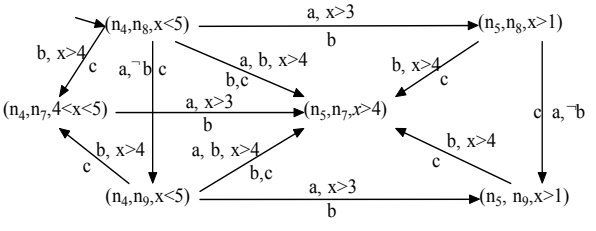

Fig. 5. Translation Timed Automata for timed Statecharts in Fig. 1

By transition label extensions added $sr(t)$ and $td(t)$, interlevel transitions can be compiled into non-interlevel transitions.

In the following, we will describe how to eliminate clock invariants for super state of timed Statecharts. Assume that two state of timed Statecharts n_1, $n_2 \in \mathcal{N}$, n_1 be a super state, i.e. type(n_1)=AND or type(n_1)=OR, and n_2 may be a arbitrary type, including AND, OR and BASIC, and let $n_2 \in \sigma(n_1)$. According to the priority of transitions for timed Statecharts, we define the priority of state invariant that if sub-state n_2 invariant is satisfied but father-state n_1 invariant is not, then the current state configuration can not be in n_2, i.e. clock invariant of state n_1 is prior to clock invariant of sub-state n_2. In order to let that clock invariants of OR-state and AND-state always are true, only clock invariant of sub-states n_2 need be updated such as $Inv(n_2)=Inv(n_2) \wedge Inv(n_1)$. More precisely, we define formally as follows.

– $\forall n \in \mathcal{N}$, and type($n$)=AND or type($n$)=OR, $Inv(n)$=true;
– $\forall n \in \mathcal{N}$, $n \in \sigma^k(root)$, type(n)=BASIC, $Inv(n)=Inv(n) \wedge \bigcup_{1 \le i \le k} \sigma^{-i}(n)$.

where *root* is a unique root state and has no ancestor. $\sigma^k(root)=\sigma(\sigma^{k-1}(root))$, σ^{-1} that gives the father-state is a inverse of σ, and $\sigma^{-k}(n)=\sigma^{-1}(\sigma^{-(k-1)}(n))$.

5.4 Model Checking Timed Statecharts

Given a timed Statecharts TS, and TCTL formulae ϕ, the model checking timed Statecharts problem that we are interested in is to check whether TS satisfies ϕ, abbreviated TS $\models \phi$. According to the last translation rules, the equivalence model TA

that results from timed Statecharts TS is called its timed automata. Thus, roughly speaking, model checking timed Statecharts against a TCTL-formula amounts to model checking its timed automata against a TCTL-formula. Formally, for any timed Statecharts TS, we have:

$$TS \models \phi \text{ if and only if } TA \models \phi$$

In summary we obtain the scheme for model checking the TCTL-formula ϕ over the timed Statecharts TS:

1. Construct the flat timed automata model TA =(S, C, s_0, L, Inv, $\rightarrow$);
2. The model checking problem for TCTL, deciding whether TA, $s_0 \models \phi$, can be solved by constructing the region automaton $\mathcal{R}$(TA) under the time equivalence classes under $\approx$;
3. Apply the CTL model checking procedure on $\mathcal{R}$(TA).

Actually, the problem for model checking timed Statecharts can be converted to the classical problem for model checking timed automata [23][24].

6 Conclusion

Timed Statecharts is an extension of the visual specification language Statecharts with real-time constructs, and can efficiently specify explicit dense time. The timed Statecharts serves better the modeling of complex reactive real-time systems. The paper presented a new approach for formalizing timed Statecharts semantics, which is centered on the compositional principle. Based on timed Statecharts term syntax and formal operational semantics, and description of parallel behavior by process algebra, each timed Statecharts is mapped to a timed automaton. This makes it possible to translate our hierarchical structure to a flat one and thus provide a framework for formal verification of a real-time extension of Statecharts.

References

1. D. Harel. Statecharts: a Visual Formalism for Complex Systems. Science of Computing, 8(1987) 231-274
2. F. Jahanian and A.K. Mok. A Graph-theoretic Approach for Timing Analysis and its Imple-mentation. IEEE Transactions on Computers, C-36(1987) 961~975
3. Leveson NG, Heimdahl M, Hildreth H, Reese JD. Requirements Specification for Process-Control Systems. IEEE Transactions on Software Engineering, 20(1994) 684~707
4. E.M. Clarke, O. Grumberg, D.A. Peled. Model Checking. The MIT Press (2000)
5. E. M. Clarke, E. A. Emerson. Synthesis of synchronization skeletons for branching time temporal logic. In Logic of Programs: Workshop, Yorktown Heights. NY, LNCS 131, Springer-Verlag, (1981) 52-71
6. D. Harel, A. Pnueli, J.P. Schmidt, and R. Sherman. On the Formal Semantics of Statecharts. In Proceedings of the 2nd IEEE symposium on Logic in Computer science, Ithaca, New York, (1987) 54-64

7. G. Lüttgen, M. von der Beeck, and R. Cleaveland. Statecharts via Process Algebra. In 10th International Conference on Concurrency Theory(CONCUR '99), J. Baeten and S. Mauw, eds., Vol. 1664 of Lecture Notes in Computer Science, Eindhoven, The Netherlands, Springer-Verlag (1999) 399-414
8. A. Maggiolo-Schettini, A. Peron, and S. Tini. Equivalences of Statecharts. In 7th International Conference on Concurrency Theory (CONCUR '96), U. Montanari and V. Sassone, eds., Vol. 1119 of Lecture Notes in Computer Science, Pisa, Italy, Springer-Verlag (1996) 687-702
9. R. Heckel, J. Kuster, and G. Taentaer. Towards Automatic Translation of UML Models into Semantic Domains. In Proc. AGT 2002: Workshop on Applied Graph Transformation, Grenoble, France, (2002) 11-21
10. G. Lüttgen, M. von der Beeck, and R. Cleaveland. A Compositional Approach to Statecharts Semantics. NASA/CR-2000-2100086, ICASE Report No. 2000-12 (2000)
11. A. Pnueli and M. Shalev. What is in a Step: on the Semantics of Statecharts. In Theoretical Aspects of Computer Software (TACS '91), T. Ito and A. Meyer, eds., Vol. 526 of Lecture Notes in Computer Science, Sendai, Japan, Springer-Verlag (1991) 244-264
12. E. Mikk, Y. Lakhnech, and M. Siegel et al. Implementing Statecharts in PROMELA/SPIN. In: Proc of Workshop on Industrial-Strength Formal Specification Techniques(WIFT'98). BocaRaton, Florida: IEEE Computer Society (1998)
13. D. Latella, I. Majzik, and M. Massink. Automatic Verification of UML Statechat Diagrams Using the SPIN Model-checker. Formal Aspects of Computing, 11(1999): 637-664.
14. S. Gnesi, and D. Latella. Model Checking UML Statechart Diagrams Using JACK. In Proceedings of the 4th IEEE international Symposium on High-Assurance Systems Engineering (1999) 46-55
15. Z. Pap, I. Majzik, and A. Pataricza. Checking General Safety Criteria on UML Statecharts. In U. Voges, editor, SAFECOMP 2001, LNCS 2187, Springer-Verlag, (2001) 46-55
16. W. Chan, R. Anderson, P. Beame, and S. Burns et al. Model Checking Large Software Specifications. IEEE Transactions on Software Engineering, 24(1998) 498-520
17. A. Schmidt and D. Varro. CheckVML: A Tool for Model Checking Visual Modeling Languages. In the 6th International Conference on the Unified Modeling Language, LNCS 2863, Springer-Verlag, (2003) 92-95.
18. A. David, M. Oliver Möller, and Wang Yi. Formal Verification of UML Statecharts with Real-Time Extensions. In Proceedings of the 5th International Conference on Fundamental Approaches to Software Engineering, Vol. 2306 of Lecture Notes in Computer Science, Springer-Verlag, (2002) 218-232
19. H. Giese and S. Burmester. Real-Time Statechart Semantics. Technical Report tr-ri-03-239, Computer Science Department, University of Paderborn, (2003)
20. R. Alur and D. Dill. A Theory of Timed Automata. Theoretical Computer Science, 126(1994) 183-235.
21. G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, (1981)
22. W. Fokkink. Introduction to Process Algebra. Springer, (2000)
23. R. Alur, C. Courcoubetis and D. Dill. Model Checking in Dense Real-time. Information and Computation, 104(1993) 2-34
24. S. Yovine. Model Checking Timed Automata. In Embedded Systems, LNCS, 1494, (1998)