김일성종합대학학보
(자연과학)
주체103(2014)년 제60권 제6호

JOURNAL OF KIM IL SUNG UNIVERSITY
(NATURAL SCIENCE)
Vol. 60 No. 6 JUCHE103(2014).

A Research on Operational Semantics of UML Statechart Diagrams

Sin Chun Ok, Pak Chol Jin

The first step in application of model checking to UML design is generation of verification model by defining its formal semantics and implementing semantics mapping [1 — 3]. Therefore, we have considered operational semantics of UML state diagram in this paper.

First, we abstracted UML state diagrams as hierarchical labeled transition system (LTS) and justified its operational semantics as deduction system on Kripke structure.

Hierarchical LTS is expressed by LTS's set and refinement function that characterized those hierarchy and concurrency

Definition 1 LTS is 4-tuple LTS $A=(States_A, S_A^0, Labels_A, Tr_A)$, where $States_A$ is finite set of states starting from S_A^0, $Labels_A$ is finite set of transition labels and $Tr_A \subseteq States_A \times Labels_A \times States_A$ is transition relation. Label $l_t \in Labels_A$ for a transition $t \in Tr_A$ is triple (ev, g, ac). ev is the trigger event, g is the guard, ac is the actions list.

We use the following functions *source, target, event, guard, action* for transition $t=(s, (ev, g, ac), s')$; *source*(t)=s, *target*(t)= s', *event*(t)=ev, *guard*(t)=g, *action*(t)=ac. But there are the following functions: function *sr(t)* constraint to source state, function *tg*(t) giving to destination state

Definition 2(Hierarchical LTS) A hierarchical LTS is a $H=(F, R, rf, L)$,*where* F is LTS's set, i.e. $\forall A_1, A_2 \in F : States_{A_1} \cap States_{A_2} = \emptyset$ and E is a finite set of event ev.

The refinement function $rf : \bigcup_{A \in F} States_A \to 2^F$ imposes a tree structure to F.

(i) there exists a unique root LTS $A_{\text{root}} \notin \bigcup \text{rng} rf$.

(ii) every non-root LTS has exactly one ancestor state: $\bigcup \text{rng} rf = F \setminus \{A_{root}\}$ and: $\forall A \in F \setminus \{A_{root}\}: \exists_1 s \in \bigcup_{A' \in F \setminus \{A\}} States_{A'} : A \in (rf(s))$.

(iii) there are no cycles:

$$\forall S \subseteq \bigcup_{A \in F} States_A : \exists s \in S : S \cap \bigcup_{A \in F} States_A = \phi, \quad L = \bigcup_{A \in F} Labels_A .$$

A is the root for LTS that is the hierarchical LTS for a refinement function *rf*.
S whose state is $rf(s) = \emptyset$ is called a basic state.

For $A \in F$, LTS, *states*, *labels* and transitions under A are defined respectively as

$$\Gamma_A = \{A\} \cup (\bigcup_{A' \in (\bigcup_{s \in State_A} rf_A(s))} \Gamma_{A'}), \qquad \Theta_A = \bigcup_{A' \in \Gamma_A} States_{A'},$$

$$\Lambda_A = \bigcup_{A' \in \Gamma_A} Labels_{A'}, \qquad \Delta_A = \bigcup_{A' \in \Gamma_A} Tr_{A'}$$

We consider LTS as partial hierarchical LTS $subH_A$ of H characterized by LTS or A.

We define the notions of conflicting transitions, transition priority and orthogonal states as follows to model state transition.

Definition 3 (state precedence) For $s,\ s'\in\Gamma_H$, $s\prec s'\Leftrightarrow s'\in\Gamma_{rf(s)}$. Let also $\leq$ denote the reflexive closure of $\prec$.

Definition 4 (conflicting transition) For $t,\ t'\in(\Delta_H)$, t is conflicting with t', written $t\#t'$, if $t\neq t'$ and $((source(t)\leq source(t'))\vee(source(t')\leq source(t)))$.

Definition 5 (priority schema) A Priority Schema is a triple $(\Pi,\lhd,\pi)$ with $(\prod,\lhd)$ a partial order and $\pi:\Delta_H\to\Pi$ such that: $\forall t,\ t'\in(\Delta_H):\pi(t)\lhd\pi(t')\wedge t\neq t'\Rightarrow t\#t'$.

Definition 6 (orthogonal state) Two states $s,\ s'\in\Theta_H$ are orthogonal, written $s\parallel s'\Leftrightarrow\exists s''\in(\Theta_H):A,\ A''\in(rf(s'')):A\neq A'\wedge s\in\Theta_A\wedge s'\in\Theta_{A'}$.

It is based on state precedence and generalizes the requirement that transitions originating from "inner" states have priority over those higher in the state hierarchy as required in UML statechart diagrams.

So we considered the followings.

H of orthogonal states: $\{S\subseteq(\Theta_H)\mid\forall s,\ s'\in S:(s\neq s'\Rightarrow s\parallel s')\}$ and $\leq^S$ is the of all the sets of pair

As $f(t)=\{s\mid s\in(source(t)\wedge sr(t)=\emptyset\}\cup(sr(t))$ is the priority function assigning priority to transitions, it comes $\forall t,\ t'\in(\Gamma_H.(f(t)\leq^S f(t')\wedge t\neq t'\Rightarrow t\#t'))$.

Δ semantics of UML state chart

By partial LTS's local state of a hierarchical LTS we define the global state of LTS

Definition 7 (state configuration) State configuration of hierarchical H is the set $C\subseteq(\Theta_H)$ satisfying the following condition.

① $\exists_1 s\in States_{Aroot}:s\in C$

② $\forall s,\ A:s\in C\wedge A\in rf(s)\Rightarrow\exists_1 s'\in A:s'\in C$

Set of all state configuration of A are denoted as $Conf_A$ for $A\in F$.

To abstract event pools, we provide three pools, i.e., completion event pool (ε_c), deferred event pool ($\varepsilon_{\mathrm{def}}$) and normal event pool (ε_n) and when dispatching events, the three event pools are inspected in order $\varepsilon_c\prec\varepsilon_{\mathrm{def}}\prec\varepsilon_n$.

We use the two functions, $outeven(\varepsilon)$ and $merge(e,p)$ for $\varepsilon=\varepsilon_c\cup\varepsilon_{\mathrm{def}}\cup\varepsilon_n$.

$$outeven(\varepsilon)=\begin{cases}outeven(\varepsilon_c) & \text{if } \varepsilon_c=\emptyset\\ outeven(\varepsilon_{\mathrm{def}}) & \text{if } \varepsilon_c=\emptyset\wedge\varepsilon_{\mathrm{def}}\neq\emptyset\\ outeven(\varepsilon_n) & \text{if } \varepsilon_c,\ \varepsilon_{\mathrm{def}}=\emptyset\wedge\varepsilon_n\neq\emptyset\end{cases}$$

$$merge(e,\ \varepsilon_x)=\varepsilon_x\wedge e$$

where ε_x is one of ε_c, $\varepsilon_{\mathrm{def}}$, ε_n and $\wedge$ represents the operation merging event e to ε_x.

In the above concepts we define the operational semantics of UML statechart modeled with the hierarchical LTS by the Kripke structure reflecting the transition relation.

The state of UML statechart is defined by the state configuration of hierarchical LTS and environment ε that the hierarchical LTS is acting mutually.

Definition 8 The operational semantics of an hierarchical LTS H is Kripke structure $K_H = (\mathrm{S},\ \mathrm{s}^0, \xrightarrow{\Delta})$, where

(i) $S = Conf_H \times \varepsilon$ is the set of statuses of $\mathrm{K_H}$,

(ii) $s^0 = (\beta_0,\ \varepsilon_0) \in S$ is the initial status,

(iii) $\xrightarrow{\Delta} \subseteq S \times S$ denote a maximal set of non-conflicting transitions of LTS of H with respect to priorities.

Promise: The relation $A \& P :: (\beta,\ \varepsilon) \xrightarrow{\Delta} (\beta',\ \varepsilon')$ models labeled transition of the hierarchical LTS and Δ denote the transitions of A's LTSs

For state s and set $S \subseteq \Gamma_{rf(s)}$, such that $s \le s''$ for all $s'' \in S$, the closure of S, $\mathrm{c}(s, S)$, is defined as the set $\{s' \mid \exists s' \in S : s \le s' \le s''\}$

The predicate $is\_join^n_{j=1}\varepsilon_j\ G$ states that G is a possible join of $\varepsilon_1, \cdots, \varepsilon_n$, $(New\ r)$ is the structure containing elements of r.

Definition 9 (enabled transition set) Set $Enabled_A(\beta,\ \varepsilon)$ of transitions enabled in the system state $(\beta,\ \varepsilon)$ is defined as following.

$$Enabled_A(\beta,\ \varepsilon) = \{t \in State_A \mid \{source(t)\} \cup (sr(t)) \subseteq \beta \wedge event(t) \in \varepsilon \wedge (\beta,\ \varepsilon) \models guard(t)\}$$

And when considering A as a hierarchical LTS, set of all enabled transitions of A in (β, ε) including A's descendants $AllEnabled_A(\beta,\ \varepsilon)$ is defined as follows

$$AllEnabled_A(\beta,\ \varepsilon) = \bigcup_{A' \in \Gamma} Enabled_{A'}(\beta,\ \varepsilon).$$

The inference rules of $\mathrm{K_H}$ are as follows

① Progress rule

$$\frac{\begin{array}{l} e = outeven(\varepsilon),\ \varepsilon' = \varepsilon \setminus \{e\} \\ t \in Enabled_A(\beta,\ \varepsilon) \qquad (1) \\ \neg \exists t' \in P \cup AllEnabled_A(\beta,\ \varepsilon) : \pi(t) \lhd \pi(t') \qquad (2) \end{array}}{A \& P :: (\beta,\ \varepsilon) \xrightarrow{\{t\}} (\mathrm{c}(taget(t),\ (td(t)),\ merge(new(action(t),\ \varepsilon')}$$

② Composition rule

$$\frac{\begin{array}{l} e = outeven(\varepsilon),\ \varepsilon' = \varepsilon \setminus \{e\} \\ \{s\} = \beta \cap State_A \qquad (1) \\ rf_A(s) = \{A_1,\ \cdots,\ A_n\} \neq \emptyset \qquad (2) \\ (\wedge^n_{j=1} A_j \ \&\ (P \cup Enabled_A(\beta,\ \varepsilon) :: (\beta,\ \varepsilon) \xrightarrow{\Delta_j} (\beta_j,\ \varepsilon_j)) \wedge isjoin(^n_{j=1}\varepsilon_j,\ \varepsilon'') \qquad (3) \\ (\bigcup^n_{j=1}\Delta_j = \emptyset) \Rightarrow (\forall t \in Enabled_A(\beta,\ \varepsilon).\exists t' \in P : \pi(t) \lhd \pi(t') \qquad (4) \end{array}}{A \& P :: (\beta,\ \varepsilon) \xrightarrow{\bigcup^n_{j=1}\Delta_j} (\{s\} \cup \bigcup^n_{j=1}\beta_j,\ \varepsilon'')}$$

③ Stuttering rule

$$e = outeven(\varepsilon),\ \varepsilon' = \varepsilon \setminus \{e\}$$
$$\{s\} = \beta \cap State_A \quad (1)$$
$$rf_A(s) = \emptyset \quad (2)$$
$$\forall t \in Enabled_A(\beta,\ \varepsilon)\exists t' \in P : \pi(t) \lhd \pi(t') \quad (3)$$
$$A \,\&\, P :: (\beta,\ \varepsilon) \xrightarrow{\emptyset} (\{s\},\ \varepsilon')$$

The progress rule establishes that if there is a transition of A enabled and the priority of such a transition is high enough then the transition fires and accordingly reaches a new status.

The composition rule stipulates how LTS delegates the execution of transitions to its sub-LTS and these transitions are propagated upwards.

If there is no transition of A enabled with priority that is "high enough" and no sub-LTS exist to which the execution of transitions can be delegated, then A has to "stutter", as enforced by the stuttering rule.

Theorem $A \,\&\, P :: (\beta,\ \varepsilon) \xrightarrow{\Delta}$ if and only Δ is a maximal set, under inclusion, which satisfies all the following properties:

(1) $\forall t,\ t' \in \Delta : \neg t \# t'$

(2) $\Delta \subseteq AllEnabled_A(\beta,\ \varepsilon)$

(3) $\forall t \in \Delta : \neg\exists t' \in AllEnabled_A(\beta,\ \varepsilon) : \pi(t) \lhd \pi(t')$

(4) $\forall t \in \Delta : \neg\exists t' \in P : \pi(t) \lhd \pi(t')$

References

[1] M. E. Beato et al.; Electronic Notes in Theoretical Computer Science, 127, 4, 3, 2005.

[2] S. Zhang et al.; In 4th International Conference on Secure Software Intergration and Reliability Improvement Companion, IEEE Computer Society, 1～6, 2010.

[3] Steffen Helke; Formalizing Statecharts using Hierarchical Automata, May, 24, 2012.