The Formal Semantics of an UML Activity Diagram

LIANG Yi-zhi (梁义芝)\(^1\), WANG Yan-zhang (王延章)\(^1\), LIU Yun-fei (刘云飞)\(^3\)
1. Institute of Information and Decision Technology, Dalian University of Technology, Dalian 116024, P. R. China
2. Combat Software Research Center, Dalian Naval Academy, Dalian 116018, P. R. China
3. Education Technology Center, Dalian Naval Academy, Dalian 116018, P. R. China

Abstract Due to lack of strictly defined formal semantics, an UML activity diagram is unsuitable for the tasks of formal analysis, verification and assertion on the system it describes. In this paper, Petri net is used to define the formal semantics of an UML activity diagram containing object flow states, laying a foundation for the precise description and analysis of a workflow system.

Key words UML activity diagrams, formal semantics, system analysis.

1 Introduction

An UML activity diagram is a special state machine intended to model computations and workflows\(^1\). Among the five diagrams supported by UML to describe the behavior of a dynamic system, the activity diagram is the most suitable one for modeling a software process\(^2\). The UML activity diagram emphasizes the control flow from activity to activity, it also represents the flow of object values. An object flow state represents an object, which can be the input or output of an activity.

The formal definitions of an UML state machine have so far neither taken into account the characteristics of an activity diagram, nor included object flows\(^3\).\(^4\).\(^5\).\(^6\). The formal definition of an UML activity diagram in Refs. \([7, 8]\) has not given much significance to objects in an activity diagram, and thus is very difficult to model computations and workflows. Therefore, we give a formal semantics of an UML activity diagram, and fully take into account the role an object flow plays in an activity diagram. Based on the formal semantics, we analyzed the correctness of an activity diagram.

2 The Elements in an UML Activity Diagram

The elements in an UML activity diagram include activity states, objects and transitions. An activity state is a state in the execution of an activity (procedure) rather than the state of a normal object. When an activity finishes, the control flow is transmitted to its successor activities. An object can be either an input or output of an activity, and thus is the condition or result of the activity. A transition describes the transfer of control flow between activities, and shows the change of the process of an object flow, and the state and attribute values of an object. The transitions in an activity diagram are completion transitions.

2.1 Activity state

An activity state of an UML activity diagram can be an action state, OR state, AND state or pseudo-state. The action state is atomic and cannot be decomposed. The execution of an activity in this state cannot be interrupted. Action states are the leaf-nodes in an activity state tree. OR state and AND state can be decomposed into subactivity states. They are non-leaf nodes in the activity state tree. A pseudo-state is one of the following: INIT, FORK, JOIN, BRANCH, MERGE, and FINAL.

Definition 1 The hierarchical structure of an UML activity diagram is an activity state tree, which can be expressed as a 4-tuple \(SH = (S, CHILD, \text{root}, \text{TYPE})\), where

(1) \(S\) is a non-empty finite set of activity states;
The Formal Semantics of an UML Activity Diagram

CHILD: $S \rightarrow 2^S$, is an activity state map, describing the parent-children relationship between states in the activity state tree. $\forall s \in S$, $\text{CHILD}(s)$ is a set of sub-states of $s$.

Let $\text{CHILD}^+ = \bigcup_{i \geq 1} \text{CHILD}^i$ be the transitive closure of relation $\text{CHILD}$, then, $\forall s \in S$, $\text{CHILD}^+(s)$ is the set of children of $s$.

$SH$ is a special tree which satisfies the following constraints:

1. $\forall (s) (s \in S \rightarrow s \notin \text{CHILD}^+(s))$;
2. $\forall (s1) (\forall s2) (s1 \in S \land s2 \in S \land s1 \notin \text{CHILD}^+(s2) \land s2 \in \text{CHILD}^+(s1) \cup \text{CHILD}^+(s2)) = \emptyset$;
3. $\forall (s1) (s1 \in S \land s1 \neq \text{root} \rightarrow (\exists s2)(s2 \in S \land s1 \in \text{CHILD}(s2)))$;
4. $\text{root} \in S$, is the sole root node of an UML activity state tree. $\forall s \in S$, $\text{root} \notin \text{CHILD}^+(s)$;
5. $\text{TYPE}: S \rightarrow \{\text{ACTION, OR, AND, PSEUDO}\} = \{\text{INIT, FORK, JOIN, BRANCH, MERGE, FINAL}\} \cup \{\text{ROOT}\}$, is the activity state type map, where
   1. If $(\exists s)(s \in S \land \text{TYPE}(s) = \text{ACTION})$, $s$ is an action state and a leaf node in the activity state tree. The execution of an activity in this state cannot be interrupted. $\text{CHILD}(s) = \emptyset$;
   2. If $(\exists s)(s \in S \land \text{TYPE}(s) = \text{OR})$, $s$ is an OR state and an OR node (non-leaf node) in the activity state tree. The execution of an activity in this state can be interrupted. $\text{CHILD}(s) \neq \emptyset$. The activities whose states are in $\text{CHILD}(s)$ cannot be triggered concurrently. The relationship between them is disjunctive;
   3. If $(\exists s)(s \in S \land \text{TYPE}(s) = \text{AND})$, $s$ is an AND state and an AND node (non-leaf-node) of activity state tree. The execution of an activity in this state can be interrupted. $\text{CHILD}(s) \neq \emptyset$. The activities whose states are in $\text{CHILD}(s)$ cannot be triggered concurrently. The relationship between them is conjunctive;
   4. If $(\exists s)(s \in S \land \text{TYPE}(s) = \text{PSEUDO})$, $s$ is a pseudo-state and $\text{CHILD}(s) = \emptyset$;
   5. $\text{TYPE(\text{root})} = \text{ROOT}$, $\text{CHILD(\text{root})} \neq \emptyset$.

2.2 Object

In an UML activity diagram, an object may be the target of a transition or the source of a completion transition to an activity, that is, it may be the output of one activity and the input of many other activities. Object flow state is a state that represents the existence of an object of particular class at a point within an activity process. The state comprises the attribute values and the processing state of an object, the latter of which indicates whether the object is being processed or has been processed.

If the object flow state is followed by a completion transition to an activity, then the activity can be performed as soon as the object value is available, that is, the creation of data in the right form is the trigger for performing the activity. In other words, the proper creation of an object and the correct change of its state are the necessary conditions for a transition to be triggered in an UML activity diagram.

2.3 Transition

In an UML activity diagram, a transition means the flow of control. When an activity state finishes, its successor activity will be triggered immediately, forming a path from one activity state to its successor activity state. All transitions in activity diagram are triggerless transitions, also called completion transitions.

Let $T$ be the transition set in a UML activity diagram, a transition $t \in T$ has a source state set, a guard condition and a target state set. The source state set is a set of activity states affected by the transition $t$. When the activity, resulting in the object flow state, finishes and the guard condition is satisfied, the transition will be triggered. The guard condition is a Boolean expression and must be evaluated true for the transition to be triggered. If there is no guard condition in a transition, it is assumed to be true. The target state set is the activity state when the transition finishes.

3 Formal Semantics of UML Activity Diagram

Definition 2 An UML activity transition net is a 3-tuple $NA = (S, T; F)$, satisfying the following conditions:

1. $S$ is a non-empty finite set of activity states;
2. $T$ is a non-empty finite set of transitions;
3. $S \cap T = \emptyset$ and $S \cup T \neq \emptyset$;
4. $F \subseteq S \times T \cup T \times S$, describes the control flow between activity states and transitions, $\times$ is Cartesian product;
5. $\text{dom}(F) \cup \text{cod}(F) = S \cup T$