Modular and Distributed Verification of SysML Activity Diagrams

Messaoud Rahim¹, Ahmed Hammad² and Malika Ioualalen³

¹Sciences and Technology Faculty, Yahia Fares University, Medea, Algeria
²Institut FEMTO-ST, UMR CNRS 6174, Besançon, France
³LSI, Computer Science Department, USTHB, Algiers, Algeria

Keywords: SysML, Activity Diagram, Places Bordered Petri Nets, Distributed Model-checking.

Abstract: Model-based development for complex system design has been used to support the increase of systems complexity. SysML is a modeling language that allows a system description with various integrated diagrams, but SysML lacks formality for the requirement verification. Translating SysML-based specification into Petri nets allows to enable rigorous system analysis. However, for complex systems, we have to deal with the state space explosion problem. In this paper, we propose new approach to allow a modular and distributed verification of SysML Activity Diagram basing on the derived Petri net.

1 INTRODUCTION

The System Modeling Language (SysML) is UML profile that can be used to specify graphically all aspects of complex systems(Friedenthal et al., 2008). Nevertheless, despite the various advantages of SysML, it remains a semi-formal language without possibilities of formally verifying the models described by it. Industrial safety-related standards strongly recommend the use of formal methods to validate critical systems. For that purposes, it is needed to use SysML in conjunction with formal method to provide formal verification of the specified system. Several approaches based on mapping SysML behavioural diagrams to Petri nets have been proposed (Carneiro et al., 2008; Andrade et al., 2009; Linhares et al., 2007). The aim of these approaches was to provide a way to verify the specified system with a Model checking technique. However, in the case of complex systems, we have to deal with the state space explosion problem to analyse the resulting Petri net. A way to overcome the state space explosion is the use of modular analysis. Another way that had gained interest, in the recent years is the use of distributed processing (Kristensen and Petrucci, 2004; Barnat and Rockai, 2008).

In this paper, we propose a global approach for performing a modular and distributed verification of the SysML activity diagram. Basing on composite activities, we derive places-bordered Petri net module for each activity. The verification of the system can concern only one simple activity or the global SysML activity diagram. For the second case, and in order to deal with the state space explosion problem, we propose to adapt the distributed verification process using a cluster of computing nodes(Boukala and Petrucci, 2011; Abid and Zouari, 2007) for verifying the derived modular Petri net. For mapping a SysML activity diagram into places-bordered Petri net, we propose a translation rule for the call behavior action. The translation of the other basic SysML activity constructs is inspired from previous works.

The rest of this paper is organized as follows: in Section 2, we discuss related works. In Section 3, we present the SysML activity diagram. In Section 4, we give a definitions of places bordered Petri net. In section 5, we present the mapping technique. We present the modular and distributed verification process in Section 6. Finally, in Section 7, we conclude and we outline some ideas for future works.

2 RELATED WORKS

The most proposed approaches concerning the formal specification of SysML diagrams have used Petri net models due to their expressiveness and formality (Linhares et al., 2007; Carneiro et al., 2008; Andrade et al., 2009). To our knowledges, this work is the first that considers the composite structure of the SysML activity diagram for a verification purposes.
The composite and modular verification approaches aim to take benefit from some information about the components of the system and the way they communicate. Modular Petri nets allow designers to specify a system as communicating modules. Modules communicate using shared transitions or fusion places. The work presented in (Valmari, 1994) proposes a compositional verification method for Petri net composed of place bordered subnets. The verification approach used in this work is based on Model checking technique. However, for complex system, we have to overcome the state space explosion problem. Several recent approaches use distributed processing environments to extend the size of the state space to be constructed (Boukala and Petrucci, 2011). This work proposes to adapt these approaches combined with a modular analysis to verify a SysML activity diagrams.

3 THE SysML ACTIVITY DIAGRAM

In SysML (OMG, 2010), an activity is a formalism for describing behaviour that specifies the transformation of inputs to outputs through a controlled sequence of actions. The basic constructs of an activity are actions and control nodes as illustrated in Figure 1. Actions are the building blocks of activities, each action can accept inputs and produces outputs, called tokens. These tokens can correspond to anything that flows such as information or physical item (e.g., water, signal). Control nodes include fork, join, decision, merge, initial, activity final, and flow final. A call behavior action permits to invoke an activity when it starts, and passes the tokens from its input pins to the behavior action permits to invoke an activity when it starts, and passes the tokens from its input pins to the output parameter nodes of the invoked activity.

4 PLACES-BORDERED PETRI NETS

Formally a Petri net is (Valmari, 1994):


Where: P is finite set of places, T is finite set of transitions, (P ∩ T = ∅) and W : (P×T) ∪ (T×P) → N is a weight function, W(p,t) (resp. W(t,p)) gives the weight of the arc from p to t (resp. from t to p).

Usually, an initial marking is associated with the Petri net:

Definition 2. A marked Petri net (PN,M₀), is a Petri net PN with an initial marking M₀ : P → N. The initial marking of a place p ∈ P is M₀(p).

To compose a large Petri net from smaller pieces, we define a places-bordered Petri (Valmari, 1994). A places-bordered Petri net modules interface with each other via common places, called border places.

Definition 3. A places-bordered Petri net module is the 4-tuple NC = (PN,B,t,Ctl). where : P, T, and W are as in a Petri net, B ⊆ P is the set of border places.

5 THE MAPPING TECHNIQUE

Basing on the previous works that propose a mapping of UML and SysML activity diagram to Petri nets (N. Yang and Qian, 2010; Andrade et al., 2009; Staines, 2008), our technique defines a mapping for the call behavior actions and propose to map the SysML activity diagram to modular Petri net with border places. The mapping we propose is activity based decomposition. The decomposition is guided by the call behavior actions which permits to facilitate the mapping of a SysML activity diagram even it includes several composite activities. The Petri net derived from the SysML activity diagram is a set of places-bordered Petri net modules, each one represents an activity instance.

5.1 Mapping Initial and Final Nodes

Initial node represents the start point of an activity. As illustrated in Figure 2, to map the initial node, we use one transition (t_in_Act) with one input place and two output places. The input place (en_Act) is used to enable the execution of the activity. The first output place (on_Act) is used to indicate that we are executing the activity and the second output place (Ctl_out) is used to represent the control flow.

Activity final represents the end point of an activity. As illustrated in Figure 2, to map final node, we use one transition (t_out_Act) with two input places and one output place. The first input place (on_Act) represents that we are executing the activity and the second input place (Ctl_in) represents the output flow enabling the termination of the activity. The output place (end_Act) is used to indicate that the activity is terminated.
5.2 Mapping Actions and Object Flows

As illustrated in the figure 3, for mapping an action with control and data flow into Petri net, places are used to represent the input and the output flows \((\text{in}_A, \text{Ctl}_\text{in}_A, \text{out}_A, \text{Ctl}_\text{out}_A, \text{out}_A)\) and one transition \((\text{Exec}_A)\) is used to represent the action. For mapping an object flow that connects output pin of one action A to the input pin of another action B, we fusion the place \((\text{Out}_A)\) that represents the output pin of the action A with the place \((\text{In}_B)\) which represents the input pin of the action B.

5.3 Mapping Routing Object Flows

For mapping a fork node 4 we use a transition \(t_{A\text{fork}}\) that represents the split operation with \(\text{out}_A\) as input place and \(\text{in}_B\) with \(\text{in}_C\) as output places. For mapping a join node 4 we use a transition \(t_{AB\text{join}}\) that represents the synchronisation between \(\text{out}_A\) and \(\text{out}_B\) as input places and \(\text{in}_C\) as output place.

5.4 Mapping Call Behavior Action

To map A call behavior action we consider that the invoked activity is already mapped into places bordered Petri net module. As presented in Figure 5, the mapping of a call behavior action A that invokes an activity Act with one input and one output flow is places bordered Petri net module which has \(\text{in}_\text{Act}, \text{en}_\text{Act}, \text{out}_\text{Act}\) and \(\text{end}_\text{Act}\) as border places with the Petri net module that represents the calling activity. The transition \(t_{\text{Abact}}\) is used to pass all input flows of the call behavior action to the invoked activity and to enable its execution. When the called activity terminates, we use the transition \(t_{\text{Aeact}}\) to pass all output flows of the invoked activity to the call behavior action.
when we create the tasks. The Figure 6 presents an example using four (4) computing nodes.

The state space is the basic model on which most verifications are built. The constructed state space is modular. It can be used to verify behavioural properties in the hole activity diagram or just on some activities. The properties to verify can be basic such as reachability, deadlocks, liveness and home state. We can adopt the approach presented in (Boukala and Petrucci, 2011) to verify such properties in the distributed and modular state space. Functional properties have to be extracted and translated from the SysML requirements diagram to temporal logic such as LTL and CTL. Various works have been proposed to verify LTL and CTL formulas in distributed and modular state space (Latvala and Makela, 2004).

7 CONCLUSIONS

The paper presents a modular and distributed verification approach for formally verifying complex systems described by SysML activity diagrams. A technique for mapping the SysML activities to Petri net have been proposed. The mapping is guided by the call behavior actions. The modular verification is enabled by analysing each activity using its related Petri net module. For enabling the verification of complex and composite activities a modular and distributed verification technique have been proposed to overcome the state space explosion problem. As future works, it is important to consider the process of extracting properties as temporal logic formulas from the SysML requirement diagram to complete the approach presented in this paper.

REFERENCES


