Liveness Verification of RGPS Process Layer Meta-model

Hao Yang

College of Media Engineering, Zhejiang University of Media and Communications, China

415360889@qq.com

Keywords: OWL-S; Promela; LTL formula; Liveness verification

Abstract: With the related theory and technology development of network software, network software has been widely used. Based on the characteristics of network software and framework of RGPS requirement meta-model, this paper proposes liveness verification of RGPS process layer meta-model. Firstly, it uses OWL-S language to describe RGPS process layer meta-model. Then, it uses Promela language to achieve modeling of OWL-S model. Next, it uses LTL formula to describe the properties of the Promela model and carry on liveness verification analysis of RGPS process layer meta-model. Finally, liveness verification of RGPS process layer meta-model is proved by an urban traffic system.

1. Introduction

Along with cloud computation, coordinated computation and general computation constantly emerging, it promotes software engineering to network software time. The core of network software system is requirement engineering. In order to better-targeted guidance network software requirement modeling, it proposes RGPS requirement meta-model frame[1]. RGPS requirement meta-model frame has already passed ISO certification and become an international standard[2]. However, RGPS is only a requirements modeling framework and does not provide a requirements validation scenario. RGPS is also a lack of correctness support for the final result. In order to improve the efficiency and reliability of verification, formalization method is being implemented internationally.

Model checking is a common formal analysis and verification method[3]. Through the state space search method, it detects a given model whether satisfies a formalized representation particular property. It includes three key steps for system modeling, protocol ingestion, and system verification. The advantage of model checking is safe to automatically verification. It is due to effective software tool support, such as Spin, SWV, CWB.

This paper is organized as follows. In section 2, it introduces Spin model checking tool. In section 3, it gives OWL-S process model. In section 4, it implements OWL-S process model to Promela model. In section 5, it introduces liveness verification. In section 6, an urban transport system example proves the validity of liveness verification method.

2. Spin Model Checking Tool

Spin is a model checking tool and is used to verify the model whether satisfy description property. Spin is a widely used software tool for analyzing logical consistency of concurrent systems. For the wrong result, the error path is returned. It can detect system design logic error. The model is designed to describe the interaction process in the system and to minimize the details within the process in order to reduce the size of the system. The processes in the system pass the message primarily through the channel or by sharing a global variable.

Spin verification process is as follows. It describes system model. After analyzing no syntax errors, the system's interaction process is simulated. Until the model is confirmed to have the expected behavior and then a language-written validator is generated by command. The compiler compiles and

  DOI: 10.25236/meici.2019.037

generates an executable verification file that can be verify. If a breach is found, feedback is given to the interactive simulator.

Spin typical operation mode starts with a description high-level standard model or starts with a distributed algorithm and after syntax error checking. A simulation to confirm that the system is designed with the expected behavior. Then it produces an optimization validator from the high-level specification model. The verification procedure describes with the language in the limited translation time. After choice use selling and buying of real esgate within the same family algorithm,verification program can be executed by the compiler.

3. OWL-S Process Model

The predecessor of Web Ontology Language for Service(OWL-S) is DAML-S. OWL-S has defined the Web service subclass process model. Depending on the level of granularity, the process is divided into atomic process, combination process and simple process. There is no control structure inside the atomic process. The composition process consists of a series of sub-processes implemented by controlling flow and data flow connections. Simple process mainly describes process model from abstract angle.

Control flows defined in OWL-S have Sequence, Split, Split-Join, Choice, If-Then-Else, Repeat-Until. Sequence defines a set of sequential processes. Split defines a set of simultaneous processes. Split-Join defines a set of partial synchronized execution processes. Choice a set of select n processes to perform in m processes. If-Then-Else defines a set of based on the criteria select the appropriate implementation processes. Repeat-Until defines a set of processes that are cycled under certain conditions.

Each process consists of three parts. It respectively is input, carry out condition and result. Result includes input and carry out effect. When satisfies the process carried out the condition, according to the input and at that time moves the environment condition to produce output and carries out effect. Is performes from the former state transition to the state after the execution.

The transportation requirement unceasing growth and the transportation network day by day complex proposed the new challenge to the urban traffic system development. The urban traffic system gradually shows the characteristics of the typical complex software system. Based on the requirement use case of urban traffic systemand on the basis of RGPS requirement meta-model framework, the process layer meta-model of urban traffic system is constructed.

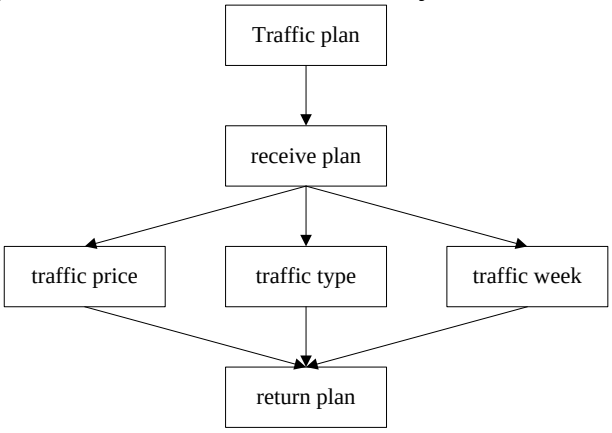

Figure 1. An urban traffic system

User traffic plan can be divided into atomic process generation preference, combined process preparation traffic, combined process traffic plan and atomic process support traffic. Four processes are associated by order. Combined process traffic plan consists of atomic process query lines and atomic process confirmation lines.

Partial code of process model which describes by OWL-S language is as follows.

<process: CompositeProcess rdf: resource = “Traffic Plan”>

```
    <process: composed of>
      <process: sequence>
        <process: components rdf: parseType = “collection”>
        <process: AtomicProcess rdf: resource = “traffic price”>
        <process: AtomicProcess rdf: resource = “traffic type”>
        <process: AtomicProcess rdf: resource = “traffic week”>
      </process: components>
     </process: sequnce>
    </process: composed of>
  </process: CompositeProcess>
```

4. Promela Modeling of OWL-S Process Model

Promela is an intuitive design protocol language provided by Spin which is a model checking tool. It is a formal descriptive language used to model finite state systems. It uses the advancement definition system behavior and through the channel for inter-process communication and data exchange. The basic elements described by the process include assignment statements, conditional statements, communication statements, non-deterministic selection, and circular statements.

Promela language supports all traditional programming language basic data types, such as integers, characters, booleans and arrays. Some variables represented in the pre- and post-conditions in OWL-S can be defined as global or local variables. Promela language supports the mtype enumeration type and may use mtype to define the message type. mtype = {nil, done} indicates whether a process is executed or whether the value of a variable is assigned.

The communication between messages can use global variables or channel. In the OWL-S description process model mainly involves the control to flow with the data stream. The control circulates has defined the syncChan channel in each combination process to control the sub-process the movement. The process control channel syncchan as a binary group, such as chansyncChan = [1] of {int, mtype}, where int indicates the sending process, mtype indicates the message type process name. It establishes the dataChan channel expression advancement the data stream.

Transform each process in the OWL-S process model into a process in the Promela language and the process is instantiated through run actions.

(1) Atomic process/simple process modeling

Atomic processes and simple processes are non-divided and are implemented in the Promela model with atomic{}.

```
proctype AtomicProcess(chan syncChan, dataChan){
atomic{…}
syncChan!_pid,done;}
```

(2)Combination process modelling

The composition process is a combination of a set of processes. In Promela model, control multiple sub-processes in a combination process implemented in channel mode. It defined as channel childSync and childData.

```
proctype CompositeProcess(chan syncChan, dataChan) {
chan childSync = [1] of {int, mtype};
chan childData = [1] of {int};
…
syncChan!_pid, done;}
```

OWL-S process model Promela modelling algorithm false code is as follows.

```
OWLS2PROMELA(M)
define_Type(M)
define_NeverClaim(M.Prop)
For each P in M{
define_proctype(P)
```

```
define_proctypeMain(P) }
init{ define_Channel(M)
run proctypeMain()}
```

5. Liveness Verification

LTL is an important formal method of describing system constraints. It first developed by Manna and Pnueli to describe the concurrency characteristics of the system. It uses the path as the object of the proposition and interprets its true value on the state sequence. Linear temporal logic may facilitate describes the system important property, such as safety and liveness. Safety is used to show that bad things never happen. Liveness is used to show good things will happen eventually. Spin supports all may use the accurate verification request which LTL expressed. LTL includes &&,||,->,！ and ◇,□,○. ◇p indicates that the p in all future status is true. □p indicates that p in future all states will be really.

A liveness verification tool achieves in the Eclipse platform. The main accessibility tools involved are Spin model checking tools, OWL-S API, OWL-S file resolution with OWL-S API. Firstly, user needs to describe the process layer meta-model, form the specification of the OWL-S file and use the OWL-S API to parse it. Secondly, it performs Promela modeling, generates .pml files and uses LTL to describe model property. Finally, it carries on Spin to check and return result. This tool contains the main function include Process level meta-model OWL-S description, OWL-S process model Promela modeling and model liveness verification.

6. Experiments

RGPS process layer meta-model partial code which is describes by Promela is shown as follows.

```
proctype ReadytoTravel(chan syncChan, dataChan)
  {chan childSync = [1] of {int, mtype};
   chan childData = [1] of {bool};
   pid x1, x2, x3;
   x1 = run Confirmpreference(childSync, childData);
   if
   ::childSync??eval(x1),done->
   atomic{x2 = run TravelPlan(chan syncChan, dataChan);
          if
          ::childSync??eval(x2),done->
             atomic{
                x3 = run SupportTravel(chan syncChan, dataChan);
                if
                ::childSync?/eval(x3),done->skip;
                fi;}
          fi;
          if
         ::childSync?/eval(x2),done->skip;
         fi;}
   fi;
   if
   ::childSync?/eval(x1),done->skip;
   fi;
   syncChan!_pid, done;}
```

It uses SPIN tool to verify liveness.

Input LTL formula []<>p&&[]<>q, it obtains the following result.

```
Full statespace search for:
never claim –
```

assertion violations +(if within scope of claim)
cycle checks-(disabled by –DSAFETY)
invalid end states +(disabled by never claim)
State-vector 57 byte, depth reached 123, errors: 0

Inpute LTL formula []<>q&&[]<>r, it obtains the following result.

Full statespace search for:
never claim –
assertion violations +(if within scope of claim)
cycle checks –(disabled by –DSAFETY)
invalid end states –(disabled by never claim)
State-vector112 byte, depth reached 135, errors: 1

7. Summary

Because model checking is highly automated, simple and fast, and provides counterexamples, it has received widespread attention. The complexity of model checking depends mainly on the size of the system state space. The biggest problems faced by model checking method is state space explosion and lack of memory. Spin offers a number of optimization technologies, such as partial sequencing protocols and on-the-fly, for detecting increasingly complex Web services. This article introducew how use Promela language modelling OWL-S process level meta-model and developes liveness verification platform. Finally, combined with the concrete example, liveness of RGPS process layer meta-model is verified.

Acknowledgments

This work is the final result of Introduction of Zhejiang University Of Media and Communications Scientific Research Grants Project(Z301B15521).

References

[1] J. Wang, K. He, B. Li, RGPS: A Unified Requirements Meta-Modeling Frame for Networked Software, Proceedings of the 3rd international workshop on Applications and advances of problem frames,(2008), pp. 29-35.

[2] K. He, R. Peng, Design methodology of Networked software evolution growth based on software patterns, Journal of System Science and Complexity, vol. 19,no.2, (2006), pp. 157-181.

[3] A. Meski, W. Penczek ,G. Rozenberg ,Model checking temporal properties of reaction systems, Information Sciences, vol. 313, (2015), pp. 22-42.

[4] A. Sistla, Employing symmetry reductions in model checking, Computer Languages Systems & Structures, vol. 30, no.3-4, (2004), pp. 99-137.

[5] M. Mundhenk, F. Wei, An AC complete model checking problem for intuitionistic logic, Computational Complexity, vol. 23, no.4,(2014), pp. 637-669.

[6] Y. Shi, C. Tian, Z. Duan, Model checking Petri nets with MSVL, Information Sciences, vol. 363,(2016), pp. 274-291.

[7] A. Donaldson, A. Miller, Automatic Symmetry Detection for Promela, Journal of Automated Reasoning. vol.41, no.3-4, (2008), pp.251-293.

[8] V. Cheval, V. Cortier and S. Delaune, Deciding equivalence-based properties using constraint solving, Theoretical Computer Science, vol. 492,(2013), pp. 1-39.

[9] M. Shen, Hinfin filtering of continuous Markov jump linear system with partly known Markov modes and transition probabilities, Journal of the Franklin Institute, vol. 350, no. 10, (2013), pp. 3384-3399.