Verification Tool of Software Requirement for Network Software

Tao He[1], Liping Li[2], and Huazhong Li[1]

[1] Software Engineering Department, Shenzhen Institute of Information Technology, Shenzhen, China
[2] Computer and Information Institute, Shanghai Second Polytechnic University, Shanghai, China
he_tao@foxmail.com, llping2000@yahoo.com.cn, lihz@sziit.com.cn

Abstract. A model checking tool OWLSVerifyTool is proposed, designed and developed to verify Web service composition model of software requirement in this paper. It can convert OWL-S documents into Petri nets document described in PNML, then analysis and verify it with Petri nets with engine in dynamic context. Compositing the DL reasoning engine Pellet and F-logic-based reasoning engine Flora-2, it can play their respective advantages to reason and verify static model in static context of software requirement. The automated validation tool can effectively verify software requirement meta-model based on Web service described with OWL-S.

Keywords: Web Service, Software Requirement, Verification Tool.

1 Introduction

The network software disposed in network environment is a kind of special ultra-large service-oriented computation of complex software system. Requirement engineering of Network software are facing many problems at present [1, 2], because of its dynamic topology, uncertainty of users, and its continuous increasing requirements. However, present software requirement modeling and verification technique lack enough support to service-oriented computation, and are unable to gather the Web service resources in the network effectively, provide high dependable Web service resources to enhance development of information system efficiency [3].

Requirement verification is an important process in software requirement engineering. If without it, the project may lead to be unsuccessful. The design and development of verification tool is very important, for its enhancing efficiency of verification, improving software development process, and guarantee software quality. Semantic Web service language has carried on the clear description to Metamodel various levels of software requirement, and carries on the formalized modeling by Petri net and F-logic, and verifies the uniformity of Web service semantic restraint.

Verification of Web service combination has two kinds of research methods at present: verification based on work flow BPEL4WS and based on semantic OWL-S. Now regarding the latter there are few research achievements. In literature [4] for controls flow and the data flow on the modeling, it transforms directly the OWL-S process model into the simpler Promela modeling, and verifies it with SPIN. However, its data flow modeling is too simple to verify whether the input /output type match.

This paper has designed OWL-S model examination prototype tool OWLSVerifyTool. It takes OWL-S storage documents (*.owl or *.xml) as the input, simultaneously carries on dynamic model and static model verification. The dynamic model, transforms with the document format switch to the PNML document, then directs or transforms the PNML document to corresponding form, then input to Petri net verification DiNAMiCS and Tina engine to verify it. The static model, combining description logic DL inference and Flora-2 rule, takes OWL-S storage documents(*.owl or *.xml) as input, simultaneously unifies DL inference (Pellet) and the Flora-2 rule to carry on inference alternately, carries on the analysis verification of the static model, and output the results.

2 General Organization of Verification of Verification Tools

OWL-S verification tool OWLSVerifyTool is mainly composed of dynamic model verification and static model verification modules. The designing structure of verification tool is shown in Fig. 1.

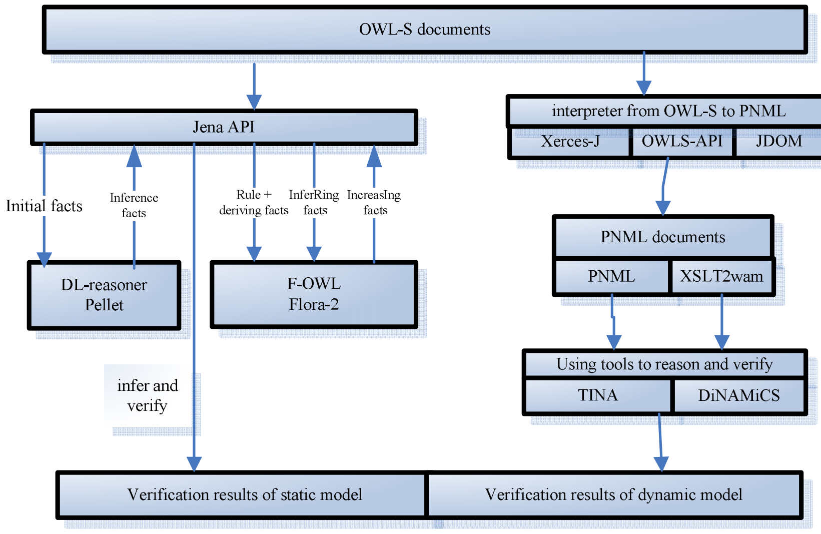

Fig. 1. General architecture diagram of validation tool

OWL-S is the Web service frame described in OWL language, but OWL represents ontology by class, attribute, value, and concept relations and so on. When describing

RGPS software requirement, it often uses essential factor and relational of standard ontology, a series of axioms as well as the formal restraint semantic to build its model. This paper calls it “static model”. A static model, often not evolving, can only express invariant and special condition. formalism methods such as Z language, DL, VDM, F-logic, which are based on set theory and the first-order predicate calculus, may use for modeling it. This paper uses DL and F-logic. To verification of static model, its realization of definition standard, accuracy, uniformity and completeness of inference must be considered.

In the static model verification module it transforms the OWL-S documents through Jena API. First, inputs it with rules to the Pellet engine to infer for obtaining the new fact, then combine the new fact and the original fact, transforms it to the form that the engine Flora-2 can accepts. Eventually, inputs it with rules to engine Flora-2 to infer and verify. The output is the verification result of static model.

Uses the Petri net in the dynamic model aspect to take the verification model, and carries on the verification with DiNAMiCS and Tina engine; mainly verify its accuracy (activeness, boundedness), Reachability, final state, security, and so on.

As shown in Fig. 1, first inputs OWL-S documents in unified user interface. Transforms the OWL-S documents into the PNML documents in the dynamic model verification module with interpreter respectively, simultaneously carries on the PNML documents verification. Because the input form of DiNAMiCS engine is different from PNML slightly, which is the wam form, first transforms it into wam form by XSLT, and then inputs it to DiNAMiCS to carry on inference and model checking. Tina may directly input PNML form documents to carry on it. After verification by two engines, merge the results to obtain dynamic model verification result of the OWL-S documents.

3 Petri Nets Verification Module

3.1 Interpreter of the Transforming from OWL-S to PNML

This interpreter can carry on analysis of OWL-S service and transform it into PNML form. For the transformation from OWL-S service to Petri net can use reuse many methods, algorithms, and reuse tools to inspect the equivalence of Petri net (for example literature [5,6]). Narayanant and McIlraith have first defined the Petri net semantics of DAML-S in [7] (the OWL-S preceding edition). However, their semantics is not the combinatorial property for it is unable to process any-order control structure. DaGen tool [8] transforms Petri net semantics of DAML-S description in [7] to referring Petri net. DaGen inserts to a Reference Net Workshop (Renew), and causes the Petri net simulator execution of Reference Network as well as the graph draw. However, DaGen has not had the intermediate Petri net file of transform. The interpreter described in this paper can transform OWL-S process model expressing service behavior to Petri network described by PNML format.

When execute the reasons, how to share the input/output data in different process? In fact, when input and output data are shared by many processes, the OWL-S process model may be defined by the input/output binding mechanism. [9, 10] The interpreter deals with this problem by carrying out a suitable analysis sentence of OWL-S process model.

This paper comes through the XML resolver to transform OWL-S documents to the PNML form. The interpreter from OWL-S to the PNML is a Java Servlet. Input a URL pointing to OWL-S service description (or file system path) from Web client of the interpreter, and sends it to the serve, analyzing by the Servlet in the background conversion, and returning the Petri net describing in PNML of OWL-S service.

3.2 PNML Verification and XSLT Transformation

1) PNML Correctness Verification module: To change the default, adjust the template as follows.

Through to the OWL-S documents' transformation, the Web service which the OWL-S documents describe definitely may use the Petri net simulation and indicate by the PNML document that like this may verify the Web service operation flow which using the Petri net's correlation analysis method and the tool the OWL-S documents describe whether to have in the flowage structure design question. Therefore the next stage is transforms, but results in the PNML document hands over by PNML Correctness the Verification module processes. PNML Correctness Verification module construction is like chart 2.

PNML Correctness the Verification module contains the PNML resolver, the accurate verification, the security, the Reachability, the deadbolt lock, finally the shape, and the durable verification and so on. PNML Parser is responsible for the PNML document which analyzes transmits. The accurate verification, the security, the Reachability, the deadbolt lock, the shape, the durability through the execution coverage diagram, may reach analysis methods separately finally and so on chart, incidence matrixes, condition agenda, migration matrix to carry on the verification. But the PNML document's proving program is first starts by the accuracy and the secure nature, next is the Reachability nature, and finally is the deadbolt lock, the final state and the durable nature. So long as this verification step has an item will be unable through to transmit makes a mistake harms the information and stops the entire proving program.

PNML Correctness Verification module contains:

*PNML decoder
PNML Parser is responsible to receive the PNML document after OWL-S file conversion. First analyzes the Web service operation flow which describes in the PNML document, again storehouse which describes the PNML document, migration and arc by array way storage. PNML Parser through analyzes the PNML document and separately the storehouse, the migration and the arc by the array form storage, will then transmit these three objects by the parameter way for the nature proving program. The nature proving program after receiving the Petri net model the image parameter the basis itself uses again the analysis method, constructs by the parameter in information may reach the tree, the incidence matrix and so on mathematics type.

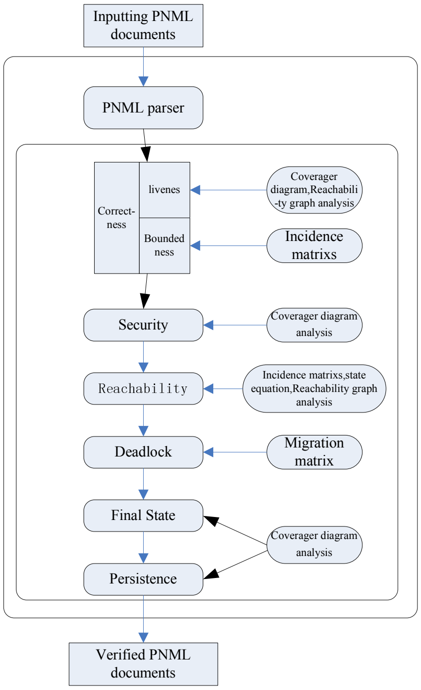

Fig. 2. PNML Correctness Verification module structure

* The secure verification
For Web service composition, the tool will use cover tree analysis to verify the Petri-Net model of the security property. Therefore realizes the cover tree and the coverage diagram method application procedure analysis in the PNML accuracy verification proxy service combine the Petri-Net model after PNML the resolver analysis Web.

* Reachability verification
The Petri-Net model regarding the Web services the Reachability nature to use the incidence matrix, the equation of state or may reach the chart the analysis method verification. Therefore is by realizes the incidence matrix and the equation of state method application procedure analysis in the PNML accuracy verification proxy service combines the Petri-Net model after PNML the resolver analysis Web.

3.3 Calculation Method Verification Tool DaNAMiCS and Tina

DaNAMiCS is a modeling verification tool, can use for to analyze the Petri net and to color the Petri net, and can reduce the modeling grid complexity. DaNAMiCS includes suppressing the arc the support to help a system model the foundation. The DaNAMiCS important merit is that it supports some analysis tool and the method, for instance matrix invariant and migration matrix, structure analysis, as well as some simple and advanced performance analysis. Because DaNAMiCS has compared to other OWL-S verification tool more analysis tools, we use XSLT to transform the PNML documents are the wam forms, like this can induct them to DaNAMiCS.

Tina (time Petri net analyzer) is a tool that analyzes the Petri net and a time Petri net. It may construct and reach the chart and can carry on the analysis to the Petri net's structure:

The accurate analysis confirmed that a system's integrity is maintained, it including analyzes net's activeness and the boundedness. We will use the accuracy to represent the net to be live and have, and with the accuracy explained that this model net expressed the correct system.

Besides these essential attributes, we must confirm some other attributes, including net whether to contain the final state, net whether safe (security), whether it is lasting (durability) and so on.

1) Analysis tools and methods:

Coverage diagram
We must inspect the construction algorithm of the coverage diagram. It will need to renew, so that processes increases newly suppresses the arc and has the capacity storehouse institute order of complexity. Carry out the algorithm that must be correct and the most superior movement.

* Analyze the correctness with invariant
DaNAMiCS will be able to calculate the incidence matrix. It will be able to determine P- and the T- invariant from the incidence matrix. This may use for surveying the boundedness and the deadbolt lock that does not exist.

* Analyze the correctness with coverage analysis
The user will be able to choose the option in DaNAMiCS, which domain through assigns the functional analysis to need to investigate. The coverage diagram will be produced for the accuracy and tests.

Coverage analysis

In many situations, the invariant analysis cannot produce about the Petri net model accurate conclusion. It needs to carry on the spreadability analysis. It is a two stepped process. The first stage is the coverage diagram construction. The coverage diagram is all may reach marking the set. The second section is this stage analysis. The first stage is called the coverage diagram production, has the greatest time consumption and the complexity; the second section analyzes the section, it will process afterward.

2) The nature analysis of Coverage diagram:
After discussion chart product, now it will analyze the coverage diagram to appraise the following nature, such as activeness, boundedness, final state, security and durable process and so on.

a) Activity

A Petri net most important nature is an activeness, this relates judges some system whether to collapse or whether systematic some part infinite loop. A Petri net's live performance through the following two rules, but determined from the coverage diagram:

* If a net is live and has, then it has the strong connection coverage diagram.

* A net has, the net is lives, and when only all migration in the coverage diagram connects in the module the demonstration is a label finally at least. A strong connection chart is the random point may arrive in the chart from the chart through a

series of ways other all point charts. A chart's final module is a series of points. A strong connection chart has a correct final module. Loads the above two principles to carry on the spreadability analysis to a useful form, we may say, if each migration can cause a coverage diagram all final module's marking to enable, then the net is active. Thus, the definite active question became finds connects a module's question finally, this was the algorithm question which easy to understand. Abbreviate this algorithm specific code here.

b) Boundary analysis

In DaNAMiCS, permits an expansion network's class use. Not only it is not always possibly uses these expansions to determine a Petri net's accuracy, This paper therefore the mark storehouse institute and does not have, moreover, if this storehouse were considered that possibly does not have, uses a different mark to mark. If a storehouse institute possibly does not have, we send a letter the number to the user to point out that corresponding storehouse institute possibly does not have. From this start, the user will be able not but to use the heuristic method to determine whether that storehouse institute accurate doesn't have.

c) Terminal analysis

The final state is that in Petri net chart a marking may arrive from each other. This is very important to the software, namely, regardless of the current condition is anything, can always arrive at the ultimate objective. The final state existence is easy to calculate. If final, strong connects module's quantity to be equal to 1, then this Petri net contains the final state.

d) Security analysis

If a Petri net does not have the storehouse to be able in the net the packet of energy including an above request to be willing, then calls it safely. This is the Petri net very important attribute; net's storehouse in the system is the condition mark. If the storehouse contains a request to be willing, then the condition is effective, otherwise the condition is untenable. A storehouse contains is more than a request to be willing not to have the logical significance in these net's type, usually the expression somewhere has a mistake in the design. As mentioned above, if in a Petri net's storehouse institute the request is willing quantity are most, then it is safe. The security may be determined by all mark of linear search chart simply. If has not met has the storehouse contains an above request to be willing a marking, then this net is safe.

e) Durability

If a Petri net enables the migration to random two, an initiation's migration ever does not forbid other migration to enable, then calls it lastingly. If a lasting net's migration enables, it will maintain enables to initiate until it.

4 Conclusion

This paper has realized the integration static model and the dynamic model inspection is a body's automated verification tool prototype; In the static model aspect, gives the method which DL description logic reasoning (Pellet) and the F-logic rule (Flora-2) unifies, has used two kind of system forward reasoning fully and latter to the inference merit, combined based on DL inference engine Pellet and based on F-logic inference engine Flora-2, displays its respective superiority to carry on the inference and the verification. An OWL DL subset may transform is F-logic, and also provides the reverse support.

Should automate the verification tool prototype to be able service to provide the effective verification support to OWL-S the description Web, have the important theory and the practical significance, and has the widespread application prospect.

Acknowledgments. This work was supported in part by a grant from Basic Research Program of Shenzhen Science and Technology Research and Development under grants JC201006020791A and JC201006020820A, Natural Science Foundation of Guangdong Province, China, under grant S2011040000672, Guangdong Provincial Education and science project of the 11th "five-year plan", Research Fund of Shenzhen Institute of Information Technology.

References

1. He, K., Liang, P., Li, B., et al.: Meta-modeling of Requirement for Networked Software, An Open Hierarchical & Cooperative Unified Requirement Framework URF. DCDIS2B 14(S6), 293–298 (2007)
2. Wang, J., He, K., Li, B., et al.: Meta-models of Domain Modeling Framework for Networked Software. In: Proceedings of The Sixth International Conference on Grid and Cooperative Computing, GCC 2007, Urumchi, China, pp. 878–885 (August 2007)
3. Matthias, K., Benedikt, F., Katia, S.: OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services. Web Semantics: Science, Services and Agents on the World Wide Web 7(2), 121–133 (2009)
4. Morimoto, S.: A Survey of Formal Verification for Business Process Modeling. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008, Part II. LNCS, vol. 5102, pp. 514–522. Springer, Heidelberg (2008)
5. Qi, G., Tianshi, C., Haihua, S., Yunji, C., Weiwu, H.: On-the-Fly Reduction of Stimuli for Functional Verification, ats. In: 2010 19th IEEE Asian Test Symposium, pp. 448–454 (2010)
6. He, F., Le, J.: Hierarchical Petri-Nets Model for the Design of E-Learning System. In: Hui, K.-c., Pan, Z., Chung, R.C.-k., Wang, C.C.L., Jin, X., Göbel, S., Li, E.C.-L. (eds.) EDUTAINMENT 2007. LNCS, vol. 4469, pp. 283–292. Springer, Heidelberg (2007)
7. Hallé, S., Hughes, G., Bultan, T., Alkhalaf, M.: Generating Interface Grammars from WSDL for Automated Verification of Web Services. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 516–530. Springer, Heidelberg (2009)
8. Moldt, D., Ortmann, J.: DaGen: A Tool for Automatic Translation from DAML-S to High-Level Petri Nets. In: Wermelinger, M., Margaria-Steffen, T. (eds.) FASE 2004. LNCS, vol. 2984, pp. 209–213. Springer, Heidelberg (2004); Kornack, D., Rakic, P.: Cell Proliferation without Neurogenesis in Adult Primate Neocortex. Science 294, 2127–2130 (2001)
9. Staats, M., Heimdahl, M.P.E.: Partial Translation Verification for Untrusted Code-Generators. In: Liu, S., Araki, K. (eds.) ICFEM 2008. LNCS, vol. 5256, pp. 226–237. Springer, Heidelberg (2008)
10. Emilia, O., Tomi, J.: A Translation-based Approach to the Verification of Modular Equivalence. Journal of Logic and Computation 19(4), 591–613 (2008)