ACPN-based Verification Method of Web Service described by OWL-S

GuangHao Zhang
School of Computer Science and Engineering
Beihang University
Beijing, China
zhangguanghao@cse.buaa.edu.cn

YuQing Lan
School of Computer Science and Engineering
Beihang University
Beijing, China
lanyuqing@buaa.edu.cn

***Abstract*—Web services are an important part of the Semantic Web. User and software agent should be able to discover, invoke, compose, and monitor web services with a high degree of automation. OWL-S is an ontology of services that makes these functionalities possible. However, OWL-S doesn't provide verification method for Web services. A graphical and formal modeling tool, CPN (Colored Petri Net), is suitable for solving this problem. This paper proposes a more practical verification method of Web services. We firstly transform Web services described by OWL-S to CPN models. Then we take advantage of CPN theory to analyze Web services and conclude that the CPN model must satisfy three properties for ensuring the correctness of the Web service. In addition, our approach can automatically invoke Web service in CPN model, making the verification more accurate. We also present an implementation of our approach for modeling and verifying Web Services. This implementation needs an OWL-S description of a Web service, automatically generates a CPN model and performs the desired analysis. Furthermore, the results of this research will lay the foundation for the Web services composition and evaluation.**

Keywords-OWL-S; CPN; Web Service; Verification

I. INTRODUCTION

The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. The main purpose of the Semantic Web is driving the evolution of the current Web by enabling users to find, share, and combine information more easily. Humans are capable of using the Web to carry out tasks such as finding the Estonian translation for "twelve months", reserving a library book, and searching for the lowest price for a DVD. However, machines cannot accomplish all of these tasks without human direction, because web pages are designed to be read by people, not machines. The semantic web is a vision of information that can be readily interpreted by machines, so machines can perform more of the tedious work involved in finding, combining, and acting upon information on the web[1].

Web services are an important part of the Semantic Web. In recent years, Web service is becoming the mainstream for distributing applications with Service Oriented Architecture (SOA). That is a software system designed to support interoperable machine-to-machine interaction over a network[2]. User and software agent should be able to discover, invoke, compose, and monitor web services with a high degree of automation. Powerful tool should be enabled by service descriptions in the Web service lifecycle. OWL-S (formerly DAML-S) is an ontology of services that makes these functionalities possible. However, OWL-S doesn't provide verification method for web services[3].

CPN (Colored Petri Net) is suitable for solving this problem[6]. That is an extended version of petri net. Petri net is a visual formal method for modeling business processes with rigid mathematical ground. It is usually used to verify distributed systems. As a graphical formal model, CPN is suitable for modeling communication, concurrency and synchronization, which has been used successfully in specification of workflow systems and communication protocols[4][5]. CPN provides formal semantics and a number of analysis techniques for modeling and verifying composite Web service[7].

In this paper, we present a more practical approach for verifying Web services descried by OWL-S. We firstly transform OWL-S to CPN models. Wecan run simulations to accurately locate the error position of Web service when there are problems in the CPN model. Then we take advantage of CPN theory to analyze Web services and give three necessary conditions to verify the correctness of web services.We also present an implementation of our approach for modeling and verifying Web Services. Furthermore, the results of this research will lay the foundation for the Web services composition and evaluation.

The remainder of this paper is organized as follows. Section 2reviewsseveral approaches for the analysis and verification of composite Web service. Section 3 introduces the principal concepts of OWL-S and CPN. We describe in detail the method for modeling Web Service based on CPN in section 4. We discuss the analysis and verification of Web services in section 5. Section 6 shows our implementation of a software tool for performing the proposed method. The conclusions are presented in section 7.

II. RELATED WORK

Several related approaches have been developed for the analysis and verification of composite Web service, including AI planning, Petri Net[8-9], FSP[10] and Pi-calculus[11]. The concept of task decomposition in AI planning is very similar to the concept of composite process decomposition in OWL-S process ontology. AI planning firstly creates model for the goal problem, then verifies the model formally, and selects services dynamically. As the model is built statically, it can't be evolved automatically with the changes of Web services resources. FSP is a textual notation for concisely describing and reasoning about concurrent programs. The constructed FSP can be used to model the exact transition of workflow processes, but it isn't as expressive as Petri Net. At present, researchers usually use Petri Net to execute, analyze and verify Web service Composition. As a graphical formal model, Petri Net is suitable for modeling communication, concurrency and synchronization, as well as, provides abundant analysis and verification techniques for services composition. However, previous studies focus on the formal correctness of composite structures and fail to consider whether web service can be invoked truly.

In this paper, we propose a CPN-based approach for verifying Web services descried by OWL-S. Comparing to previous studies, we not only focus on the parameter checking and structure transformation, but also focus on how to get the output from the input. In addition, Web service can be automatically invoked in CPN model. So we can run simulations to accurately locate the error position of Web service when there are problems in the CPN model.

III. OWL-S AND CPN

A. OWL-S

OWL-S is an ontology, within the OWL-based framework of the Semantic Web, for describing Web services[3]. The service ontology can be structured to provide three essential types of knowledge about a service, as shown in figure 1. The properties *presents*, *describedBy*, and *supports* are properties of Service. The classes*ServiceProfile*,*ServiceModel*, and *ServiceGrounding* are the respective ranges of those properties.

To give a detailed perspective on how to interact with a service, it can be viewed as a process. Specifically, OWL-S defines a subclass of *ServiceModel*, *process*. A process can have two sorts of purpose. First, it can generate and return some new information based on information it is given and the world state. Information production is described by the inputs and outputs of the process. Second, it can produce a change in the world. This transition is described by thepreconditions and effects of the process. If a process has a precondition, then the process cannot be performed successfully unless the precondition is true. The performance of a process may result in changes of the state of the world, defined as effects.

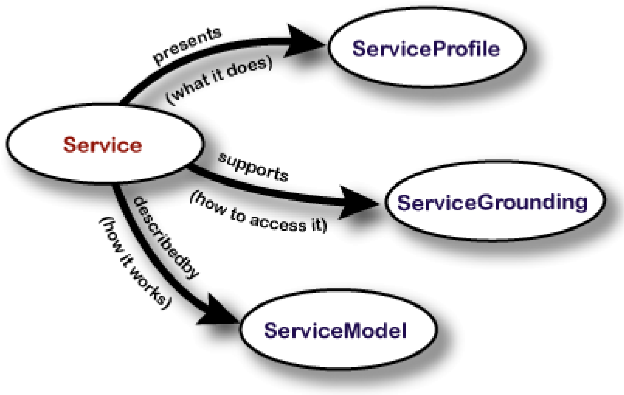


Figure 1. Top level of the service ontology.

In OWL-S, the processes are divided into atomic process and composite process. The atomic process corresponding to the atomic service, like it's literal meaning, has no sub processes and execute in a single step. The composite process corresponding to the composite service is decomposable process, their decomposition can be specified by using control structure including Sequence, Split, Split+Join, Choice, Any-Order, Condition, If-Then-Else, Iterate, Repeat-While, and Repeat-Until.

B. CPN

CPN models are executable and are used to model and specify the behavior of concurrent and distributed systems. A CPN model of a system is both state and action oriented. It describes the states of the system and the transitions that can cause the system to change state[12][13]. By performing simulations of the CPN model, it is possible to investigate different scenarios and explore the behavior of the system. CPN can be simulated automatically. Automatic simulation is typically used for the analysis and verification of Web services.

The Colored Petri Net is a nine-tuple as formula 1.

$$\mathrm{CPN} = (\mathrm{P}, \mathrm{T}, \mathrm{A}, \textstyle\sum, \mathrm{V}, \mathrm{C}, \mathrm{G}, \mathrm{E}, \mathrm{I}) \qquad (1)$$

Where:

1) P is a finite set of places.
2) T is a finite set of transitions T such that $\mathrm{P} \cap \mathrm{T} = \emptyset$.
3) $\mathrm{A} = \mathrm{P} \times \mathrm{T} \cup \mathrm{T} \times \mathrm{P}$is a set of directed arcs.
4) $\sum$is a finite set of non-empty color sets.
5) V is a finite set of typed variables such that $\mathrm{Type}[\mathrm{v}] \in \sum$ for all variables$\mathrm{v} \in \mathrm{V}$.
6) $\mathrm{C}: \mathrm{P} \rightarrow \sum$is a color set function that assigns a color set to each place.
7) $\mathrm{G}: \mathrm{T} \rightarrow EXPR_V$is a guard function that assigns a guard to each transition t such that $\mathrm{Type}[\mathrm{G}(\mathrm{t})] = \mathrm{Bool}$
8) $\mathrm{E}: \mathrm{A} \rightarrow EXPR_V$is an arc expression function that assigns an arc expression to each arc a such that$\mathrm{Type}[\mathrm{E}(\mathrm{a})] = C(p)_{MS}$, where p is the place connected to the arc a.
9) $\mathrm{I}: \mathrm{P} \rightarrow EXPR_\emptyset$is an initialization function that assigns an initialization expression to each place p such that $\mathrm{Type}[\mathrm{I}(\mathrm{p})] = C(p)_{MS}$.

IV. MODELING WEB SERVICES BASED ON CPN

To take advantage of CPN theory to validate Web services, we transform Web services described by OWL-S to CPN models. We define the transformation from Web service to CPN model as $\mathrm{Transform} = (\mathrm{P}, \mathrm{CPN}, \mathrm{I}, \mathrm{O})$, whereP is a process of web service; CPN is colored petri nets model; I is the set of input places; O is the set of output places.

Then we discuss the details about transformation from two aspects: atomic Web service and composite Web service[14].

A. *Atomic Web Service*

The rules for transforming atomic Web service described by OWL-S to CPN model are presented as follows: the input and output of an atomic Web Service is transformed to input place and output place of CPN model, the atomic process is transformed to a transition of CPN model, the precondition is transformed to an arc from the input place to the transition, the effect is transformed to an arc from the transition to the output place.

The atomic process corresponds to the atomic service. It can generate and return some new information based on information it is given and the world state. Information production is described by the inputs and outputs of the process. Previous studies only focus on type checking of inputs and outputs. However, for Web services with the condition, if not actually calling a Web service, it is possible that some branches will never be able to reach. Some deadlock issues of the original Web service will not be found.

To solve this problem, we automatically invoke a Web service in CPN model, because CPN model is executable in CPN Tools. Firstly codes about Web service calling are embedded in the transition of CPN model. Secondly we use a server to listen requests from CPN Tools. When the transition of CPN model is executing, the server receive the input data, invoke Web service, and get output. Finally the output is sent back to the CPN Tools.

A communication protocol and some communication interface were defined, include openConnection(), send() and receive(). Figure 2 illustrates how to transform the atomic Web service to CPN model.

```
Algorithm ModelingAtomicWS(WS)
Input: an atomic Web service WS
Output: a CPN model
1.  generate a basic CPN model
2.  for each input in WS
3.      do generate input place in CPN model
4.          generate input()
5.  for each output in WS
6.      do generate output place in CPN model
7.          generate output()
8.  generate action(send(conn,input,StringEncode) ......
    receive(conn,output,StringEncode))
9.  add input(), output(), action() to the transition of CPN
    model
```

Figure 2. Algorithm ModelingAtomicWS(WS).

B. *Composite Web Service*

Having introduced the transforming from the atomic Web service to CPN model, we now turn to modeling composite Web service as CPN model.

Some rules for transforming composite Web service to CPN model are same as atomic Web service: the input and output is transformed to input place and output place of CPN model, the precondition is transformed to an arc from the input place to the transition, the effect is transformed to an arc from the transition to the output place.

The difference is composite process, corresponding to composite Web service. Composite processes are decomposable into atomic or composite processes; their decomposition can be specified by using control constructs. The OWL-S control constructs: Sequence, Split, Split-Join, Choice, Any-Order, Condition, If-Then-Else, Iterate, Repeat-While, and Repeat-Until, will be modeled based on CPN, which are discussed below.

Figure 3 shows the CPN model of Sequence construct. Sequence has a list of component sub-processes that specify the body. A list of control constructs to be done in order.

Figure 4 shows the CPN model of Split+Join construct. A Split+Join composite process consists of concurrent execution of a bag of sub-processes. The process consists of concurrent execution of a bunch of process components with barrier synchronization. That is, Split+Join completes when all of its components processes have completed.

Figure 5 shows the CPN model of If-Then-Else construct. An If-Then-Else composite process is a simple construct that has a relation whose domain is a process and whose range is a binary value. The internal process usually corresponds to one or more test actions, but it may alternatively be some evaluation of world state, resource levels, timeouts or other conditions that affect the evolution of processes.

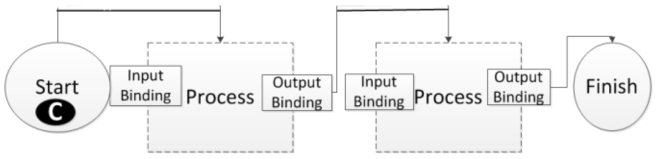


Figure 3. Sequence CPN model.

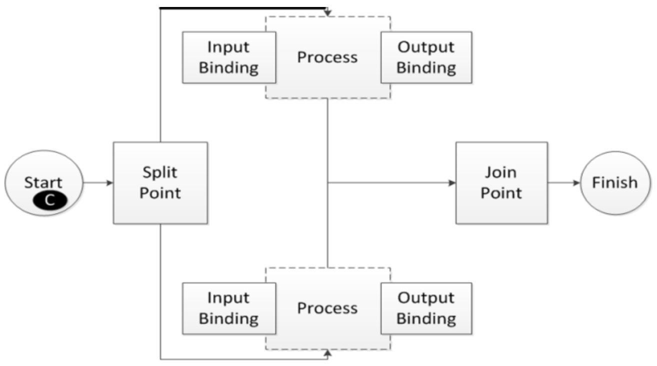


Figure 4. Split+Join CPN model.

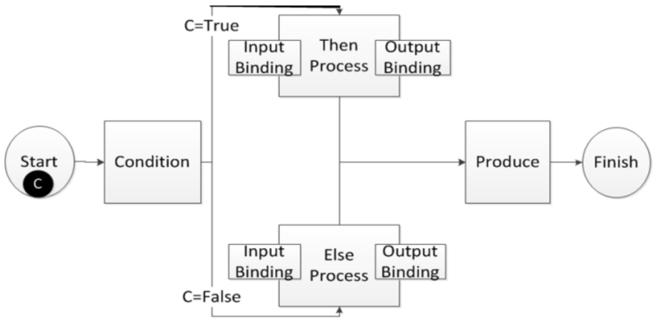


Figure 5. If-Then-Else CPN model.

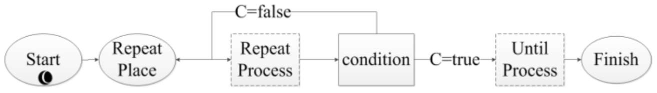
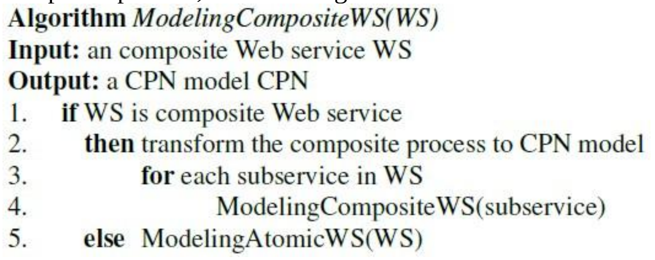


Figure 6. Repeat-Until CPN model.

Figure 6 shows the CPN model of Repeat-Until construct. An Repeat-Until specializes the Control Construct class with properties untilCondition(range is of type Condition) and untilProcess(range is of type Repeat). Repeat-Until does the operation, tests for the condition, exits if it is true, and otherwise loops.

Having illustrated the algorithm of modeling atomic process, we can propose the recursive algorithm of modeling composite process, shown in Figure 7.



```
Algorithm ModelingCompositeWS(WS)
Input: an composite Web service WS
Output: a CPN model CPN
1.  if WS is composite Web service
2.     then transform the composite process to CPN model
3.          for each subservice in WS
4.               ModelingCompositeWS(subservice)
5.     else  ModelingAtomicWS(WS)
```

Figure 7. Algorithm ModelingCompositeWS(WS).

V. VERIFICATION METHODOF WEB SERVICES

A. *Behavioural Properties and State Space Method*

The state space method of CPN makes it possible to validate and verify the functional correctness of systems[16]. The state space method relies on the computation of all reachable states and state changes of the system, and is based on an explicit state enumeration. By means of a constructed state space, behavioral properties of the system can be verified. The properties of CPN to be checked include Reachability, Boundless, Dead Transitions, Dead Marking, Liveness, Home, and Fairness. We will review some important properties in verifying Web services:

Reachability -- The possibility of reaching a given state. By formulate application specific properties as reachability of CP-net models, we can validate whether the models of the composition process can achieve the desired result.

Dead Transitions -- The transitions which will never be enabled. There are no activities in the process that cannot be realized. If initially dead transitions exist, then the composition process is bad designed.

Dead Marking -- Markings having no enabled binding element. The final state of process instance is one of dead marking. If the number of dead markings reported by state space analysis tool is more than expected, then there must be errors in the design.

To illustrate what properties of CPN model should be verified with state space method, the following theories are essential[15]:

1) A **directed graph** with arc labels from a set L is a tuple:$\mathrm{DG} = (\mathrm{N}, \mathrm{A})$ where N is a set of nodes, A is a set of arcs.
2) Let$\mathrm{DG} = (\mathrm{N}, \mathrm{A})$ and $\mathrm{DG}' = (\mathrm{N}', \mathrm{A}')$be directed graphs. $\mathrm{DG}^{'}$ is a **subgraph** of DG if and only if $\mathrm{N}^{'} \in \mathrm{N}$ and $\mathrm{A}' \in \mathrm{A}$.
3) Let $\mathrm{DG} = (\mathrm{N}, \mathrm{A})$ be a directed graph. A set of nodes N' is **strongly connected** if and only if $\forall \mathrm{n} \in \mathrm{N}^{'}$: n is reachable from $\forall \mathrm{n}' \in \mathrm{N}^{'}$.
4) Let $\mathrm{DG} = (\mathrm{N}, \mathrm{A})$ be a directed graph.The **Strongly-Connected Component graph(SCC graph)** for DG is a directed subgraph $\mathrm{SG} = (N_{SG}, A_{SG})$, where N_{SG} is strongly connected.

B. *necessary properties of CPN model*

According to the above theory, we can draw a conclusion. The CPN model of effectiveWeb service described by OWL-S must satisfy the following properties:

1) All SCCs are trivial, in other words, every strongly connected component consists of a single node and no arcs. This property ensures the CPN model can terminated in some finite steps.
2) The count of the terminal SCCs must be equal to the count of the output of Web service. This property ensures that CPN model has the same outputs as the Web service.
3) There are no dead transition, in other words, all transitions occurred. This property ensures the model will be terminated as expected.

Then we can give the verification results by our implementation.

VI. IMPLEMENTATION

We developed a tool WSM&VT (Web Services Modeling and Verifying Tool) to implement our method for modeling and verifying Web Services. This tool needs an OWL-S description of a Web service, automatically generates a Colored Petri Net and performs the desired analysis. This tool allows for interactive simulation. We can accurately locate the error position of Web service when there are problems in the CPN model. Figure 8 shows the architecture of WSM&VT, which includes the following components.

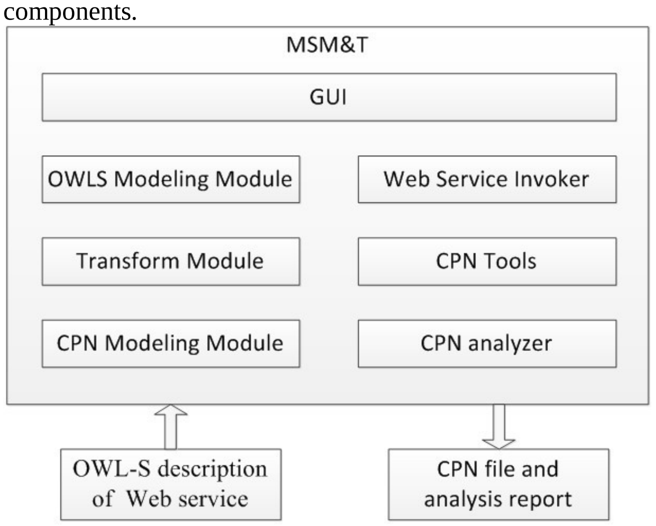


Figure 8. The architecture of WSM&VT.

The *OWLS Modeling Module* receives an OWLS description of Web service and generates the OWLS model of Web service.

The *Transform Module* implements the transforming algorithms of atomic Web service and composite Web service.

The *CPN Modeling Module* generates the CPN model of Web service.

The *Web service Invoker* generates the simulation invoke code of Web service in the CPN model and uses Web stub to receive the request of Web service.

The *CPN Tool* can execute the CPN model and locate the error position of Web service. It is developed based on access/cpn framework[17][18].

The *CPN analyzer* implements the state space method of CPN model and generates the CPN model of Web service.

We have already used our implements to model a variety of the existing Web services described by OWL-S. An example Web service is about buying books. The generated CPN model is illustrated in Figure 9.

Then we can get the analysis report as shown below:

State SpaceNodes: 10Arcs: 9

Scc GraphNodes: 10Arcs: 9

Dead Markings[3,6]

Dead Transition InstancesNone

From the above analysis report, the following conclusions can be drawn:

1) The nodes and arcs of SCCs and state space graph are same, which indicates that SCCs are trivial. Moreover, it also explains that in composite service, the combinational logic of process model does not include the rings and has the normal termination quality.
2) The number of Dead Markings indicates the colored Petri net model has two final states, which is consistent with the number of combinational service's output.
3) Without the Dead transition, it is supposed that all the transitions of the CPN model have been occurred and the model is just equipped with the normal state except other termination status.

Depending on the above results, it can be assumed that the structure of this composite service is effective.

VII. CONCLUSION

In this paper, we investigate the verification problem of Web service described by OWL-S.We propose a more practical approach for verifying Web services descried by OWL-S. We not only focus on the parameters and structure transformation, but also concentrate on how to get the outputs from the inputs when we transform Web services to CPN models. According to the executable character of CPN model, we can run simulations to accurately locate the error position of Web service when there are problems in the CPN model. The modeling approach is described by the following two aspects that are the atomic Web service and composite Web service. Based on the properties and state space method of CPN model, we conclude that the CPN model must satisfy three properties for ensuring the correctness of the Web service. Finally, we present a tool to implement our approach and introduce its effectiveness with an example. The OWL-S description file of Web service and the CPN model with its analysis results are respectively the tool's input and output. Furthermore, the results of this research will lay the foundation for the Web services composition and evaluation.

ACKNOWLEDGMENT

We would like to thank all those who have helped us. Without their careful guidance and help, we couldn't have accomplished this paper successfully. This paper is supported by National Natural Science Foundation of China-"The basic theory and key technologies of the low power and secure embedded processor" under Grant No. 61232009.

REFERENCES

[1] "semantic web", http://en.wikipedia.org/wiki/Semantic_Web.

[2] "web service", http://en.wikipedia.org/wiki/Web_service.

[3] W3C Member Submission 22 November 2004, "OWL-S:Semantic Markup for Web Services", http://www.w3.org/Submission/OWL-S.

[4] K. Jensen, "A Brief Introduction to Coloured Petri Nets", in Tools and Algorithms for the Construction and Analysis of Systems, ed. E. Brinksma, pp.203-208, Springer-Verlag, Enschede, 1997.

[5] K. Jensen, "An Introduction to the Practical Use of Coloured Petri Nets", in Lectures on Petri Nets II, ed. W. Reisi, and G. Rozenberg, pp:237-292, Springer-Verlag, Enschede, 1998.

[6] S. Narayanan, and S. A. Mcllraith, "Simulation, verification and automated composition of web services", Proc. the 11th international conference on World Wide Web, New York, 2002.

[7] Y. cheng, Z. wang, C. wang, and L. tang, "Modeling and Verifying Composite Semantic Web Service Based on Colored Petri Nets", Proc. the Sixth International Conference on Advanced Language Processing and Web Information Technology, HeNan, 2007.

[8] B. Yu, and D. Li, "Verifying web services of OWL-S with Petri net", Proc, 5th International Conference on Computer Science and Education, Hefei, 2010.

[9] R. Hamadi, and B. Benatallah, "A Petri Net-based Model Web Service Composition", Proc. the 14th Australasion Database Conference, 2003.

[10] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Model-based verification of web service compositions", Proc. the 18th IEEE International Conference on Automated Software Engineering Conference, 2003.

[11] G. Salaun, L. Bordeaux, and M. Schaerf, "Describing and reasoning on Web Services using process algebra", International Journal of Business Process Integration and Management, Volume 1, Number 2/2006, June 2006.

[12] "The CPN group at Aarhus University", http://cs.au.dk/CPnets/.

[13] L. M. Kristensen, S. Christensen, and K. Jensen, "The Practitioner's Guide to Coloured Petri Nets", International Journal on Software Tools for Technology Transfer, Volume 2, Issue 2, pp 98-132, December 1998.

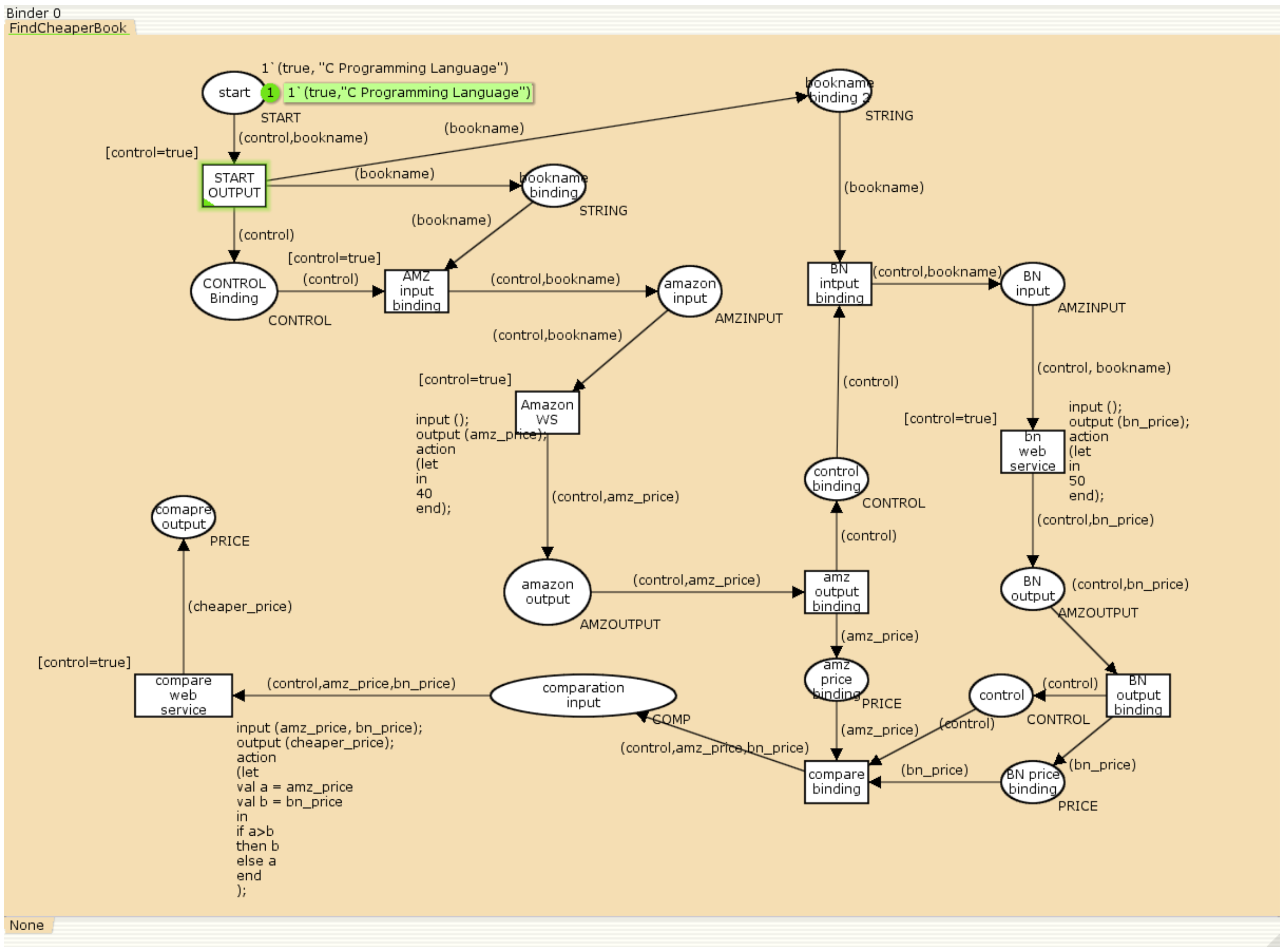
[14] A. Brogi, S. Corfini, and S. Iardella, "From OWL-S Descriptions to Petri Nets", in Service-Oriented Computing - ICSOC 2007 Workshops, Lecture Notes in Computer Science Volume 4907, pp 427-438, 2009.

[15] L. M. Kristensen, "State Space Methods for Coloured Petri Nets", University of Aarhus, 2000.

[16] J. B. Jorgensen, and L. M. Kristensen, "Verification of Coloured Petri Nets Using State Space with Equivalence Classes", in Petri Net Approaches for Modelling and Validation, ed. B. Farwer, D. Moldt ,M-O. Stehr.Petri Nets, University of Hamburg, 1997.

[17] M. Westergaard, "Access/CPN 2.0: A High-Level Interface to Coloured Petri Net Models", in Applications and Theory of Petri Nets, Lecture Notes in Computer Science Volume 6709, pp 328-337, 2011.

[18] M. Westergaard, and L. M. Kristensen, "The Access/CPN Framework: A Tool for Interacting with the CPN Tools Simulator", in Applications and Theory of Petri Nets, Lecture Notes in Computer Science, Volume 5606, pp 313-322, 2009.



```
input ();
output (amz_price);
action
(let
in
40
end);
```

```
input ();
output (bn_price);
action
(let
in
50
end);
```

```
input (amz_price, bn_price);
output (cheaper_price);
action
(let
val a = amz_price
val b = bn_price
in
if a>b
then b
else a
end
);
```

Figure 9. The generated CPN model.