Verifying Temporal and Epistemic Properties of Web service Compositions*

Alessio Lomuscio, Hongyang Qu, Marek Sergot, Monika Solanki

Department of Computing, Imperial College London, UK
E-mail: {alessio, hongyang, mjs, monika}@doc.ic.ac.uk

Abstract. Model checking Web service behaviour has remained limited to checking safety and liveness properties. However when viewed as a multi agent system, the system composition can be analysed by considering additional properties which capture the knowledge acquired by services during their interactions. In this paper we present a novel approach to model checking service composition where in addition to safety and liveness, *epistemic* properties are analysed and verified. To do this we use a specialised system description language (ISPL) paired with a symbolic model checker (MCMAS) optimised for the verification of temporal and epistemic modalities. We report on experimental results obtained by analysing the composition for a Loan Approval Service.

1 Introduction

Web services are now considered as one of the key paradigms underlying application integration. Several research efforts – both from industry and academia – have addressed varied aspects of service composition including verification via model checking. Most of the approaches [11,13] take BPEL [9] as the language for development and use model checkers such as SPIN [6] and NuSMV [3] for checking safety and liveness properties. These model checkers are limited to temporal modalities in the scope of properties they can analyse. However as we argue below, in addition to verifying temporal properties it is also necessary to predict and verify the knowledge gained by services during the composition.

In this paper we propose an alternative yet complementary approach to verifying service behaviour. As proposed by the W3C consortium: " A Web service is an abstract notion that must be implemented by a concrete **agent**. The agent is the concrete piece of software or hardware that sends and receives messages.", a composition of Web services can be viewed as a multi agent system [12].

There is a tradition in the multi agent systems (MAS) community to use rich logic-based languages to analyse the behaviour of agents in the system. In particular not only is temporal logic used but also, among others, epistemic (to reason about knowledge of the processes), deontic (to reason about obligation of the processes), cooperation (to reason about strategies of the agents) modalities. These

* The research described in this paper is partly supported by the European Commission Framework 6 funded project CONTRACT (IST Project Number 034418).

logic-based languages can be used to specify formally and unambiguously the behaviour of the system. Recent developments in the verification of MAS via model checking techniques [2,10] allow for the first time the verification of not only plain temporal languages but also a variety of modalities describing the informational and intentional state of the agents. In particular reasoning about the agents' knowledge is demonstrably of interest in a variety of applications, including coordination, security, communication, fault-diagnosis, networking, etc. This work has not yet been extended to the challenges of service composition. The aim of this paper is to make a step in this direction. In particular in this paper we show how MCMAS [8] can be used to model check rich specifications based on temporal-epistemic logic representing compositions of web-services.

The rest of the paper is organised as follows. In Section 2 we introduce the trace-based semantics of interpreted systems. Section 3 introduces a motivating example and some of its key specifications. In Section 4 we introduce MCMAS, a symbolic model checker for semantics of interpreted systems. The encoding of the example in a specialised language is also shown in this section, its key properties are checked automatically, and experimental results are discussed. We discuss related work in Section 5 and conclude in Section 6.

2 Preliminaries

The first class citizen within an interpreted system as applied to Web services is an agent that represents the concrete counterpart of a service in the composition. Below we summarise the framework of interpreted systems [4] as implemented in MCMAS. Every agent i ($i \in \{1, \ldots, n\}$) is characterised by a finite set of local states L_i for the service and a finite set of actions Act_i that the agent performs on behalf of the service. A Protocol defines the actions that may be performed by an agent in each of its local states and is defined as $P_i : L_i \rightarrow 2^{Act_i}$. The environment is modelled as a special agent with a set of local states (L_e), a set of actions (A_e) and a protocol (P_e). The set of global states of the composition can be defined as a non-empty subset of the Cartesian product $L_1 \times L_2 \times L_3 \ldots \times L_n \times L_e$. A global state of the system at a particular instant in time is therefore represented by a tuple $(l_1, l_2, \ldots l_n, l_e)$.

The evolution (transition) of the agents' local states is described by a function $t_i : L_i \times \ldots \times L_n \times L_e \times Act_i \times \ldots \times Act_n \times Act_e \rightarrow L_i$ that defines the next local state of an agent given the current local state and the action(s) that are performed in that state as per the protocol. The evolution of all the agents' local states describes a set of runs over the set of reachable states. It is assumed that in every state the agents perform simultaneous actions. Note that some agents may perform "null" actions. The evolution of the global states of the system is described by a function $t : S \times Act \rightarrow S$ where $S = L_1 \times \ldots \times L_n \times L_e$ and $Act = Act_1 \times \ldots \times Act_n \times Act_e$. Given a set $I \subseteq S$ of possible initial global states, the set $G \subseteq S$ of reachable global states is generated by all possible runs of the system. Finally, the definition includes a set of atomic propositions AP together with a valuation function $h : AP \rightarrow S$.

We adopt the syntactical constructs and semantic model for the interpretation of temporal-epistemic formulae in interpreted systems as presented in [8] to analyse composite Web services. Of particular interest to us is the formula $K_i\varphi$ for expressing epistemic properties. The formula is read as "Agent i knows φ". Epistemic properties capture knowledge that the agents and their environment acquire as the system evolves. Verification of epistemic properties ensures the correctness of this knowledge at various states within the system as interaction progresses. In terms of verification via model checking, in our approach, this can be defined as establishing whether or not $M_s \vDash K_i\varphi$. We can also verify complex specifications like $K_iK_j\varphi$ which informally expresses "Agent i knows that agent j knows φ".

3 A Motivating Example

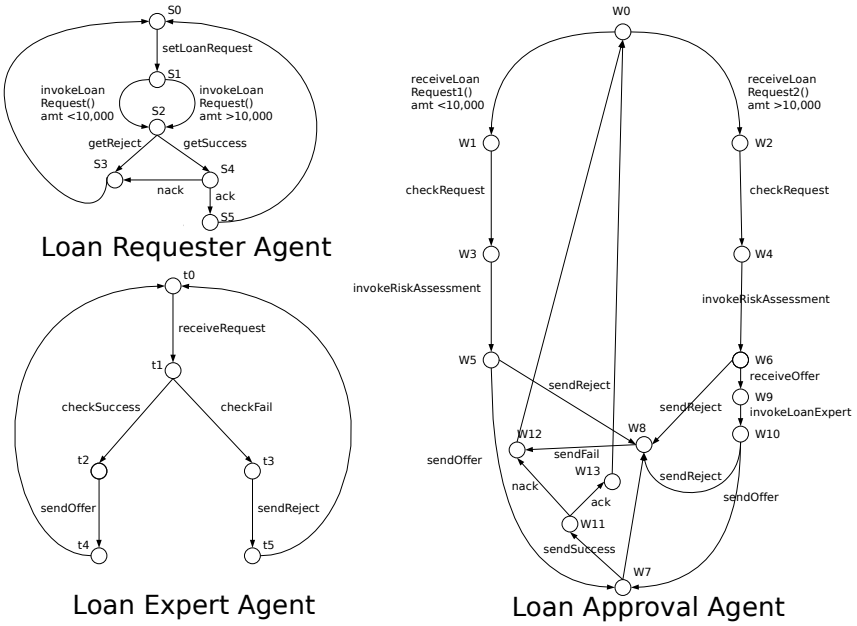

Fig. 1. Protocols for Agents in the Loan Approval Service

We take as our reference example a composition of services for Loan Approval as outlined in the WSBPEL specification [9]. Figure 1 shows the interaction protocols for the various services. At a high level of abstraction, these protocols can be viewed as individual BPEL representations of the processes. For simplicity in this paper, we do not model explicit communication between the agents. We assume that the underlying network for sending and receiving messages is reliable, communication is synchronous and message delivery is instantaneous. Asynchronous

communication can be easily modelled by allowing the agents to "wait" or do "nothing". It is also possible in our framework to model channels as environment for the agents in the systems and reason about their correct behaviour for e.g. coordination and synchronisation. However in this paper we abstract from modelling these.

3.1 Formalisation

We represent the above example using the formalism of interpreted systems. In order to verify a system with MCMAS, we need to translate the system into a model written in ISPL, which includes the following components:

- The definition of agents which describes the local behaviour of every agent, such as states, actions and protocols.
- The global evaluation function of the system which define atomic propositions held over global states, the combinations of local states.
- The local initial state of agents.
- Specifications to be checked. They are expressed as temporal-epistemic formulae.

In the example, we define four agents "Loan Requester (LRA)", "Loan Service (LSA)","Risk Assessor (RAA)" and "Loan Expert (LEA)". Each of them is modelled using their local states, local actions, protocols and transition functions.

For example, for the LRA the local states are $\{s_0, s_1, s_2, s_3, s_4, s_5\}$. The set of actions for the LRA includes *setLoanRequest*, *invokeLoanRequest1*, *invokeLoanRequest2*, *ack*, *nack*, *nothing*, *return1*, among which *invokeLoanRequest1* represents a request with amount less than 10,000 GBP, *invokeLoanRequest2* one with amount greater than 10,000 GBP, *nothing* is just a dummy action (corresponding to no-op) and *return1* is used to move to the initial state. The Protocol function in the definition explicitly specifies possible actions at each state: for example, at state s_1, only *invokeLoanRequest1*, *invokeLoanRequest2* are possible. If no action can be enabled, *nothing* is assigned to the state.

Finally the evolution function defines the behaviour of the agent, i.e., when and how the agent moves to another state. For example, LRA proceeds to the state s_1 if and only if it is in the state s_0 and executes the action *setLoanRequest*. In addition, the agent can jump to other states without firing a "local" action. This is done by following actions of other agents. For instance, LRA moves to state s_3 from state s_2 when agent LSA executes action *sendFail*. In this way, we can easily model synchronisation between agents. A typical scenario of synchronisation is when an agent sends a request to another and the latter has to receive it. Moreover, this mechanism allows us to reduce the total number of actions and thus the number of Boolean variables needed to encode the system which speeds up the verification. Asynchronous communication can be easily modelled as explained earlier. The Loan Service may choose not to receive the request sent by the Loan requester till the send operation is complete. In this case, the transition of the Loan service from state w_0 to state w_1 happens only after the transition *invokeLoanRequest1* of the loan requester from state s_1 to state s_2.

As observed, the evolution function provides a simple means of modelling coordination and synchronisation/asynchronisation between agents for the purposes of the paper. It also allows us to reduce complexity, while focusing on our core objective of verifying temporal-epistemic properties. More elaborate models of coordiantion and synchronisation are possible but will not be presented here.

4 Model Checking The Loan Approval Composition

MCMAS [8] is an OBDD based symbolic Model Checker for Multi Agent Systems. In addition to temporal modalities MCMAS allows the verification of epistemic, correctness and cooperation modalities. Input to the model checker is defined in ISPL. The evaluation function in ISPL maps atomic propositions to states, which specifies for every atomic proposition the set of states in which the proposition holds. For example the proposition *loanApproved* holds if the loan requester is in state s_4 or it is in state s_5.

We check the following epistemic properties: (1) if the loan request is approved by LAA, then LRA *knows* the fact that LSA *knows* that the request of LRA has low risk; (2) if the amount of the loan requested is greater than 10,000, the customer *knows* that his request will be directed to a Loan Expert. They are formalised as follows:

$$\begin{aligned} AF\ loanApproved \rightarrow (amountLess10000 \rightarrow K_{LRA}K_{LSA}LowRisk1) \\ \wedge(amountGreater10000 \rightarrow K_{LRA}K_{LSA}LowRisk2) \\ AF\ amountGreater10000 \rightarrow K_{LRA}expertInvoked \end{aligned}$$

We also tested two CTL formulae: $AF\ (loanFail \vee loanSucceed)$, which stands for eventually in all paths, a loan request would fail or succeed, i.e., LSA must make decision for every load request, and $AF\ amountGreater10000 \rightarrow EF\ loanReject$ which means that for all paths in which the loan amount is greater than 10,000 GBP, the request would fail in some paths.

MCMAS used 13 Boolean variable to encode local states, 12 for actions. It returned the result immediately, as the model is not complex. It is obvious that the four properties are true for the model. It is easy to produce a false property as well, for example, change "EF" into "AF" in the fourth formula. Due to space restrictions we do not present the complete ISPL code for the example; it is available on request.

5 Related Work

Several research efforts have addressed the problem of model checking Web service specification, however to the best of our knowledge this is one of the first papers to address the verification of epistemic properties of agents that represent Web services. Pistore et al [11] present a technique based on "Planning as Model Checking" for planning under uncertainty for composition and monitoring

of BPEL4WS processes. The Model checking approach uses the MBP Planner [1]. Fu et al [5] presents a framework where BPEL specifications are translated to an intermediate representation, using guarded automata as XPath expressions. This is followed by the translation of the intermediate representation to a verification language "Promela", input language of the model checker SPIN. Hu Huang et al [7] presents an approach using the BLAST model checker to verify the process models of OWL-S

6 Conclusions

In this paper we show that along with temporal modalities, epistemic properties for agents representing the services can be verified. We use the symbolic model checker MCMAS and verify temporal-epistemic properties for Loan Approval composition. As part of our future work we intend to investigate the explicit modelling of coordination and synchronisation between agents which are abstracted in this paper.

References

1. P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. MBP: a model based planner. In *In Proc. of the IJCAI'01*, 2001.
2. R. Bordini, M. Fisher, C. Pardavila, W. Visser, and M. Wooldridge. Model checking multi-agent programs with CASP. In *(CAV'03)*, volume 2725 of *LNCS*, pages 110–113. Springer-Verlag.
3. A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model verifier. In *Proc. CAV'99*, pages 495–499.
4. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.
5. X. Fu, T. Bultan, and J. Su. Analysis of interacting BPEL web services. In *WWW'04*, pages 621–630. ACM Press.
6. G. J. Holzmann. The model checker SPIN. *IEEE Trans. on Software Eng.*, 23(5):279–295, 1997.
7. Hai Huang, Wei-Tek Tsai, Raymond Paul, and Yinong Chen. Automated model checking and testing for composite web services. In *ISORC '05*, pages 300–307 IEEE Computer Society.
8. A. Lomuscio and F. Raimondi. MCMAS: A model checker for multi-agent systems. *TACAS'06*, volume 3920, pages 450–454. Springer Verlag.
9. OASIS Web service Business Process Execution Language (WSBPEL) TC. Web service Business Process Execution Language Version 2.0, 2007.
10. W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.
11. Marco Pistore, F. Barbon, Piergiorgio Bertoli, D. Shaparau, and Paolo Traverso. Planning and monitoring web service composition. In *AIMSA*, pages 106–115, 2004.
12. M. Wooldridge. *An introduction to multi-agent systems*. John Wiley, England, 2002.
13. X. Fu T. Bultan and J. Su. Conversation Protocols: A Formalism for Specification and Verification of Reactive Electronic Services. In *CIAA'03*, pages 188–200.