# An AUML State Machine Based Method for Multi-agent Systems Model Checking

Dapeng Zhang, Xiang Ji, Xinsheng Wang

# An AUML State Machine Based Method for Multi-agent Systems Model Checking

Dapeng Zhang[1,2], Xiang Ji[1], Xinsheng Wang[1]

[1]Institute of Information Science and Engineering, Yanshan University, Qinhuangdao, 066004, China
[2]Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, 541004, China

daniao@ysu.edu.cn, 907887343@qq.com, wxs@ysu.edu

ABSTRACT. This paper firstly proposes a Multi-agent System Model Checking Framework, which is based on AUML (Agent Unified Modeling Language) state machine model and temporal logics of knowledge and provides a method using AUML state machine for Multi-Agent Systems formal modeling. Then a method for the conversion from AUML state machine formal description to ISPL language is proposed. Finally a simulation is accomplished with the conversion tool AUML2ISPL.

Keywords: Model Checking, MCMAS, AUML state machine, ISPL, Multi-agent System

## 1 Introduction

Currently, agent and Multi-agent has become the focus of the research in artificial intelligence. Multi-agent System has been used more widely in computer and related fields than before. So it has become a very urgent issue to ensure the correctness and reliability for Multi-agent System. As a method of automatic verification for finite-state system, model checking [1] has drawn wide attention. The basic idea for model checking is to establish finite state models for the system that is to be verified, using some logical formula to describe the specification properties of the system which are expected, then put them into model checking tools to verify whether the model meets the specification properties by automatic checking. Model checking has a high degree of automation implementation and it can provide a counter-example path when the system does not meet the specification properties. So it can decrease the workload for researchers and it is also convenient for researchers to trace the error path for the system.

## 2 Related Work and the issue

Since the idea of model checking has been proposed, many researchers have paid a lot of attention for the consistency of model checking for Multi-agent System. Dong [2] has promoted a method about formal description of state machine use UML to defining the extension hierarchical automata and the label transfer system, then verified the correctness of the UML state machine model based on this method. Luo [3] has provided a method for automatic detection of the interaction Web service, and modeled the MTELG by 5-tuple, then proposed a rule to make the 5-tuple formal description automatic convertion to the input language for the model checking tool MCTK. Du [4] has promoted a method for UML consistency and tested case generation based on model checking. Abdel and Gherb [5] have provided a consistency test framework for UML/SPT model. This framework gives a method to verify the model state chart for UML/SPT, Which includes the consistency of syntax and semantics.

For Multi-Agent System, although UML is a popular modeling language, it can't describe the agents' knowledge and their collaborations well. Therefore scientists have extended the UML language and proposed a language for modeling Multi-agent System, which is called AUML (Agent Unified Modeling Language). Although scientists have developed a number of model checking tools, such as MCK [6], MCMAS [7] and MCTK [8], the input language for those tools is incompatible with UML, especially AUML.

## 3 A Model Checking framework for Multi-Agent Systems

In order to solve the problem proposed in section 2, we proposed a Multi-agent Systems Model Checking Framework that is based on AUML state machine model and temporal logics of knowledge [9]. Fig.1 shows this framework.

In this framework, we choose MCMAS as the Model Checking tool. It is because MCMAS is more efficiency, which is based on the logic of CTLKD-$A^{DC}$, It can verify the knowledge for Multi-agent System, and it has the ability to verify the correctness behaviors of agents and the collaboration among agents for Multi-agent System. MCMAS uses ISPL (Interpreted Systems Programming Language) [10] as the input language.

This Model Checking framework can be achieved in the following steps:

(1)Symbolic the Multi-agent System model to AUML state chart.

(2)Formalize the AUML state chart into AUML state machine formal description.

(3)Making a syntax conversion between AUML state machine formal description and the ISPL language.

(4)Formalized system specifications into CTLKD-A$^{DC}$ formulas, respectively put the CTLK-A$^{DC}$ formulas and the ISPL description that the third step obtained into MCMAS to check whether this model satisfied the formulas.

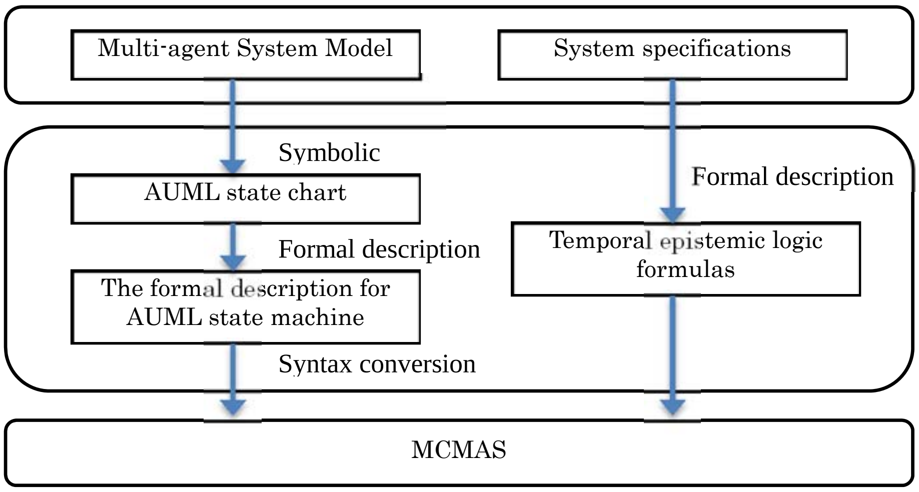


**Fig. 1.** The Model Checking framework for Multi-agent System

## 4 The conversion from AUML to ISPL language

### 4.1 The formal description of AUML state machine

AUML state machine model is very important. It can reflect specific circumstances from the interaction of agents. We extend the AUML state machine description syntax in reference [11]. Firstly we add the observed variable into AUML state machine description. Secondly we redefine the transition rules. The syntax for AUML state machine description is as follows.

$StateChartModel \triangleq agentname: String; states: StateSet;$
$\quad actions: Actionset; goals: GoalSet; transitions: TransitionSet$
$StateSet \triangleq \{State\}$
$ActionSet \triangleq \{Action\}$
$GoalSet \triangleq \{Goal\}$
$TransitionSet \triangleq \{Transition\}$
$State \triangleq statename: String; statetype: StateType;$

$boolformula: BoolFormulaSet$

$BoolFormulaSet \triangleq \{BoolFormula\}$

$BoolFormula \triangleq var: Var;\ relation: Relation;\ value: Value$

$Var \triangleq \{Variable\}$

$Relation \triangleq \{' \leq ';\ ' \geq ';\ ' \neq ';' > ';' < '\}$

$Value \triangleq \{Int; String; Float; Bool\}$

$StateType \triangleq \{Initial; Final; Common; FaultState\}$

Transition description is the instance of the connection of two states. The rules consist of the triggered state transitions preconditions and the results after transitions. The preconditions include variable assignments for the state before the state transition and the other agent actions that triggered state transition (represented by the message receiving events), while the results include variable assignment after state transition and the agent collection that other agents receive messages from. Their syntax is described below.

$Rule \triangleq condition: Cdt; agentaction: AtentAction; result: Rst$

$Cdt \triangleq \{BoolFormula\}$

$AgentAction \triangleq agent: Agent; action: Action$

$Rst \triangleq \{varassign: VarAssignSet\}; \{agent: Agent\}$

$VarAssignSet \triangleq \{VarAssign\}$

$VarAssign \triangleq var: Var; value: Value$

### 4.2 The conversion from AUML state machine formal description to ISPL language

Here we decompose the description of state transition for AUML state machine, and give protocol description and evolution description for ISPL respectively. Finally we propose a conversion algorithm between AUML state machine and ISPL language.

The description of protocol is represent by protocol functions: protdef::= Protocol : protdeflist. Here protdef is composed of a condition Protocol and a list of executable actions protdeist.

The evolution is represent by evolution functions: evline::= boolresult if gboolcond. Here each evolution function is composed of a group of boolresultes and the preconditions for evolution gboolcond.

The conversion algorithm is given in Algorithm 1.

## 5 Simulation

### 5.1 The state machine formal description for transmission system

The bit transmission system [12] contains two agents (Receiver, Sender), which is shown in Figure 2. One protocol to achieve communication is as follows: S immediately starts sending a bit to R, and continues to do this until it receives an acknowledgement from R. R does nothing until it receives the bit. Then it sends acknowledging to S. S stops sending the bit to R when it receives the first acknowledgement from R, and the protocol terminates. We encode each agent for the bit transmission system in AUML state machine formal description respectively. Here we give part of each agent encoding result in AUML state machine formal description.

Algorithm 1 conversion from AUML state machine formal description to ISPL language

```
1. Input: StateChartModel
2. Output: Protocol, Evolution
3. For each transition ∈ StateChartModel.transitions
4.    if transitions.source = state, then protdeflist = protdeflist + ' ' +transition.rule.act.name + ',';
5. For each boolformula ∈ state.boolformula, order protocol = protocol + ' ' + boolformula.vas.name + ' ' + relation + ' ' + String (boolformula.vas.value) + ' and ';
6. Add protdef into Protocol;
7. For each transition ∈ StateChartModel.transitions
8.    Output protocol;
9. For each vasassign ∈ transition.rule.result. vasassign, order boolresult = boolresult + '( ' + vasassign.vas.name + ' = ' + String (vasassign.value) + ' ) ' + ' and ';
10. gboolcond = ' if ', for every boolformla ∈ transition.rule. condition.boolformula, order gboolcond = gboolcond + '( ' + boolformula.vas.name + ' ' + relation + ' ' + String(boolformula.value) + ' and ';
11. gboolcond = gboolcond + transition.rule.condition.action + ' )';
12. evline = boolresult + gboolcond;
13. Add evline into Evolution;
14. Output: Evolution
```

For the Sender, its state is expressed by two variables: bit and ack. An emulation type bit encodes the value of the bit the Sender wants to send, which consists b0 and b1. Ack is a boolean variable encoding whether or not an ack has been received from Receiver. If the Sender receives the ack, ack equals true. The actions collection for Sender is expressed by Actions which consists sb0, sb1 and noting, respectively indicating the value of the bit the Sender sends. There are two initial state situations for the Sender: transmit b0 or transmit b1. Here we encode one state transition for Sender as follows.

Transition = S00, S1; rule : Rule;

Cdt = { boolformula: b0 }

Agent : Environment, Receiver;

Action : ( Receiver.Action = sendack and Environment.Action = R ) or ( Receiver.Action = sendack and Environment.Action = SR );

Rst = {(b0, ack = false) }; { agent: Receiver, Environment }

For Receiver, one enumeration variable is declared for the agent, representing whether the bit has been received or not and the value it's received, the collection of enumeration variable is {r0, r1, empty}. The collection of actions is {sendack, noting}. Sendack indicates Receiver send an acknowledgement to Sender.noting indicate Receiver don't send anything. Here we encode one state transition for Receiver as follows.

Transition = R00, R0 ; rule : Rule;

Cdt = { boolformular : empty };

Agent : Sender , Environment;

Action : ( Sender.Action = sb0 and Receive.state = R00 and Environment.Action = S ) or ( Sender.Action = sb0 and Receiver.action = noting and Environment.Action = SR )

Rst = { boolformula: r0 }; { agent: Environment, Sender }

## 5.2 Specification Verification

We use the CTLK-$A^{D,C}$ formula to formalize the specifications of bit transmission System. It can describe the correctness of actions and the knowledge for the Agents. The specification is usually based on system requirement. Here we manually input the following specifications.

1、 E((bit0 or bit1) U recack);

2、 EF(EG(recbit and !recack));

3、 AF(recack);

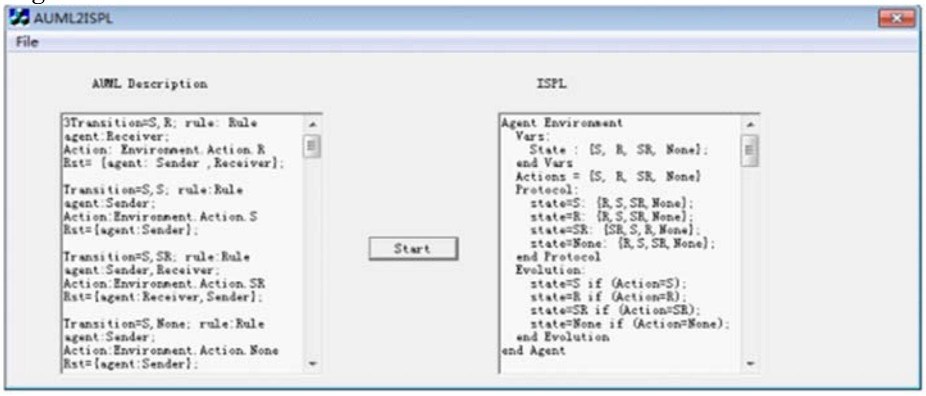
4、 AF(recbit -> AF (recack));

5、 AF(recbit -> K(Sender,K(Receiver,bit0) or K(Receiver,bit1)));

Formulas 1-4 are CTL formulas and they can describe the correctness actions for agents. Formula 5 is CTLK-$A^{D,C}$ formula and it can describe the reasoning ability for agents. For example, formula 1 means if agent Sender sends message, it can get the acknowledgement. Formula 5 means it is always true. If an acknowledgement is received, then the sender knows that the receiver knows the value of the bit.

### 5.3 Simulation

We realize the conversion tool AUML2ISPL according to algorithm 1. Then we put the whole AUML state machine formal description for bit transmission System into this tool, and get the conversion result as Fig.2 shows.



```
3Transition=S,R; rule: Rule
agent:Receiver;
Action: Environment.Action.R
Rst= {agent: Sender ,Receiver};

Transition=S,S; rule:Rule
agent:Sender;
Action:Environment.Action.S
Rst={agent:Sender};

Transition=S,SR; rule:Rule
agent:Sender,Receiver;
Action:Environment.Action.SR
Rst={agent:Receiver,Sender};

Transition=S,None; rule:Rule
agent:Sender;
Action:Environment.Action.None
Rst={agent:Sender};
```

```
Agent Environment
  Vars:
    State : {S, R, SR, None};
  end Vars
  Actions = {S, R, SR, None}
  Protocol:
    state=S:  {R,S,SR,None};
    state=R:  {R,S,SR,None};
    state=SR:  {SR,S,R,None};
    state=None:  {R,S,SR,None};
  end Protocol
  Evolution:
    state=S if (Action=S);
    state=R if (Action=R);
    state=SR if (Action=SR);
    state=None if (Action=None);
  end Evolution
end Agent
```

**Fig. 2.** The conversion result from AUML2ISPL

We put the conversion result and the system specifications into MCMAS and run a simulation. The model checking result shows that out modeling is correct.

## 6 Conclusion

This paper uses AUML language for the modeling language of Multi-agent System, proposes a method for AUML state machine formal description and the conversion algorithm from the AUML formal description to ISPL, and the conversion tool is realized. Simulation shows that out method is feasible.

## Acknowledgements

This work is supported by the Natural Science Foundation of China (No. 61303129) and Open Foundation of Guangxi Key Laboratory of Trusted Software (No. 201211)

## References

1. Lin huimin. Zhang wenhui. Model Checking: Theories, techniques, and applications, Acta Electronica Sinica, 12(30), 1906-1912 (2002)

2. W. Dong, J. Wang, X. Qi, Z.C. Qi. Model checking UML statecharts. In: Proc. of 8th Asia-Pacific Software Engineering Conference (2001)
3. XiangYu Luo, Zheng Tan, ShengRong Dong. Automatic Detection Method for Web Services Feature Interaction. Computer Science, 37(12), 106-110 (2010)
4. Du. UML consistency and test case generation based on model checking. College of Computer Science and Technology, NanJing, Master Dissertation (2011)
5. Abdel Gherbi, Ferhat Khendek. Consistency of UML/SPT Models. In: E.Gaudin, E.Najm, and R.Reed (Eds.): SDL 2007, LNCS 4745, pp. 203-224, Springer-Verlag Berlin Heidelberg (2007)
6. P.Gammieand, R.V.Meyden. Mck: Model Checking the logic of Knowledge. In: CAV 2004, 478-483, Springer-verlag (2004)
7. Lomuscio,A, Raimondi, F. MCMAS: A model checker for multi-agent systems. In: H.Hermanns and J.Palsberg (Eds.): Proc. of TACAS 2006, Vienna, volume 3920, pp. 450-454 (2006)
8. Su Kaile, Luo Xiangyu, Abudul Scatter. The Interpreted System Model of knowledge, belief, desire and intention. In: The fifth international Joint Conference on Autonomous Agents and Multi-Agent Systems, ACM, 220-222 (2006)
9. Chen Bin, Wang Zhixue. Symbolic Model Checking Algorithm for Temporal Epistemic logic CTL*K. Computer Science, 36(5), 214-219 (2009)
10. R.Fagin, J.Y Halpen, Y. Moses and M.Vardi. Reasoning about Knowledge. MIT Press, Cambridge (1995)
11. ZhiKun Zhao. Research on Agent Unified Modeling Language. Institute of Computing Technology, Chinese Academy of Sciences, Beijing, PHD Dissertation (2003)
12. Franco Raimoudi. Model Checking Multi-Agent Systems, Department of Computer Science London University, London, PHD Dissertation (2006)