VNU Journal of Science, Natural Sciences and Technology 28 (2012) 1-10

Automated analysis of consensus protocol in specifcation of multi-agents coordination

Trinh Thanh Binh[1,*], Truong Ninh Thuan[2], Nguyen Viet Ha[2]

[1]*Haiphong University, 171 Phan Dang Luu, Kien An, Haiphong, Vietnam*
[2]*VNU University of Engineering and Technology, 144 Xuan Thuy, Hanoi, Vietnam*

Received 10 January 2012

Abstract. Formal specification and reasoning techniques in software modelling are needed to ensure the correctness of the system at the design phase. Event-B is a formal method with support tools that allows the specifcation and verifcation of reactive systems. In this article, we propose an approach to specify capabilities of a number of software agents. We also verify whether these capabilities help the agents to accomplish a task using a support tool of Event-B. In our previous paper, we have presented about the specifcation and verifcation of sequential protocols. We extend in this article the one of combination between the sequential and parallel protocols of multi-agents software.

1. Introduction

Coordinated consensus problems [1] have a long history of study in computer science and their solutions have become an important foundation of reactive systems such as multi-agent systems. These problems can now be formally and efficiently analyzed thank to the development of formal methods in software specification and verification.

A multi-agent system [2, 3] is a collection of subsystems in which each subsystem, called an agent, updates itself in accordance with the information it gathers from some of the other agents, i.e from its *neighbors*. In general, the neighbors of an agent are subject to change in time which introduces a switching behavior to the dynamics of the system through communication links. It has been proved that it is very important to study and understand the effect of this varying communication topology on some common task to be accomplished (i.e reaching a *consensus*) by the agents composing the system.

The B method [4] is a formal software development method, originally created by J.-R. Abrial. The B notations are based on the set theory, generalized substitutions and the first order logic. Event-B [5] is an evolution of the B method that is more suitable for developing large reactive and distributed systems. Software development in Event-B begins by abstractly specifying the requirements of the whole system and then refining them through several

* Corresponding author. Tel: 84-988681275.
E-mail: binhtt.di07@vnu.edu.vn

steps to reach a description of the system in such a detail that can be translated into code. The consistency of each model and the relationship between an abstract model and its refinements are obtained by formal proofs. Rodin platform [5] is the tool supports for Event-B specification and proof.

In this article, we propose an approach to build a specification of a multi-agent system and then to prove the coordinated consensus of agents in the specification using Event-B. In our approach, each agent is specified by an abstract machine which sees its context machine. The context and abstract machines of agents are later composed to form general ones as the whole systems according to the rules of the protocol consensus algorithms. The support tools provided by Event-B enable to formally analyze the coordinated consensus of agent specifications through the composed machines. In our previous work [6], we have been working with sequential protocol, this artcile extends the previous work to analyse protocols contained both sequential and concurrent events.

The rest of this paper is organized as follows. Section 2 presents our main contribution of using Event- B to specify multi-agent systems and to prove the consensus of the coordination of some agents. We also illustrate in this section the proposed approach by a case study, the multi-agent system for calculations of binary numbers. Section 3 discusses related works. We conclude the paper and give some future works in Section 4.

2. Our approach of coordinated consensus analysis using Event-B composition

In multi-agent systems, each agent and their capabilities is provided to perfom particularly tasks. These agents can be coordinated to solve a problem which is impossible or otherwise difficult for an individual agent or monolithic system to solve. However, the cooperation of agents has been lacked the consensus analysis in specification and design tools. As a result, we propose an approach to analyse the cooperation of agents to accomplish a task using Event-B notation and tools.

Note that, “consensus” means to reach an agreement between agents regarding a certain quantity of interest that depends on the state of all agents. A “consensus algorithm” (or protocol) is an interaction rule that specifies the way of information exchange between an agent and all of its neighbors [7]. Before introducing the approach of coordinated consensus analysis using Event-B, we give some definitions and their corollary related to Event-B specification, they are useful in the analysis process.

Defnition 1: $\hat{e} @\{e =< g, a > | [a]_{\rightarrow false}\}$

An iterative event $e = <g, a>$ is called convergence when the guard g is hold and then their list of actions a executed until g is not hold in a finite of execution steps. Note that, when the guard g of an convergent event is unsatisfied, proof obligations of deadlock freeness defined in a model machine will be unproved. Thus, to prove the convergence of the model, we have to add a new event $e' = <g', a'>$ to the model such that $g \vee g'$ is always hold under the execution of a or a'.

Corollary 1. If $e= <g, a>$ then $\exists e' = <g', a'>$ such that $g \wedge g' = true$

This corollary can be stated that the value of a convergent iterative event can be obtained by a new added event.

Proof. Suppose that $e = <g, a>$ is a convergent event. According to the Definition

1, the guard g will be unsatisfied in a finite of execution steps. Then we can define additionally a new event e' = <g', a'> in the machine in order to get results of variables in event e where $g' = \neg g$, in a simple case.

Defnition 2:

$$\hat{E} @\{S(e_i =< g_i, a_i >) \mid [S(e_i)] \vee g_i \rightarrow false\}$$

The interaction of events which conforms to a sequential protocol execution is convergent when:

• The order of events is conformed to the protocol execution,

• The disjunction of all guards of related events will not be hold in a finite steps of execution.

The disjunction of all guards of related events will not be hold, then deadlock freeness of the model is violated. We have to add a new event e' = <g';a'> such that $g_1 \vee g_2 \vee ... \vee g_n \vee g'$ hold.

Corollary 2. If $\hat{E} = S(e_i)$ then ∃e′ = <g′, a′> such that $\vee g_i \vee g' = true$

When the interaction of events is convergent, we can obtain the convergent value by a new added event. The proof of the Corollary 2 is similarly as the one of the Corollary 1. In order to analyze the coordinated consensus of the composition of the agents in a multi-agent system, we first compose the Event-B specification of the system. The architecture of the specification is depicted in Figure 1.

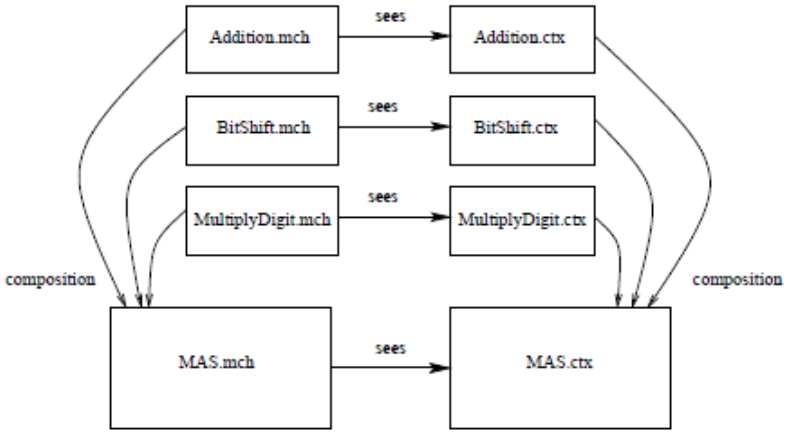

Figure 1. Composition of agent machines.

In this specification, the context machines of different agents will be combined into a context machine of the system called MAS.ctx, while the abstract machines of different agents will be combined into a general abstract machine called MAS.mch.

Note that composition in Event-B has been proposed earlier by Poppleton in [8]. His approach of composition is roughly described as follows. Let $M1$ and $M2$ be two models which are proposed to be fused. The variables and events from each model will be combined

with the ones from the other model. That is, they concatenate the variable lists and events, conjoin those events with common names (in a manner to be defined) in a new model M. The variable list $\boldsymbol{v}$ in $M1$ comprises the list x of actioned variables and the list y of skipping variables for each event. Similarly variables list w in $M2$ comprise the list z of actioned variables and the list a of skipping variables for each event. They define $xz = x \cap z$, the common actioned variables, and $ya = y \cap a$, the common skipping variables. Note that the other intersecting variable lists yz and xa are both empty, to enable meaningful composition definitions [8].

This approach focuses on the composition of independent events and variables of machines and on assuring the correctness of proof in the composition machine. It means that, the events in systems do not affect results of the execution after the composition. In addition to the above, the composition mechanism in Event-B has to ensure that the overall behavior of the abstract model is kept and that the concrete model does not get into two states:

- Divergence: this situation occurs when the system behaves chaotically, it happens whenever some events are aborted.
- Deadlock: this situation occurs when no event is enabled and as a consequence, the system's state never change since it happens.

The first constraint (non-divergence), imposes to exhibit a variant V, which is a well-founded structure (e.g. $N,\leq$), is proved to be decreased by a wellfounded relation. The second constraint (deadlock freeness) is proved by proof obligations which state that the disjunction of the event guards always hold under the properties of the constant and the invariant. The absence of the divergence and the deadlock can be proved by the support tool of Event-B. This leads us to the thought of applying the idea of machine composition to compose machines of agents and to analyze the coordinated consensus between agents. In this composition, the interaction between events plays an important role in accomplishing a task.

Note that, in order to avoid the ambiguity in the case that an agent may have many capabilities, we decompose it to several model machines, each of them corresponds to one capability. The decomposition process is applied to a model machine which specifies an agent with more than one capabilities [9]. Alternatively, we can also specify each capability of the agent by a model machine at the beginning.

Suppose that, in our model, an abstract machine specifies an agent's capability, which is expressed by a quatro-tuple $Mi = <v_i , Init_i , ec_i , ee_i >$. where v_i is the list of variables, $Init_i$ is the initial event of the agent's machine i , ec_i is the list of events which completely specify the agent capability, and eei is the event used to obtain the result (see the Corollary 1).

Defnition 3. *A Mult is a quatro-tuple Mult =<Ag; Mact; α; Γ> where:*

- *Ag is a fnite set of agents (in a MAS),*
- *Mact is the set of capabilities possible in Mult,*
- *α: Mact → Ag assigns to each capability of Mact the agent that performs it,*
- *Γ is the execution protocol between capabilities toaccomplish a task.*

Then, let $M = <V, Init, ec, ee, eeM>$ be the composed machine for the agent capabilities Mi, $i = 1,...,n$. Depending on the protocol execution of agents which contains only

sequential events or with parallel events, we establish the composition machine.

2.1. Sequential protocols

In the case that the protocol contains only sequential events, without the loss of generality, we suppose that the execution protocol between events in sequential protocols of multi-agents system is the ordering $T @[ec_1, ec_2, ec_3, ..., ec_n]$, visually presented in Figure 2 using protocol diagram of AUML.

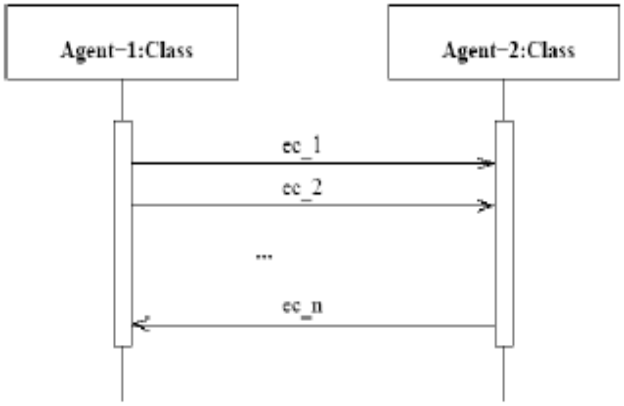

Figure 2. Sequential protocol.

The construction of the composition machine *M* of Event-B notation has to be conformed to the following principle:

• $V = \cup v_i$, the list of variables of the composition machine contains variables of agent machines

• $ec = \cup ec_i$, the composition machine contains all events of the agent machines

• $Init = Init_1$, the *Init* event of the composition is defined as the *Init* event of the first capability machine in protocol execution.

• $ee = \cup ee_i$ where $ee_i = Init_{i+1} \cup ee_i$, to activate the events of the next capability, a part of its event *Init* is combined to the get result event of the previous capability.

• ee_M is the new event added to the model to get final result of computation process.

After executing the composition, we have to optimize the MAS model machine and its context by eliminating some constants and variables which are redundant or unnecessary. The principle proposed above is reasonable because it can make the events executed in the order of the protocol consensus. The event ec_1 is executed first via the definition *Init* of the machine, then the next event will be executed via a part of the definition of the get result event of the previous event.

2.2. Parallel protocols

In the case that the protocol contains not only sequential but also concurrent events, we suppose that the execution protocol between events of agents Γ defined formally as follows:

$$\Gamma \quad ::= scenario$$
$$(1) \quad e \quad event$$

(2) $|\Gamma; e$ *sequence*
(3) $|\Gamma||e$ *parallel*

The protocol contains concurrent events in an agent system may be visually presented in Figure 3. Note that, in the Event-B model, the events are fire concurrently if the guards of these events satisfy an environment condition together. The protocol is thus convergent when each of event must be convergent. As a same way as sequential scenario, if the execution of concurrent events are convergent, we also add an event to the scenario to get results of these events.

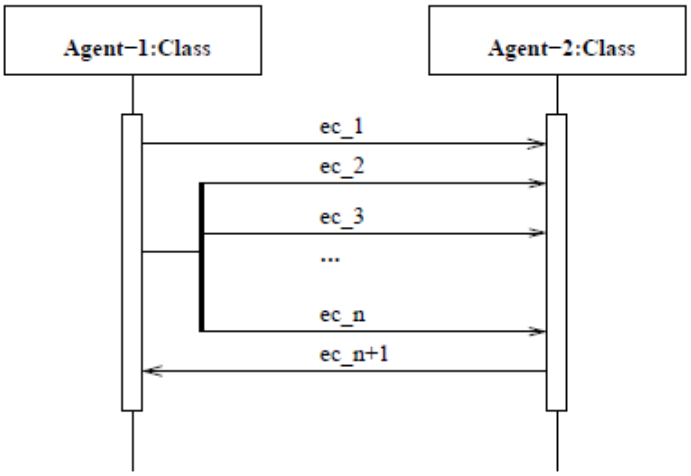

Figure 3. Parallel protocol.

The construction of the composition machine *M* in this case is the merging between sequential and parallely event protocol. The sequential part is done in similar manner as the one presented above. The concurrent events part are constructed as the following principle:

1. From the previous sequential event, activate the guard of all executed concurrent events such that they work at the same time;

2. With each event ee_i executed parallely, we add a get result event ee_{is} (we can do it because each of this one is convergent);

3. Add an event ee_P to get the final result of papallel process, this event will be enabled by ee_{is} ;

4. The get result event ee_P is responsible to activate the next sequential event in the protocol.

The convergence of each event and of the interaction between events will be automatically proved by the tool support of Event-B [5].

2.3. Description of a case study

Supposing that in a multi-agent system, an organisation contains agents used to calculate the results of some operations for binary numbers: BitShift, Sum and MultiDigit agents.Multiplication for binary numbers works in the same way as for the decimal numbers. In our multi-agent system, we have three

capabilities: *multiplyWithOneDigit*, *shiftLeft* , and *addition*.

These capabilities respectively belong to the MultiDigit agent, the BitShift agent, and the Sum agent. In the BitShift operation, the digits are shifted left or right. The shiftLeft capability of the BitShift agent enables digits in a binary number move a number of bits left by the operation with the same number of right bits being filled up by zero. For instance, if we apply the shift left operator by one position to the binary number 00011011, we obtain the number 00110110. The BitShift agent can be specified by AUML as depicted in Figure 4.

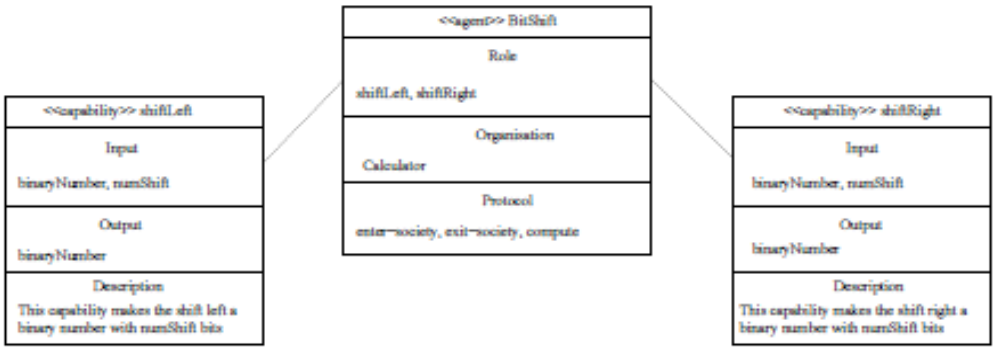

Figure 4. Bit shift agents specification.

The Sum agent is used to add or subtract two binary numbers. The input of the Sum agent are two binary numbers and the output is a binary number as the result of the operation. For example:

$$\begin{array}{r} 00011011 \\ +\ 00011011 \\ \hline 00110110 \end{array}$$

The capabilility of the MultiDigit agent is to multiply a binary number by one digit number (0 or 1). The input is a binary number and a binary digit, the output is a binary number as the result. The specification of the case study in Event-B notation can be found in [10, 6].

3. Related work

In the literatures, there are many papers proposed to formalize multi-agent systems using different formal methods in order to support the formal verification of the system. Hilaire [7] proposed a general framework for modelling multi-agent systems based on Object-Z and statecharts. This framework focused on rganisational aspects in order to represent agents and their roles. Similarly, Regayeg [11] combined Z notations and linear temporal logic to specify the internal part of agents and the specification of the communication protocols between agents. They proposed general patterns and the use of Z

support tools to modelcheck their specifications.

H. Fadil and J. Koning presented a work [12] involving the use of classical B to model agents roles and interactions. The goal of the paper is to model the interaction between agents with a formal method that is able to check and then prove their initial UML specification. The paper [13] also focused on the interaction protocol between agents using Event-B. Some patterns for the B specification of fault tolerance protocols are proposed in the case of agent communication.

A. Lanoix [14] proposed an approach to report their experience with the Even-B stepwise development of a situated MAS which study the displacement of vehicles in a convoy. In the case study, they suppose that all the vehicles move in a simultaneous movement using Event-B to ensure a safety property of the system: no collision must occur between a vehicle and its predecessor.

The papers above try to specify protocol execution between tasks of agents using Z, classical B, Event- B, etc. but it did not provide an approach to check if tasks are coordinated consensus, that our paper proposes. The coordinated consensus problems in multi-agent systems have been also considered in [1,15, 16].

R. Carli [1] discussed a work concerned a group of autonomous mobile agents in order to analyse a common task, communications constraints impose limits on the achievable control performance. Analysing the consensus or state agreement problem, the authors characterize the relationship between the amount of information exchanged by the agents and the rate of convergence to the agreement.

Another approach to the coordinated consensus problem of multi-agent systems is presented in [15]. In this approach, the authors introduce a characterization of contraction for bounded convex set. For discrete-time multi-agent systems, the authors provide an upperbound on the rate of convergence to a consensus under some assumptions.

However, these papers focused on analysing coordinated consensus problems using mathematical models, this is still impossible to use support tools to prove the convergence of tasks.

4. Concluding remarks

Multi-agent systems play an important role in developing complex or distributed information systems. As each agent of the system usually be designed to be autonomous and does not aware of other agent existences, it is difficult for developers to ensure the coordinated task of these agents will be accomplished.

In this article, we proposed an approach to specify multi-agent systems and then verify the consensus property of agents using Event-B. In our approach, each agent is specified by an abstract machine which sees its context machine. A context machine here refers to the environment of the agents. The interactions between agents are specified as protocols or algorithms that modify machine states. The context and machines of agents are then composed to general ones as the whole systems according to the rules of the protocol consensus algorithms. Then, we can use Event-B tools to formally analyze the coordinated consensus of agent specifications through the composed machines. We have provided the rules for

specifying the protocol contained both sequential and parallel events.

We illustrated our approach by an example of a binary multiplication system. In this system, the result of a multiple operation is accomplished by the collaboration of different agents. Using Rodin platform, we proved that the system will reach the state that provide the result. However, this case study just illustrated only the specification and verification of sequential scenarios.

Our approach is based on the proving ability of Event-B tools so it cannot cope with large multiagent systems which have a large number of agents and complex interactions. Another limitation is that it only works with simple consensus problems that can be specified by Event-B. We are working to extend our approach to check the plan of agents.

Acknowledgments

This work is partly supported by the research project No. QG.11.32 granted by Vietnam National University, Hanoi.

References

[1] R Carli et al. "Communication constraints in coordinated consensus problems". In: *American Control Conference*. IEEE, 2006.

[2] Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artifcial Intelligence*. The MIT Press, 2000.

[3] Michael Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2002.

[4] J.R. Abrial. *The B-Book, Assigning Programs to Meanings*. Cambridge University Press,1996.

[5] http://event-b.org.

[6] Ninh-Thuan Truong, Thanh-Binh Trinh, and Viet-Ha Nguyen. "Coordinated consensus analysis of Multi agent systems using Event-B". In: *SEFM'09: Proceedings of the Formal Methods and Software Engineering*. IEEE Computer Society, 2009, pp. 201–209.

[7] V. Hilaire et al. "Formal specification approach of role dynamics in agent organisations: Application to the Satisfaction-Altruism Model". In: *Int. Jour. of Software Engineering and Knowledge Engineering* 17.5 (2007), pp. 615–641.

[8] Michael Poppleton. "The Composition of Event-B Models". In: *ABZ '08: Proceedings of the 1st international conference on Abstract State Machines, B and Z*. London, UK: Springer-Verlag, 2008, pp. 209–222.

[9] Michael Butler. "Decomposition Structures for Event-B". In: *IFM'09: Proceedings of the 7th International Conference on Integrated Formal Methods*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 20–38.

[10] T. Binh Trinh. "Verifying Java concurrent components". Phd Thesis. Vietnam National University, Hanoi, 2011.

[11] A. Regayeg, A.H. Kacem, and M. Jmaiel. "Specification and verification of multi-agent applications using temporal Z". In: *Intelligent Agent Technology Conference*. IEEE Computer Society, 2004, pp. 260–266.

[12] H. Fadil and J. Koning. "A formal approach to model multiagent interactions using the B formal method". In: *International Symposium on Advanced Distributed Systems*. Vol. 3563. LNCS. Springer Verlag, 2005, pp. 516–528.

[13] E. Ball and M. Butler. "Event-B patterns for specifying fault-tolerance in Multi-Agent interaction". In: *Proceedings of Methods, Models and Tools for Fault Tolerance*. Vol. 5454. LNCS. Springer, 2009.

[14] Arnaud Lanoix. "Event-B Specification of a Situated Multi-Agent System: Study of a Platoon of Vehicles". In: *TASE '08: Proceedings of the 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*. IEEE Computer Society, 2008.

[15] Sezai Emre Tuna and Rodolphe Sepulchre. "Quantitative convergence analysis of multiagent systems". In: *7th IFAC Symposium on Nonlinear Control Systems*, 2007.

[16] R. Olfati-Saber, J. A. Fax, and R. M. Murray. "Consensus and Cooperation in Networked Multi-Agent Systems". In: *Proceedings of the IEEE,* 2007, pp. 215–233.

Phân tích tự động sự đồng thuận trong đặc tả sự phối hợp của tác tử

Trịnh Thanh Bình, Trương Ninh Thuận, Nguyễn Việt Hà

Trường Đại học Hải Phòng, 171 Phan Đăng Lưu, Kiến An, Hải Phòng, Việt Nam
Trường Đại học Công nghệ, ĐHQGHN, 144 Xuân Thủy, Hà Nội, Việt Nam

Các phương pháp đặc tả hình thức và suy luận thường được sử dụng nhằm bảo đảm tính đúng đắn của hệ thống phần mềm tại pha thiết kế. Event-B là một phương pháp hình thức được cung cấp sẵn các công cụ hỗ trợ cho phép đặc tả và kiểm chứng tự động các hệ thống phản ứng lại (reactive system). Trong các bài báo trước chúng tôi đã đề xuất một phương pháp sử dụng Event-B để đặc tả và chứng minh tự động sự tương tác (giao thức tuần tự) giữa các tác tử phần mềm để cùng nhau thực hiện một nhiệm vụ. Trong bài báo này, chúng tôi tiếp tục mở rộng các kết quả để đặc tả và phân tích sự tương tác giữa các tác tử thông qua sự kết hợp giữa giao thức tuần tự và song song.