Safety Property Analysis Techniques for Cooperating Embedded Systems Using LTS

Woo Jin Lee[1], Ho-Jun Kim[1], and Heung Seok Chae[2]

[1] EECS, Kyungpook National University, Sangyeok-dong, Buk-gu, Daegu, South Korea
woojin@knu.ac.kr, sisqo00@nate.com
[2] Department of Computer Science and Engineering, Pusan National University,
30 Changjeon-dong, Keunjeong-gu, Busan, 609-735, South Korea
hschae@pusan.ac.kr

Abstract. Safety issues of cooperating embedded systems are very important since they are closely related to our living. In this research, modeling techniques and safety analysis techniques for cooperating embedded systems are provided. Behaviors of embedded systems and safety properties are described by Labeled Transition Systems (LTS). For convenient and effective analysis, we provide a slicing method of the state space of a system according to a property. Based on the slice models, we provided an equivalence algorithm of LTS models and a compositional analysis technique of safety properties.

Keywords: safety property analysis, embedded system, LTS, slice model.

1 Introduction

Recently, cooperating embedded systems via internets have been widely used in our lives. The main task of embedded software is to engage the physical world, interacting directly with sensors and actuators in distributed processing nodes. Since even a simple failure of software may lead to severe consequences, safety properties of embedded software should be checked before delivery.

Various static analysis techniques have been proposed for verifying properties of distributed systems, which include model checking [1], inequality-necessary conditions analysis [2], data flow analysis [3,4], explicit state enumeration [5,6,7,8], and compositional reachability analysis[9, 10]. Among these analysis techniques, our approach focuses on compositional reachability analysis techniques, especially based on property automata [10] due to its scalability.

In this paper, we propose an efficient approach to verifying safety properties of cooperating embedded systems using Labeled Transition Systems (LTS). We introduce a slice model concept for easily checking behavioral equivalence between a system model and safety properties. After LTS models and safety properties are transformed into slice models, respectively, safety analysis for each property is performed in the incremental and iterative manner.

The remainder of the paper is organized as follows. Related works for LTS modeling and compositional analysis techniques are described in Section 2. Section 3

R. Obermaisser et al. (Eds.): SEUS 2007, LNCS 4761, pp. 114–124, 2007.

provides a description technique of system behaviors and safety properties by LTS. Section 4 presents a slice model concept and an algorithm for checking equivalence between two LTS models using slice models. In Section 5, a compositional safety analysis technique and its procedure are described. Section 6 evaluates the generated state space in the compositional safety analysis. Conclusion and future work appear in Section 7.

2 Related Works

LTS computation model has been widely used for specifying and analyzing distributed systems. To perform analysis based on LTS, it is necessary to construct the whole behavior model from the specification of the primitive processes. The whole behavior of the system can be described by the composite LTS which is constructed by composing the LTS1, LTS2, …, and LTSn of its constituent processes. This approach is generally known as reachability analysis. A major problem with reachability analysis is that the search space involved can grow exponentially with the increase in the number of concurrent processes.

To cope with this problem reduction techniques have been proposed by reducing the search space. These reduction techniques can be categorized into two classes; reduction by partial ordering and reduction by compositional minimization. In the reduction techniques by partial ordering, the search space is reduced by excluding the paths formed by the interleaving of the same set of transitions [6]. In techniques by compositional minimization, also known as compositional reachability analysis, the search space is reduced by compositionally constructing the composite LTS where globally observable actions are abstracted out [9,11,12,13].

We will adopt and extend the compositional reachability analysis since it is amenable to automation and can reflect the architecture of distributed software. In the compositional safety analysis method [10], safety properties are described by state machines, called a property automata, which is augmented with a special undefined state (π). A property automata is automatically transformed to its corresponding image property automata by adding the π state for capturing potential violation of safety properties. For example, we want to check a safety property which an event 'on' should be followed by event 'off' in all cases. Fig. 1 (b) and (c) show examples of a property automata and its image property automata, respectively.

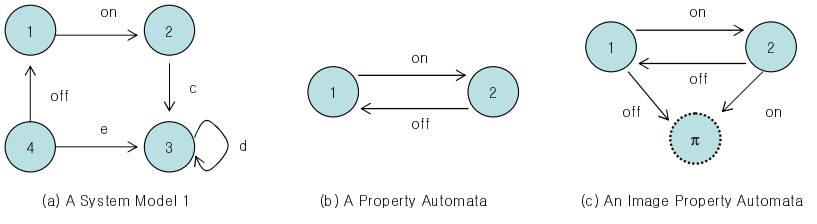

Fig. 1. Examples of compositional safety analysis

Fig. 1 (a) shows a simplified system model, whose main behaviors include on → c → d*. In the example system, behaviors of the system do not have the safety property. However, the violations of the safety property in the model are not detected by the image property automata. For rigorously checking safety properties, the equivalence checking between safety properties and the system model should be enforced.

3 Modeling System Behaviors and Properties

Suppose that we have a gas oven that can be remote-controlled at home or outside using mobile devices. This remote control system may be useful for turning off the gas oven when we forgot to turn it off at going outside or when we want to control the oven remotely at home. However, it is unsafe to control a gas oven remotely since we can not check its status such as gas leakage and inflammable materials on it. Therefore, for safety, we need some complementary devices such as a flame detection sensor, which can be monitoring the status of the gas oven. Fig. 2 shows the overall structure of the gas oven that can be remote-controlled. Now, is the gas oven system safe ?

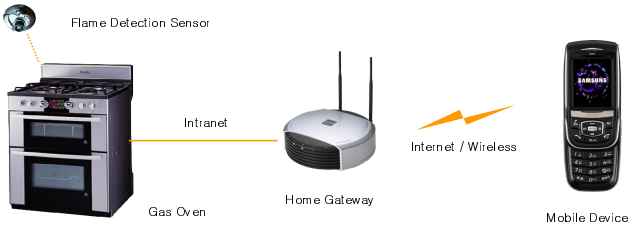

Fig. 2. An example of remote-controlled gas oven system

Fig. 3 represents a block diagram of the remote-controlled gas oven system. For simplicity, we abstractly describe only core components. The gas oven system is composed of a gas oven controller, a valve controller, a flame sensor, a communication media, and mobile devices.

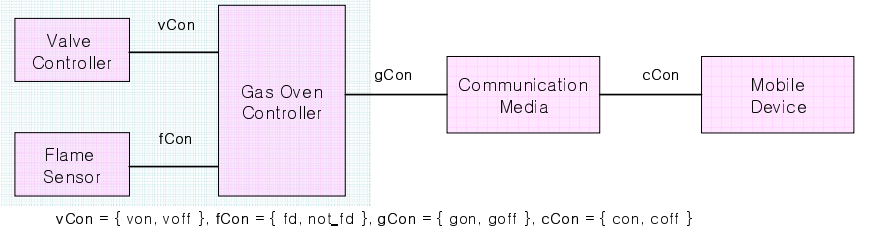

Fig. 3. A block diagram of the remote-controlled gas oven system

Each component of the block diagram is described by LTS. Fig. 4 shows the LTS models of the remote-controlled gas oven system. Communicating channels between components such as vCon and cCon are described by shared labels. In a LTS, all the states are considered as accepting states. The parallel composition of two LTS models, denoted by P ‖ Q, models the synchronized behavior of shared labels. Local events behave independently while the shared labels should be synchronized.

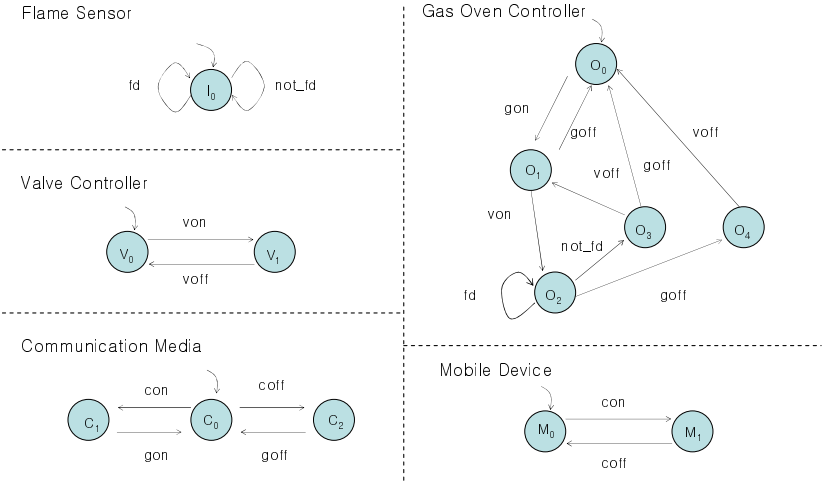

Fig. 4. The LTS models of the gas oven system

Safety properties can be represented by a sequence of events or be related with system states. And they can be described in positive form or negative form. In this paper, we support a state-based property and an event-based property in both the descriptions by extending property automata description technique [10]. Safety properties are also represented by LTS. But, a safety property model has several accepting states not all accepting ones. Followings are two types of safety properties: a state-based one and an event-based one.

- State-based safety property: safety properties are described based on state variables.
  Safety Property 1 (SP1): When the valve controller component is in the "V_1" state, the event "con" must not be occurred (!(State(V_1) → con))
- Event-based safety property: safety properties are described as a sequence of events.
  Safety Property 2 (SP2): After a gas valve is opened, it should be closed (von → voff).

Fig. 5 represents the safety property models of SP1 and SP2. In the figure, double circled states means the accepting states.

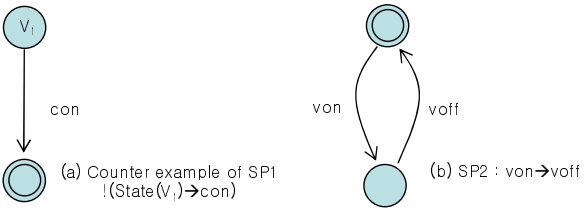

Fig. 5. Examples of describing the safety properties

Analysis of safety properties is performed in two ways. Negative safety properties can be checked whether the corresponding positive behaviors of the negative safety property can be occurred in the system model or not. In the case of positive safety properties, the behavior of a safety property should be always satisfied in the system model. Therefore, the satisfaction of the safety property can be checked whether its abstracted system behaviors are equivalent to behaviors of the property.

4 Slicing System Model Based on State Variables

It is not easy to compare two large LTS models. In order to simply compare the structures of LTS models, we slice the LTS models by restructuring them in the perspective of each state variable for effectively checking safety properties.

Definition 1. *Slice models of LTS*
A slice model of LTS is an LTS model that has only two boolean system states related and their related transitions, which has only four types of transitions (00, 01, 10, 11).

Fig. 6 shows an example of representing a slice model in a graphical and a tabular form, which represents the valve controller component shown in Fig. 4.

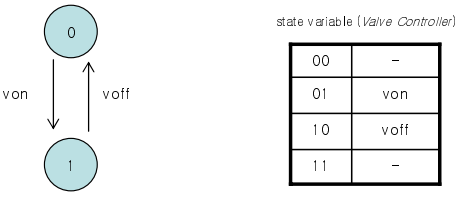

Fig. 6. Graphical and tabular representations of the slice models

For transforming an LTS model into the slice models, the states in the LTS model are represented by state variables. For each state variable, one or more slice models can be generated. If a state variable has several enumerated values, it is represented by several slice models. Followings show the steps for transforming a LTS model into the slice models.

Step 1: If there are the same labels in a LTS model, rename all the same transitions for differentiating all the transitions. For example, the 'voff' transition that appears severally in the gas oven controller component is renamed into $voff_1$, $voff_2$, and $voff_3$.

Step 2: For each transition in the LTS model, record the transition label in the each pattern of changes for each state variable in the slice models. As shown in Fig. 7, the 'con' transition from the state O_0 to the state O_1 is transformed into the '10' transition of S_0, the '01' transitions of S_1, respectively.

Fig. 8 shows the slice models of the communication media component. There are two state variables: GON(command on) and GOFF(command off). The equivalence of an original LTS model and the composition of the slice models can be easily checked. Since we assume that all the transitions in the LTS model are different, there is one-to-one and onto mapping between two transition sets. The transition information between the corresponding transitions is equivalent since the transition rules preserve the transition information. The memory space for representing slice models in the tabular form is equivalent to the original LTS representation.

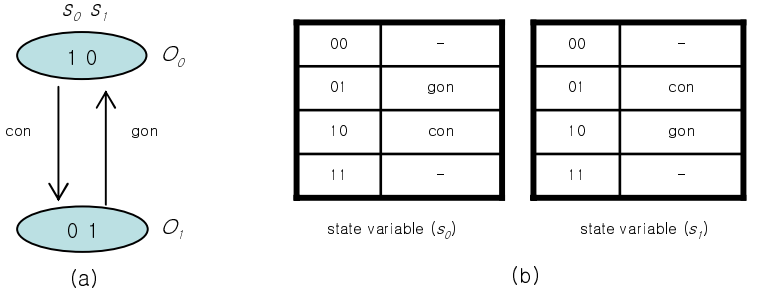

Fig. 7. An example for transformation rules

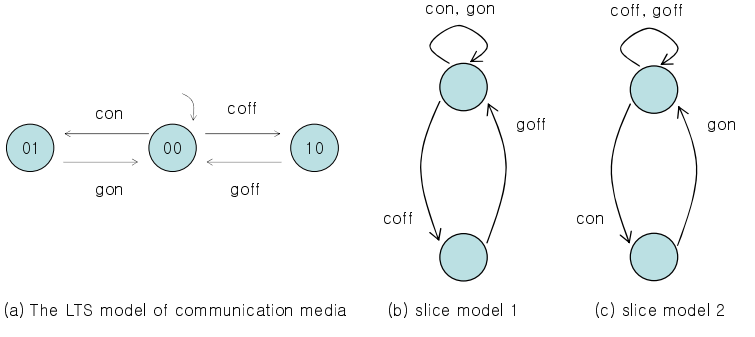

Fig. 8. Two slice models of the communication media component

Fig. 9 shows simple reduction rules for slice models. The event which is always occurred at any state can be reducible since it has no effects in enabling the other events.

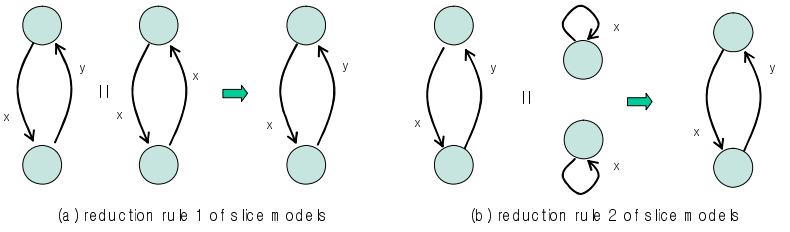

Fig. 9. Reduction rules for the slice models

Checking equivalence of two finite state machines is generally not easy since it is difficult to find the corresponding parts between different models. As shown in Fig. 10 (a) and (b), two finite state machines have the same equivalent behavior. But, their structures are different. In the observation equivalence [15], the behavioral equivalence of two systems is checked by composing the corresponding states. But, this approach needs an additional space for recording the corresponding states information. In the slice models, behavior equivalence is checked by comparing each slice model in pair-wise manner. Fig. 10 (c) and (d) show the slice models of LTS2 model and its reduced model by sequentially applying reduction rules. We can easily find out that the transformed slice models are the same.

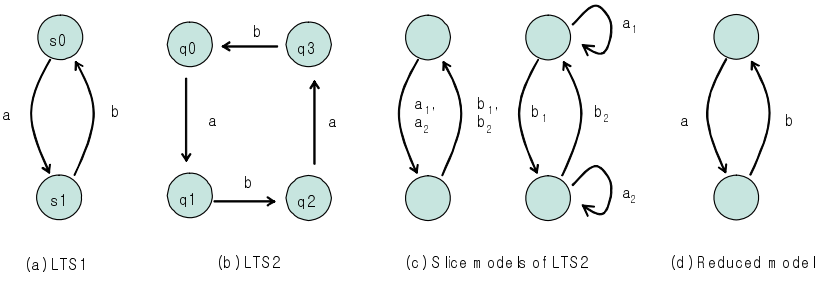

Fig. 10. Checking equivalence of two LTS models

5 Compositional Verification of Safety Properties

For effective analysis, it is important to minimize the state space of a system model by localizing and reducing features unrelated to the safety property. During making an reduced model by the compositional approach, local transitions are abstracted by the λ elimination rules of transformations from the λ–acceptor to the λ–free machine [14]. Fig. 11 shows the overall procedure of our algorithm. In the start of analysis

procedure, the system model and the safety property are composed since we need the same reference points between two models for easily finding corresponding ones. During reduction procedure, the state variables and transitions of the property model are preserved.

Safety properties are categorized into a positive form and a negative form. Safety analysis is differently performed according to its form. Followings are overall explanation of two safety analysis approaches.

- Negative safety property: A safety property in the negative form describes that a situation should not be occurred. For checking these properties, we check whether the reversed positive situation is occurred in the system model or not. If the situation occurs, the property is not satisfied.
- Positive safety property: A safety property in the positive form means that the property should be always satisfied in the system model. In this case, we check the equivalence of the property model and the abstracted system model against the property model.

Main analysis procedure is performed on the slice models. Therefore, the reduced system model and the property model are transformed to the slice models. Inclusion of two models is decided by checking whether each slice model of the property model is equivalent to the corresponding slice model of the system model. Equivalence of two models is checked by the equivalence of all the corresponding slice models.

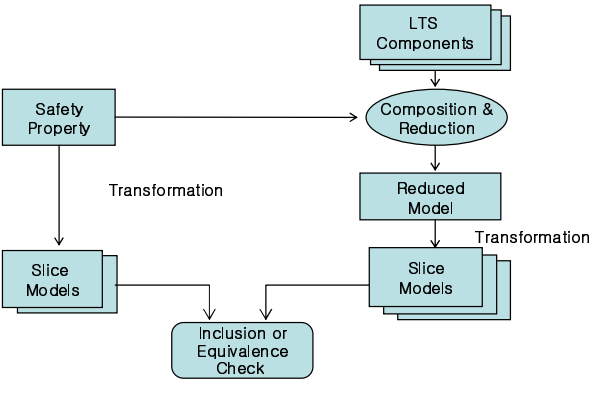

Fig. 11. Safety analysis procedure of LTS models

Fig. 12 shows the analysis steps of the safety property (SP2) using the compositional analysis technique. Fig. 12 (a) shows the abstract model of (communication media || mobile device), called C1. Fig. 12 (b) represents the composed model of C1 and the gas oven controller component. In Fig. 12 (b), local transitions such as gon and goff are transformed into the λ transition and eliminated by the λ–elimination rules [14] such as the λ–loop elimination and the λ–transition reduction ($q_0 =\lambda=> q_1 -s-> q_1 \rightarrow q_0 -s-> q_1$) to become the model shown in Fig. 12 (c). Through several composition and reduction steps, the final composed model C4 is

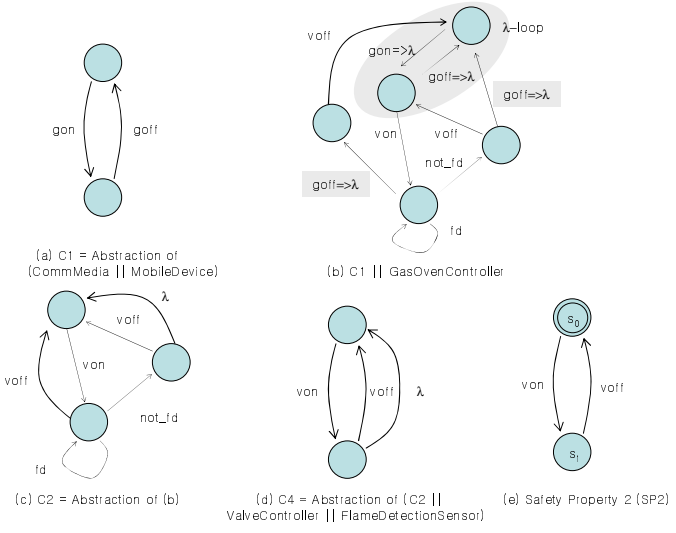

Fig. 12. Analysis steps of the safety property SP2

generated as shown in Fig. 12 (d). Finally, we compare the final model and the safety property model after transforming into the slice models. In this example, we can easily find their differences of two models. In the consequence of analysis, we conclude that the safety property 2 is not satisfied in the system behavior due to the modeling error in the gas oven controller.

6 State Space Evaluation of Slice Model Approach

In this section, we evaluate the state space for the slice models and the compositional safety analysis. At first, we calculate the state space of the slice models. Let n and m be the number of the states and transitions of a system, respectively. Since the number of the slice models is dependent on the number of system states, it is calculated by $\log_2(n)$. For each slice model, there can be m transitions in 00, 01, 10, and 11 slots at worst cases. Therefore, the slice model approach needs the state space of $\log_2(n)$ * m * 2 (bits for identifying 4 transition slots). This number is the same to that of the FSM approach in which, for each transition, the source and destination states information ($\log_2(n)$) should be recorded.

Next, we consider the state space for performing equivalence checking based on the slice models. In the observational equivalence approach, additional state space (the same or more size of the original model) is needed since the mapping information between corresponding states in two models should be recorded, while the slice model approach needs no additional space due to pair-wise comparison of each slice model.

Table 1 shows the generated state spaces for checking the safety property SP2. As shown in Table 1, the compositional approach is more efficient than the FSM

approach. Our compositional safety analysis method can fully utilize these merits of the compositional approach.

Table 1. The generated analysis spaces for checking the safety property SP2

Approaches Composed models	State diagrams (number of states and transitions)	Compositional approach		
		Original Models	Reduced Models	Reduction Rates (%)
C1 = S1 ‖ S2	4 (4)	3 (4)	3 (2)	0.0 (50.0)
C2 = C1 ‖ S3	10 (19)	5 (8)	3 (6)	40.0 (25.0)
C3 = C2 ‖ S4	10 (19)	3 (6)	2 (3)	33.3 (50.0)
C4 = C3 ‖ S5	14 (23)	2 (3)	2 (3)	0.0 (0.0)
Total	38 (65)	13 (21)	10 (14)	23.1 (33.3)

Legends: S1: Communication Media, S2: Mobile Device, S3: Gas Oven Controller, S4: Flame Detection Sensor, S5: Valve Controller.

7 Conclusion and Future Work

Safety issues are very important in the embedded system literature. In this paper, cooperating embedded systems such as the remote-controlled embedded system are described and analyzed by LTS. For convenient and effective compositional analysis of safety properties, we provide a slicing method of the system state space based on the system property, which is obtained by restructuring the LTS model. Based on the slice models, we provided an equivalence algorithm of LTS models and a compositional analysis technique of safety properties.

Currently, we are developing a modeling and analysis tool that helps to describe LTS models and to automatically partition a LTS model into the slice models. In future work, we will add timing concepts in our analysis approach.

Acknowledgments. This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by IITA (Institute of Information Technology assessment).

References

1. McMillan, K.L.: Symbolic model checking. Kluwer Academic Publishers, Dordrecht (1993)
2. Avrunin, G.S., et al.: Automated analysis of concurrent systems with the constrained expression toolset. IEEE trans. software engineering 17(11), 1204–1222 (1991)
3. Cheung, S.C., Kramer, J.: Tractable dataflow analysis for distributed systems. IEEE trans. software engineering 20(8), 579–593

4. Dwyer, M.B., Clarker, L.A.: Data flow analysis for verifying properties of concurrent programs. In: Proc. of the 2nd ACM SIGSOFT Symposium on the foundation of software engineering, pp. 62–75. ACM Press, New York (1994)
5. Cheung, S.C., Kramer, J.: Context constraints for compositional reachability analysis. ACM trans. software engineering and methodology 5(4), 334–377 (1996)
6. Godefroid, P., Wolper, P.: Using partial orders for the efficient verification of deadlock freedom and safety properties. In: Proc. of the 3rd international conference on computer aided verification (1991)
7. Long, D., Clarke, L.: Task interaction graphs for concurrency analysis. In: Proc. of the 11th ICSE, pp. 44–52 (1989)
8. Valmari, A., et al.: Putting advanced reachability analysis techniques together: The 'ARA' tool. In: Larsen, P.G., Woodcock, J.C.P. (eds.) FME 1993. LNCS, vol. 670, pp. 597–616. Springer, Heidelberg (1993)
9. Yeh, W.J., Young, M.: Compositional reachability analysis using process algebra. In: Proc. of ACM SIGSOFT, pp. 49–59 (1991)
10. Cheung, S.C., Kramer, J.: Checking Safety Properties using Compositional Reachability Analysis. In: ACM TOSEM, pp. 49–78 (1999)
11. Malhotra, J., et al.: A tool for hierarchical design and simulation of concurrent systems. In: Proc. of the BCS-FACS workshop on specification and verification of concurrent systems, pp. 140–152 (1988)
12. Sabnani, K.K., et al.: An algorithmic procedure for checking safety properties of protocols. IEEE trans. communication 37(9), 940–948 (1989)
13. Tai, K.C., Koppol, P.V.: An incremental approach to reachability analysis of distributed programs. In: Proc. of the 7th international workshop on software specification and design, pp. 141–150 (1993)
14. Denning, P.J., et al.: Machines, Languages, and Computation. Prentice-Hall, Englewood Cliffs (1978)
15. Milber, R.: Communication and Concurrency. Prentice Hall, Englewood Cliffs (1989)