Enhanced Compositional Safety Analysis for Distributed Embedded Systems using LTS Equivalence

HUI GUO, YOUNGSUL SHIN, WOO JIN LEE
School of Electrical Engineering and Computer Science
Kyungpook National University
1370 Sangyeok-dong, Buk-gu, Daegu
SOUTH KOREA

Abstract: - Real-time systems such as aeronautic systems, medical systems, and nuclear power plant systems are generally operated in a standalone mode. In the home network and ubiquitous computing systems, integrated services related with several embedded systems are focused, which is called distributed embedded systems. Safety issues of distributed embedded systems are very important since they are closely related to our living. In this research, distributed embedded systems and its safety properties are described by Labeled Transition Systems (LTS). For efficiently checking safety issues, we enhance the existing compositional safety analysis technique [10] using LTS equivalence concept.

Key-Words: - Safety analysis, embedded systems, compositional analysis, LTS equivalence

1 Introduction

Distributed embedded software has been widely used in our lives. The main task of distributed 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 catastrophic consequences, distributed embedded software must be extremely reliable. Safety issues of these systems are very important.

The architecture of distributed embedded system can be described by a collection of primitive processes, which communicate with each other in order to provide the global behavior of the system. Behavior of a primitive process can be described by a state machine whose transitions are labeled by the actions that the process can perform. Specifically speaking, labeled transition system (LTS) is used to specify the behavior of each primitive process. LTS is often used to model the behavior of a synchronous communicating process in distributed software.

Various static analysis techniques have been proposed for verifying properties of distributed systems. These 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]. We adopt and extend Cheung's compositional safety analysis technique. In Section 2, problems of current approaches are discussed.

In this paper, LTS is also adopted to specify a safety property. We propose an efficient approach to specifying and verifying safety properties of distributed embedded systems. An LTS component is used to specify the behavior for each component which comprises a system. The global behavior of the system is defined by the composite LTS by composing LTS models of the constituent components. A typical problem with the generation of the global behavior of a system is that the analysis complexity exponentially grows as the number of the components is increased. To cope with such problem, we adapt the compositional reachability analysis technique with reducing unrelated local transitions. At first, LTS models are composed and reduced in the perspective of a given property model. We provide an equivalence checking algorithm and a compositional safety analysis based on LTS equivalence concept.

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 provides system and safety property description techniques. In Section 4, we describe an algorithm for checking equivalence and inclusion between two LTS models. In Section 5, compositional safety analysis

technique and procedure are described. In Section 6, we perform experiments for gas oven systems with several burners. Conclusion and future work appear in Section 7.

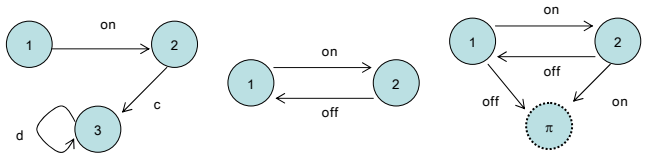
2 Related Works

LTS computation model has been widely used for specifying and analyzing distributed systems [11,12,13,14,15]. To perform analysis based on LTS, it is necessary to construct the whole behavior model from the specification of the primitive processes. For example, consider a system consists of n processes whose behavior are specified by LTS_1, LTS_2, …, and LTS_n. The whole behavior of the system can be described by the composite LTS which is constructed by composing the LTS_1, LTS_2, …, and LTS_n 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 [6,16]. 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 [9,17,18,19].

We will adopt and enhance the compositional reachability analysis since it is amenable to automation and can reflect the architecture of distributed software. In compositional safety analysis method [10], safety properties are described by state machines, called 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 property automata and its image property automata, respectively.

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. We will provide an equivalence checking method and a compositional safety analysis technique based on LTS equivalence concept. Detailed analysis methods and procedures are described in Section 5.



(a) A System Model 1 (b) A Property Automata (c) An Image Property Automata

Fig.1 Examples of compositional safety analysis

3 Modeling System Behaviors and Safety 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 near it. Therefore, for safety, we need some complementary devices such as a flame detection sensor, which can monitor 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 ?

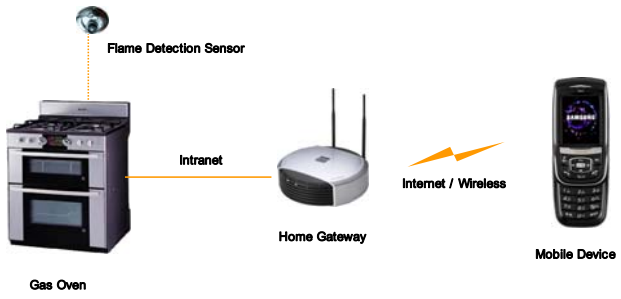


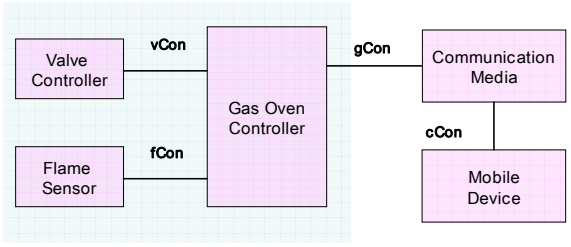
Fig.2 An example of remote-controlled gas oven system

Definition 1 *A labeled Transition System (LTS)*
An LTS is denoted by a 4-tuple (Q, Σ, δ, q_0) where

- Q is a set of states,
- Σ corresponds to the set of event labels of the LTS, which represents internal events or communicating labels,
- $\delta \subseteq Q \times \Sigma \times Q$, denotes a transition relation that maps from a state and an event onto another state,
- q_0 is an initial state.

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. Local events behave independently while the shared labels should be synchronized.

Fig. 3 represents the block diagram of the remote-controlled gas oven system. For simplicity, we describe only core components in abstract form. The gas oven system is composed of a gas oven controller, a valve controller, and a flame detection sensor. Fig. 4 (a) through (e) show a LTS model of the remote-controlled gas oven system. Each component of the system is described by LTS.



vCon = { von, voff }, **fCon =** { fd, not_fd }, **gCon =** { gon, goff }, **cCon =** { con, coff }

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

Safety properties should be always satisfied in a system model. When they are not satisfied, there may be catastrophic consequences such as lost of lives and money and threat to environment. 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 state-based property and event-based property in both the descriptions by extending property automata description technique [10]. Safety properties are also represented by LTS. But, there is a difference of LTS system modeling and property modeling. That is, a safety property has not all accepting states but some accepting states. Safety properties are described as a sequence of events. For example, after gas valves are opened, they should be closed (SP1 : von → voff).

Analysis of safety properties is performed in two ways. Negative safety properties can be checked whether the corresponding positive behaviors of a 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 a safety property can be checked whether abstracted system behaviors are equivalent to its behaviors. Fig. 4 (f) represents the safety property models of SP1. In the figure, double circled states means the accepting states.

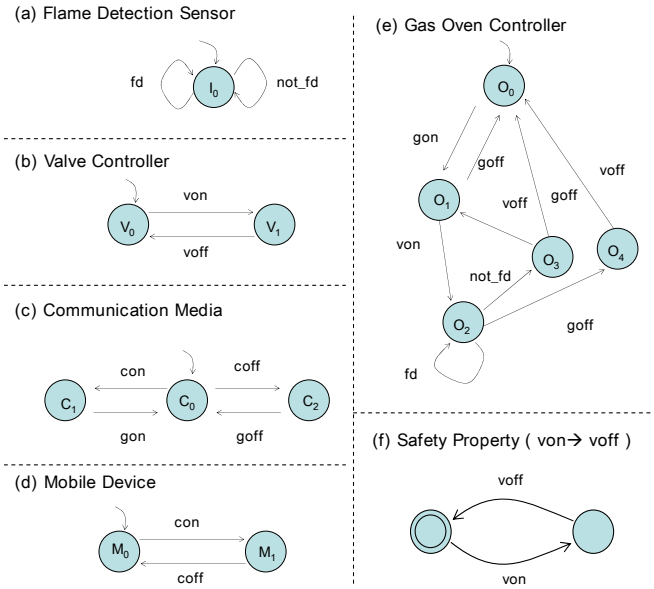


Fig.4 An LTS model of a gas oven system

4 Equivalence Checking of LTS models

In this section, we provide algorithms for checking observational behavior inclusion and equivalence between two LTS models, which is based on the weak bi-simulation concept of Milner [21]. Inclusion and equivalence of LTS models are checked by comparing the reachable trace sets of two models. Definition of reachable traces is given as follows.

Definition 2 *A Reachable Trace*
A reachable trace is composed of a sequence of consequent transitions which may be terminated or have an inner loop to a previous transition.

Since reachable trace can be terminated or make a loop, they are finite. And a system model has a finite set of reachable traces. Reachable traces can be generated by depth-first traversal algorithms. The set of reachable traces corresponding to the gas oven controller, as shown in Fig. 4(e), is represented as follows. Each

trace is described as a consequent pair of a state and a transition.

Tr(Gas Oven Controller) = { O_0 –con-> O_1 –coff-> O_0, O_0 –con-> O_1 –von-> O_2 –fd-> O_2, O_0 –con->O_1 –von-> O_2 -not_fd-> O_3 -voff-> O_1, O_0 –con->O_1 –von-> O_2 -not_fd-> O_3 -coff-> O_0, O_0 –con->O_1 –von-> O_2 -coff-> O_4 -voff-> O_0}

For checking inclusion of two model, ***tr***(A) ⊆ ***tr***(B), each reachable trace of model A should be appeared in the model B. Appearance of a reachable trace is easily checked by traversing the trace in the model B. Behavioral equivalence between A and B models can be checked by performing inclusion checking both of ***tr***(A) ⊆ ***tr***(B) and ***tr***(B) ⊆ ***tr***(A)**.**

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 safety property. During making a reduced model by compositional approach, local transitions except the referenced transitions in the safety property are abstracted by the λ–elimination rules of transformations from a $\tilde{\lambda}$–acceptor to a $\tilde{\lambda}$–free machine [20]. Fig. 5 shows the overall procedure of our algorithm. In the start of analysis procedure, system model and safety property are composed since we need the same reference points between two models for easily finding corresponding ones. During reduction procedure, state variables and transitions of the property model are preserved.

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

- Negative safety property: For checking these properties, we check whether the reversed positive situation is occurred in the abstracted system model against the safety property or not. If the situation occurs, the property is not satisfied. That is, we check ***tr***(¬SafetyProperty) ⊆ ***tr***(SystemModel ↑ α(¬SafetyProperty))
- Positive safety property: A safety property in positive form means that the property should be always satisfied in the system model. In this case, we check the equivalence of property model and abstracted system model against the property model. That is, equivalence of ***tr***(SystemModel ↑ α(SafetyProperty)) and ***tr***(SafetyProperty) is checked.

Checking inclusion and equivalence relations between the system model and its safety property model can be performed by generating and comparing their reachable trace sets as described in Section 4.

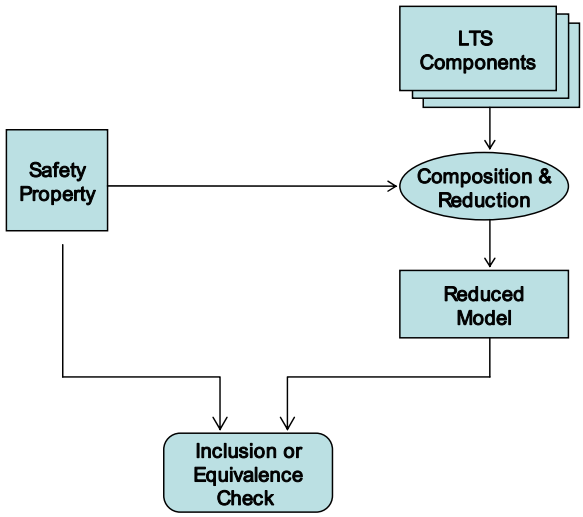


Fig. 5 Safety analysis procedure of LTS models

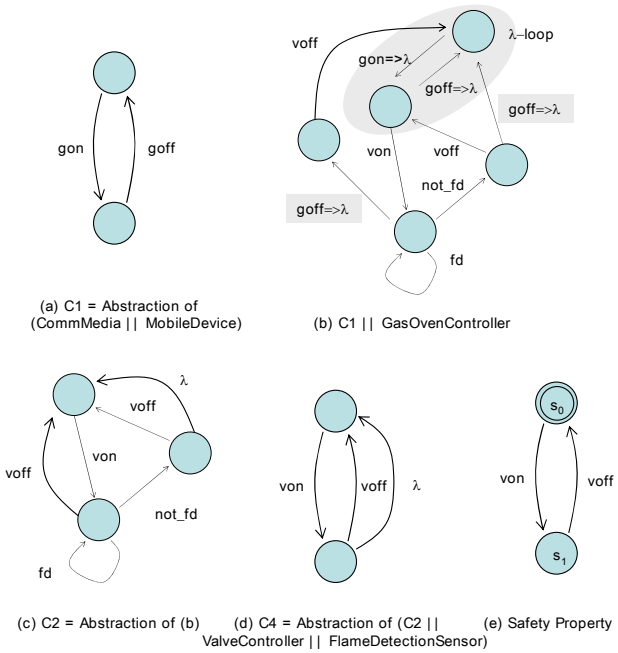


Fig. 6 Analysis steps of the safety property SP1

Fig. 6 shows the analysis steps of safety property (SP2) using compositional analysis technique. Fig. 6 (a) shows the abstract model of (communication media || mobile device), called C1. Fig. 6 (b) represents the composed model of C1 and gas oven

controller component. In Fig. 6 (b), local transitions such as gon and goff are transformed into λ transition and eliminated by λ–elimination rules [20] such as λ –loop elimination and λ–transition reduction (q0 =λ=> qt –s->q1 ➔ q0 –s-> q1) to become a model shown in Fig. 6 (c). Through several composition and reduction steps, as shown in Fig. 6 (d), the final composed model C4 is generated. Finally, we compare the final generated model and the safety property model by comparing reachable trace sets. As shown in Fig. 6 (d) and (e), two models have different reachable trace sets. In consequence of analysis, we conclude that safety property 1 is not satisfied in the system behavior.

6 Experimental Result

For checking applicability of our approach, we compared the numbers of generated states and transitions of both FSM approach and our approach. For scalability, we incrementally added the burner controller to the original model. Table 1 shows the comparative results. As show in Table 1, our approach is more applicable to the large-scale systems than existing FSM approach.

Table 1

# of Burner	# of Comp	FSM approach States(Trans)	Our approach (SP1)
1	5	76(180)	2(3)
2	8	4104(15836)	27(79)
3	11	102320(534848)	131(474)
4	14	-	542(2320)
5	17	-	2231(10809)
6	20	-	8101(44816)

7 Conclusion

Safety issues are very important in embedded system literature. In this paper, distributed embedded systems such as remote-controlled embedded system are described and analyzed by Labeled Transition Systems. We enhanced the existing compositional safety analysis technique using LTS equivalence concept with preserving merits of original compositional approach. In the future work, we will add the timing concepts in our analysis approach.

References:

[1] K. L. McMillan, *Symbolic model checking*, Kluwer Academic Publishers, 1993

[2] G. S. Avrunin, et al., "Automated analysis of concurrent systems with the constrained expression toolset," *IEEE trans. software engineering*, Vol. 17, No. 11, 1991, pp. 1204-1222

[3] S. C. Cheung, J. Kramer, "Tractable dataflow analysis for distributed systems," *IEEE trans. software engineering*, Vol. 20, No. 8, pp. 579-593

[4] M. B. Dwyer, L. A. Clarker, "Data flow analysis for verifying properties of concurrent programs," *Proc. of the 2nd ACM SIGSOFT Symposium on the foundation of software engineering*, 1994, pp. 62-75

[5] S. C. Cheung, J. Kramer, "Context constraints for compositional reachability analysis," *ACM trans. software engineering and methodology,* Vol. 5, No. 4, 1996, pp. 334-377

[6] P. Godefroid, P. Wolper, "Using partial orders for the efficient verification of deadlock freedom and safety properties," *Proc. of the 3rd international conference on computer aided verification*, 1991

[7] D. Long, L. Clarke, "Task interaction graphs for concurrency analysis," *Proc. of the 11th ICSE*, 1989, pp. 44-52

[8] Al Valmari, et al. "Putting advanced reachability analysis techniques together: The 'ARA' tool," *Proc. of the FME*, 1993, pp. 597-616

[9] W. J. Yeh, M. Young, "Compositional reachability analysis using process algebra," *Proc. of ACM SIGSOFT*, 1991, pp. 49-59

[10] S. C. Cheung, J. Kramer, "Checking Safety Properties using Compositional Reachability Analysis," *ACM TOSEM*, 1999, pp. 49-78

[11] S. Uchitel, et al., "Synthesis of behavioral models from scenarios," *IEEE trans. on software engineering*, Vol. 29, No. 2, 2003, pp. 99-115

[12] C. Damas, et al., "Generating annotated behavior models from end-user scenarios," IEEE trans. on software engineering, Vol. 31, No. 12, 2005, pp. 1056-1073

[13] H. Foster, et al., "Compatibility verification for Web Service Choreography," *Proc. of ICWS*, 2004

[14] J. Magee, et al., "Behavior analysis of software architectures," *Proc. of the 1st Working IFIP Conference on Software Architecture*, 1999

[15] J.H.Lee, et al., "Formal verification of protocol specified in LTS for railway signaling systems," *Computers in Railways*, 2004

[16] G. Holzmann, et al., "Coverage preserving reduction strategies for reachability analysis," *Proc. of the PSTV*, 1992, pp. 349-364

[17] J. Malhotra, et al., "A tool for hierarchical design and simulation of concurrent systems," *Proc. of the BCS-FACS workshop on specification and verification of concurrent systems*, 1988, pp. 140-152

[18] K. K. Sabnani, et al., "An algorithmic procedure for checking safety properties of protocols," *IEEE trans. communication,* Vol. 37, No. 9, 1989, pp. 940-948

[19] K. C. Tai, P. V. Koppol, "An incremental approach to reachability analysis of distributed programs," *Proc. of the 7th international workshop on software specification and design*, 1993, pp. 141-150

[20] P. J. Denning, et al., *Machines, Languages, and Computation*, Prentice-Hall, 1978

[21] Robin Milner, *Communication and Concurrency*, Prentice Hall, 1989