The Formal Transformation of AADL Based on Z-CoIA

Fugao Zhang(✉) and Zining Cao

College of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
luwayzhg@163.com, caozn@163.com

Abstract. The Architecture Analysis and Design Language (AADL) is a component-based semi-formal language. This paper proposes an expanded component-interaction automaton with Z language (Z-CoIA) based on the characteristics of AADL, introducing the formal specification Language Z into the component-interaction automata, then the formal transformation rules from AADL to the Z-CoIA is given, which is good for describing the data during system interaction and the attributes in state transitions and data constraints. Finally, a concrete example is shown.

Keywords: AADL · Component-interaction automata · Z language · Model transformation

1 Introduction

With the development of the AADL, the research about the formal verification of the AADL model has become a hot topic. Transforming from the AADL model into the expanded automata is one of the main directions. The literature [1] proposed a series of formal transformation rules converting the subset of the abstract AADL model into a timed automata, and developed the AADL verification tool based on the OSATE. The literature [2] transformed the AADL model into a generalized stochastic Petri net, and evaluated the reliability of the model based on stochastic Petri nets analysis technology. To specify some other aspects which are not limited to behavior and communication, the literature [8] translates AADL to timed abstract state machine for timing and resource aspects of a system. Some other researches such as literature [9] pays attention to the formal verification of distributed real-time systems, it transforms AADL model to another specification formalism like LNT language which is an input to CADP toolbox for formal analysis.

Some early automata like I/O automata and interface automata do not describe the interaction between components, which do not support the interface interaction between components naturally. The literature [3] proposed the component interaction automata, the advantages and disadvantages of component interaction automata and the verification method based on the temporal logic are also mentioned. The literature [4] mentioned a modeling method based on the time component interaction automata and introduced the definition, combination and verification algorithm of the time component interaction automata. The literature [5] introduced the qos constraints into the component interaction automata, and put forward a component model based on constraint interaction automata.

AADL is the industry standard language for embedded real-time systems, and is suitable for the specification of hierarchical components architecture. Embedded real-time systems with data constraint are computing system with time constraints and variable data constraints at the same time. Constraints between data variables are certainly included in the requirements of these systems, so in order to describe the constraints between data variables, we use the Z language to describe the states sets, state transition and properties in the model. Z language is a formal specification language based on the first order predicate logic and set theory, and is well suited for describing the states transitions and data constraints. The literature [6] combined Z language and interface automata, and put forward the definition of ZIA. ZIA is in a style of interface automata but its states and operations are described by Z language.

Moreover, AADL is semi-formal and cannot be verified by the formal methods directly, so this paper combines the Z language and the component interaction automata based on the characteristic of AADL. And then transforms the AADL model into the Z-CoIA model based on the formal transformation rules, so as to AADL can be verified by logic formula or any other formal methods in the future work.

2 Component Interaction Automata with Z Language

The formal specification language Z is based on the first-order predicate logic and set theory. Z language adopted strict mathematical theory, thus can produce concise, precise, unambiguous, and provable specifications.

The schema [7] is a kernel notation in Z language that to aid the structuring and modularization of the specification. A schema is constitute of variable declarations and predicates represent the limitation of these variables, and the schema has vertical and horizontal two forms.

Where S is the name of the schema, $D_1; \ldots; D_m$ are the declarations part and $P_1; \ldots; P_n$ are the predicates, and the horizontal form will be: $S \triangleq [D_1; \ldots; D_m | P_1; \ldots; P_n]$.

Furthermore, Z language use the identifier decorations to encode intended interpretations. A variable ending with "?" represents an input variable, and the "!" represents the output. The decoration "'" behind the variable represents the next state. With the decorations, Z language can depict the next state variables and the current state of the relationship between variables. Then the definition of component interaction automata with Z language is as follows, which can be abbreviated by Z-CoIA.

Definition 1. The component interaction automata with Z language can be depict as $M = (Q, Q_0, A, V, \Gamma, F^A, F^V, H)$, where,

(1) Q is a set of states that is finite and not empty, $Q_0 \subseteq Q$ represents the set of the initial states;
(2) A is a set of the actions in the automata, and include the input, output and internal actions respectively;

(3) V is a set of the variables in the automata, and include the input, output and internal variables respectively;
(4) $\Gamma \subseteq Q \times \sum \times Q$ is a set of transitions between states;
(5) F^A is a map, and maps any states in A to a operation schema in Z language, and specifically, the input action maps the input operation schema in Z language, the output action maps an output operation schema in Z language, and the internal maps operation schema in Z language;
(6) F^V is a map, which maps any states in V to a state schema in Z language;
(7) H is a tuple corresponding to a hierarchy of component names.

A component interaction automata with Z language M consists of the following elements, which can be represented by Fig. 1:

(1) $Q = \{q_0, q_1, q_2, q_3\}$;
(2) $Q_0 = \{q_0\}$;
(3) $A^I = \{a, c\}, A^O = \{d\}, A^H = \{b\}$;
(4) $V^I = \{x_1, x_2\}, V^O = \{y\}, V^H = \{z\}$;
(5) $\Gamma = \{(M, a, +), (M, b, M), (M, c, +), (M, d, -)\}$;
(6) $F^V(p_0) = S_0 \triangleq [z : N | z = 0]$,
$F^V(p_1) = S_1 \triangleq [x_1? : N | x_1 \in N; z = 0]$,
$F^V(p_2) = S_2 \triangleq [x_1 : N; z : N | z = x_1]$,
$F^V(p_3) = S_3 \triangleq [x_2? : N; y! : N | x_2 \in N; z = x_1]$,
(7) $F^A(a) = A_a \triangleq [x_1? : N | x_1 \in N]$,
$F^A(b) = A_b \triangleq [x_1? : N; z : N | z' = z + x_1?]$,
$F^A(c) = A_c \triangleq [x_2? : N | x_2 \in N]$,
$F^A(d) = A_d \triangleq [x_2 \in N; y! : N; z : N | y! = z * x_2]$;
(8) $H = \{M\}$;

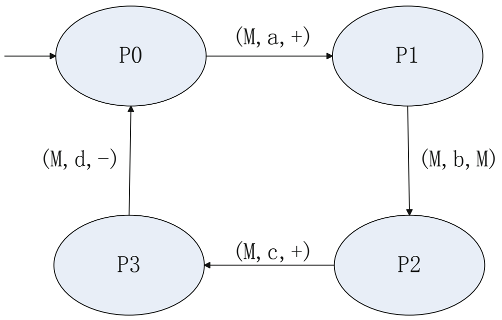

Fig. 1. A component interaction automata with Z language.

The compointernal actions of each component automata is disnent automata is hierarchical and composable if the joint with the action sets of all other component automata.

Definition 2. $S = (Q_C, Q_{C0}, A_C, V_C, \Gamma_C, F_C^A, F_C^V, H_C)$, the Combination of Component Interaction Automata with Z Language is combined by the sets of components where,

(1) $i \in N$ and the set of component name $\{(H_i)\}$ is disjoint of each other;
(2) Q_C, Q_{C0}, is the combined automata states set and the initial states; and the set of actions and variables of the combined automata $A_C = \cup_{i \in N} A, V_C = \cup_{i \in N} V$;
(3) $\Gamma_C \subseteq \{X \times A_C \times (X \cup \{\pm\})\}, X \in \{(H_i)\}$ is the set of transitions between states of the combined automata;
(4) F_C^A, F_C^V are maps of actions and states of the combined automata;
(5) $H_C = \{(H_i)\}, i \in N$ is the corresponding to a hierarchy of component names.

When combining this kind of automata, the question that whether the components are matched each other or not should be taken into account. When two components are input and output module with each other, the input and output actions can be matched for an internal action, marked as (X, τ, X). The input part is $M_1 = (Q_1, Q_{10}, A_1, V_1, \Gamma_1, F_1^A, F_1^V, H_1)$, $M_2 = (Q_2, Q_{20}, A_2, V_2, \Gamma_2, F_2^A, F_2^V, H_2)$. The output part is $S = (Q_C, Q_{C0}, A_C, V_C, \Gamma_C, F_C^A, F_C^V, H_C)$. The synchronous actions are combined into the internal action, and the others are interleaving. And there will be the combined set of states Q_C, the set of actions A_C and the set of transitions Γ_C, then the new states and actions can be remapped into the Z language, at last, the component name sets $H_C = (H_1, H_2)$, then the work of the combination of the Z component interaction automata is finished.

3 The Transformation of AADL to Z-CoIA

AADL is a modeling language that supports early and repeated analyses of a system's architecture with respect to performance-critical properties through an extendable notation, a unified framework, and precisely defined semantics. AADL subset includes the components of thread, process, process group, system and the behavior annex. In order to construct data constraint, we use Z schema to extend AADL based on its behavior annex. The formal transformation rules is given in Table 1.

The main transformation rules include: one basic component in AADL to one single Z-CoIA; the in-ports and out-ports in the feature of AADL component to the set of interaction in the Z-CoIA; the states set in the behavior annex to the sets of states in Z-CoIA; the transitions of states in AADL to the sets of states transitions in Z-CoIA; the guard of the states transitions to the input actions of the Z-CoIA; the actions in the behavior annex to the output actions of Z-CoIA; the transitions without the output

Table 1. The transformation rules of the AADL to Z-CoIA.

AADL model	Z-CoIA model
A single component	One component interaction automata
The features of the component	The ports of the Z-CoIA
The states in behavior annex	The sets of states in Z-CoIA
The transitions of the states	The sets of transitions in Z-CoIA
The action of transitions	The actions of Z-CoIA
The component consist of subcomponents	The combined Z-CoIA

actions to the internal actions in Z-CoIA. And the Table 1 is the formal transformation rules of the AADL to Z-CoIA.

4 The Example of AADL Model Transformation

We present a specific metro control system to show our method to transform the AADL to Z-CoIA. The architecture of metro control system can is shown in Fig. 2. It includes metro control system, the metro door control system and the platform door control system. Control system receives the command from the metro control system, and send command to control the open or closed of the metro door after data processing, and at the same time, send the status of the metro door to the metro control system.

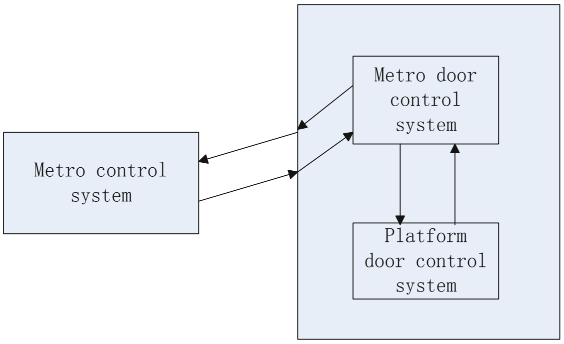

Fig. 2. The architecture of metro control system.

The entire system can be described by the system component metro_system in AADL like Table 2, and consist of two subcomponents, the metro_controller process component and door_system component. Whereby, the system component is consist of door_controller and door process components, and each process component consist of the corresponding thread components. Based on the transformation rules and the specific AADL model, we can transform the AADL model of metro control system into the Z-CoIA model.

The process component metro_controller can be transformed to the Z-CoIA M_1:

$$M_1 = (\{p_0, p_1\}, \{p_0\}, \{a, b\}, \{(M_1, a, +), (M_1, b, -)\}, F^A, F^V, (M_1)).$$

The process component door_controller can be transformed to the Z-CoIA M_2:

$$M_2 = (\{p_0, p_1, p_2, p_3\}, \{p_0\}, \{a, b, c, d\}, \{(M_2, a, +), (M_2, b, -), (M_2, c, +), (M_2, d, -)\}, F^A, F^V, (M_2)).$$

Table 2. The AADL model of metro control system.

```
process door
        features
                x: in data port;
end door;

process door_controller
        features
                current_data: in data port;
                new_feature: out data port;
                new_feature2: out data port;
end door_controller;

process metro_controller
        features
                current_data: out data port;
                new_feature: in data port;
end metro_controller;

system implementation door_system.impl
        subcomponents
                door:process door;
                door_controller: process door_controller;
end door_system.impl;

system implementation metro_system.impl
        subcomponents
                door:system door_part;
                metro_controller: process metro_controller;
end metro_system.impl;
```

The process component door can be transformed to the Z-CoIA M_3:

$$M_3 = (\{p_0, p_1\}, \{p_0\}, \{a, b\}, \{(M_3, a, +), (M_3, b, -)\}, F^A, F^V, (M_3)).$$

The system component can transformed by the combined of the Z-CoIA. The door_system component is combined by the door_controller and door process components, and can be described by M_4:

$$M_4 = (\{p_0, p_1\}, \{p_0\}, \{a, b\}, \{(M_4, a, +), (M_4, b, -)\}, F^A, F^V, (M_4)).$$

5 Conclusion

Based on the modeling language AADL, this paper presents an expended component interaction automata with Z language named Z-CoIA. So as to formally describe large amount of data and data constraints in the system and expend the advantages of Z language with automata, and take advantages of the two formal methods. It can be used for the formal verification of the AADL in the future research work. Then the formal transformation rules from AADL to Z-CoIA are described formally, and apply our method to a specific example of metro control system.

AADL is the industry standard language for embedded real-time systems. When an embedded real-time systems with data constraint is a computing system, we can combine the Z language, at the same time, we can also generate the time component interaction automata to describe the time constraints in the system, which both with time constraints and variable data constraints. Similarly, when combine the component automata, we can expend the process algebraic language to add the description of the relationship between the components. In addition, we can also develop the tool to realize the automatic transformation from AADL to Z-CoIA as a plug of the AADL tool OSATE in the future work.

Acknowledgments. This paper was supported by the Aviation Science Fund of China under Grant No.20150652008, the National Basic Research Program of China (973 Program) under Grant No.2014CB744903 and the National Natural Science Foundation of China under Grant No.61572253.

References

1. Yang, Z., Hu, K., Ma, D., et al.: From AADL to timed abstract state machines: a verified model transformation. J. Syst. Softw. **93**(2), 42–68 (2014)
2. Wu, Y.: The study of formal verification of embedded software based on AADL. Shanxi Normal University (2014)
3. Zimmerova, B., Vařeková, P., Beneš, N., Černá, I., Brim, L., Sochor, J.: Component-interaction automata approach (CoIn). In: Rausch, A., Reussner, R., Mirandola, R., Plášil, F. (eds.) The Common Component Modeling Example. LNCS, vol. 5153, pp. 146–176. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85289-6_7
4. Yangli, J., Zhenling, Z.: Formal model of component real-time interaction behavior based on automata theory. J. Comput. Sci. **37**(9), 151–156 (2010)
5. Yuyu, Z.: The study of component behavior consistency based on constraints interaction automata. Harbin Engineering University (2012)
6. Zining, C.: Temporal logics and model checking algorithms for ZIAs. In: International Conference on Software Engineering and Data Mining, New York, pp. 57–62. IEEE Press (2010)
7. Jonathan, B., Bowen, A.J.: Formal specification and documentation using Z: a case study approach (2003)
8. Hu, K., et al.: Exploring AADL verification tool through model transformation. J. Syst. Archit. **61**(3–4), 141–156 (2015)
9. Mkaouar, H., Zalila, B., Hugues, J., Jmaiel, M.: From AADL model to LNT specification. In: de la Puente, J.A., Vardanega, T. (eds.) Ada-Europe 2015. LNCS, vol. 9111, pp. 146–161. Springer, Cham (2015). doi:10.1007/978-3-319-19584-1_10