基于 Z-AADL 模型的形式化转换

高 正,曹子宁
(南京航空航天大学 计算机科学与技术学院,江苏 南京 210016)

摘 要:结构分析与设计语言(Architecture Analysis and Design Language,AADL)是针对嵌入式实时系统领域中的软件开发复杂度问题而提出的一种基于模型驱动开发的体系结构建模语言,可用于设计和分析一些安全关键嵌入式实时系统的软硬件体系结构。但是 AADL 严格来说只是一种半形式化的建模语言,虽然也描述了模型中构件的属性,但是对模型的非功能属性并没有明确的形式化描述,因此对 AADL 系统模型的非功能属性进行形式化验证,对于保证系统的正确性和可靠性具有重要意义。针对 AADL 模型中对非功能属性及数据性质等描述的不足之处,将其与形式规格说明语言 Z 语言相结合,提出了一种新的描述能力更加强大的形式规范语言—Z-AADL,并且定义了将 Z-AADL 模型转换为形式化模型—ZIA 模型的转换规则,以实现 Z-AADL 模型到 ZIA 形式化模型的转换。并通过一个转换实例进行说明。

关键词:Z-AADL;ZIA;CT-ZIA;模型转换
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2017)03-0023-06
doi:10.3969/j.issn.1673-629X.2017.03.005

Formal Transformation Basedon Z-AADL Model

GAO Zheng,CAO Zi-ning
(School of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing 210016,China)

Abstract:Architecture Analysis and Design Language (AADL) is a kind of system structure modeling language based on model driven development in view of the problem of software development complexity in the field of embedded real-time system,which can be used to design and analyze the software and hardware architecture of some security critical embedded systems. But AADL is a semi-formal modeling language seriously. Although it also describes the properties of the components,the non-functional properties of the model are not clearly defined. Therefore,the formal verification of the non-functional properties of the AADL system model is of great significance to guarantee the correctness and reliability of the system. Aiming at the shortage of non-functional and data property of AADL model,combining the AADL specification with the Z language,a new specification—Z-AADL is put forward. Then the transformation rules between Z-AADL and ZIA is given,which define that how the Z-AADL model can be transformed to the ZIA model. An example is given to illustrate the transformation.

Key words:Z-AADL;ZIA;CT-ZIA;model transformation

0 引 言

嵌入式实时系统在航空航天、汽车控制、机器人等安全关键系统领域得到愈来愈广泛的应用。但由于计算精度、实时响应、资源限制等要求的提高,系统开始变得越来越复杂,因此采用更加严格和可靠的设计与实现规范可以实现具有高可靠性、高质量的复杂嵌入式实时系统[1-2]。针对这一越来越紧迫的问题,2004 年,美国汽车工程师协会(SAE)提出了嵌入式实时系统体系结构分析与设计标准-AADL[3]。它以传统的建模语言 MetaH、UML[4] 为基础,能够更加精确地设计与分析嵌入式实时系统的软硬件体系结构及功能属性等。但是,AADL 仍然存在一些问题,有待进一步的研究与完善。AADL 采用自动机的形式对线程、进程等构件的执行状态和动作进行了语义描述,但这种语义并不是严格的形式化语义描述,而且对于状态及状态转换之间的数据约束关系也没有形式化语义描述,因此难以直接进行一致性检验。结合国内外对 AADL 建模语言多年的研究现状,现在对 AADL 形式语义的

收稿日期:2015-12-03 **修回日期**:2016-03-17 **网络出版时间**:2017-01-04
基金项目:国家"973"重点基础研究发展计划项目(2014CB744900)
作者简介:高 正(1991-),男,硕士研究生,研究方向为形式化方法;曹子宁,博导,研究方向为形式化方法、人工智能。

研究多采用形式化转换的方式,主要包括两种:

(1)用某种具有精确语义的形式化语言来定义 AADL 的语义,然后通过这种语义来进行形式化的转换;

(2)对目标系统进行 AADL 建模,然后将这种模型直接转换为另一种形式化模型。

Z 语言是一种严格的形式化规范描述语言,可以对状态及状态转换之间的数据约束等进行严格的形式化描述。因此文中提出在 AADL 的基础上扩展 Z,形成 Z-AADL 规范,从而可以在不考虑具体设计细节的情况下,从更高的抽象级别用 Z-AADL 刻画 AADL 模型。

1 AADL 的 Z 扩充

1.1 AADL 简介

AADL 是一种功能十分强大的分析与设计语言,越来越广泛地应用于航空电子、飞行控制等航空航天领域的嵌入式实时系统的体系结构设计与分析中。它主要包括三大类组件,即:软件组件、执行平台组件、系统组件。并且,在系统的建模过程中,通过包来组织所定义的组件类型和组件实现等内容,这是一种类似于高级开发语言的组织形式。其中,软件组件主要包括进程组件、线程组件、线程组组件、数据组件、子程序组件、子程序组组件等。软件组件的主要功能是用来表示系统的进程、线程组、线程等应用程序所具有的任务架构,可用于表示模型所属的软件体系组成部分,例如数据组件中的数据类型、程序中的可执行代码等。执行平台组件主要包括处理器组件、存储器组件、总线组件、外设组件等,主要功能是对系统的软硬件体系结构分层次建模,通常表现为系统中相关线程的调度、接口与接口之间的通信、接口与外部系统的通信等。系统组件的主要功能是将所建模的系统中所拥有的所有组件组合起来,形成一种分层次的系统结构。AADL 建模语言通过构件、连接等来刻画系统的软硬件体系结构;通过特征、属性等来刻画系统的性质;当系统需要在不同的工作模式之间切换时,它也可以通过模式转换来刻画系统的运行体系演变。体系结构、执行模型和行为描述构成一个完整的 AADL 模型,因此 AADL 语义也涉及这 3 个方面。文中主要是对 AADL 中的 Behavior Annex[5-6] 语义进行扩充的形式化描述。

1.2 Z 语言简介

Z 语言[7] 是一种基于一阶谓词逻辑和集合论的形式规格说明语言,它采用了严格的数学理论,可以对系统的行为特征和状态值进行简明、精确、无歧义且可证明的形式化描述。Z 模式是 Z 语言的核心,它有两种模式:状态模式和操作模式。状态模式定义目标软件系统某一部分的状态空间及其约束特征;操作模式描述了系统某一部分的行为特征,它通过描述操作前该部分的状态值和操作后该部分状态值之间的关系来定义系统该部分的一种操作特征。为了简单明确地描述状态与操作模式,Z 规范中采用下述方式来表示变量:输入变量的最后一个字符后跟随一个“?”,输出变量的最后一个字符后跟随一个“!”。前状态变量就是通常的变量,对应的后状态变量是以撇号“’”作为右上标修饰的变量。

1.3 AADL 的 Z 扩充规则

AADL 的完整语义的形式化非常庞大,因此考虑一个 AADL 的子集,包括 Thread, connection, behavior annex,系统执行模式等。文中主要对 AADL 中的 behavior annex 进行 Z 语言扩充。其中,behavior annex 包括状态集合以及状态的转换集合。约定:R 表示变量之间的约束关系,ψ 表示状态转换时所要满足的约束性质,Computation 表示状态转换时执行的操作。状态的 Z 模式描述如下:

$$\text{State} \triangleq [x_1, x_2, \cdots, x_n?:\text{Input}; y_1, y_2, \cdots, y_n!:\text{Output} \mid x_i? \in \text{Input} \wedge y_j! \in \text{Output}; x_i, y_j : x_i R y_j]$$

状态转换的 Z 模式描述如下:

$$\text{Transition} \triangleq [x_1, x_2, \cdots, x_n?:\text{Input}; y_1, y_2, \cdots, y_n!:\text{Output}; \text{SrcState}, \text{DstState}:\text{STATE} \mid \text{SrcState}.x_i \text{SrcState}.x_i = \text{DstState}.y_i; \text{SrcState}.x_j = \text{DstState}.y_j; x_i, x_j \in \text{Input}; y_i, y_j \in \text{Output};]$$

状态转换时的约束性质的 Z 模式描述如下:

$$\text{Guard} \triangleq [x_1, x_2, \cdots, x_n?:\text{Input}; y_1, y_2, \cdots \cdot, y_n!:\text{Output} \mid x_i? \in \text{Input} \wedge y_j! \in \text{Output}; \varphi \in \psi]$$

状态转换时所进行的操作模式描述如下:

$$\text{Computation} \triangleq [x_1, x_2, \cdots, x_n?:\text{Input}; y_1, y_2, \cdots, y_n!:\text{Output} \mid x_i? \in \text{Input} \wedge y_j! \in \text{Output}; \varphi \in \psi]$$

1.4 基于连续时间的 ZIA(CT-ZIA)

1.4.1 CT-ZIA 的定义

ZIA[8-9] 可以刻画系统的行为和状态,但是它并不能刻画系统在实时方面的性质;时间自动机[10] 是用来刻画实时系统的一种自动机模型。文中将连续时间 ZIA 与时间自动机相结合,形成一种针对嵌入式实时系统的能够同时刻画行为和状态的形式规范 CT-ZIA[11]。

定义 1:一个基于连续时间的 ZIA(CT-ZIA) $P = \langle S_P, S_P^i, A_P^I, A_P^O, A_P^H, V_P^I, V_P^O, V_P^H, F_P^V, F_P^A, X, I, T_P \rangle$

其中:

(1) S_P 是状态集合。

(2) $S_P^i \subseteq S_P$ 是初始状态集合。如果 $S_P^i = \varnothing$,那么 P 就为空。

(3) A_P^I, A_P^O 和 A_P^H 分别是不相交的输入动作集合、输出动作集合和内部动作集合。记所有动作集合 $A_P = A_P^I \cup A_P^O \cup A_P^H$。

(4) V_P^I, V_P^O 和 V_P^H 分别是不相交的输入变量集合、输出变量集合和内部变量集合。记所有变量集合 $V_P = V_P^I \cup V_P^O \cup V_P^H$。

(5) F_P^V 是一个映射,把 S_P 中的任意一个状态映射到用 Z 语言描述的状态模式。

(6) F_P^A 是一个映射,把 A_P^I 中的任意一个输入动作映射到用 Z 语言描述的输入操作模式,把 A_P^O 中的任意一个输出动作映射到用 Z 语言描述的输出操作模式,把 A_P^H 中的任意一个内部动作映射到用 Z 语言描述的内部操作模式。

(7) X 为时钟变量的非负实数有限集合, $C(X)$ 为 X 上时钟约束的集合,其语法定义如下: $\Phi ::= x < c \mid c < x \mid \varphi_1 \wedge \varphi_2$。其中, $x \in X$, $< \in \{<, \leqslant\}$, c 为非负有理数。

(8)映射 $I: S_P \to C(X)$ 为每个状态赋一时间约束,此约束称为节点不变量。

(9) T_P 是状态之间转换关系的集合, $T_P \subseteq S_P \times A_P \times C(X) \times 2^X \times S_P$。如果 $(s, a, \varphi, \lambda, s') \in T_P$,那么表示在满足转换约束条件 φ 的前提下,通过动作 $a \in A_P(s)$,状态 s 可以迁移到新的状态 s',同时 $\lambda \subseteq X$ 中的时钟被重置为 0。

下面给出 CT-ZIA 所对应的一种时序逻辑及其语法和语义。

1.4.2　RT-DCL 时序逻辑

文献[12]通过在时序操作符上增加一个下标用于约束时间的范围,从而得到一种时序逻辑 RT-DCL,用这种时序逻辑可以作为 CT-ZIA 规范所对应的时序逻辑。

RT-DCL 的语法也就是逻辑合法公式,定义如下:

定义 2:RT-DCL 的逻辑合法公式。

(1)若有 φ 形如 $p(x_1, x_2, \cdots, x_n)$,则有 $\varphi \in$ RT - DCL。其中, $x_1, x_2, \cdots, x_n$ 表示变量, p 表示 n 元谓词。

(2)若有 $\varphi_1, \varphi_2 \in$ RT - DCL,则有 $\varphi_1 \vee \varphi_2 \in$ RT - DCL。

(3)若有 $\varphi_1, \varphi_2 \in$ RT - DCL,则有 $\varphi_1 \wedge \varphi_2 \in$ RT - DCL。

(4)若有 $\varphi_1 \in$ RT - DCL,则有 $(\forall x:T)\varphi_1 \in$ RT - DCL。

(5)若有 $\varphi_1 \in$ RT - DCL,则有 $(\exists x:T)\varphi_1 \in$ RT - DCL。

(6)若有 $\varphi_1, \varphi_2 \in$ RT - DCL,则有 $E\varphi_1 U_{\sim c}\varphi_2 \in$ RT - DCL。其中, $\sim \in \{<, \leqslant, =, >, \geqslant\}$, $c \in N$。

(7)若有 $\varphi_1, \varphi_2 \in$ RT - DCL,则有 $A\varphi_1 U_{\sim c}\varphi_2 \in$ RT - DCL。其中, $\sim \in \{<, \leqslant, =, >, \geqslant\}$, $c \in N$。

下面给出计算路径 CT-ZIA 上的定义。

定义 3:CT-ZIA P 上的一个计算路径可以表示为具有无限状态的序列 $\pi' = (s_0, s_1, \cdots)$,假设状态 s_i 上存在某个转移 $(s_i, a, \varphi, \lambda, s_i') \in T_P, i \in N$。假设存在某个计算路径 $\pi' = (s_0, s_1, \cdots)$,令 $\pi'[k] = s_k, \pi'_k = (s_0, s_1, \cdots, s_k)$, $k \in N$。将 P 中存在的全部的无限计算路径放入一个集合,计算路径的起点为 s,这个集合记为 $\prod'(s)$。

定义 4:RT-DCL 的语义。

如果由 Z-AADL 模型转换后得到的 CT-ZIA 模型是带有数据约束的,刻画如下:

$P = \langle S_P, S_P^i, A_P^I, A_P^O, A_P^H, V_P^I, V_P^O, V_P^H, F_P^V, F_P^A, X, I, T_P \rangle \varphi$ 所表示的是一个 RC-DCL 公式, $s \in S_P$,且满足关系 $(P, s) \models \varphi$,则归纳定义为:

(1) $(P, s) \models p(x_1, x_2, \cdots, x_n)$ 当且仅当存在任意的 $s \in S_P, F_P^V(s) \to p(x_1, x_2, \cdots, x_n)$;

(2) $(P, s) \models \varphi_1 \vee \varphi_2$ 当且仅当 $(P, s) \models \varphi_1$ 或 $(P, s) \models \varphi_2$;

(3) $(P, s) \models \varphi_1 \wedge \varphi_2$ 当且仅当 $(P, s) \models \varphi_1$ 且 $(P, s) \models \varphi_2$;

(4) $(P, s) \models (\forall x:T)\varphi_1$ 当且仅当有满足式 $(P, s) \models \cap_{v \in T}\varphi_1\{v/x\}$;

(5) $(P, s) \models (\exists x:T)\varphi_2$ 当且仅当有满足式 $(P, s) \models \cup_{v \in T}\varphi_2\{v/x\}$;

(6) $(P, s) \models E\varphi_1 U_{\sim c}\varphi_2$ 当且仅当存在有些计算路径 $\pi' \in \prod'(q)$, $t \sim c, \pi'[t] \models \varphi_2$,并且对于任意 $0 < t' < t, \pi'[t'] \models \varphi_1$;

(7) $(P, s) \models A\varphi_1 U_{\sim c}\varphi_2$ 当且仅当对任意计算路径 $\pi' \in \prod'(q)$, $t \sim c, \pi'[t] \models \varphi_2$,并且对于任意的 $0 < t' < t, \pi'[t'] \models \varphi_1$。

2　Z-AADL 到 CT-ZIA 的转换

2.1　Z-AADL 模型的刻画

下面将实时系统用 Z-AADL 规范刻画为一个七元组 $T = < S_T, N_T, P_T, C_T, \mathrm{TB}, A_T, \mathrm{TR}_T >$,其中:

(1) $S_T = \{s_i : i \in \mathbb{N}\}$ 表示系统中状态的集合,即 1.3 节中 Behavior annex 中的状态 States,即扩充了的 Z 状态模式的集合。

(2) N_T 表示系统中关于数据性质描述的集合。性质包括但不限于 1.3 节中的 Guard 等数据性质。

(3) P_T 表示系统中接口的集合,包括线程、进程

等构件中的输入输出接口。

(4) C_T 表示系统中时钟变量的非负实数集合,状态转换时所要满足的时钟约束;TB 为 C_T 上的时钟基,时钟基是时钟变量的有序可数集合: TB = $\{[c_i,c_j] \mid c_i < c_j\}$,其中 $c_i,c_j \in C_T$。

(5) A_T 表示系统中动作的集合,即 1.3 节 Transition 中扩充了的操作模式 Computation 等。

(6) TR_T 表示系统中状态转换的集合,即 1.3 节中 Behavior annex 中的状态转换 Transitions,即扩充了的状态转换模式 Transition 的集合。

2.2 Z-AADL 到 CT-ZIA 的转换规则

从 Z-AADL 到基于连续时间 CT-ZIA 模型的转化规则如下:

假设 $T = < S_T,N_T,P_T,C_T,TB,A_T,TR_T >$ 是由 Z-AADL 规范所建模的带数据约束的实时系统,将它转换到(CT-ZIA)模型 $P = \langle S_P,S_P^i,A_P^I,A_P^O,A_P^H,V_P^I,V_P^O,V_P^H,F_P^V,F_P^A,X,I,T_P\rangle$,其中:

(1) $S_P = \{s_i : s_{Ti} \in S_T, i \in \mathbb{N}\}$,其中 $s_i \in S_P$ 是 CT-ZIA 模型 P 中的状态, $s_{Ti} \in S_T$ 是 Z-AADL 模型 T 中的状态。

(2) $A_P = \{a_i : a_j \in A_T, a_j \rightarrow a_i, i,j \in \mathbb{N}\}$,其中 $a_j \in A_T$ 是 Z-AADL 模型 T 中的动作, $a_j \rightarrow a_i$ 表示将模型 T 中的每一个动作 a_j,映射为模型 CT-ZIA 中的一个动作 a_i,根据动作执行者的不同,将它们分别归入内部动作 A_P^H、输入动作 A_P^I 和输出动作 A_P^O 集合中。

(3) $V_P = \{v_i : p_i \in P_T, p_i \rightarrow v_i, i \in \mathbb{N}\}$, $p_i \rightarrow v_i$ 表示将 Z-AADL 模型 T 中的接口用到的变量 v_i 根据变量的拥有者分别把它们归入到输入变量 V_P^I、输出变量 V_P^O 和中间变量 V_P^H 的集合中。

(4) F_P^V 就是将 Z-AADL 模型 T 中对数据性质进行描述的每一个非功能属性 $n_i \in N_T, i \in \mathbb{N}$ 映射到 Z 模式的集合。

(5) F_P^A 是把 CT-ZIA 模型 P 中的动作 $A_P = A_P^I \cup A_P^O \cup A_P^H, A_P^I$ 中的任意一个输入动作映射到用 Z 模式描述的输入操作模式,把 A_P^O 中的任意一个输出动作映射到用 Z 模式描述的输出操作模式,把 A_P^H 中的任意一个内部动作映射到用 Z 模式描述的内部操作模式。

(6) $X = \{x_i : c_i \in C_T, i \in \mathbb{N}\}$, $C(X) = \{c_x : [c_i,c_j] \in TB, i,j \in \mathbb{N}, [c_i,c_j] \rightarrow x_i, x_i \in X, i \in \mathbb{N}\}$。其中 $c_i \in C_T$ 是 Z-AADL 模型 T 中的一个时钟变量,映射为 CT-ZIA 模型 P 中的一个时钟变量 $x_i \in X$, $[c_i,c_j] \in$ TB 是 Z-AADL 模型 T 中的一个时钟基, $[c_i,c_j] \rightarrow x_i$ 表示将时钟基 $[c_i,c_j] \in$ TB 映射为 CT-ZIA 模型 P 中对于 x_i 的时钟约束。

(7) $I = \{s_i \mid \rightarrow [c_i,c_j] : s_i \rightarrow x_i, [c_i,c_j] \rightarrow x_i, s_i \in S, [c_i,c_j] \in TB\}$,其中 $s_i \rightarrow x_i$ 表示将 Z-AADL 模型 T 中的时钟基 TB 中的时钟对 $[c_i,c_j]$ 转换为 CT-ZIA 模型 P 中状态 s_i 上的时钟约束。

(8) $T_P \subseteq S_P \times A_P \times C(X) \times 2^X \times S_P$,其中 $s_i \in S_P$, $s_i' \in S_P$ 表示 Z-AADL 模型 T 中的状态, $t_i \in S_T, t_i' \in S_T$ 分别表示状态的开始和结束。每一个状态转换 $(s_i \xrightarrow{a_i} s_i')$,其中 $a_i \in A_P$ 表示 Z-AADL 模型 T 中的状态转换 $tr_i \in TR_T, tr_i \rightarrow (t_i \xrightarrow{a_j} t_i')$, $a_j \in A_M$,时钟约束 $C(X)$ 表示 Z-AADL 模型 T 中的一个时钟基 TB。

在系统内部状态变迁过程中,CT-ZIA 中的状态转换是严格按照所要转换的扩充了 Z 语言的 AADL 系统中的状态转换映射过来的,这样也就保证了在转换过程中,系统内部行为的一致性;而关于对时钟的刻画[13],通过把实时系统中的时钟变量提取出来,CT-ZIA 中的时钟变量是根据实时系统中的时钟变量以及时间事件来进行时钟不变量的描述,这就确保了实时性的一致性描述;对于数据的约束能力,因为实时系统中非功能属性可以映射到 Z 语言模型,而 CT-ZIA 就是把数据用 Z 模式来进行约束,这样在数据约束能力方面它们采用的是一样的 Z 模式,所以从实时系统转换到 CT-ZIA 之后,它的行为和性质与转换之前保持了一致性。文献[13]对模型转换的正确性进行了归纳证明,文中的证明与此类似。

3 模型转换实例

锅炉装置在发电厂、船舶等领域应用广泛,文献[14]中对一个蒸汽锅炉控制系统进行了自动机建模,文中参考此模型,建立了一个简化的蒸汽锅炉模型,它包括温度控制器、供热系统以及压力控制器,如图 1 所示。其中,温度是由温度控制器控制,压力由阀门自动

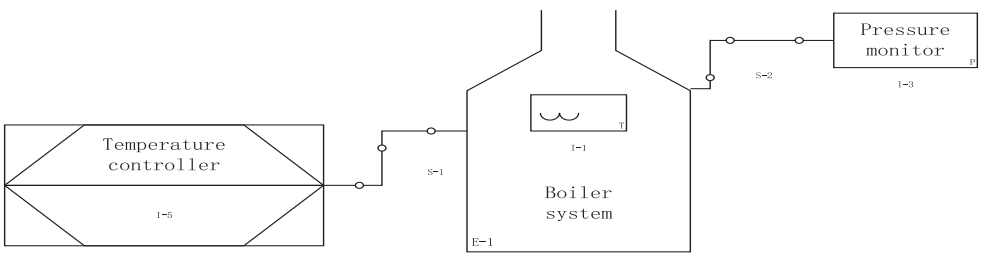

图 1 锅炉系统

控制,供热系统会向压力控制器发送压力值。

图 2 展示了系统中温度和压力的状态转换关系,用变量 x 和 y 分别表示温度和压力。$x?$ 表示供热系统从温度控制器接收一个输入变量,$y!$ 表示供热系统向压力控制器发送一个输出变量。假设初始温度和压力分别为 20 °C 和 100 Kpa,初始状态为 l_0。当压力大于等于 700 时状态会跳转到 l_1,并将压力置为 700,压力导数置为 30,此后的状态跳转与此类似。

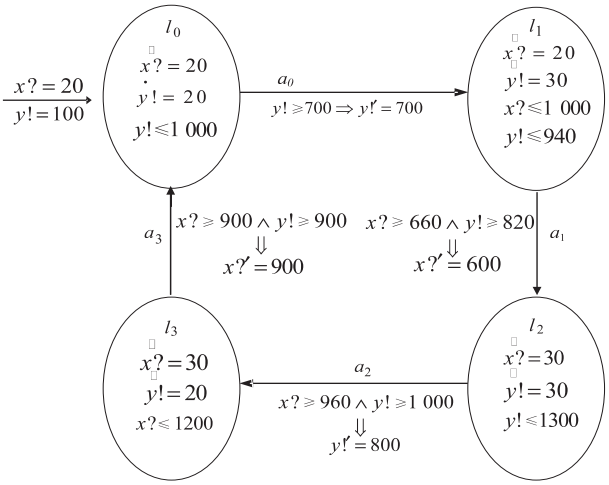

图 2　状态转换关系

锅炉系统中的压力会随着温度自动变化,因此下面只给出温度控制器的 AADL 建模描述:

```
Thread implementation temp. impl
Features
x :in data port temperature;
y :out data port pressure;
Properties
Dispatch _Protocol=>Periodic;
Period=>50 ms;
Compute_Execution_Time=>15 ms. . 15 ms;
Deadline=>50 ms;
annex behavior_specification * *
states
l0:initial complete state;
transitions
l0-[ ]→l1 { a0 (30 ms ); x' :=20, y' := y' +10};
l1-[ ]→l2 { a1 (15 ms ); x' := dx' +10, y' := y' };
l2-[ ]→l3 { a2 (15 ms ); x' := x' , y' := y' -10};
l3-[ ]→l0 { a3 (15 ms ); x' := x' -10, y' := y' };
* * ;
end temp . impl ;
```

根据 1.3 节中的 Z 扩充规则,对 AADL 模型中的状态、操作以及状态变迁等进行 Z 模式的扩充。以 l_0,a_0 以及通过 a_0 所做的状态变迁为例,分别给出 Z 模式描述如下,后面的情形与此类似。

$l_0 \triangleq [x?:N;y!:N \mid x? < 1\,000;y! \geqslant 20]$;

其中,$x?$ 表示状态 l_0 时的温度,$y!$ 表示状态 l_0 时的压力。

transition $\triangleq [x?:N;y!:N;l_0;l_1 \mid x? < 1\,000 \wedge y'! \leqslant 940;x? = 20 \wedge y'! \leqslant 1\,000]$;

其中,l_0;l_1 表示状态变迁时的前状态与后状态。

$a_0 \triangleq [x?:N;y!:N \mid x? = 700;x'? = 30]$;

根据上述状态转换关系,可以刻画锅炉系统的 Z-AADL 模型,即 2.1 节中的七元组模型 $T = < S_T, N_T, P_T, C_T, \mathrm{TB}, A_T, \mathrm{TR}_T >$。

(1) $S_T = \{l_0, l_1, l_2, l_3\}$ 表示系统中状态的集合;

(2) $N_T = \{?\ (y! \geqslant 700), (x? \geqslant 660 \wedge y! \geqslant 820), (x? \geqslant 960 \wedge y! \geqslant 1\,000), (x? \geqslant 900 \wedge y! \geqslant 900)\}$;

(3) $P_T = \{\mathrm{temperature}, \mathrm{pressure}\}$ 表示系统中接口的集合;

(4) $C_T = [0,75]$,TB 为 C_T 上的时钟基,$\mathrm{TB} = \{[0,30],[30,45],[45,60],[60,75]\}$;

(5) $A_T = \{a_0, a_1, a_2, a_3\}$ 表示系统中动作的集合;

(6) $\mathrm{TR}_T = \{(l_0, a_0, l_1), (l_1, a_1, l_2), (l_2, a_2, l_3), (l_3, a_3, l_0)\}$ 表示系统中的状态转换的集合。

根据 2.2 节的模型转换规则,可将上述的 Z-AADL 模型转换为 CT-ZIA 模型:

$P = \langle S_P, S_P^i, A_P^I, A_P^O, A_P^H, V_P^I, V_P^O, V_P^H, F_P^V, F_P^A, X, I, T_P \rangle$

(1) $S_P = \{l_0, l_1, l_2, l_3\}$;

(2) $S_P^i = \{l_0\}$;

(3) $A_P = \{a_0, a_1, a_2, a_3\}$;

(4) $V_P = \{x?; y!\}$;

(5) $F_P^S(l_0) \triangleq [x?:\mathbb{R}; y!:\mathbb{R}, \mathrm{clock}:C_P \mid x? = 20; y! = 100; \mathrm{clock} = 0]$,$l_1, l_2, l_3$ 的状态模式与此类似;

(6) $F_P^S(a_0) \triangleq [y!:\mathbb{R} \mid y! = 700]$,$a_1, a_2, a_3$ 对应的操作模式与此类似;

(7) $T_P = \{(l_0, \mathrm{a}_0, l_1), (l_1, \mathrm{a}_1, l_2), (l_2, \mathrm{a}_2, l_3), (l_3, a_3, l_0)\}$。

4　结束语

文中对 AADL 建模语言进行了 Z 语言扩充,形成了新的规范 Z-AADL,从而使其不仅保留了原来的描述能力,而且具备了形式化的描述状态及状态转换之间的数据约束方面的能力,为转换为 CT-ZIA 形式化模型打下了基础;然后研究了 Z-AADL 模型与 CT-ZIA 模型之间的转换,使得转换后的模型可以采用形式化方法进行分析与验证等。

在 AADL 的扩展和形式化模型转换方面取得了一定的成果,但是转换后的模型是基于连续时间的无穷状态模型,因此需要将其转换为有限的域自动机模型

来验证。未来的主要工作是研究基于连续时间的 CT-ZIA 如何转换为有限的域自动机模型,从而实现模型检测。

参考文献:

[1] 杨志斌,皮 磊,胡 凯,等. 复杂嵌入式实时系统体系结构设计与分析语言:AADL[J]. 软件学报,2010,21(5):899-915.

[2] 禚百田,付秀敏,郑永果,等. 基于 UML 的嵌入式实时系统开发方法[J]. 信息技术与信息化,2010(1):56-59.

[3] SAE Aerospace. SAE AS5506:Architecture Analysis and Design Language (AADL)[EB/OL]. 2004. http://www.aadl.info/aadl/currentsite/.

[4] 王立杰,刘昌禄,俞烈彬. 基于 MDA 的 MARTE 模型形式化转换[J]. 指挥控制与仿真,2012,34(6):128-133.

[5] SAE Aerospace. SAE AS5506 annex:behavior_specification V1.6. [EB/OL]. 2006. http://www.aadl.info/aadl/documents/Behaviour_Annex1.6.pdf.

[6] 李振松,顾 斌. 基于 UPPAAL 的 AADL 行为模型验证方法研究[J]. 计算机科学,2012,39(2):159-161.

[7] 倪水妹. 面向混成系统的 ZIA 形式化模型及其自动验证方法研究[D]. 南京:南京航空航天大学,2014.

[8] 李广元,唐稚松. 带有时钟变量的线性时序逻辑与实时系统验证[J]. 软件学报,2002,13(1):33-41.

[9] 狄杨思,曹子宁,王 辉. 一种基于形式规范 ZIA 的自动验证方法[C]//火力控制技术/航空电子系统综合技术 2011 年学术年会. 出版地不详:出版者不详,2011:113-120.

[10] 孙全勇. 时间自动机及其应用研究[D]. 哈尔滨:哈尔滨工程大学,2007.

[11] 倪水妹,曹子宁,李心磊. 带数据约束实时系统的模型检测[J]. 计算机科学,2014,41(5):254-262.

[12] Lin Huiming, Zhang Wenhui. Model checking: theories, techniques and applications[J]. Acta Electronica Sinica, 2002, 30(S1):1907-1912.

[13] 钱俊彦,赵岭忠,古天龙. 一种基于时间自动机的时钟等价性优化方法[J]. 计算机工程,2005,31(18):71-73.

[14] Henzinger T A, Wong-Toi H. Using HyTech to synthesize control parameters for a steam boiler[M]. Berlin: Springer, 1996.

(上接第 22 页)

孟海军,译. 第 2 版. 北京:电子工业出版社,2004.

[2] 姜 文,刘立康. 现代应用软件的维护与技术支持[J]. 计算机技术与发展,2015,25(4):116-120.

[3] 姜 文,刘立康. 应用软件维护中的补丁开发与管理[J]. 计算机技术与发展,2015,25(11):11-16.

[4] 程友清. 嵌入式网络设备软件热补丁技术研究[J]. 微电子学与计算机,2013,30(1):28-31.

[5] 常莉莉. 分布式系统热补丁技术的研究与实现[D]. 广州:中山大学,2006.

[6] 雷震宇. 嵌入式网络设备在线热升级的研究及实现[D]. 武汉:武汉邮电科学研究院,2012.

[7] River W. Vxworks 程序员指南[M]. 北京:清华大学出版社,2003.

[8] 陈智育,温彦军,陈 琪. Vxworks 程序开发实践[M]. 北京:人民邮电出版社,2004.

[9] Mehdi G. Contributors to quality during software maintenance [J]. Decision Support Systems, 1998, 23(4):361-369.

[10] Hauptmann S, Wasel J. On-line maintenance with on-the-fly software replacement[C]//International conference on configurable distributed systems. [s. l.]: IEEE Computer Society, 1996:70-80.

[11] 姜 文,刘立康. Oracle 数据库补丁问题研究[J]. 电子设计工程,2014,22(20):10-13.

[12] Shah R. Oracle on demand best practices: critical patch update [R]. [s. l.]: Oracle Corporation, 2008.

[13] 姜 文,刘立康. 基于 VxWorks 平台的软件重量级静态检查[J]. 微型机与应用,2016,35(6):79-81.

[14] 蔡建平. 嵌入式软件测试实用技术[M]. 北京:清华大学出版社,2010.

[15] Michalik B, Weyns D, Boucke N, et al. Supporting online updates of software product lines: a controlled experiment[C]// 2011 international symposium on empirical software engineering and measurement. Washington DC: IEEE Computer Society, 2011:187-196.

[16] Chen Xuhui, Zhang Dengyi, Yang Hongyun. Research on key technologies of on-line programming in embedded system [C]//2009 third international symposium on intelligent information technology application. Washington DC: IEEE Computer Society, 2009:45-47.