基于 AADL 的嵌入式系统可调度性验证

孙 健,徐 敏
(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:AADL 近年来在嵌入式实时系统领域得到了广泛的应用,相对于其他语言能够更好地描述系统的非功能属性,同时支持系统软硬件建模,在基于模型驱动的开发方法下对系统进行建模和分析,用形式化的方法对系统的相关属性进行验证,从而可以在设计阶段发现错误,对保证系统的安全运行和开发效率的提高有着重要意义。对于验证 AADL 模型可调度性的问题,文中利用时间自动机理论,根据 AADL 调度模型和时间自动机模型的语义相似性,将 AADL 模型转换成时间自动机模型,并设计转换插件,通过 Eclipse 插件开发技术将其集成到 AADL 建模与分析工具 OSATE 中。最后在 UPPAAL 工具中对转换后的时间自动机模型进行模拟和验证,利用相关性质验证语句等价地对原模型的可调度性进行验证。

关键词:嵌入式系统;体系结构分析设计语言;时间自动机;模型转换;UPPAAL;实时性
中图分类号:TP302 **文献标识码**:A **文章编号**:1673-629X(2016)03-0023-04
doi:10.3969/j.issn.1673-629X.2016.03.006

Schedulability Verification of Embedded System Based on AADL

SUN Jian,XU Min
(School of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,
Nanjing 210016,China)

Abstract:AADL has been widely used in embedded real-time system in recent years. Compared with other languages,it can better describe the system non functional attributes,and also support the software and hardware modeling of the system. AADL can model and analyze the system based on the model driven development method,and verify relevant properties using formal method. Error can be found in the design phase,and it has great significance to ensure the safe operation of the system and improve the efficiency of development. For verifying AADL model schedulability problem,according to the semantics similarity of AADL scheduling model and timed automata model,the theory of timed automata is used to convert AADL model to timed automata model,and integrate conversion plug-in into the AADL modeling and analysis tool OSATE through the development of Eclipse plug-in technology. Finally,the converted timed automaton model in the UPPAAL tool is simulated and verified,using the related verification statement to verify the schedulability of the original model.

Key words:embedded system;AADL;timed automata;model transformation;UPPAAL;real-time

0 引 言

对于安全关键的嵌入式系统来说,系统的非功能属性跟功能属性一样重要,都需要得到满足[1-2]。基于模型驱动的方法可对系统设计和安全性分析采用一致的模型,该模型贯穿整个开发过程[3]。在系统设计阶段,通过对模型分析和验证,发现错误来提高系统的安全性,保证系统能够正常运行,从而提高开发效率。

体系结构分析设计语言(Architecture Analysis and Design Language,AADL)近年来在航空电子领域以及安全性分析等方面得到了广泛应用,是一种符合模型驱动思想的建模和分析语言,在 2004 年由 SAE 基于 MetaH 和 UML 提出,并发布为 SAEAS5506 标准,能对系统的软硬件进行建模。随着 AADL 的广泛应用以及对内容的扩展,与 AADL 相关的研究工作及其工具的开发日益成为研究热点。

时间自动机建模与分析工具 UPPAAL 近年来在实时系统中得到广泛应用,具有建模功能灵活、验证效率高等特点,也在 AADL 建模与分析中得到了尝试[4-5]。其中,Johnsen 在文献[6]中提出用 UPPAAL 进行建模,并在模拟器和验证器中对模型进行模拟和

收稿日期:2015-06-09 **修回日期**:2015-09-15 **网络出版时间**:2016-02-18
基金项目:国家"973"重点基础研究发展计划项目(2014CB744900)
作者简介:孙 健(1990-),男,硕士研究生,研究方向为人工智能;徐 敏,副教授,研究方向为人工智能、软件工程。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20160218.1630.032.html

验证,且模型的调度策略是固定优先级的;刘倩等使用 UPPAAL 对 AADL 非可抢占调度模型设计了时间自动机[7]。上述研究存在的问题是调度策略单一,没有考虑到线程切换中的时间消耗,从而对于实际的 AADL 模型不能很好地进行模拟,且其考虑的问题还停在了对自动机的建模和验证上,而对于 AADL 模型转换为时间自动机模型,以及转换插件的设计都还没较多的涉及。

针对上述问题,利用时间自动机理论,根据 AADL 调度模型和时间自动机模型的语义相似性,将 AADL 模型转换成时间自动机模型,并设计转换插件,在 UPPAAL 工具中对转换后的时间自动机模型进行模拟和验证,利用相关性质验证语句等价地对原模型的可调度性进行验证。

1 AADL 与 UPPAAL

1.1 AADL 概述

在 AADL 中,组件通过类型和实现声明来定义[8]。一个组件类型说明定义了一个组件的接口元素以及外部可观察属性(即与其他组件、流程规范和内部属性值之间交互点的特性)。一个组件实现的声明定义了一个组件内部结构的子组件、子组件连接、子程序调用、模式、流实现以及属性。组件被分为应用程序软件,执行平台和复合组件,属性集和附件库使得设计者能够扩展语言和自定义一个 AADL 规范来符合项目或特定区域的需求。

组件在特定的组件类别中声明为类型和实现,有三组不同的组件类别:应用软件、执行平台和复合。一个 AADL 组件类型声明建立了一个组件的外部可见特性[7]。例如,一个声明指定了一个线程组件的接口,一个组件类型声明由一个定义句子和描述性的子句组成,特征(features)是组件的接口,流(flows)指定信息流的不同抽象通道,属性(properties)定义组件的固有特性,每个组件类别都有预定义的属性(如一个线程的执行时间)。组件的实现就是用来描述一个组件的内部结构,一个组件实现包括的子句主要有:流,属性,模式和子组件。流(flows)代表组件类型里流规范的实现或者端到端的流分析(流从一个子组件开始,经过零个或更多子组件并在另一个子组件结束);模式(modes)是对组件、连接及属性值的配置,表示一个系统或组件可达到的操作状态;属性(properties)定义组件的固有特性,每个组件的实现都有预定义的属性。

1.2 时间自动机与 UPPAAL 概述

1.2.1 时间自动机理论

时间自动机[9]理论最早是在 1992 年由 Alur R 提出,在有限状态自动机上扩充了时钟、时钟约束及不变条件而得到。时间自动机被提出是实时系统建模自动机理论的扩展,从此时间自动机理论及其验证工具一直是计算机科学研究的一个密集领域。一个时间自动机是一个配备了一组有限时钟的有限自动机,时钟是时间的连续的实值函数,精确地记录时间的流逝。所有时钟以相同的速度前进,即它们是同步的,实时系统行为使用时间自动机来捕获,通过设置时钟以及用常数比较时钟读数[10]。所有时钟都有和时间相关的相同的衍生物,且都被假定为一个相同的定义。

下面根据时间自动机的定义,做如下说明[11]:

定义 1:状态转移系统。

将状态转移系统形式化为四元组 $< T, t_0, P, E >$。其中:T 为转移系统中的状态集合;t_0($t_0 \in T$)为初始状态;P 为触发状态发生转移的事件集合;$E \subseteq T \times P \times T$ 为所有状态转移的集合。

在一个系统中,一个状态在触发事件的作用下发生状态转移,若一个状态到另一个状态的转移发生则称该状态到另一状态是可到达的,而从初始状态开始,所有可到达其他状态的转移构成了状态空间。

定义 2:时钟约束。

把 $\varphi(C)$ 作为时钟约束集合,其中 C 为时钟变量集,定义时钟约束集合为:

$\varphi := x \rhd \lhd n \mid \varphi_1 \wedge \varphi_2$

其中:x 是一个时钟变量;n 是一个自然数集 N 中的常量。

定义 3:时钟解释。

时钟解释 $v: C \rightarrow N$ 表示集合 C 到自然数的映射。

对于一个 $\delta \in R$,R 表示的时钟解释是对于集合 C 中的每一个时钟变量 x 的赋值为 $v(x) + \delta$;对于一个 $\delta \in R$,$\delta \cdot v$ 表示的时钟解释是对 C 中的每个时钟变量 x 赋值为 $\delta \cdot v(x)$;对于 $X \subseteq C$,$v[X := 0]$ 表示对于满足 $x \in X$ 的时钟 x 复位为 0,其余时钟保持增长。

定义 4:时间自动机。

在有限状态自动机的基础上,通过增加时钟变量的概念形成了时间自动机,其中时钟为整型变量,并且所有的时钟变量是同步递增的[12-14]。一个时间自动机可以表示为一个六元组,$TA = < T, t_0, C, V, E, I >$。其中:$T$ 为有向图中有穷的位置(location)集合;t_0 为自动机的初始状态;C 为时钟集合,默认从 0 开始,不断自增加 1,可以随时被重新赋值;V 为所有变量的集合;E 为有向图中所有边(edge)的集合;I 为状态转移约束函数的集合,即不变条件(invariant)。

1.2.2 验证工具 UPPAAL

UPPAAL[7]是一个基于约束求解和动态技术的,用来进行实时系统建模、仿真和验证的工具,它可以将系统建模成一系列具有有限控制结构和实值时钟的非

确定性的进程,并通过共享变量进行通信。典型的应用领域包括实时控制器和特别的通信协议,时间方面是十分关键的,所以在设计方面,主要是通过探索系统的状态空间来检查不变和可达性。

UPPAAL 工具的两个设计标准是效率和易于使用,可达性分析的限制对于工具的模型检查器的效率至关重要。对效率的另一个关键是结合符号技术的动态搜索技术的应用,减少为处理和解决简单约束的验证问题。为了方便建模和调试,UPPAAL 模型检查器会自动生成一个诊断跟踪,来解释为什么一个属性满足(或不满足)一个系统描述,由模型检查器生成的诊断跟踪可以使用模拟器来图形化显示。

2　AADL 可调度模型转换与验证

为了验证实时系统 AADL 模型中的可调度性,利用时间自动机形式化方法。首先分析 AADL 模型,将与系统调度相关的属性信息提取出来,抽象出调度模型。在非可抢占及可抢占调度策略下,分别设计了线程和调度器时间自动机模板(以下简称模板),并利用 TCTL 验证语句对 UPPAAL 中的时间自动机模型进行可调度性验证。最后设计了从 AADL 调度模型到时间自动机模型的转换插件,并自动调用 UPPAAL 打开模型对相关性质进行验证。

2.1　非可抢占调度策略下的时间自动机设计

(1)线程模板。

线程是执行的主体,线程组件也是通过线程模板来进行转换,在线程模拟的过程中,用标号 id 来区分,在非可抢占策略下设计了如图 1 所示的周期线程时间自动机模板。

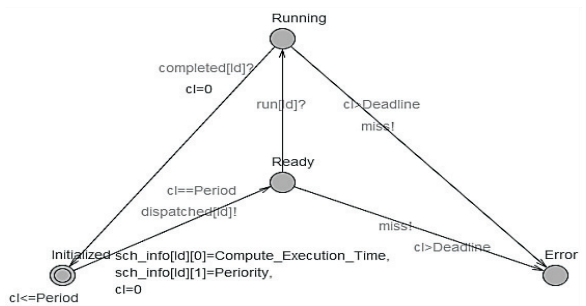

图 1　非可抢占调度策略下周期线程时间自动机模板

如图 1 所示,模板中有四个状态,分别为初始化状态(Initialized)、准备状态(Ready)、错误状态(Error)及运行状态(Running)。Initialized 状态表示线程创建成功且处于等待被派发的状态,其中不变条件 cl<=Period 代表线程处于 Initialized 状态的条件为局部时钟小于等于周期;Ready 表示线程按照优先级的高低进入到等待队列,等待被处理器执行;Running 状态对应于线程状态转换机制中的 Compute(计算)状态,表示线程正在被运行;Error 状态对应于线程状态转换机制中的 Recover(恢复)状态,表示线程由于超时而进入到一种死锁状态,且所有的状态转移在该状态下都会中止。

(2)调度器模板。

根据并发系统的特点,即线程会根据调度策略通过调度器来决定其运行状况,于是在调度器时间自动机模板的设计过程中,利用同步通道以及全局变量与线程进行通信,对组件调度属性进行模拟,设计的调度器模板如图 2 所示。

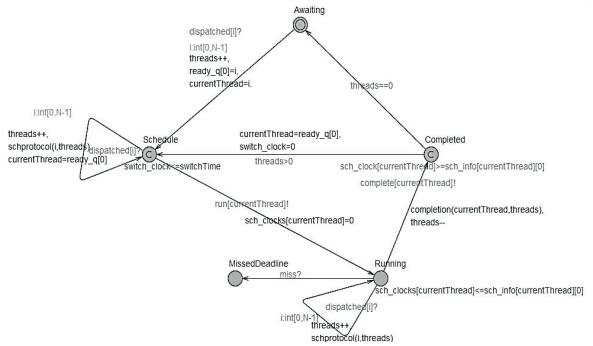

图 2　非可抢占调度策略下调度器时间自动机模板

从图中可以看出,调度器模板有如下五个状态:等待状态(Awaiting)、调度状态(Schedule)、运行状态(Running)、完成状态(Completed)、超时状态(MissedDeadline)。其中调度状态、完成状态为限制状态,即不考虑时间的流逝,为中间状态。Awaiting 状态表示在处理器中没有要调度执行的线程;Schedule 是一种中间状态,调度器会在有线程被派发时进入到该状态;Running 状态表示线程的运行状态,模拟线程在处理器上的运行;Completed 状态表示线程运行完毕,进入该状态的条件为线程实际运行时间与需要的最大运行时间相等;MissedDeadline 状态表示线程运行超时,没有在截止时间之前完成运行,当系统进入到此状态便会发生死锁,即该系统不可调度。

(3)带有线程切换时间下的非可抢占调度器模板。

线程在切换时产生了时间的消耗,并在模型中用 Thread_Swap_Execution_Time 表示线程切换过程所产生的时间消耗。因此本节扩展了模板,用 switch_clock 表示切换时钟,用 switch_clock<=switchTime 模拟某状态的时间切换。

扩充后的模板中, switchTime 若为 0,则表示该调度器模板不考虑切换时间,但在实际情况中,新增加的判断条件可能会带来很大的影响和负担,因此将其转换到对应的模板显得尤为重要。

2.2　可抢占调度策略下时间自动机设计

(1)线程模板。

相对于非可抢占调度策略,在可抢占调度策略下,

周期线程时间自动机模板增加了一个 Preempted 状态。该状态表示线程在运行时被更高优先级的线程抢占后停止运行,等到更高优先线程运行完之后,该线程恢复运行,返回到 Running 状态,其中该状态上的截止时钟是持续增长的。

(2)调度器模板。

由于引入了线程的抢占、恢复等操作,在可抢占调度策略下,调度器模板的复杂程度相对来说要大些。在非可抢占的调度器模板基础上,该模板增加了两个状态 Schedule1 和 Preemption。其中:Schedule1 是一种中间状态,调度器在有线程被派发时会进入到此状态,同时线程会在调度器中按优先级高低排成一个队列 ready_q;Preemeption 状态表示有新线程派发时 Schedule1 决定执行可抢占调度,并按优先级由高到低依次进入运行状态,且被抢占的线程会进入到被抢占状态。

(3)带有线程切换时间下的可抢占调度器模板。

线程在可抢占调度策略下,因抢占而产生时间的消耗,所以线程在恢复到执行状态时被抢占时间的计算会有所不同。

2.3 模型转换器的工具实现

对于模型的自动转换,通过 Eclipse 插件开发技术设计了转换插件,并将其集成到了 AADL 建模与分析工具 OSATE 中,实现了 AADL 模型到时间自动机模型的自动转换。插件以实例化的 AADL 模型作为输入,通过文件解析,转换生成时间自动机模型文件和性质验证查询文件,自动调用 UPPAAL 工具打开模型进行可调度性验证。

2.4 可调度性验证

文中使用 UPPAAL(时间自动机建模与验证工具)来验证设计的时间自动机模型,从而对 AADL 调度模型进行等价的可调度性验证。UPPAAL 内含有模拟器和验证器,在模拟器里可以观察时间自动机的状态转移,且可用单步或连续的方式,同时可观察状态序列的变化,从而模拟系统的运行。从前面的介绍中可知,在 UPPAAL 中,模型验证时所使用的理论是自动机可达性验证理论。同时在验证器中使用了时序逻辑 TCTL 来进行相关属性的验证,其性质查询语言包括状态公式和路径公式。而文中在验证调度模型的可调度性中,用到的主要语句见表 1。

表 1 模型可调度性性质验证语句

性质验证语句	验证性质说明
A[ ] not deadlock	判断系统是否发生死锁,是否可以连续运转
A < > Thread0. Running	判断某线程是否进入运行状态
A[ ] not Thread0. Error	判断某线程是否进入错误状态
E < > Processor. MissedDeadline	判断调度器是否进入超时状态
Thread0. Preempted → Thread0. Running	判断某线程是否由被抢占状态转移到执行状态

3 结束语

文中利用时间自动机的形式化检验方法,设计了 AADL 调度模型到时间自动机模型的转换方法,在 UPPAAL 工具中对时间自动机模型进行模拟和验证,利用相关验证语句,等价地验证原模型可调度性。同时设计了不同调度策略下的线程和调度器模板以及自动转换插件,并将其集成到工具 OSATE 中,使得嵌入式实时系统的建模、转换以及验证实现一体化。

对基于时间自动机的 AADL 模型验证工作虽然取得了一定进展,但考虑调度策略影响因素时,所关心的还不是很全面,如目前还不支持多处理器和 EDF 调度算法。在未来的研究工作中,需进一步考虑影响系统调度的因素,从而使得分析和验证的效果更加全面。

参考文献:

[1] Krishna C M. Real-time systems[M]. [s. l.]:John Wiley & Sons,Inc. ,1999.

[2] Peled D. Software reliability methods[M]. Berlin:Springer,2001.

[3] Selic B. The pragmatics of model-driven development[J]. IEEE Software,2003,20(5):19-25.

[4] Björnander S, Seceleanu C. A formal analysis framework for AADL[J]. Journal of Science and Technology,2011,49(5):1-13.

[5] Bae K, Ölveczky P C, Meseguer J, et al. The SynchAADL2 Maude tool[M]. Berlin:Springer,2012.

[6] Johnsen A. Architecture-based verification of dependable embedded systems[D]. Sweden:Mälardalen University,2013.

[7] 刘 倩,桂盛霖,李 允,等. 基于 UPPAAL 的 AADL 模型可调度性验证[J]. 计算机应用,2009,29(7):1820-1824.

[8] 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.

[9] Alur R. Timed automata[M]//Computer aided verification. Berlin:Springer,1999.

[10] 谯婷婷,王 乐,耶国栋. 基于 AADL 的软件可靠性验证[J]. 计算机应用,2012,32:92-95.

[11] 童 超. 基于时间自动机的 RBC 控车流程研究[D]. 成都:西南交通大学,2009.

[12] 朱雪阳,唐稚松. 基于时序逻辑的软件体系结构描述语言 XYZ/ADL[J]. 软件学报,2003,14(4):713-720.

[13] 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.

[14] 周清雷,姬莉霞,王艳梅. 基于 UPPAAL 的实时系统模型验证[J]. 计算机应用,2004,24(9):129-131.

基于AADL的嵌入式系统可调度性验证

万方数据 WANFANG DATA 文献链接

作者： 孙健， 徐敏， SUN Jian， XU Min
作者单位： 南京航空航天大学 计算机科学与技术学院,江苏 南京,210016
刊名： 计算机技术与发展 ISTIC
英文刊名：
年，卷(期)： 2016,26(3)

引用本文格式：孙健.徐敏.SUN Jian.XU Min 基于AADL的嵌入式系统可调度性验证[期刊论文]-计算机技术与发展 2016(3)