AADL 行为模型时间一致性验证方法

刘 骁[1],谢红梅[2]
(1. 南京航空航天大学 计算机科学与技术学院,江苏 南京 211100;
2. 南京航空航天大学 继续教育学院,江苏 南京 211100)

摘 要:任务关键的实时系统对时序及安全要求的特殊性,在实时系统开发的早期阶段进行体系结构的时间一致性分析,能够尽早发现系统设计时出现的有关时序的潜在问题,为此提出了基于时间约束的 AADL 行为模型的验证方法。针对带有时间约束的 AADL 行为模型时序验证问题,提出了一种基于节点转换规则的 AADL 行为模型分解规则,将实时系统的行为模型转换成运行时的路径集合;设计了路径集合到 Prolog 事实的转换算法;将实时系统的隐式时间约束和显式时间约束转换成 Prolog 规则,支持实时系统对两种时间约束进行验证。最后应用 AADL 行为模型时间一致性验证方法对船舶指控系统进行实例验证,验证了实时系统的隐式时间约束和显式时间约束,为实时系统的时序分析提供了一种新的有效途径。

关键词:AADL 行为模型;实时系统;时间一致性;Prolog
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2017)07-0001-05
doi:10.3969/j.issn.1673-629X.2017.07.001

A Time Consistency Validation Approach of AADL Behavior Model

LIU Xiao[1], XIE Hong-mei[2]
(1. College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211100, China;
2. College of Continuing Education, Nanjing University of Aeronautics and Astronautics, Nanjing 211100, China)

Abstract: In the early development of real time system, a time consistency of system structure is analyzed due to the specialty of time series and security for real time system with critical tasks, which can discover the potential issue related time series in system design as soon as possible. Therefore, the validation method of AADL behavior model based on time restraint is proposed. In view of correctness verification problems of the AADL behavior model with time restriction, a kind of model decomposition rule based on node switching rule is presented to transform the AADL behavior model into execution trace set. Also a transformation algorithm from the trace set to Prolog language is designed, transforming the implicit and explicit time constraint into Prolog rules, supporting the validation of the real-time system with two time constraints. Finally, a special example of shipborne combat system with time restriction is verified by means of the time consistency validation approach of AADL behavior model. The approach verifies the implicit and explicit time constraint and provides a new effective way for the real time system of time series analysis.

Key words: AADL behavior model; real-time system; time consistency; Prolog

0 引 言

随着通信、微电子等技术的飞速发展,嵌入式实时系统广泛应用于汽车、航空航天等任务关键领域中。为了保证实时系统的安全性及可靠性,基于模型驱动架构(Model Driven Architecture, MDA)[1]思想的体系结构分析与设计语言—AADL[2]广泛应用于实时系统的建模及验证中。实时系统的复杂性逐渐增加,随之 AADL 规范及附件内容[3]也在不断扩展,围绕其展开的模型验证工作也即成为了热点。在系统开发的早期阶段,就对系统模型的时间一致性、安全性等进行验证,可以避免开发阶段对模型进行的反复修改,大大提高开发效率。基于逻辑程序语言 Prolog,借助 SWI-Prolog 工具对实时系统行为模型的时间一致性进行前期验证是一种可行的方法。

AADL 行为模型[4]是对 AADL 标准语言所构建模型的进一步精化,是以 AADL 的行为附件[5]为基

收稿日期:2016-08-29 **修回日期**:2016-12-02 **网络出版时间**:2017-06-05
基金项目:"十三五"重点基础科研项目(JCKY2016206B001)
作者简介:刘 骁(1991-),男,硕士研究生,研究方向为软件工程、系统建模与仿真;谢红梅,硕士,讲师,研究方向为软件工程。

础、描述 AADL 标准组件内部详细行为的模型。文献[6]采用时间抽象自动机对实时系统进行建模分析,给出了 AADL 子集和 TASM 的抽象语法,并基于语义函数和类 ML 的元语言形式定义转换规则,并在此基础上对实时系统的隐式时间约束进行验证,但对于时间区间并没有做详细验证。文献[7-9]在实时系统建模阶段采用 Petri 网,其具有图形化描述、精确的语义和强大的表达能力,但 Petri 网具有很高的空间复杂度,若为了降低复杂度限制 Petri 网有界,则会降低其表达能力。Ricardo[10]通过扩展 AADL 行为附件增强 AADL 行为附件的执行语义,借此对航电系统进行详细的定性描述,但是对实时系统的定量描述较欠缺。

文中提出一种基于逻辑程序语言 Prolog[11]的实时系统时间一致性的验证方法,通过将模型与时间约束转换成 Prolog 语言中的事实,定义 Prolog 规则,对系统进行时间一致性验证。

1 AADL 行为模型及时间约束

AADL 行为模型定义了各节点的详细动作,是以 AADL 的行为附件为基础,对 AADL 标准语言所构建模型的进一步细化。AADL 的行为附件基于扩展自动机理论[12],通过 6 个可选块进行描述:

```
annexbehavior_specification {* *
<state variables>?
<initialization>?
<states>?
<transition>?
<connections>?
<composite_declaration> *
* *};
```

<state variables>为声明类型表示符。这里的类型代表 AADL 模型的数据分类器。<initialization>代表状态变量的初始化。<states>声明了自动机的状态,也即构件内部行为状态的变化,状态被分为初始态(initial)、完成态(complete)、返回态(return)、紧急态(urgent)和复合态(composite)。<transition>定义了状态间迁移过程的描述。基于以上理论形式地给出如下定义:

定义 1:带时间约束的 AADL 行为模型为一个多元组 $B=(N,\Gamma,\tau,\Sigma,\Upsilon,E,C)$。其中,$N$ 表示状态变量的集合,即代表模型中的各个节点;$\Gamma=\{\text{Activity},\text{Event},\text{EXCL},\text{INCL}\}$ 表示节点类型,包括活动、时间和两种入口元素(或元素和与元素);$\tau:N\rightarrow\Gamma$ 是将节点标注成某一节点类型的函数;$\Sigma=\{\text{initial},\text{complete},\text{return},\text{urgent},\text{composite}\}$,表示节点状态集合;$\Upsilon:N\rightarrow\Sigma$ 为将节点标注成某一节点状态的函数;$E\subseteq N\times N$ 是模型中节点之间边的集合;C 是定义在该行为模型上的时间约束集合。

1.1 隐式时间约束

实时系统的时间约束[13-14]一般分为两种:显式时间约束和隐式时间约束。隐式时间约束是由实时系统结构控制间的时序关系确定。显式时间约束根据实时系统活动间的时间距离约束确定。

实时系统中状态的变迁是由实时系统的执行控制结构决定的。这种执行时的时序约束没有明确的定义,但在执行过程中必须要满足,因此称为隐式时间约束。为给出隐式时间约束的形式化表达,引入时序算子"@",其中"$\oplus$"、"$\diamond^{+}$"分别代表下一时刻(at the next moment)和未来某一时刻(some time in the future)。下面给出隐式时间约束的形式化表达:

定义 2(隐式时间约束):在行为模型 $B=(N,\Gamma,\tau,\Sigma,\Upsilon,E,C)$ 中,n_i、$n_k\in N$ 任意节点且满足 $\tau(n_i)=\text{Activity}$、$\tau(n_k)=\text{Activity}$,则 $(\text{LB}=n_i\wedge P)\Rightarrow@(Q\wedge\text{LB}=n_k)$。

其中,LB 是控制变量,代表当前系统的状态;P、Q 是一阶逻辑公式,代表系统处于当前状态下为真的条件(文中默认考虑为真的情况);"$\Rightarrow$"代表逻辑蕴含。

式子的直观含义是:控制流 LB 走到 n_i 节点且前置条件 P 为真,那么下一时刻(或未来某一时刻) Q 为真且控制流 LB 转到 n_k 节点。

1.2 显式时间约束

显式时间约束根据实时系统活动间的时间距离约束确定。状态间的时间距离约束:在实时系统中,各不同节点间往往也存在着时间距离约束,其形式化表述如下:

定义 3:在行为模型 $B=(N,\Gamma,\tau,\Sigma,\Upsilon,E,C)$ 中,n_i、$n_k\in N$ 是任意节点且满足 $\tau(n_i)=\text{Activity}$、$\tau(n_k)=\text{Activity}$,$n_i$、$n_k$ 之间存在时间距离约束[MinT,MaxT],即 $\text{MinT}\leqslant E(n_i)-S(n_i)+E(n_{i+j})-S(n_{i+j})+\cdots+E(n_k)-S(n_k)\leqslant\text{MaxT}$(其中 n_{i+j} 为经过 n_i 和 n_k 路径上且在两点间的节点)。

例如,在船舶指控对空作战系统中,从舰空弹目标解算(n_5)到舰空弹发射(n_{10})之间需要在时间区间[65,75]内完成。以 $S(n_5)$ 表示状态节点 n_5 的开始时刻,$E(n_{10})$ 表示状态节点 n_{10} 的结束时刻,节点 n_5 和 n_{10} 间时间距离约束为[65,75],即 n_5 和 n_{10} 两活动节点之间满足:$65\leqslant E(n_{10})-S(n_{10})+E(n_8)-S(n_8)+E(n_5)-S(n_5)\leqslant 75$。

2 基于 Prolog 的模型验证方法

作为一种描述型语言,Prolog 只需描述问题中的

对象和对象关系等已知的事实与规则,确定对象间的逻辑关系。给出待验证的实时系统的事实与规则,作为 Prolog 的数据(即程序,数据与程序是统一的)进行性质验证。

因实时系统在实际中选择与并行(即 AADL 行为模型中的与或元素)结构的出现,对于实时系统在一个周期内多条执行路径的情况,也相应给出了解决方案。下面先介绍路径概念,定义如下:

定义 4:在行为模型 $B=(N,\Gamma,\tau,\Sigma,Y,E,C)$,$B$ 的一条执行路径是指从开始事件到结束事件仅包含活动节点的集合,用 $T=<N_1,N_2,\cdots,N_n>$ 表示,N 表示路径 T 中包含的活动节点的集合。

2.1　AADL 行为模型的分解算法

AADL 行为模型中的节点(N)类型分为活动节点(Activity)、事件节点(Event)、与元素(EXCL)、或元素(INCL)。根据节点类型的不同,模型在经过此类节点时所采用的方式也应有所区别,除开始事件、普通节点与结束事件外,其余类型的节点应遵循以下规则:

·或元素,如图 1 所示。

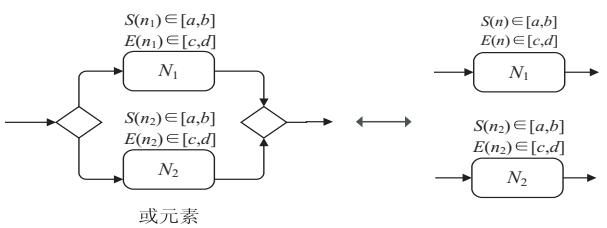

图 1　或元素

或元素中包含的两个活动节点 N_1、N_2,分别对应两条不同的执行路径。

·与元素,如图 2 所示。

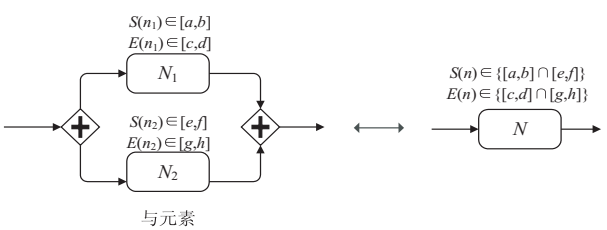

图 2　与元素

与元素要求模型在某一事件内同时执行节点 N_1 与 N_2,可将两活动节点等价地转换成一个活动节点加入到路径中。则新活动节点的开始时间 $S(n)$ 取 $[a,b]$ 和 $[e,f]$ 的交集,结束时间 $E(n)$ 取 $[c,d]$ 和 $[g,h]$ 的交集。

根据以上节点转换规则,得到 AADL 行为模型的执行轨迹集合 $T=\{T_0,T_1,\cdots,T_n\}$。

2.2　AADL 行为模型到 Prolog 的转换

AADL 行为模型到 Prolog 转换的思路是将 AADL 模型分解得到的执行路径集合 T 及每条执行路径中的约束集合转换为 Prolog 语言中的事实,实现用 Prolog 语言来对该 AADL 行为模型的描述。

算法的输入为 AADL 行为模型分解后得到的执行路径集合。用 Facts(B)表示转换成 Prolog 事实的集合。具体算法(Trace_To_Prolog)如下:

输入:执行路径集合 $T=\{T_1,T_2,\cdots,T_n\}$
输出:Facts(B)
For each T_i in T
For each $N_j$in T_i
S_{min} <-GetStartMin(N_j)
S_{max} <-GetStartMax(N_j)
E_{min} <-GetEndMin (N_j)
E_{max} <-GetEndMax(N_j)
S_{statue} <-GetStatue(N_j)
N_{next} <-FindNextActivity(N_j)
If FindN(activity(N_j, N_{next}, S_{min}, S_{max}, E_{min}, E_{max}, S_{statue}), Facts)
Goto next(N_i)
else
Add(Facts, activity(N_j, N_{next}, S_{min}, S_{max}, E_{min}, E_{max}, S_{statue}))
End if
End for
For each C_j in T_i
N_{for} <-FindHeadActivity(C_j)
N_{next} <-FindTailActivity(C_j)
MinTime<-GetMinTime(C_j)
MaxTime<-GetMaxTime(C_j)
If FindC(TimeCons(N_{for}, N_{next}, MinTime, MaxTime), Facts)
Goto next(C_i)
Else
Add(Facts, TimeCons(N_{for}, N_{next}, MinTime, MaxTime))
End if
End for
End for

算法分为两步:首先将执行路径中的各个活动节点及活动节点本身的时间约束转换成 Prolog 事实;其次将 AADL 行为模型的不同活动节点间的时间距离约束转换成 Prolog 事实。

2.3　Prolog 验证规则

通过前面算法转换将 AADL 行为模型转换为 Prolog 事实。对于规则部分,根据实时系统的隐式时间约束与显式时间约束,进而转换为 Prolog 规则。定义 Prolog 规则对其一致性进行验证,提出使用 Prolog 来描述模型中时序限制的转换规则。

规则 1。

通过使用函数 borderlink 及 routelink 对 AADL 行为模型中的状态节点进行时序限制方面的判定。Behavior 事实中给定了相邻两个状态节点间的时序逻

辑,borderlink(A,B)判断给定的两个状态节点 A 和 B 是否满足这种邻接的时序关系(其中 A 和 B 代表两状态节点的编序,后文的 C 和 D 也一样, Z 代表临时变量);routelink(C,D)函数通过递归的思想将这种邻接的时序关系延长,从而判断出给定的状态节点 C 和 D 是否存在时序关系。

```
borderlink( A,B ):-
not( A = B ),
activity( A , B ,_,_,_,_,_).
routelink( C,D ):-
not( C = D ),
activity( C,D ,_,_,_,_,_).
routelink( C,D ):-
activity( C , Z ,_,_,_,_,_).
routelink( Z , D )
```

规则2。

使用 minbordercons(A,B,T)来计算给定的相邻状态节点 A 和 B 之间(包括状态节点 A 和 B)所需要花费的最短时间,并将计算后的结果赋值给 T;minroutecons(A,B,T)函数则通过递归的思想完成不相邻状态节点之间最短时间距离的计算,为计算显式时间约束提供了保障。Prolog 程序如下:

```
minbordercons( A,B,T ):-not( A = B ),
activity( A , B , S1 ,_, E1 ,_,_),
activity( B ,_, S2 ,_, E2 ,_,_),
T is S1 + E1 + S2 + E2
minroutecons( A,B,T ):-
not( A = B ),activity( A , B , S1 ,_, E1 ,_,_),
activity( B ,_, S2 ,_, E2 ,_,_),
T is S1 + E1 + S2 + E2
minroutecons( A,B,T ):-
activity( A , Z , S1 ,_, E1 ,_,_),
minroutecons( Z,B,T1 ),
T is S1 + E1 + T1
```

规则3。

使用 maxbordercons(A,B,T)计算给定的相邻状态节点 A 和 B 之间(包括状态节点 A 和 B)所需要花费的最长时间,并将计算后的结果赋值给 T;maxroutecons(A,B,T)函数则通过递归的思想完成不相邻状态节点之间最长时间距离的计算,为计算显式时间约束提供了保障。Prolog 程序如下:

```
maxbordercons( A,B,T ):-
not( A = B ),activity( A , B , S1 ,_, E1 ,_),
activity( B ,_,_, S2 ,_, E2 ,_),
T is S1 + E1 + S2 + E2
maxroutecons( A,B,T ):-
not( A = B ),activity( A , B , S1 ,_, E1 ,_),
activity( B ,_,_, S2 ,_, E2 ,_),
T is S1 + E1 + S2 + E2
maxroutecons( A,B,T ):-
activity( A , Z ,_, S1 ,_, E1 ,_),
maxroutecons( Z,B,T1 ),
T is S1 + E1 + T1
```

实时系统的时间约束(隐式时间约束和显式时间约束),在 Prolog 中做如下说明:

(1)隐式时间约束。

针对任务关键的实时系统,各个任务的执行应该满足一定的先后关系。结合隐式时间约束的形式化表述:(LB= n_i ∧ P) ⇒ @(Q ∧ LB= n_k),一阶逻辑公式 P 、Q 默认为真,LB 对应 Facts(B)集合中的状态节点,“@”使用规则1中的 borderlink 函数以及 routelink 函数,对算法转换后的 AADL 行为模型进行隐式时间约束的验证,验证代码如下:

```
verify:-
borderlink( A,B ),write( A ),write( 'and' ),write( B ),write
( 'is borderlink' )
routelink( A,B ),write( A ),write( 'and' ),write( B ),
write( 'isroutelink' )
```

(2)显式时间约束。

除了任务间的时序关系,系统任务之间具有一定的时间约束。结合显示时间约束的形式化表述,针对状态间的时间距离约束: MinT ≤ $E(n_i)$ − $S(n_i)$ + $E(n_{i+j})$ − $S(n_{i+j})$ + ⋯ + $E(n_k)$ − $S(n_k)$ ≤MaxT,使用规则2中的 minroutecons(A,B,T)函数和规则3中的 maxroutecons(A,B,T)函数进行显示时间约束的验证。验证代码如下:

```
verify:-
TimeCons( A,B ,MinT,MaxT),
minroutecons( A,B , T1 ),
maxroutecons( A,B , T2 ), T1>MinT,
T2<MaxT,write( A ),
write( B ),write( T1 ),write( '>' ),
write(MintT),write( T2 ),
Write( '<' ),wirte(MaxT).
```

3 实例验证与分析

以船舶自控系统中的对空作战子系统为例,介绍基于 Prolog 并采用 AADL 行为模型描述原模型的验证方法。

文中将 AADL 行为模型进行验证,分析其时间一致性。船舶指控系统中拥有大量的子系统,系统中存在大量的行为流程,这类的行为流程对实时性都有较高的要求。例如,状态活动的开始时间、执行时间以及网络时延等达到秒级甚至毫秒级,所以对于此类实时系统,保证其实时性至关重要。

利用实时系统描述语言,文中描述船舶指控系统中一个典型的实时子系统—对空作战的实时系统,如图 3 所示。

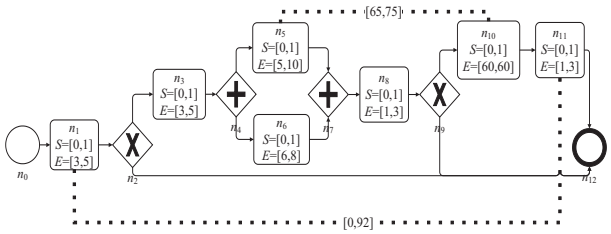

图 3　对空作战实例

从图 3 中可知,该对空作战模型中含有 7 个分别带有各自时间约束的状态节点,根据 2.1 提出的模型分解算法,分解后得到该模型的执行路径集合 T 的表达式为:$\{(n_1,n_3,n_5,n_8,n_{10},n_{11}),(n_1,n_3,n_6,n_8,n_{10},n_{11})\}$。执行路径集合作为输入应用到 2.2 中的模型转换算法中,可以得到 Prolog 的事实(包含活动节点及时间约束)。

将上述 Prolog 事实与 2.3 节中的验证规则作为输入运行在 SWI-Prolog 中,得到表 1 的结果。

表 1　实验结果

行为节点	行为节点	时序条件	实验结果
舰空弹目指(n_3)	目标数据加载(n_6)	$n_3 \Rightarrow \oplus n_6$	True
舰空弹发射(n_{10})	舰空弹目标解算(n_5)	$n_{10} \Rightarrow \diamond^+ n_5$	False
舰空弹目标解算(n_5)	舰空弹发射(n_{10})	[65,75]	[66,76]
舰空弹准备(n_1)	舰空弹停止射击(n_{11})	[0,92]	[74,92]

从表 1 的第一和第二条实验结果可以看出,当行为执行过程不满足实时系统控制结构的内在要求时,系统必定出错,舰空弹目标解算(n_5)必须在舰空弹发射(n_{10})前完成。从第三及第四条实验结果可以看出,虽然对空作战系统在整个行为执行过程中能满足[0,92]的实时性能指标,但在其子过程中,即舰空弹目标解算到舰空弹发射(n_5-n_{10})的时间约束超出既定的时间约束:maxroutecons($E(n_{10})-S(n_{10})+E(n_8)-S(n_8)+E(n_5)-S(n_5)$)>75。

4　结束语

为解决实时系统时间一致性问题,提出了扩展的 AADL 行为附件与两类实时系统时间一致性验证方法,对时间一致性进行了分类描述,并使用逻辑编程语言 Prolog 进行验证,设计了 AADL 行为模型的分解算法及分解后的路径集合到 Prolog 的转换算法,并对实时系统的时间一致性进行了形式化描述。

Prolog 语言通过事实对 AADL 行为模型的行为节点及时间约束进行描述,利用规则刻画实时系统的一致性。随着实时系统的扩充限制在模型上的时间约束形式(隐式时间约束和显式时间约束)也会有所变化,通过 Prolog 语言则不需要改变事实部分,只需要针对变化后的时间约束对规则进行修改。即在新的时间约束情况下,可通过增加相应的规则来实现,将已有的事实和既定的规则分离作为对外提供的事实库与规则引擎并封装其内部的实现,在需要验证或扩展现有的时间约束时调用相应接口,从而实现了与具体实例分离,达到了通用性验证与可扩展性验证的效果。

参考文献:

[1] 刘　静,何积丰,缪淮扣. 模型驱动架构中模型构造与集成策略[J]. 软件学报,2006,17(6):1411-1422.

[2] 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.

[3] 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.

[4] Liu W, Liu S. Research on verification method of AADL behavior model based on BIP[C]//International conference on computational & information sciences. [s. l.]:[s. n.],2013:1987-1990.

[5] 刘　博,李蜀瑜. 基于 NuSMV 的 AADL 行为模型验证的探究[J]. 计算机技术与发展,2012,22(2):110-113.

[6] 杨志斌,胡　凯,赵永望,等. 基于时间抽象状态机的 AADL 模型验证[J]. 软件学报,2015,26(2):202-222.

[7] 刘　铭,张国印,姚爱红,等. 基于层次实时有色 Petri 网的实时系统建模与分析方法研究[J]. 电子与信息学报,2011,33(3):580-586.

[8] Maciel P, Barros E. Capturing time constraints by using Petri-nets in the context of hardware/software codesign[C]//International workshop on rapid system prototyping. [s. l.]:IEEE,1996:36-41.

[9] 周　航,黄志球,祝　义,等. 基于 Time Petri Net 的实时系统冲撞检测与消解[J]. 计算机研究与发展,2012,49(2):413-420.

[10] Frana R B, Bodeveix J P, Filali M, et al. The AADL behaviour annex - experiments and roadmap[C]//International conference on engineering complex computer systems. [s. l.]:IEEE,2007:377-382.

[11] Shapiro E, Garrett R. The art of prolog[J]. IEEE Expert,1987,2(2):106-107.

[12] 陈文宇. 形式语言与自动机理论若干问题研究[D]. 成都:电子科技大学,2009.

[13] 杨　林. 基于 UML 的实时系统建模及顺序图时间约束研究[D]. 长沙:湖南大学,2007.

[14] 白晓颖,汪　明,陆　皓,等. 实时系统时间约束验证[J]. 清华大学学报:自然科学版,2012,52(9):1286-1292.