ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.1, January 2006, pp.48−58
DOI: 10.1360/jos170048

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于场景构件式实时软件设计的一致性检验*

胡 军[1,2+], 于笑丰[1,2], 张 岩[1,2], 李宣东[1,2], 郑国梁[1,2]

[1](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

[2](南京大学 计算机科学与技术系,江苏 南京 210093)

Scenario-Based Consistency Verification of Component-Based Real-Time System Designs

HU Jun[1,2+], YU Xiao-Feng[1,2], ZHANG Yan[1,2], LI Xuan-Dong[1,2], ZHENG Guo-Liang[1,2]

[1](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

+ Corresponding author: Phn: +86-25-83593671, Fax: +86-25-83594683, E-mail: hujun@seg.nju.edu.cn, http://www.nju.edu.cn

Hu J, Yu XF, Zhang Y, Li XD, Zheng GL. Scenario-Based consistency verification of component-based real-time system designs. *Journal of Software*, 2006,17(1):48−58. http://www.jos.org.cn/1000-9825/17/48.htm

Abstract: For real-time software systems, this paper considers the problem of checking component-based designs for timing scenario-based specifications, which is one of the challenges in real-time computing domain. Firstly the timing scenario-based specifications are specified by UML sequence diagrams with a set of boolean expressions, then the interface automata for modeling real time systems through adding time intervals on the actions is extened. The component-based designs are modeled by a real-time interface automaton network which contains a set of real-time interface automata synchronized by shared actions. Based on analyzing the compatible integer state space of a real-time interface automata network, a corresponding reachability graph is constructed and finally an algorithm for checking the consistency between the real-time component-based designs and the timing scenario-based specifications is developed.

Key words: real-time software; component-based design; model checking; interface automata; sequence diagrams; unified modelling language

摘 要: 在复杂的实时软件系统中使用构件式设计方法,已成为目前软件工程中的研究热点.如何有效地验证实时软件的设计是否满足给定的时间规约,是实时计算领域中的主要挑战之一.通过在接口自动机模型中添加时间区间标记,来扩展其对实时系统接口行为的表达能力;使用实时接口自动机网络来描述实时软件系统的构件式设计模型;使用带布尔不等式时间约束的 UML 顺序图表示基于场景的需求规约,对系统设计阶段实时软件构件的动态行为进行形式化分析与检验.通过对实时接口自动机网络状态空间的分析,构造了其可兼容的整型状态等价类空间的可达图,并在此基础上给出了验证算法,以检验构件式实时软件系统的设计与带时间约束的场景式规约之间的一致性.

* Supported by the National Natural Science Foundation of China under Grant Nos.60425204, 60233020, 60273036 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312001 (国家重点基础研究发展规划(973)); the Natural Science Foundation of Jiangsu Province of China under Grant Nos.BK2003408, BK2004080 (江苏省自然科学基金)

Received 2005-04-30; Accepted 2005-08-25

关键词: 实时软件;构件式设计;模型检验;接口自动机;顺序图;统一建模语言
中图法分类号: TP311 文献标识码: A

实时系统是一类设计、实现和验证工作都非常复杂的系统,其系统行为与时间紧密相关.近年来,在实时系统领域中,随着系统中软件规模和复杂度的迅速增加,整个系统的质量和可靠性极大地依赖于其软件系统的实时行为.如何有效地验证实时软件的设计是否满足给定的时间规约,是实时计算领域中的主要挑战之一.构件式软件系统设计方法目前已成为软件工程中的研究热点.它通过设计和复用软件模块来构建复杂的软件系统,提高开发效率和软件质量.系统的整体行为通过构件之间的交互来完成,这种交互通常描述为某种基于场景的规约.在复杂的实时软件系统设计中使用构件式设计方法,是提高实时系统开发效率和可靠性的有效途径之一.

接口自动机(interface automata,简称 IA)[1]是一种基于自动机的形式化语言,可以用来描述软件构件接口动态行为的某些性质,包括构件对外界环境的输入假设和输出保证.统一建模语言(unified modeling language,简称 UML)[2]是目前软件工业界的一个可视化的标准建模语言.通常在构件式系统设计中,可以使用 UML 顺序图描述用户的需求规约.它通过构件之间的消息发送和接收序列来表达系统中的一个交互场景.本文的工作是在 IA 模型中添加时间区间标记,以扩展其对实时系统中构件接口行为的表达能力.使用实时接口自动机网络(real-time interface automaton networks,简称 RIAN)来描述实时软件系统的构件式设计模型;使用带布尔不等式时间约束的 UML 顺序图表示基于场景的需求规约,对系统设计阶段软件构件的动态行为进行形式化分析和检验.通过对 RIAN 状态空间的分析,构造其可兼容的整型状态等价类空间的可达图,并在此基础上给出了验证算法,以检验构件式实时软件系统的设计与带时间约束的场景式规约说明之间的一致性.

本文第 1 节给出带时间约束不等式的 UML 顺序图模型的形式化描述.第 2 节给出实时接口自动机的形式化定义以及作为系统组合行为描述的 RIAN.第 3 节对 RIAN 状态空间进行分析,给出实时行为一致性的检验算法.最后是结束语和相关工作.本文中,为叙述方便,凡提到实时系统均指实时软件系统.

1 带实时约束的 UML 顺序图模型

在构件式系统设计中,可以使用 UML 顺序图模型描述系统运行过程中的一个行为场景.它通过构件之间的消息发送和接收的序列,来表达在这个场景中这些构件是如何进行交互的.图 1 所示为一个简单的带时间约束不等式的 UML 顺序图模型.它描述了机载防撞告警系统(TCAS)中一个可能的飞行冲突探测场景.

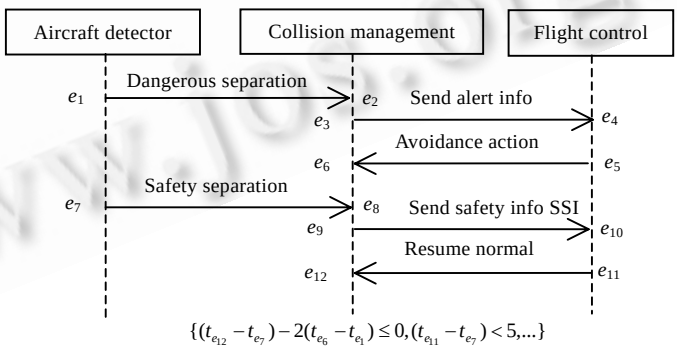

Fig.1 A simple UML sequence diagram for the TCAS scenario

图 1 TCAS 系统中飞行冲突探测场景的 UML 序列图

顺序图中的消息发送以及消息接收使用事件来表示.每个顺序图的语义可以表示为一个或多个可视事件对的序列[3].在图中的两个事件 e_1,e_2 当出现满足下述关系时,将事件 e_1 视为在事件 e_2 之前发生,并记为一个可视事件对(e_1,e_2):

- 因果关系:e_1 为消息发送事件,e_2 为此消息对应的接收事件;

- 控制关系:在同一个构件的生命线轴上,事件 e_1 出现于事件 e_2 的上方,且 e_2 是消息发送事件;
- FIFO 顺序关系:在同一个构件的生命线轴上,接收事件 e_1 出现于接收事件 e_2 的上方,并且相对应的发送事件 e'_1 和 e'_2 同时出现在另外一个构件的生命线轴上,且 e'_1 位置高于 e'_2.

带实时约束的 UML 顺序图模型(以下简称顺序图)的形式化定义如下:

定义 **1**. 一个顺序图是六元组 $D=(C,E,M,F,W,S)$,其中:

- C:有穷的构件集;
- E:有穷的事件集;
- M:有穷的消息集;对每一个消息 $m\in M$,分别使用 $m!$和 $m?$来表示消息 m 的发送和接收事件;
- $F:E\mapsto C$ 是一个标记函数,将每一个事件 $e\in E$ 分配给一个构件 $F(e)\in C$;
- W:元素形式为(e,e')的有穷集,其中 $e,e'\in E$ 且 $e\neq e'$,表示 D 中的每一个可视事件对.
- S:有穷的布尔表达式的集合,表示顺序图中的时间条件约束.

S 中的布尔表达式形如: $c_0(t_{e_0}-t_{e'_0})+c_1(t_{e_1}-t_{e'_1})+...+c_n(t_{e_n}-t_{e'_n})\sim b$;其中 $t_{e_0},t_{e'_0},t_{e_1},t_{e'_1},...,t_{e_n},t_{e'_n}$ 表示相应事件发生的时间值,$c_0,c_1,\ldots,c_n,b$ 为实数,且 $b\neq\infty$,$\sim\in\{\leq,<\}$.此外,对任意一个事件 $e\in E$,都对应一个消息 $m\in M$ 的发送和接收事件,我们用 $\phi(e)=m!$或者 $\phi(e)=m?$来表示.

顺序图刻画了系统运行的一个场景,其运行过程就表现为可视事件的一个序列 $e_0\wedge e_1\wedge\ldots\wedge e_m$,其中事件 e_{i+1} 在事件 e_i 之后发生$(0\leq i\leq m-1)$.由于在控制关系中,可能存在不确定的接收消息次序,因此一个顺序图允许有多个事件序列.

定义 **2**. 事件序列 $e_0\wedge e_1\wedge\ldots\wedge e_m$ 是顺序图 $D=(C,E,M,F,W,S)$的一个有效事件序列当且仅当以下条件满足:

- $\{e_0,e_1,\ldots,e_m\}=E$,且对于任意 $i,j(i\neq j,0\leq i,j\leq m)$,有 $e_i\neq e_j$;
- $e_0,e_1,\ldots,e_m$ 满足 W 中定义的可视事件对,即:对于任何 e_i 和 e_j,当$(e_i,e_j)\in W$ 时,有 $0\leq i<j\leq m$.

我们使用带时间戳的事件序列来表示实时系统的行为.序列中每个事件都带有一个相对时间的标记,形如:$(e_0,\tau_0)\wedge(e_1,\tau_1)\wedge\ldots\wedge(e_m,\tau_m)$,其中 e_i 表示事件,相对时间戳 τ_i 是一个非负实数.其语义为:事件 e_0 在系统启动后的第 τ_0 个时间单位处发生;事件 e_1 在事件 e_0 发生之后的第 τ_1 个时间单位处发生;序列中其余的元素如此类推.注意,定义 1 的时间约束不等式中 t_{e_i} 则是表示事件 e_i 发生的绝对时间,即: $t_{e_i}=\sum_{j=0}^{i}\tau_j(0\leq i\leq m)$.

定义 **3**. 带时间戳的事件序列$\sigma=(e_0,\tau_0)\wedge(e_1,\tau_1)\wedge\ldots\wedge(e_m,\tau_m)$是顺序图 $D=(C,E,M,F,W,S)$的一个行为当且仅当以下条件满足:

- $(e_0\wedge e_1\wedge\ldots\wedge e_m)$是顺序图 D 的一个有效事件序列;
- $\tau_0,\tau_1,\ldots,\tau_m$ 满足 D 中的时间约束.即:对于 S 中的任意一个不等式 $\sum_{i=0}^{n}c_i(t_{f_i}-t_{f'_i})\sim b$,都有不等式 $\sum_{i=0}^{n}c_iA_i\sim b$ 成立,其中,对任何 $i(0\leq i\leq n)$,当 $f_i=e_j$, $f'_i=e_k$ 时, $A_i=\begin{cases}\tau_{k+1}+\tau_{k+2}+...+\tau_j(j>k)\\-(\tau_{j+1}+\tau_{j+2}+...+\tau_k)(j<k)\end{cases}$.

2 实时接口自动机与构件式实时系统设计

2.1 实时接口自动机(RIA)

接口自动机(IA)是用来刻画软件构件接口时序特征的一种形式化语言.它描述了构件使用时对外界环境的输入假设和输出保证,如:构件内方法被调用的先后次序以及构件向环境输出调用信息或结果的次序.IA 模型采用乐观的(optimistic)构件组合兼容性思想[1],认为构件的设计总是基于一定的环境假设:当构件组合时,它们的环境假设也应该同时组合在一起;只要存在某个合法环境使组合假设得以满足,则可兼容性成立.IA 模型明确地表示了当前系统状态上只有哪些输入动作是可接收的;而其他的输入动作是不可接收的,即为非法的输入.在本节中,我们对 IA 进行实时方面的扩展,以满足其对实时系统的描述,并称扩展之后的模型为实时接口自动机(real-time interface automata,简称 RIA).具体来看,在接口自动机的每一个转换上面添加相应的时间区间约束,以

表示此转换发生的最小、最大时限.RIA 的形式化定义如下:

定义 **4**. 实时接口自动机(RIA)是一个五元组: $P=(V_P,v_P^{init},A_P,I_P,\Gamma_P)$,其中:

- V_P 为有穷状态集,其中每一个状态为 $v\in V_P$;
- v_P^{init} 为初始状态, $v_P^{init}\in V_P$;
- A_P 为有穷的动作集,包括输入、输出和内部动作集: A_P^I, A_P^O 和 A_P^H,且两两互不相交;
- I_P 为有穷的时间区间集;每一个区间都形如$[x,y]$,其中 x,y 为非负整数$(x\le y)$,y 可以取值为∞;
- $\Gamma_P\subseteq V_P\times A_P\times I_P\times V_P$ 为有穷的转换集.

当动作 $a\in A_P^I, A_P^O$ 或者 A_P^H 时,相应的转换$(v,a,[x,y],v')$分别称为输入、输出或内部转换;对于某一状态 $v\in V_P$,当存在转换$(v,a,[x,y],v')\in\Gamma_P$ 时,称动作 $a\in A_P$ 在此状态上是可激活的.IA 模型并不要求对任一状态 $v\in V_P$,都满足 $A_P^I=A_P^I(v)$;即:并非在每一个状态上,所有的动作都是可激活的.

我们用带时间戳的状态转换序列表示 RIA 的行为: $v_0\xrightarrow{a_0,\tau_0}v_1\xrightarrow{a_1,\tau_1}\ldots\xrightarrow{a_{m-1},\tau_{m-1}}v_m\xrightarrow{a_m,\tau_m}v_{m+1}$.表示系统从状态 v_0 开始,经过 τ_0 个时间单位后,通过动作 a_0 转换到状态 v_1,并在 v_1 上停留 τ_1 个时间单位,又通过动作 a_1 转换到状态 v_2;如此类推.RIA 的行为定义如下:

定义 **5**. 一个带时间戳的状态转换序列 $v_0\xrightarrow{a_0,\tau_0}v_1\xrightarrow{a_1,\tau_1}\ldots\xrightarrow{a_{n-1},\tau_{n-1}}v_n\xrightarrow{a_n,\tau_n}v_{n+1}$ 是实时接口自动机 $P=(V_P,v_P^{init},A_P,I_P,\Gamma_P)$ 的一个行为当且仅当以下条件满足:

- $v_0=v_P^{init}$,且对每一个 $i(0\le i\le n)$,有$(v_i,a_i,[x_i,y_i],v_{i+1})\in\Gamma_P$;
- $\tau_0,\tau_1,\ldots,\tau_n$ 满足Γ_P 中转换的时间约束,即:对任何$(v_i,a_i,[x_i,y_i],v_{i+1})$,有 $x_i\le\tau_i\le y_i$.

2.2 实时接口自动机网络(RIAN)

我们使用实时接口自动机网络(RIAN)作为构件式实时系统的设计模型.系统中每个构件的行为都表示为相应的 RIA.整个系统由一个接口自动机集组成,并通过构件接口之间共享动作来进行同步组合.图 2 所示即为 TCAS 系统的 RIAN,它由 3 个构件的 RIA 组成:冲突探测器构件 RIA、飞行控制构件 RIA 和冲突管理构件 RIA.接口自动机组合的相关细节可参看文献[1].RIAN 的形式化定义如下:

定义 **6**. 实时接口自动机网络(RIAN)是一个二元组 $N=(K,Z)$,其中:

- $K=\{P_1,P_2,\ldots,P_n\}$为可组合的实时接口自动机集,其中 $P_i=(V_{P_i},v_{P_i}^{init},A_{P_i},I_{P_i},\Gamma_{P_i})\ (0\le i\le n)$;
- $Z=\{a\ shared\ (P_i,P_j)\mid 1\le i,j\le n,i\ne j\}$,为所有的共享动作集,其中任何 P_i 和 P_j 的共享动作集为 $shared(P_i,P_j)=A_{P_i}\cap A_{P_j}=(A_{P_i}^O\cap A_{P_j}^I)\cup(A_{P_i}^I\cap A_{P_j}^O)\ (1\le i,j\le n,i\ne j)$.

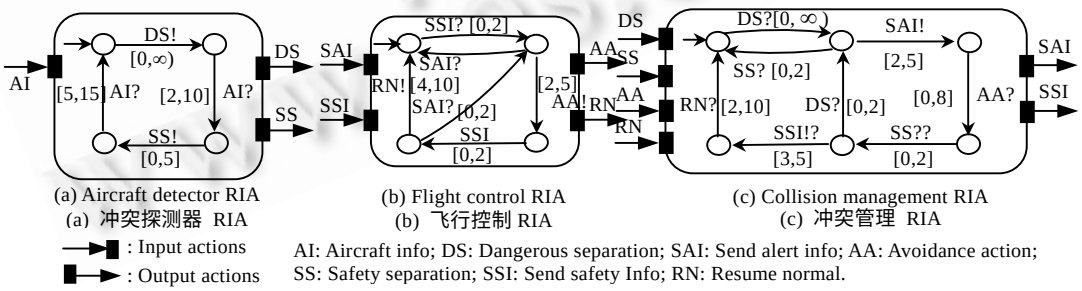

Fig.2 A real-time interface automaton networks for TCAS system

图 2 TCAS 系统的实时接口自动机网络图

RIAN 的状态集、动作集等定义如下:

定义 **7**. $N=(K,Z)$是一个 RIAN,其中 $K=\{P_1,P_2,\ldots,P_n\}$,且 $P_i=(V_{P_i},v_{P_i}^{init},A_{P_i},I_{P_i},\Gamma_{P_i})\ (0\le i\le n)$;

- N 的一个无时间状态为: $\bar{v}=(v_1,v_2,...,v_n)$, $(v_i\in V_{P_i},1\le i\le n)$;即: $\bar{v}\in(V_{P_1}\times V_{P_2}\times...\times V_{P_n})$, 其中无时间的初始

状态为 $v_N^{init}=(v_{P_1}^{init},v_{P_2}^{init},...,v_{P_n}^{init})$;

- N 的每个状态 u 都是一个形如 $(\bar{v},c)$ 的二元组,其中: $\bar{v}=(v_1,v_2,...,v_n)$ 是 N 的一个无时间状态; $c:\{v_i\mid 1\le i\le n\}\mapsto R^+\cup\{0\}$ 是一个时钟函数(R^+表示正整数集),它将每一个 v_i 映射为一个非负整数用以表示相对 P_i 而言系统在其状态 v_i 上已经停留了 $c(v_i)$个时间单位.N 的初始状态为 $u_N^{init}=(\bar{v}_N^{init},c^{init})$,其中: $\bar{v}_N^{init}$ 是 N 的无时间初始状态,且 $c^{init}(\bar{v}_{p_i}^{init})=0$ $(0\le i\le n)$.N 的状态集合我们用 U_N 来表示.
- N 的动作集 $A_N=A_N^I\cup A_N^O\cup A_N^H$;输入动作集 $A_N^I=\left(\bigcup_{1\le i\le n}A_{P_i}^I\right)-Z$;输出动作集 $A_N^O=\left(\bigcup_{1\le i\le n}A_{P_i}^O\right)-Z$;内部动作集为 $A_N^H=\left(\bigcup_{1\le i\le n}A_{P_i}^H\right)\cup Z$;
- N 的时间区间集合为 $I_N=\bigcup_{1\le i\le n}I_{P_i}$.

RIAN 的转换和行为定义如下:

定义 **8**. $N=(K,Z)$是一个 RIAN,其中 $K=\{P_1,P_2,\ldots,P_n\}$,且 $P_i=(V_{P_i},v_{P_i}^{init},A_{P_i},I_{P_i},\Gamma_{P_i})$ $(0\le i\le n)$;设 $u=(\bar{v},c)$ 和 $u'=(\bar{v'},c')$ 为 N 的两个不同的状态,其中: $\bar{v}=(v_1,v_2,...,v_n)$, $\bar{v}'=(v_1',v_2',...,v_n')$.当满足以下条件之一的时候,系统 N 可以通过动作 a 从状态 $\bar{v}$ 到达状态 $\bar{v}'$,用 $\bar{v}\xrightarrow{a,d}\bar{v}'$ 来表示:

- 对于一个动作 $a\notin Z$,在 $P_k(1\le k\le n)$中存在一个转换 $(v_k,a,[x_k,y_k],v_k')\in\Gamma_k$, 且满足 $x_k\le c(v_k)+d\le y_k$, $c'(v_k')=0$.同时,对任何的 $i(i\ne k,1\le i\le n)$,有 $v_i=v_i'$, $c'(v_i)=c(v_i)+d$;
- 对于一个动作 $a\in shared(P_i,P_j)(1\le i,j\le n,i\ne j)$,在 P_i 中存在一个转换$(v_i,a!,[x_i,y_i],v'_i)\in\Gamma_i$($a!$表示 a 是输出动作),在 P_j 中存在一个转换$(v_j,a?,[x_j,y_j],v'_j)\in\Gamma_j$($a?$表示 a 是输入动作),且满足 $x_i\le c(v_i)+d\le y_i, x_j\le c(v_j)+d\le y_j$, $c'(v_i')=0$ 和 $c'(v_j')=0$.同时,对任何的 $k(k\ne i,\ j,1\le k\le n)$,有 $v_k=v_k'$, $c'(v_k)=c(v_k)+d$.

N的一个行为用带时间戳状态转换序列来表示,形如: $u_0\xrightarrow{a_0,d_0}u_1\xrightarrow{a_1,d_1}\ldots\xrightarrow{a_{n-1},d_{n-1}}u_n\xrightarrow{a_n,d_n}u_{n+1}$.其中,$u_0$是 N 的初始状态.

3 检验基于场景规约的构件式实时系统设计

现在,构件式实时系统设计建模为一个实时接口自动机网络(RIAN),场景式规约使用带时间约束的顺序图来表示.我们所关心的问题就是:检验构件式实时系统设计模型是否满足顺序图所表示的场景规约.这需要对 RIAN 行为中的事件序列和顺序图中的事件序列进行比较.为此,以下给出相关行为的轨迹以及事件序列之间的投影关系.

设状态转换序列 $\zeta=u_0\xrightarrow{a_0,d_0}u_1\xrightarrow{a_1,d_1}\ldots\xrightarrow{a_{n-1},d_{n-1}}u_n\xrightarrow{a_n,d_n}u_{n+1}$ 是 N 的一个行为.其相应的带时间戳动作序列为$(a_0,d_0)\wedge(a_1,d_1)\wedge\ldots\wedge(a_n,d_n)$,替换其中的每一个内部动作$(a_i,d_i)(0\le i\le n)$为一对输出和输入动作$(a_i!,d_i)\wedge(a_i?,0)(0\le i\le n)$,其余的保持不变.这样我们就得到一个只有输入/输出动作形成的带时间戳的序列.在本文的验证问题中,所谓一个输入/输出的动作 a,事实上就是一个输入/输出事件 e.这样,可以得到一个带时间戳的且只包含输入/输出事件的序列,形如:$(e_0,\tau_0)\wedge(e_1,\tau_1)\wedge\ldots\wedge(e_r,\tau_r)(r\ge n)$,称其为行为$\zeta$的轨迹.此外,注意到顺序图中的消息实际上就对应于 N 中的某一个共享动作;消息的发送/接收事件就对应于某个构件的输出/输入动作.只是在顺序图的形式化定义 $D=(C,E',M,F,W,S)$中,对于任一个 $e'\in E'$,是一个事件的名字标记,事件的真正内容则是 $\phi(e')$;而在 RIA 中的输入/输出动作就是真正的事件;因此在进行事件比较时,实际上我们需要比较$\phi(e')$和 e.

设σ是 N 的一个行为ζ的轨迹,σ_1 是σ的任意一个子串,形如$(e_p,\tau_p)\wedge(e_{p+1},\tau_{p+1})\wedge\ldots\wedge(e_m,\tau_m)$.对于一个有效事件序列形如 $f_0\wedge f_1\wedge\ldots\wedge f_n$ 的顺序图 D,如果存在 $e_{k_0},e_{k_1},...,e_{k_n}$ 同时满足以下条件:(1) $p=k_0<k_1<\ldots<k_n=m$;(2) 对于所有的 $i(0\le i\le n)$,有 $\phi(f_i)=e_{k_i}$;(3) 对于任何的 $i(0\le i\le n)$,任何的 $j\ne k_q\ (p\le j\le m,0\le q\le n)$,有 $\phi(f_i)\ne e_j$,则称子串σ_1是轨迹σ中关于 D 的一个投影.事实上,子串σ_1正好表示了顺序图 D 所描述的场景(以下简称为场景 D)

在 N 的行为ζ中的一个出现.

定义 **9**. N 为一个实时接口自动机网络(RIAN),其满足顺序图 $D=(C,E,M,F,W,S)$当且仅当以下条件同时成立:

- 场景 D 在 N 中的一个行为中出现;
- 在 N 中任一个行为的轨迹中,其关于 D 的投影(不妨设其形如:$(e_k,\tau_k)\wedge(e_{k+1},\tau_{k+1})\wedge\ldots\wedge(e_m,\tau_m)$)均满足 D 中时间约束,即:对于 S 中任何布尔不等式 $\sum_{i=0}^{n}c_i(t_{f_i}-t_{f_i'})\sim b$,都有一个不等式 $\sum_{i=0}^{n}c_i\Delta_i\sim b$ 成立.其中:当 $\phi(f_i)=e_u$, $\phi(f_i')=e_v$ $(k\le u,v\le m)$ 时,有 $\Delta_i=\begin{cases}\tau_{v+1}+\tau_{v+2}+\ldots+\tau_u(u>v)\\-(\tau_{u+1}+\tau_{u+2}+\ldots+\tau_v)(u<v)\end{cases}$.

显然,一个 RIAN 的状态空间是无穷的,其行为也是无穷的,我们需要找到一个与此检验问题等价的有穷形式的验证方式.下面引入整型状态与整型行为.

3.1 构造整型状态空间的可达图

设 $\zeta=u_0\xrightarrow{a_0,d_0}u_1\xrightarrow{a_1,d_1}\ldots\xrightarrow{a_{n-1},d_{n-1}}u_n\xrightarrow{a_n,d_n}u_{n+1}$ 是 N 的一个行为,当满足对所有的 $i(0\le i\le n)$,d_i 都是一个整数时,则称ζ为 N 的一个整型行为.当整型行为中的每一状态$((v_1,v_2,\ldots,v_m),c)$都满足 $c(v_j)(1\le j\le m)$是一个整数时,则称其为整型状态;仅由整型状态构成的系统空间称为 N 的整型状态空间.

定理 **1**. N 为一个实时接口自动机网络(RIAN),它满足顺序图 $D=(C,E,M,F,W,S)$当且仅当以下条件同时成立:

- 场景 D 在 N 中的一个整型行为中出现;
- 在 N 中任一个整型行为的轨迹中,其关于顺序图 D 的投影均满足其中的时间约束.

由于篇幅原因,定理证明从略.由定理 1 可知,对于本文的验证问题,只需考虑 RIAN 的整型状态空间和整型行为.但由于整数集仍然是无穷集,N 的整型行为仍然是无穷的,还需要进一步处理.

考虑到 N 中所有的时间区间约束可分成两种形式:$[x,y](x\le y,y\ne\infty)$和$[x,\infty)$,且 N 中的时间区间约束集 I_N 为有穷,定义一个关于 N 的时间边界 K:$K=\max\{X_{\max},Y_{\max}\}$,其中:$X_{\max}=\max\{x_i\mid[x_i,\infty]\in I_N\}$,$Y_{\max}=\max\{y_i\mid[x_i,y_i]\in I_N,(x_i\le y_i,y_i\ne\infty)\}$.基于 N 的时间边界 K,在 N 的状态集 U_N 上定义一个时间区域关系 R_t 如下:

$$R_t=\left\{\langle u,u'\rangle\middle|\begin{array}{l}\bar{v}=\bar{v'};\text{并且对于所有的}v_i(1\le i\le n),\text{有}\\c(v_i)=c'(v_i)\text{或者}c(v_i)>K\wedge c'(v_i)>K\end{array}\right\}.$$

很容易验证时间区域关系 R_t 满足自反、对称、传递性,因此是一个在 U_N 上的等价关系.U_N 的商集 U_N/R_t 则是 U_N 的一个有穷划分.商集 U_N/R_t 中所包含的等价类是有穷的,虽然其中某些等价类仍是无穷集.我们使用$[u]$来表示包含状态 u 在内的等价类,即:$[u]=\{u'\mid uR_tu'\}$;使用 $u\equiv u'$表示两个状态等价;使用$[N]$表示由 R_t 确定的有穷等价类集所构成的系统.由 R_t 的定义,可以得到如下定理:

定理 **2**. N 为一个实时接口自动机网络(RIAN),u_1 和 u_2 是其两个满足时间区域等价关系的整型状态,即 $u_1\equiv u_2$,则有:存在一个整型状态 u'满足 $u_1\xrightarrow{a,d}u_1'$ 的充要条件是存在一个整型状态 u_2' 满足 $u_2\xrightarrow{a,d}u_2'$ 且 $u_1'\equiv u_2'$.

上面定理表明:$[N]$中的整型行为与 N 中的整型行为是一致的.基于定理 1 和定理 2,按照以下构造方法可以得到 N 的一个有穷可达图 $G=(V_G,L_G)$:(1) 对于 N 中的起始状态 u_N^{init},$[u_N^{init}]$ 是 V_G 中的节点,称其为 G 的起始节点;(2) 对于 V_G 中的任一个节点$[u]$,当 N 中存在 $u\xrightarrow{a,d}u'$ $(d\in[0,K+1])$时,$[u']$也是 V_G 中的一个节点;且 $[u]\xrightarrow{a,d}[u']$成为 L_G 中的一条边 l.当 $a\in A_N^I,A_N^O$ 或者 A_N^H 时,分别称为输入边、输出边或内部边,相应的边集合分别用 L_G^I, L_G^O 和 L_G^H 来表示.

在构造可达图 G 时,考虑到,当出现 $[u]\xrightarrow{a,K+1}[u']$时,表明在 v_i(其中 $v_i\in\bar{v}\wedge(\bar{v},c)\in[u]$)上存在一个可激活的动作,且其相应的时间区间约束形如 $[x,\infty)$;这意味着同时有无穷多个形如 $[u]\xrightarrow{a,d'}[u']$ 的成立,其中,$d'\in[K+2,\infty]$.这样,在可达图 G 中,$[u]$和$[u']$之间将存在无穷多条边.为保证边集为有穷,在这种情况下,构造可达图

过程中,我们只选取$[u]\xrightarrow{a,K+1}[u']$作为无穷条边集$[u]\xrightarrow{a,d'}[u']$($d'\in[K+1,\infty]$)的代表.

一般地,可达图 G 的任一条路径$\rho=l_0\wedge l_1\wedge\ldots\wedge l_n(l_i\in L_G,0\le i\le n)$定义为从 G 的起始节点开始的一个边的序列.ρ的子路径定义为形如 $l_i\wedge l_{i+1}\wedge\ldots\wedge l_{i+k}(0\le i\le n-k)$的边序列.显然,路径$\rho$也可以用一个带时间戳的状态转换序列来表示,其形如: $\rho=[u_0]\xrightarrow{a_0,d_0}[u_1]\xrightarrow{a_1,d_1}\ldots\xrightarrow{a_{n-1},d_{n-1}}[u_n]\xrightarrow{a_n,d_n}[u_{n+1}]$,其中:$[u_0]$是 G 的起始节点.其相应的带时间戳的动作序列为$(a_0,d_0)\wedge(a_1,d_1)\wedge\ldots\wedge(a_n,d_n)$.采用前面同样的方法,替换其中的每一个内部动作$(a_i,d_i)(0\le i\le n)$为一对输出和输入动作$(a_i!,d_i)\wedge(a_i?,0)(0\le i\le n)$,其余的保持不变;并将动作视为事件.最后我们可以得到一个带时间戳的事件序列,形如:$(e_0,\tau_0)\wedge(e_1,\tau_1)\wedge\ldots\wedge(e_s,\tau_s)(s\ge n)$;称其为路径$\rho$的轨迹$\sigma_\rho$.这样,路径$\rho$的每条边上的标记要么对应于一个事件,要么对应于一对输入/输出事件.我们用$\psi(l_k)$来表示在边 l_k 上的事件对.此外,用$\sigma(l_i,l_j)$来表示子路径 $l_i\wedge l_{i+1}\wedge\ldots\wedge l_j(i\le j)$的轨迹;路径 ρ 的轨迹也可以表示为$\sigma(l_0,l_n)$.

3.2 消除整型空间中的非法状态

定义 4 表明:在每一个状态上,并不是所有的输入动作都是可以激活的.因此,在 RIAN 中可能会存在非法状态.所谓非法状态,即表示在任意两个具有共享动作的 RIA 之间,在输入和输出动作的交互次序上的矛盾.其形式化定义如下:

定义 **10**. $N=(K,Z)$是一个 RIAN,其中 $K=\{P_1,P_2,\ldots,P_n\}$; $P_i=(V_{P_i},v_{P_i}^{init},A_{P_i},I_{P_i},\Gamma_{P_i})$ $(0\le i\le n)$;其非法状态集合为

$$illegal(N)=\left\{((v_1,v_2,...,v_n),c)\in N\left|\begin{array}{l}\exists v_i\in V_{P_i},\exists v_j\in V_{P_j}(i\ne j;1\le i,j\le n),\exists a\in shared(P_i,P_j)\\ \left(\begin{array}{l}a\in A_{P_i}^O(v_i)\wedge a\notin A_{P_j}^I(v_j)\\ \vee\\ a\in A_{P_j}^O(v_j)\wedge a\notin A_{P_i}^I(v_i)\end{array}\right)\end{array}\right.\right\}.$$

由定理 2 和非法状态的定义,很容易得到:当 $u\in illegal(N)$时,$[u]$一定是一个非法的时间区域等价类;在非法的时间区域等价类$[u]$中的所有状态都是非法状态.在可达图 G 中可能存在由非法时间区域等价类构成的节点集,我们使用 $illegal(V_G)$来表示.其他节点称为可兼容等价类状态节点.

基于可达图 G,使用文献[1]中逆向可达性算法,可以计算$[N]$中可兼容等价类状态集$\pi_{[N]}$.基本思想为:在 V_G 上定义算子 $OH_{pre}:2^{V_G}\mapsto 2^{V_G}$:对于集合$\Lambda\subseteq V_G$,有 $OH_{pre}(\Lambda)=\{\mu\in V_G\mid\exists(\mu\xrightarrow{a,d}\upsilon)\in L_G^O\cup L_G^H;\upsilon\in\Lambda\}$.然后计算 OH_{pre} 的最大不动点,Λ的初值赋为 $illegal(V_G)$.最后,$\pi_{[N]}=V_G\backslash Fixpoint(OH_{pre}(\Lambda))$.当$\pi_{[N]}$为空集时,表示不存在任何合法环境使得 N 在运行过程中能够避免进入那些非法状态.显然,在这种情况下,本文所讨论的一致性问题回答为否.当$\pi_{[N]}$不为空集时,表明至少存在一个合法环境使得 N 能够正常工作.因此下面只考虑$\pi_{[N]}$不为空集的情况.我们使用 $com([N])$表示只包含可兼容等价类状态集$\pi_{[N]}$的系统,那么由定理 1、定理 2 以及上述讨论可知,对于本文的一致性检验问题,只需对 $com([N])$中的整型行为进行检验即可.

定理 **3**. N为一个实时接口自动机网络(RIAN),其满足顺序图$D=(C,E,M,F,W,S)$当且仅当以下条件同时成立:

- 场景 D 在 $com([N])$中的一个整型行为中出现;
- 在 $com([N])$中任意一个整型行为的轨迹中,它关于顺序图 D 的投影均满足其中的时间约束.

在可达图G中只保留可兼容等价类状态节点所得到的可达图,称为兼容的可达图$com(G)$.显然,$com(G)$中的任意一条路径就表示 $com([N])$中的一个整型行为.但由于在构造可达图时对 d 的限制,$com(G)$中的所有路径集只是 $com([N])$中所有整型行为集的一个子集,并非一一对应.为此,任取 $com(G)$的一条路径ρ,其轨迹为σ_ρ形如:$(e_0,\tau_0)\wedge(e_1,\tau_1)\wedge\ldots\wedge(e_m,\tau_m)$,可构造与$\rho$相关的带时间戳事件序列$\theta(\sigma_\rho)$如下:

$$\theta(\sigma_\rho)=\left\{(e_0,\tau_0')\wedge(e_1,\tau_1')\wedge\ldots\wedge(e_m,\tau_m')\left|\begin{array}{l}\text{对所有}i(0\le i\le m),\text{当}\tau_i=K+1\text{时}\\ \tau_i'\in[K+1,\infty);\text{否则}\tau_i'=\tau_i\end{array}\right.\right\}.$$

这样,$\theta(\sigma_\rho)$中任意一个带时间戳的事件序列,一定是 $com([N])$中的某个整型行为的轨迹;同时,对于 $com([N])$中的任意一个整型行为的轨迹σ',在 $com(G)$一定存在某一条路径ρ',使得 $\sigma'\in\theta(\sigma_{\rho'})$.由此,我们可以通过检查兼容的可达图 $com(G)$中的每一条路径来完成一致性检验问题.

3.3 一致性验证

任取 $com(G)$ 中的一条路径 ρ,其轨迹形如 $\sigma_\rho=(e_0,\tau_0)\wedge(e_1,\tau_1)\wedge\ldots\wedge(e_m,\tau_m)$.设带时间戳的子事件序列 $(e_i,\tau_i)\wedge(e_{i+1},\tau_{i+1})\wedge\ldots\wedge(e_{i+j},\tau_{i+j})(0\le i\le m-j)$为 σ_ρ中关于 D 的一个投影.当在不等式 $\sum_{i=0}^{n}c_i\Delta_i \sim b$ (参看定义 9)的某个 $\Delta_p(0\le p\le n,c_p\Delta_p\ge 0)$中存在 $\tau_v=K+1(i\le v\le i+j)$时,意味着在 $\theta(\sigma_\rho)$中存在无穷多个带时间戳的子事件序列是关于 D 的一个投影,但其相应位置上的 τ_v 的取值区间为 $[K+2,\infty]$.这样,就一定存在某个足够大的 τ_v 的取值使得上述的时间不等式不成立.当出现这种情况时,我们说 σ_ρ及 $\theta(\sigma_\rho)$是不满足 D 的时间约束.因此,对于集合 $\theta(\sigma_\rho)$而言,只需要检查 σ_ρ中是否存在关于 D 的投影及其是否满足 D 的时间约束.

由于 $com(G)$ 中的路径数量和路径的长度有可能是无穷的,为此,以下引入简单路径.对顺序图 $D=(C,E,M,F,W,S)$而言,若 $com(G)$中的一条路径 $\rho=l_0\wedge l_1\wedge\ldots\wedge l_n$ 满足以下条件时,称其为关于 D 的简单路径:(1) 存在一个子路径的轨迹 $\sigma(l_i,l_j)(0\le i\le n)$是 σ_ρ中关于 D 的一个投影;(2) 对任何的 $j,k(0\le j<k<i)$,有 $l_j\ne l_k$;(3) 对任何相邻的 $j,k(i\le j<k\le n,\psi(l_j)\in E,\psi(l_k)\in E)$,有 $l_g\ne l_h(j<g<h<k)$.第 1 个条件表明:场景 D 正好在路径 ρ的后半段上出现;第 2 个和第 3 个条件则分别对路径 ρ中的两段子路径中可能出现的环给出了相应的限制.当 $\sigma(l_i,l_n)$进一步满足 D 的时间约束时,则称简单路径 ρ满足 D.显然,简单路径的长度是有穷的,$com(G)$中简单路径的数目也是有穷的.

考虑环路的影响.设任一子路径 $\rho_1=l_j\wedge l_{j+1}\wedge\ldots\wedge l_k$,其轨迹为 $\sigma(l_j,l_k)$,其中,$l_i:[u_i]\xrightarrow{a_i,d_i}[u_{i+1}](j\le i\le k)$.当 $l_j=l_k$ 时,称 ρ_1 为一个环; $\sum_{i=j+1}^{k}d_i$ 为环 ρ_1 上的时间流逝值,用 $T(\rho_1)$表示.若进一步对于任意 $p,q(j\le p<q<k)$,有 $l_p\ne l_q$,则称 ρ_1 为简单环.若对于顺序图 $D=(C,E',M,F,W,S)$而言,任一个 $e'\in E'$都不在 $\sigma(l_j,l_k)$中出现,则称 ρ_1 为与 D 无关的平凡环.

设 $com(G)$中的任一条简单路径为 $\rho=l_0\wedge l_1\wedge\ldots\wedge l_m$,若在 $com(G)$中存在一个包含 l_i 在内的与 D 无关的平凡环,且 $l_i:[u_i]\xrightarrow{a_i,d_i}[u_{i+1}](0\le i\le m)$ 中的 d_i 在 $\Delta_p(0\le p\le n,c_p\Delta_p\ge 0)$(参看定义 9)中出现,则称其为一条异常的简单路径;l_i 称为简单路径 ρ 中的一个瑕点.显然,只要在简单路径中存在瑕点,就可以通过这个瑕点在 $com(G)$中构造出一条路径,它满足场景 D 的出现,但是不满足其中的时间约束.至此,我们给出以下定理:

定理 **4**. N 为一个实时接口自动机网络(RIAN),$com(G)$是其兼容的可达图.N 满足带时间约束的顺序图 D 当且仅当以下条件同时成立:

- $com(G)$中存在一条关于 D 的简单路径;
- $com(G)$中所有的简单路径都满足 D;
- $com(G)$中不存在异常的简单路径.

定理 4 的证明从略.基于上述定理,我们可以给出相应的检验算法.

实时接口自动机网络与带时间约束序列图的一致性验证算法.

current_path:=⟨[u_0]⟩; *loopset*:= ϕ

repeat *node*:=the last node of *current_path*;
  if *node* has no new successive node **then** delete the last node of *current_path*
  else begin *node*:=a new successive node of *node*;
    if *node* is such that there is a simple flat loop for D in the corresponding path of *current_path*∧*node*
    then put the loop into *loopset*
    else append *node* to *current_path*;
  end
until *current_path*=⟨⟩;
current_path:=⟨[u_0]⟩; *is_simple_path*:=**false**;
repeat *node*:=the last node of *current_path*;
  if *node* has no new successive node **then** delete the last node of *current_path*
  else begin *node*:=a new successive node of *node*;
    if *node* satisfies that the corresponding path of *current_path*∧*node* is a simple path

then begin check if the simple path satisfies *D*;
if no, **then return false**; *is_simple_path*:=**true**;
check if the simple path is a flaw simple path for *D*;
if yes, **then return false**;
end
if *node* satisfies that the corresponding path of *current_path*∧*node* is a prefix of simple path
then append *node* to *current_path*;
end
until *current_path*=〈〉
if *is_simple_path* **then return true else return false**.

在此算法框架中,需要判断一条简单路径ρ是否成为一个异常的简单路径,即需要判断ρ中是否存在瑕点.下面先给出在 *com*(*G*)中基于边 l_i 的简单平凡环集合$\lambda(l_i,D)$的递归定义:(1) 任何包含 l_i 在内与 *D* 无关的简单平凡环在$\lambda(l_i,D)$中;(2) 对于$\lambda(l_i,D)$中任一个环 $\rho'=l'_p\wedge l'_{p+1}\wedge\ldots\wedge l'_q$,任何包含 $l'_r(p\le r\le q)$ 在内的与 *D* 无关的简单平凡环也都在$\lambda(l_i,D)$中.当存在一个$\rho'\in\lambda(l_i,D)$满足 $T(\rho')>0$)时,则在 *com*(*G*)中一定存在一个包含 l_i 在内的与 *D* 无关的平凡环ρ_1 满足 $T(\rho_1)>0$;反之也是如此.因此,判断简单路径ρ是否具有瑕点,可以检查每一个 $l_i(0\le i\le m)$是否与 *D* 的时间约束不等式相关,以及是否存在一个$\rho'\in\lambda(l_i,D)$满足 $T(\rho')>0$),如果二者都成立,则 l_i 一定是瑕点,ρ一定是一条异常的简单路径.

验证算法内容描述如下:我们采用深度优先的方式从初始节点开始对 *com*(*G*)进行遍历.使用变量 *current_path* 来存放当前所检查的路径;布尔变量 *is_simple_path* 表示 *com*(*G*)是否存在一个简单路径;集合变量 *loopset* 用来保存所有与 *D* 无关的简单平凡环.整个验证算法对 *com*(*G*)进行两次深度优先遍历:首次遍历找出所有与 *D* 无关的简单平凡环,目的是用来判断简单路径中的瑕点;然后重新开始一次新的遍历.通过不断地检查当前路径是否成为一个简单路径的前缀,来找出 *com*(*G*)中所有的简单路径并检查其是否满足 *D*.所谓一个简单路径的前缀是指,对 *com*(*G*)中任意一条路径ρ而言,若存在一个子路径ρ'使得$\rho\wedge\rho'$成为一个简单路径,则称ρ为一个简单路径的前缀.在第 2 次遍历过程中,每发现一个新节点,首先检查 *current_path* 所对应的当前路径是否成为一条简单路径;若是,则继续检查其是否满足 *D*;若满足 *D*,则将 *is_simple_path* 赋值为 true,并进一步检查这条简单路径是否异常,这可以通过检查其中有没有包含瑕点来完成.当新节点使得 *current_path* 所对应的路径不能构成简单路径的前缀时,算法回溯;否则将此节点加入到 *current_path* 中.由于简单路径的长度和数量都是有穷的,所以此检验算法是可终止的,其复杂度与 *com*(*G*)中简单路径的数量和长度成正比.

4 相关工作比较

相关的研究工作包括:文献[4]对 UML 中带时间区间标志的状态机模型,与带时间约束的合作图的一致性问题进行了验证.其中,带时间标志的状态机模型被转换为时间自动机模型;带时间约束的合作图模型也被翻译成一个自动机,然后直接调用实时模型检验工具 UPPAAL[5]进行形式化验证;文献[6]中考虑了实时系统的需求,在 UML 顺序图中添加时间区间以扩展其描述能力,并对与时间相关的一致性性质进行了分析;文献[7]中给出了一个实例研究,用以说明如何将一个简单的 UML 顺序图转换成一系列的时间自动机,然后,使用时间自动机的相关验证工具进行验证.文献[8]中使用时钟变量、时间卫式以及时钟不变式对 UML 中的状态图进行扩展,然后转换成时间自动机进行性质验证.对接口自动机而言,Lee 等人在 PtolemyII[9]系统中使用接口自动机设计了一个扩展的类型系统框架[10],通过接口自动机之间的组合运算,对构件之间的交互进行兼容性检查,但没有进一步对系统行为是否满足某些需求规约进行检验.文献[11]中给出了一种对数据流进程网络进行模块化一致性分析的方法.其中,接口自动机作为进程模型中,架构模型和混合式组件具体模型之间进行转换的有效描述工具.Wen 等人给出了从接口自动机到 I/O 自动机的转换方法[12].由于接口描述通常比相应的实现要简单得多,通过转换前后的对比,发现接口自动机模型的状态空间的确要比通常的 I/O 状态机模型小得多,这也是使用接口

自动机进行模型检验的好处之一.文献[13]中使用接口自动机网络表示基于组件的设计模型,并对此模型与基于场景的规约之间的存在一致性和强制存在一致性等性质进行分析,给出了相关的验证算法,但没有涉及与时间相关的性质.此外,Luca de Alfaro 等人继续在接口自动机的基础上进行时间约束和资源约束表达的扩充,给出了时间接口(timed interface)[14]和资源接口(resource interface)[15],分别用于对实时环境和有限资源环境中,构件接口交互行为和组合性质进行形式化的描述与验证.其中,时间接口的相关工作与我们的工作比较类似:他们在接口自动机的状态上添加不同种类的时钟变量,时间接口之间的兼容性和复合运算是通过转换成带时间的 Game 来实现的.虽然时间接口的相关算法可以用来有效地描述和验证组件接口之间实时交互的兼容性,但对本文中所关心的在设计模型和规约模型之间带时间约束行为的一致性验证问题而言还不够.与上述研究相比,本文的工作对构件式实时软件系统设计模型与场景式规约模型之间的一致性问题,给出了一个更为完整的验证框架:一方面,我们使用轻量级的实时接口自动机这种形式化语言而不是 UML 的状态机模型,来描述在复杂实时软件系统中构件的接口行为,并通过使用时间不等式约束,使得所定义的实时接口自动机网络与带时间约束的顺序图之间的一致性问题更具有一般性;另一方面,我们的工作是直接建立在 UML 的顺序图和时间接口自动机网络之上的,避免了在大多数相关工作中所需的在不同的形式模型之间进行复杂的转换——通常,这些转换需要相当的空间和时间消耗;并且,本文中仅使用时间区间对接口自动机进行扩展,简单而有效.事实上,对于实时接口自动机网络而言,其描述能力与时间自动机是等价的.

5 结束语

本文中使用实时接口自动机网络来描述构件式实时软件系统的设计模型,使用带时间约束不等式的 UML 顺序图来表示基于场景的需求规约.通过对实时接口自动机网络的可兼容整型状态空间的分析,构造其兼容的可达图,并在此基础上给出了验证算法以检验构件式实时软件系统的设计与带时间约束的场景式规约之间的一致性.在本文的工作基础上,正在实现一个原型工具.我们进一步的工作包括:考虑复杂层次结构的顺序图,以及在实时接口自动机中引入资源的相关约束,用以描述和验证嵌入式实时软件系统.此外,还将进行更多工业界的应用实例研究.

References:

[1] de Alfaro L, Henzinger TA. Interface automata. In: Proc. of the Joint 8th European Software Engineering Conf. and 9th ACM SIGSOFT Int'l Symp. on the Foundations of Software Engineering (ESEC/FSE 2001). Austria: ACM Press, 2001. 109−120.

[2] Booch G, Rumbangh J, Jacobson I. The Unified Modeling Language User Guide. Addison-Wesley, 1999.

[3] Peled DA. Software Reliability Methods. Springer-Verlag, 2001.

[4] Alur R, Yannakakis M. Model checking of message sequence charts. In: Baeten JCM, Mauw S, eds. Proc. of the 10th Int'l Conf. on Concurrency Theory. LNCS 1664, Berlin: Springer-Verlag, 1999. 114−129.

[5] Larsen KG, Pettersson P, Yi W. UPPAL in a nutshell. Int'l Journal of Software Tools for Technology Transfer, 1997,1(1−2): 134−152.

[6] Seemann J, Gudenberg JW. Extension of UML sequence diagrams for real-time systems. In: Bézivin J, Muller P-A, eds. Proc. of the Int'l UML Workshop. LNCS 1618, Berlin: Springer-Verlag, 1998. 240−252.

[7] Firley T, Huhn M, Diethers K, Gehrke T, Goltz V. Timed sequence diagrams and tool-based analysis-A case study. In: France R, Rumpe B, eds. Proc. of the 2nd Int'l Conf. on UML (UML'99). LNCS 1723, Berlin: Springer-Verlag, 1999. 645−660.

[8] David A, Möller MO, Yi W. Formal verification of UML statecharts with real-time extensions. In: Kutsche R-D, Weber F, eds. FASE2002. LNCS 2306, Berlin: Springer-Verlag, 2002. 218−232.

[9] http://ptolemy.eecs.berkeley.edu/ptolemyII/index.htm. 2005.

[10] Lee EA, Xiong YH. System-Level types for component-based design. In: Henzinger TA, Kirsch CM, eds. Proc. of the EMSOFT 2001. LNCS 2211, Berlin: Springer-Verlag, 2001. 237−253.

[11] Jin Y, Esser R, Lakos C. Lightweight consistency analysis of dataflow process networks. In: Oudshoorn M, ed. Proc. of the 26th Australasian Computer Science Conf. (ACSC 2003). Adelaide, 2003. 291−300.

[12] Wen YJ, Wang J, Qi ZC. Bridging refinement of interface automata to forward simulation of I/O automata. In: Davies J, *et al*., eds. Proc. of the 6th Int'l Conf. on Formal Engineering Method (ICFEM 2004). LNCS 3308, Berlin: Springer-Verlag, 2004. 259−273.

[13] Hu J, Yu XF, Zhang Y, Zhang T, Wang LZ, Li XD, Zheng GL. Scenario-Based verification for component-based embedded software systems. In: Proc. of the 2nd Embedded Computing Workshop (ICPP-EC 2005). Norway: IEEE Computer Society Press, 2005. 240−247.

[14] de Alfaro L, Henzinger TA, Stoelinga M. Timed interfaces. In: Sangiovanni-Vincentelli A, Sifakis J, eds. Proc. of the 2nd Int'l Conf. on Embedded Software (EMSOFT 2002). LNCS 2491, Berlin: Springer-Verlag, 2002. 108−122.

[15] Chakrabarti A, de Alfaro L, Henzinger TA, Stoelinga M. Resource interfaces. In: Alur R, Lee I, eds. Proc. of the 3rd Int'l Conf. on Embedded Software (EMSOFT 2003). LNCS 2855, Berlin: Springer-Verlag, 2003. 117−133.

胡军(1973－),男,湖北黄冈人,博士生,主要研究领域为软件工程,形式化方法,嵌入式软件建模与验证.

于笑丰(1976－),男,博士生,主要研究领域为软件工程,模型驱动转换与验证.

张岩(1974－),男,博士生,主要研究领域为软件工程,形式化方法,面向服务的计算,软件度量.

李宣东(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,形式化方法,软件验证.

郑国梁(1936－),男,教授,博士生导师,主要研究领域为软件工程.