面向嵌入式实时软件的需求规约语言及检测方法*

舒风笛[1+], 毋国庆[2], 李明树[1]

[1](中国科学院 软件研究所,北京 100080)

[2](武汉大学 计算机学院 计算机科学系,湖北 武汉 430072)

An Embedded Real-Time Software Oriented Requirements Specification Language and Checking Methods

SHU Feng-Di[1+], WU Guo-Qing[2], LI Ming-Shu[1]

[1](Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Department of Computer Science, Computer School, Wuhan University, Wuhan 430072, China)

+ Corresponding author: Phn: +86-10-82620803, E-mail: fdshu@itechs.iscas.ac.cn

Received 2004-04-01; Accepted 2004-06-11

Shu FD, Wu GQ, Li MS. An embedded real-time software oriented requirements specification language and checking methods. *Journal of Software*, 2004,15(11):1595~1606.

http://www.jos.org.cn/1000-9825/15/1595.htm

Abstract: Aiming at the requirements specification and related checking of embedded real-time software, a visual modeling language, RTRSM* (real-time requirements specification model*), which is compositional and based on hierarchical and concurrent finite state machine, is proposed. It uses state transitions with duration and scheduled events to describe timing constraints, and can support the description of interactivity and timing constraints effectively. Additionally, RITL (real-time interval temporal logic), a kind of prepositional temporal logic, is presented to make up for RTRSM*'s defect description of global system properties, which is the drawback of operational specification languages. Interpreted over timed state sequences, RITL is able to deal with the description of both point-based and interval-based metric temporal properties, and supports the property description of RTRSM* models naturally and comprehensively. The verification and validation of the resulted requirements specification, especially issues with respect to the reachability graph of RTRSM* models with finite system states and the simulation execution of the specification, are also explored.

Key words: embedded real-time software; requirements specification language; requirements specification checking; reachability graph

摘 要: 针对嵌入式实时软件需求规约及其检测问题,提出了基于层次并发有穷状态机的可合成的图形化建模语

* Supported by the National Natural Science Foundation of China under Grant Nos.69873035, 60273026 (国家自然科学基金); the K.C. Wong Education Foundation, Hong Kong, China (香港王宽诚教育基金)

作者简介: **舒风笛**(1976－),女,湖南邵阳人,博士,主要研究领域为需求工程,形式化理论和方法;**毋国庆**(1954－),男,教授,博士生导师,主要研究领域为软件工程,需求工程,软件形式化理论;**李明树**(1966－),男,博士,研究员,博士生导师,主要研究领域为智能软件工程,实时系统.

言 RTRSM*(real-time requirements specification model*),利用转换有效期和事件预定机制来描述时间限制,能够较好地支持系统交互性和实时性的建模.为弥补 RTRSM*作为操作性规约语言不便于性质描述的问题,提出了命题时序逻辑 RITL(real-time interval temporal logic).该语言以时间状态序列为语义模型,具有基于区间和时间点的量化时间属性描述功能,能自然、全面地描述 RTRSM*模型性质.介绍并讨论了基于两种语言的规约检测方法和技术,主要包括系统状态空间有穷的 RTRSM*模型状态可达图的相关问题和规约的模拟执行.

关键词: 嵌入式实时软件;需求规约语言;需求规约检测;可达图

中图法分类号: TP311 **文献标识码**: A

形式化需求规约语言有助于准确而无二义性地理解和描述目标系统,支持规约的形式化检测,但它又应具有一定的表达能力,能自然地对现实世界实体进行建模并支持领域特征描述.嵌入式实时软件(embedded real-time software,简称 ERS)的主要领域特征为事件驱动、具有复杂动态交互行为和严格时间限制.操作性规约语言便于描述外部刺激、系统动作和状态等建模实体,却不适合于性质描述;而单纯的描述性形式系统,如逻辑,则正好相反.许多研究工作将两者结合起来,得到的软件需求规约(software requirements specification,简称 SRS)包括用操作性规约语言描述的系统模型及用断言式语言描述的期望模型满足的性质.SRS 正确性的检测内容主要包括一致性、完备性等独立于具体应用的性质和用户需求的确认.

与 ERS 的需求规约及其检测相关的代表性工作包括:基于状态转换的 Statecharts[1]、SCR[2]、时间自动机(timed automata,简称 TA)[3]、时间转换系统(timed transition system,简称 TTS)[4]、TPTL[5]、XYZ[6]等多种时序逻辑,Modechart/RTL[7]、TTM/RTTL[8]等双语言框架,以及相应的原型化和形式化检测方法和工具,如基于 Statecharts 等的 Statemate[9,10],形式化验证工具 SPIN[11]、PVS[12]等.这些研究各有其侧重点.Harel 提出的 Statecharts 面向复杂、交互式系统,通过引入状态的层次、并发和广播机制对传统状态转换图进行扩充,简洁且高度结构化,具有良好的形式化语义,但其时间限制描述能力较弱;Timed Statecharts[13]较之传统的 Statecharts,时间限制描述能力得到加强,但其形式过于复杂.实时 UML[14]是在基本 UML 基础上引入了起源于 ROOM(real-time object oriented modeling)[15]的便于设计复杂嵌入式系统的结构,其描述单一对象行为的状态图与传统的 Statecharts 没有本质区别,主要通过交互图和活动图进行时间限制建模.此外,虽然已有一些 UML 形式化语义和验证研究[16~18],但仍不成熟,尤其是时间限制方面.Modechart[7]强调的是控制和实时的描述,没有转换操作机制.TA 和 TTS 为较为成熟的实时模型,分别主要通过时钟变量和转换的上下时限来支持时间限制的描述,但两模型对于反应性的支持不够自然和直接,且其分析机制较为复杂.对于逻辑语言而言,也没有任何一种单独的逻辑可以说绝对优于其他逻辑[19].

针对 ERS 的需求规约及其检测问题,我们提出了需求建模语言 RTRSM*(real-time requirements specification model*)及其性质描述语言 RITL(real-time interval temporal logic).前者继承了 Statecharts 的层次并发状态和广播通信机制,通过增加转换有效期和事件预定机制来描述时间限制,能够较好地支持交互性、实时性和一定数据计算的建模,图形化且具有严格形式语义.RITL 为基于区间限制的命题时序逻辑,能够支持 RTRSM*模型的性质描述,尤其是时间性质.两种语言共同组成了适合于 ERS 的需求描述符号系统.

本文首先给出了两语言的语法和语义,举例说明了 RTRSM*的时间性质描述机制,介绍并讨论了相应规约检测方面所做的工作,包括模型系统状态可达图的构造算法及相应性质验证,规约一致性和完备性检查方法,两语言的 PVS 语义嵌入及规约的模拟执行等.

1 需求建模语言 RTRSM*

1.1 RTRSM*概述

RTRSM*以 RTRSM[20]的模板为基本描述单元,每个模板对应一层次并发有穷状态机 HCA,其具体表现形式为包括数据和接口定义的表格、状态转换图及其等价的规则集.HCA 继承了 Statecharts[1]的层次和并发状态,即状态包括基本状态和实际为状态机的超状态,后者又分为与、或状态.模型所有状态构成一个以初始状态为

根的树型结构.

定义 1(HCA 的规则). HCA 中的每条转换对应于一条形式如下的规则:

源状态, 触发事件(事件属性)=>目的状态, WHEN 卫士条件, DO 操作 *D*

其中,非负整数 *D* 为转换有效期,缺省为 0.含义为:当源状态活跃、触发事件发生且卫士条件为真时,该转换可执行,而在成为可执行后 *D* 时间单元内,该转换只要有效(源状态活跃且卫士条件满足),可且必定实际执行(*D* 为 0 则立即执行).转换的执行过程是:退出源状态,执行操作(可包括:变量赋值,系统内外部通信指令的发送和事件的预定、取消等[21]),然后进入目的状态,具有原子性和瞬间性.

默认转换的源状态为空,目的状态非空,*D* 为 0,卫士条件缺省为真,无触发事件,一旦进入该转换所在超状态,则立即执行.对于其他转换,源状态、触发事件和目的状态非空,其他项可选.源状态和目的状态必须为兄弟.

定义 2(转换的一致性). 两转换一致,当它们相同或源状态正交(不存在祖先关系)且操作中无变量写冲突.对于一个转换集合 *T*,若其中任何两转换都一致,则称 *T* 一致.

RTRSM*不允许不确定性.允许可能存在的不确定性便于建模,但可能隐藏错误,且判断一个确定行为是否可接受比证明所有可能的不确定行为都可接受要容易.可将允许的多种情况抽象统一,正体现了规约能且正好能区分系统所期望行为与其他性质.RTRSM*通过顺序执行以下优先级原则消除可能存在的不一致:高层转换具有高优先级;外部事件触发的转换优先级最高,被延迟转换的执行优先级最低.

1.2 时间限制的描述

嵌入式实时系统的时间限制反应在外部刺激(包括时间的流逝)及系统反应上,基于 RTRSM*,前者可描述为能触发状态转换的事件,后者则为相应由当前系统状态所决定的转换的实际执行,由于其瞬间性,即为其目的状态的进入.RTRSM*采用隐含时间信息,利用事件及其预定机制和转换有效期来描述.Statecharts 的理想同步假设[1](系统总比外部环境动作更快,在收到刺激后马上作出反应)能简化系统需求的分析,但该假设并不总能被满足,且不便于充分、完全地描述实际中常见的以时间段形式给出的时间需求.RTRSM*转换带有有效期,能有效克服以上两点.下面通过对经典的铁路交叉路口系统(railroad crossing system,简称 RCS)[22]的特例——标准 RCS——稍加修改,来说明如何用 RTRSM*来描述时间段形式的时间需求.

用软件来控制一个通常打开的铁道口控制门.由探测器 1 检测火车的临近:当火车经过探测器 1 时,该探测器发送通知 near;火车在经过该探测器后,至少 300 个时间单元后才到达该路口.探测器 2 检测火车是否通过该路口,若通过则发送通知 passed.火车通过路口与经过探测器 1 最多相隔 500 个时间单元.控制器在接收到 near 消息后 100 个时间单元时向门发送关闭的控制命令 lower,并在接收到 passed 消息后 100 个时间单元时向门发送打开的控制命令 raise.门的关闭和打开各需 20~50 个和 100~200 个时间单元才能完成.根据火车调度方案,从一列火车离开铁道口到下一列火车到达探测器 1 至少间隔 100 个时间单元.在该例中,门的关闭和打开都是持续动作,作为状态来描述,其时间需求以常见的时间段形式给出.若转换是一旦满足则立即执行,虽然所得模型也能在限制时间范围内完成门的关闭和打开,但并未完全地表达用户需求,从而可能隐藏需求中的错误.而通过转换有效期,RTRSM*能较好地描述此类需求,从而支持通过相应规约检测发现其中的错误.图 1 给出了用支持 RTRSM 的面向 ERS 的需求工程环境 SREE[20]所编辑得

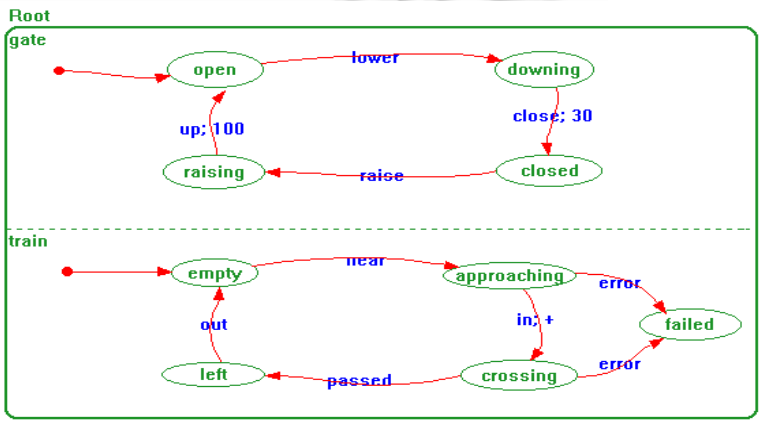

Fig.1 State transition diagram of the controlling system of railroad crossing

图 1 铁道口控制系统状态图

到的状态图,其中“up;100”中的 up 为事件名,100 为该转换有效期,“+”表示有效期正无穷大.图 2 为由 SREE 自动生成的规则集,其中第 1 条规则表示的是状态 Root 的进入导致其并发子状态 gate 和 train 的进入,对应于层次关系的分解规则[20].目的状态 gate.open 表示的是模板 gate 中的状态 open.用“||”分隔同一转换操作中的多个表达式.限于篇幅,省略该系统相应的模板信息.

```
the rule set of template Root
    Root, ___ => gate||train
the rule set of template gate
    gate, ___ => gate.open  which is a default state transition
    open, lower => gate.downing  Action  SetEvent(close,20)
    downing, close  => gate.closed ; 30
    raising, up  => gate.open  ; 100
    closed, raise => gate.raising  Action  SetEvent(up,100)
the rule set of template train
    train, ___ => train.empty  which is a default state transition
    empty, near => train.approaching  Action  SetEvent(lower,100);||SetEvent(in,300);||SetEvent(error,500)
    approaching, in => train.crossing  ; +
    approaching, error => train.failed
    left, out => train.empty
    crossing, error => train.failed
    crossing, passed => train.left  Action  SetEvent(out,100);||SetEvent(raise,100);
```

Fig.2 Rule set of the controlling system of railroad crossing

图 2 铁道口控制系统规则集

从本例可看出,RTRSM*在一定程度上吸取了 Statecharts,TA,TTS 等的长处,图形化、结构化,又具有较强的时间描述能力.

1.3 HCA模型及其合成

定义 3(HCA 模型). 一个 HCA 模型为一个四元组(V,D,θ,T),其中:

V:变量集合,包括:

(1) 模板中所定义的数据变量,包括接口变量集合 I 和私有变量集合 pr,I 又包括系统输入(由外部环境决定,系统只读)、输出(由系统决定,发送给外部环境)和共享变量(外部环境和系统都可读写);

(2) 时钟变量 t,有 $type(t)=\overline{Z^-}$,其中 $type(v)$为变量 v 的类型;

(3) 为状态机中每个状态和事件设置一个对应的布尔变量,变量名为该状态名或事件名,称为状态变量或事件变量,若为真,则表示该状态活跃或该事件发生;

(4) 记录列表 *FutureEvent*,其元素形式为(e,n),e 为事件名,n 为自然数.

D:所有变量类型的并.

θ:必须满足的初始条件.

T:转换集合,包括特殊的时钟转换 tick.

定义 4(HCA 的系统状态). HCA 模型 M 的系统状态为一个映射 $gs:V\to D$,使得对于任意一个 $v\in V$,有 $gs(v)\in type(v)$,由此构成模型的系统状态空间集合 GS.若 p 为 M 中变量的表达式,则 $gs(p)$表示 p 在 gs 的取值.

定义 5(转换). 转换 τ为一个三元组(e_τ,h_τ,u_τ),有:τ的可执行条件 $e_\tau=_{\text{def}} source(\tau)\wedge event(\tau)\wedge cond(\tau)$,其中 $source(\tau)$和 $event(\tau)$分别为 τ的源状态和触发事件所对应布尔表达式,$cond(\tau)$为其卫士条件,当 $gs(e_\tau)$为真时,τ可执行;τ有效的条件 $c_\tau=_{\text{def}} source(\tau)\wedge cond(\tau)$;$\tau$的转换函数 $h_\tau:GS\to GS$ 为一个定义在 $gs(e_\tau)$=true 的 gs 上的偏函数;非负整数 u_τ为 τ的有效期.

定义 6(转换关系). 如果存在一个转换 τ,其源系统状态为 gs_1,目的系统状态 gs_2,则定义 gs_1,gs_2 满足转换关系 ρ,记为$\rho(gs_1,gs_2)$,由 RTRSM*的确定性,该转换唯一.

定义 7(模板). 一个模板 M 可表示为一个三元组(I,hca,P),各成员分别表示模板的接口定义(记为 $I(M)$)、所对应的 HCA 模型和必须满足的性质.

定义 8(HCA 的合成). 对于 $hca_1=(V_1,D_1,\theta_1,T_1)$,$hca_2=(V_2,D_2,\theta_2,T_2)$,它们并发合成和顺序合成分别得到对应

的与状态和或状态的 $hca=hca_1||hca_2=(V,D,\theta,T)$ 和 $hca'=hca_1°hca_2=(V',D',\theta',T')$,其中顺序合成时先进入 $hca_1,\tau_1',...,\tau_m'$ 为连接它们的转换,有:$V=V'=V_1\cup V_2$;$D=D'=D_1\cup D_2$;$\theta=\theta_1\wedge\theta_2$;$T=T_1\cup T_2$;$\theta'=\theta_1$;$T'=T_1\cup T_2\cup\{\tau_j'\}$,$j\in[1,m]$.

定义 9(并发相容的模板). 两个模板 $M_1=(I_1,hca_1,P_1)$,$M_2=(I_2,hca_2,P_2)$,其中 $hca_1=(V_1,D_1,\theta_1,T_1)$,$hca_2=(V_2,D_2,\theta_2,T_2)$是并发相容的,当对于每个 $v\in I(M_1)\cap I(M_2)$在两模板中的类型匹配,且 $\theta_1\wedge\theta_2$ 和 $P_1\wedge P_2$ 可满足.

定义 10(模板的并发合成). 两并发相容的模板 $M_1=(I_1,hca_1,P_1)$,$M_2=(I_2,hca_2,P_2)$ 的并发合成体 $M=M_1||M_2=(I,hca,P)$,有:$I\subseteq I_1\cup I_2$,$hca=hca_1||hca_2$,$P=P_1\wedge P_2$.

定义 11(模板的顺序合成). 两模板 $M_1=(I_1,hca_1,P_1)$,$M_2=(I_2,hca_2,P_2)$的顺序合成体 $M=M_1°M_2=(I,hca,P)$,设 $\tau_1,\tau_2,...,\tau_m$ 为从 hca_1 到 hca_2 的转换,$\tau_1',\tau_2',...,\tau_n'$为从 hca_2 到 hca_1 的转换(HCA 中不允许出现跨层转换),有 $I\subseteq I_1\cup I_2$,$hca=hca_1°hca_2$,$P=(\wedge_{1\le i\le m}(P_1\wedge cond(\tau_i)\wedge event(\tau_i)\rightarrow P_2)\wedge\wedge_{1\le j\le n}(P_2\wedge cond(\tau_j')\wedge event(\tau_j')\rightarrow P_1))\wedge(P_1\vee P_2)$.

1.4 RTRSM*的操作语义

ERS 的目的是使计算机有效控制相应部件.鉴于绝大多数计算机控制是离散的且离散时间域便于分析,将RTRSM*定义在离散时间域上.在实践应用中则要求合理选择时间片.

定义 12(HCA 的初始轨迹). HCA 的初始轨迹 $\sigma=gs_0gs_1gs_2...$,其中 $gs_i\in GS,i\in[0,+\infty]$,$\lambda_i$ 为连接 gs_i 和 gs_{i+1} 的全局转换(包括一个或多个一致的正交转换),满足以下性质:

- 初始化:$\lambda_0=initial$ 且除此外,*initial* 不再发生,它作为系统的启动,进入稳定的系统初始状态 gs_1,其转换函数的作用包括系统根状态及相应默认状态的进入;
- 连续性:对于每个 $i,\rho(gs_i,gs_{i+1})$;
- 时间上限:对于任意的 $gs_i(e_\tau)$=true 的 $\tau\in T$,若 $u_\tau=0$,则:或者 $\tau\in\lambda_i$,或者存在一个与 τ不一致的转换 $\tau'\in\lambda_i$:$gs_i(e_{\tau'})$=true 且 τ'优先级高于 τ;否则,必存在一个 $j>i$,有 $(gs_j(c_\tau)$=false$\vee\tau\in\lambda_i)\wedge(gs_j(t)\le gs_i(t)+u_\tau)$;
- 时钟滴答(前进):σ中有无穷多个 $\lambda_{i>0}$=tick.

定义 13(HCA 的合法轨迹). 一个 HCA 模型 *hca* 的合法轨迹集合 Σ_{hca} 由其所有初始轨迹及其后缀组成.

RTRSM*所描述的系统行为是一组可能的由其环境产生的外部刺激(包括时间的流逝)所导致的反应序列(如图 3 所示),即对应的 Σ_{hca}.没有引发状态迁移的事件发生的系统状态称为稳定状态(如图 3 中的 $C_{1,k}$),从一个稳定状态出发,直至进入下一个稳定系统状态的全局转换执行序列称为一宏步,每个全局转换的执行称为一微步[1],规定每一微步的执行效果只在下一微步有效.事件的"真"只维持一微步.宏步的开始用显式全局时钟滴答来标识,可作为一个外部事件,所引发的系统状态迁移即为时钟转换 tick.因此,RTRSM*的完全操作语义定义应包括每一宏步的执行过程和宏步间的连接,限于篇幅,此处不予详述[21].

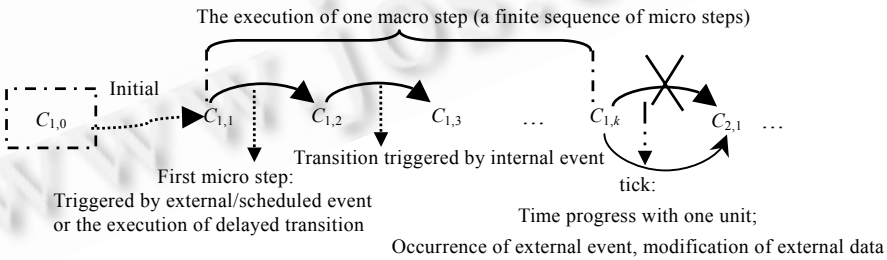

Fig.3 System dynamic behavior of RTRSM* model

图 3 RTRSM*模型系统动态行为

2 时序逻辑语言 RITL

RTRSM*作为操作性规约语言,难以描述系统性质.因此,根据 RTRSM*的操作语义,我们提出了基于命题逻辑的时序逻辑 RITL,以支持 RTRSM*模型性质的描述及相应验证,包括时间性质.RITL 基于离散时间域,采用特定算子支持 RTRSM*的微步和宏步概念;为过去和将来设置对称的时序算子;采用线性时间结构,与 RTRSM*不

允许不确定性一致.时序的存在和任意量词使其能描述活性和安全性性质;用区间来限制时序算子,又允许"准时"的描述,可描述系统基于区间和时间点的性质.

2.1 语 法

定义 14(区间). 区间 I 为一个非空非负整数集合.RITL 公式中出现的区间形式包括$[a,b]$,$[a,+\infty)$,其中 a,b 为非负整数,$b\geq a$,a 和 b 分别称为左、右端点,记为 $l(I),r(I)$.

除了括号,布尔常元 true,false 和通常的真值连接词$\neg,\wedge,\vee,\rightarrow,\leftrightarrow$以外,RITL 的字母表还包括:

(1) 命题 $p_1,p_2,...$,用语法变元 p 表示;

(2) 整形区间 $I_1,I_2,...$,用语法变元 I 表示;

(3) 时序算子:$\triangle,\nabla,\mathrm{O},@,?,\mathrm{U}_I,\mathrm{S}_I,\Theta,@!,?!,\mathrm{U}!_I,\mathrm{S}!_I$;

(4) 切割算子: ;和;! .

定义 15(RITL 的公式). RITL 的公式递归定义如下,其中ϕ,φ为 RITL 公式,I 为区间,t 为非负整数:

$$\phi::=\mathrm{true}|\mathrm{false}|\mathrm{O}\phi|\Theta\phi|\neg\phi|\phi\wedge\varphi|\phi\vee\varphi|\phi\rightarrow\varphi|\phi\leftrightarrow\varphi|\phi\triangle t|\phi\nabla t|\phi\mathrm{U}_I\varphi|\phi\mathrm{S}_I\varphi|$$
$$\phi@I|\phi?I|\phi\mathrm{U}!_I\varphi|\phi\mathrm{S}!_I\varphi|\phi@!I|\phi?!I|\phi;\varphi I|\phi;!\varphi I$$

2.2 语 义

定义 16(时间状态序列). RITL 的语义模型为无穷的时间状态序列[5]$\sigma=\sigma_0\sigma_1\sigma_2\sigma_3...$,其中$\sigma_i=(s_i,t_i)$,$s_i$ 为状态,时间 t_i 为非负整数,满足以下性质:

单调性:对于所有的 $i\geq 0$,有 $t_i\leq t_{i+1}$;

前进性:对于所有的 t_i 存在一个 t_j,有 $t_i<t_j$.

定义 17(RITL 公式的语义). 第 i 个状态(位置)s_i 满足 RITL 公式ϕ当且仅当$\sigma,i\models\phi$,$\models$的递归定义如下:

$\sigma,i\models\mathrm{true}$;

$\sigma,i\not\models\mathrm{false}$;

$\sigma,i\models\neg A$,当且仅当$\sigma,i\not\models A$;

$\sigma,i\models A\vee B$,当且仅当$\sigma,i\models A$ 或$\sigma,i\models B$;

$\sigma,i\models A\rightarrow B$,当且仅当$\sigma,i\models\neg A\vee B$;

$\sigma,i\models\mathrm{O}A$,当且仅当$\sigma,i+1\models A$;

$\sigma,i\models\Theta A$,当且仅当$\sigma,i-1\models A$,当 $i\geq 1$;

$\sigma,i\models A\triangle t$,当且仅当存在一个 $j:t_j=t_i+t\wedge t_{j-1}<t_i+t$,有$\sigma,j\models A$;

$\sigma,i\models A\nabla t$,当且仅当分别存在一个 $j,k:t_j=t_i+t\wedge t_{j-1}<t_i+t\wedge t_k=t_i+t\wedge t_{k+1}>t_i+t$,对任意 $h:j<h\leq k$,有$\sigma,h\models A$;

$\sigma,i\models\phi I$,当且仅当对每个 $j:t_j-t_i\in I\wedge j>i$,有$\sigma,j\models\phi$;

$\sigma,i\models\phi?I$,当且仅当存在一个 $j:t_j-t_i\in I\wedge j>i$,有$\sigma,j\models\phi$;

$\sigma,i\models\phi_1\mathrm{U}_1\phi_2$,当且仅当存在一个 $k:t_k-t_i\in I\wedge k>i$,有$\sigma,k\models\phi_2$,且对任意 $j:i<j<k$,有$\sigma,j\models\phi_1$;

$\sigma,i\models\phi_1\mathrm{S}_I\phi_2$,当且仅当存在一个 $k:t_k-t_i\in I\wedge k>i$,有$\sigma,k\models\phi_2$,且对任意 $j:k<j\wedge t_j-t_i\in I$,有$\sigma,j\models\phi_1$;

$\sigma,i\models\phi_1;\phi_2 I$,当且仅当存在一个 $j:t_j-t_i\in I\wedge j\geq i$,对于任意 $r:i\leq r\leq j$,有$\sigma,r\models\phi_1$,且对任意 $k:k>j\wedge t_k-t_i\in I$,有$\sigma,k\models\phi_2$.

其中,A,B,ϕ_1,ϕ_2 为公式,$\wedge,\leftrightarrow$取通常的语义;由定义在将来的算子$@,?,\mathrm{U}_I,\mathrm{S}_I,;$的语义,可分别定义相应定义在过去的对称算子$@!,?!,\mathrm{U}!_I,\mathrm{S}!_I$和;!.定义在区间上的时序算子都是"严格"的,即不包括所考查位置本身.

定义 18. σ满足 RITL 公式ϕ,或σ为ϕ的模型,当且仅当$\sigma,1\models\phi$,记为$\sigma\models\phi$.

2.3 RTRSM*模型性质的RITL描述

定义 19. 将每个状态、事件和变量的每种取值对应一个命题,当所对应的状态活跃,事件发生或变量取指定值,则所对应的命题为真.系统状态用在该系统状态为真的命题的合取式表示,称为状态公式,不含时序算子.

定义 20. RTRSM*模型的一次执行过程称为一次计算,即一个时间状态序列.

用 RITL 公式来描述 RTRSM*模型性质,若公式相对于计算的第 1 个位置为真,则根据定义 18,该公式为该

RTRSM*模型的一个性质.下面给出部分性质描述:

(1) RTRSM*模型的基本实体(状态、事件和变元取值)的性质描述:

- 外部事件只在宏步开始,即一个时间单元的第 1 个位置发生,其他位置为假:

$\Box(p\rightarrow p\triangle 0\wedge\neg p\triangledown 0)$,其中 p 代表任意一个外部事件所对应的命题.

- 私有变量的值不会在时钟转换中被改变:$\Box(p\triangle 0\leftrightarrow(\Theta p)\triangle 0)$,其中 p 代表该私有变量此项取值的命题.
- 不出现无穷微步序列.设 $p_1,p_2,...,p_n$ 为该 RTRSM*模型所有事件对应的命题,终将有一个系统状态,无任何事件发生,即最终有一个稳定系统状态:$\Box\neg(p_1\vee p_2\vee...\vee p_n)?[0,\infty)$.

(2) 常见性质描述:

- 最大持续时间(一旦发生,最多持续 a 个时间单元): $\Box(\neg p?[0,a+1])$.
- 最小持续时间(一旦发生,则必定持续至少 a 个时间单元): $\Box(\neg p\wedge Op\rightarrow p@[0,a])$.
- 必须同时发生,如与状态的所有直接子状态同时活跃或不活跃: $\Box(A\wedge B\vee\neg A\wedge\neg B)$.
- 以 d 为周期,周期性发生: $\Box(p\rightarrow\neg p@[0,d-1]\wedge P?[d,d])$.

3 基于 RTRSM*和 RITL 的规约检测

作为主要的规约检测方法,形式化方法和原型化方法各有特点,前者如模型检测、定理证明,基于严格的形式语义,可靠性高,可得到(半)自动工具支持,但实现和使用难度较大;后者能较快地给用户一个直观反馈,但可靠性不高.本节首先介绍 RTRSM*模型的可达图构造方法及相应可进行的性质验证,然后对规约一致性、完备性的检查方法进行讨论,并简单介绍在定理证明和原型化验证方面所做的工作.

3.1 基于模型状态可达图的性质验证

3.1.1 RTRSM*状态可达图构造算法

定义 21(可达状态). 系统 M 的一个系统状态 s 称为可达的,当它出现在 M 的一个合法轨迹中.

定义 22(状态可达图). 一个状态可达图 G 是从初始系统状态映射开始,可以到达的所有系统状态映射,简称可达图.

定义 23. 每个系统的输入变量和共享变量(包括数据变量和事件)的一种取值称为一个有效期为 0 的单外部转换,所有这些转换的集合记为 $ExTrans$.

若除 t 外,RTRSM*模型 M 的 V 中每个变量的有效取值都有穷,则 M 的可达状态映射有穷,此时,令$|GS_r|$为所有可达系统状态映射的个数,$|GS_m|$为所有系统状态映射个数,有$|GS_r|\leq|GS_m|$.

$G=(N,E)$,N 为节点集合,E 为边集,对于任意一个 $n\in N$,有 $n=(flag,q,enabled(n),FutureEvent(n))$,其中 $flag$ 为标志位,真表示该节点是由时钟转换得到的,否则为假,q 为该节点的状态映射域,$enabled(n)$为历史记录,其中的元素 $\tau(u)$表示转换 τ在 n 可执行,u 为其剩余有效时间.列表 $FutureEvent(n)$记录到达 n 之前执行的转换所预定的尚未到期的事件,其元素为$(e,lefttime)$,表示事件 e 将在 $lefttime$ 个时间单元后发生.

定义 24. 给定节点 $n=(flag,q,enabled(n),FutureEvent(n))$,定义当全局转换 $gt\subseteq enabled(n)$执行后得到的后继节点为 $n'=(flag',q',enabled(n'),FutureEvent(n'))$,有两种情况:

1) gt 不包含时钟转换,有

$$enabled(n')=delayed(n')\cup newlyenabled(n') \tag{1}$$

其中:

$$delayed(n')=\{\tau[u]\,|\,\tau[u]\in enabled(n)-gt\wedge q'(c_\tau)=\text{true}\} \tag{2}$$

$$newlyenabled(n')=\{\tau[u_\tau]\,|\,q'(e_\tau)=true\wedge\forall u.\tau[u]\notin delayed(n')\},\text{其中 } u_\tau\text{为 }\tau\text{的有效期} \tag{3}$$

$$must(n')=\{\tau[u]\,|\,\tau[u]\in enabled(n')\wedge u=0\} \tag{4}$$

2) gt 包含且只包含时钟转换 tick:公式(1),(3),(4)仍成立,有

$$delayed(n')=\{\tau[u-1]\,|\,\tau[u]\in delayed(n)\wedge u-1>0\} \tag{5}$$

对定义 6 进行扩充,得到全局转换关系.

定义 25(全局转换关系). 如果存在一个全局转换 gt,其源系统状态为 gs_1,目的系统状态 gs_2,则定义 gs_1,gs_2 满足转换关系ρ,记为$\rho(gs_1,gs_2)$,由 RTRSM*的确定性,该转换唯一.

算法基于 RTRSM*的操作语义.

输入:一个 RTRSM*模型 $M=(V,D,\theta,T)$及其初始状态映射 q_1;

输出:$G_M=(N,E)$,N 中所有的状态映射(域)集合等于从 q_1 出发,可到达的所有状态映射集合.E 为所有满足以下条件的边(n_s,gt,n_d):$gt\subseteq T$,$\rho(q_s\ q_d)$,其中 q_s,q_d 分别为 n_s 和 n_d 的状态映射域.

(1) 准备

A. $G=(nodes(G),edges(G))$,其中 $nodes(G)=\{n\}=\{(\text{true},q_1,\varnothing,\varnothing)\}$,$edges(G)=\varnothing$,所有节点都未标记;

B. $enabled(n)=\{\tau[u_\tau]|q_1(e_\tau)=\text{true}\}$,$must(n)=\{\tau[u]|\tau[u]\in enabled(n)\wedge u=0\}$;

C. 定义由全局转换得到新节点的过程:

```
CalNewG(gt:全局转换; g:G; sts, req:bool)
{   按照优先级原则消除 gt 中的不一致;
    执行 gt,得到 q′和 FutureEvent(n′);
    若(nodes(G)中无节点同时包括 q′和 FutureEvent(n′))且 sts=req
      则{由定义 23 计算相应数据项;
         nodes(g):=nodes(g)∪{n′}; }
    edges(G):= edges(G)∪[n,gt,n′]; }
```

(2) 计算

```
WHILE  nodes(G)中有一个未标记节点  DO
{ 从 nodes(G)中任选一个未标记节点 n,并加以标记;
  IF must(n)=∅
  THEN  IF enabled(n)≠∅
        THEN     //可执行事件触发转换
        { IF flag=false     // 是否不是由 tick 进入的系统状态
          THEN FOR i=0 to |newlyenabled(n)|  DO
                 FOR 由 i 个 newlyenabled(n)中局部转换组成的全局转换 gt DO
                   { flag′=false; CalNewG(gt,G, flag, flag′) }
          ELSE FOR i=0 to |enabled(n)|  DO
                 FOR 由 i 个 enabled(n)中局部转换组成的全局转换 gt DO
                   { CalNewG(gt,G, flag, flag);flag′=false; } }
        ELSE     //执行时钟转换:
          { FutureEvent(n′)={(e,n−1)|(e,n)∈FutureEvent(n)∧n>1};
            删除 FutureEvent(n′)中 n−1=0 的项(e,n),且 q′(e):=true;
            FOR i=0 to |ExTrans|  DO     // 执行外部转换:
              FOR 由 i 个单外部转换组成的全局转换 gt DO {CalNewG(gt,G, flag, flag);flag′=true }; }
  ELSE IF flag=false
       THEN FOR i=0 to |newlyenabled(n)−must(n)| DO
              FOR 由 must(n)+i 个(newlyenabled(n)−must(n))中转换组成的全局转换 gt DO
                {CalNewG(gt,G,flag,flag); flag′=false}
       ELSE FOR i=0 to |enabled(n)−must(n)| DO
              FOR 由 must(n)+i 个(enabled(n)−must(n))中转换共同组成的全局转换 gt DO
                { flag′=false; CalNewG(gt,G, flag, flag′) } }
```

(3) 将 $nodes(n)$中的 $flag$,$enabled(n)$和 $FutureEvent(n)$列表信息删除.得到 G_M.

3.1.2 算法讨论

算法的节点标识机制保证了没有节点会被重复访问.每个节点的 q 将 V 的每个变量映射到相应值域上,共 $|V|-1$ 个,设每个变量 v_i 有 $|v_i|$ 种可能的取值,$u_{\max}$ 为所有转换有效期的最大值,$|E|$ 为内部事件个数,$n_{\max}$ 为所有预定事件 $SetEvent(e,n)$ 中 n 的最大值,则 G 的最大节点数为 $Node_{\max}=2\times\prod_{i=1}^{|V|-1}|v_i|\times|T|\times u_{\max}\times 2^{|E|\times n_{\max}}$.算法终止,则要求各变量(除 t)的取值范围有穷.算法中第(2)步的循环最多执行 $Node_{\max}$ 次.考虑其循环体,时钟转换最多执行次数为 $C^0_{|ExTrans|}+C^1_{|ExTrans|}+\ldots+C^{|ExTrans|}_{|ExTrans|}=N_1$,操作转换的最多执行次数 $C^0_{|T|}+C^1_{|T|}+\ldots+C^{|T|}_{|T|}=N_2$.因此,最坏情况下,算法执行 $Node_{\max}\times(N_1$ 和 N_2 的最大数)次全局转换,即 G 的最大边数.

以上算法可以进行优化,比如:划分等价类,将某些变量无穷的取值划分为有穷多种有意义的取值类型(见第 3.2.3 节);节点信息可只包含原子状态信息,但性质判定时则应加上相应结构信息;上述算法对外部环境无任何限制,但实际中它通常存在一定约束,而考虑该约束的最为直接的方法就是加一个条件规约:$Env\rightarrow r$,若环境满足其限制,则该模型满足其性质[8].以上算法得到的是完整的可达图,但在有些情况下,要验证的性质只针对某子图,如针对某一时间区间或由特定属性确定,对于这些情况,可先抽取出相应子图[23,24],再基于该子图进行验证.

3.1.3 举 例

为了便于说明,我们通过时间限制较简单的温控系统(TCS)来说明状态可达图的构造过程.系统自动控制空调制冷或制热以调节室内温度.系统最初处于 Halt 状态,在按下“Run”按钮后开始运行.它根据室温和房门的打开情况控制空调工作:房门打开且超过 2 个时间单元,系统将停止空调工作;其他情况:室温高于 26°C 或低于 22°C,空调制冷或制热.按钮“Stop”被按下,则整个系统将停止运行.室温和房门的开关情况由传感器得到.约定该控制系统启动时:房门关闭,空调未运行.图 4 为用 SREE[20]编辑的 TCS 的状态图(为了便于说明,图中标出了实际并未显示的转换名,为 t1~t15).限于篇幅,图 5 只给出了极小一部分系统可达图.图中每节点的 q 包括:代表状态图中状态或事件是否活跃或发生的布尔量组成的串以及事件 T 的属性 tmp(所检测到的温度),其中布尔串的每位所代表的状态或事件分别为 CController,ON,AC, Door,Halt,off,refrigeration,heating,closed,open,run,stop,TO,T,D_open,D_Close.本例中,我们实际感兴趣的是温度的 3 种情况:小于 22°C、小于等于 26°C 而大于等于 22,以及大于 26°C,分别用值 1,2,3 表示;0 表示该属性值未知,即事件 T 未发生.这样即通过划分等价类对系统状态进行了简化.

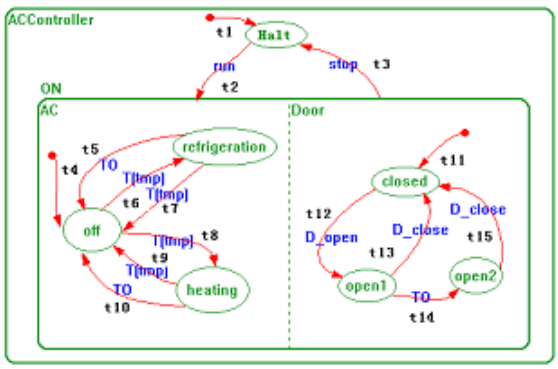

Fig.4 State transition diagram of TCS

图 4 温控系统状态图

3.1.4 性质判定

文献[23]给出了基于可达图的某些特定类型性质的判定过程.这些性质用 RITL 公式表示为

不变式:$\phi\rightarrow\varphi@[0,\infty)$; 最终式:$\phi\rightarrow\varphi?[0,\infty)$;$\phi_1\rightarrow\phi_2 U_{[0,\infty)}\phi_3$;$\phi_1\rightarrow\phi_2?[a,b]$;$\phi_1\rightarrow\neg\phi_2@[0,a)\wedge\phi_2@[a,b]$.

基于文献中的判定算法,可实现上述公式所描述的 RTRSM*模型性质的机械化验证.对于针对某一区间的性质验证,先根据 tick 的执行情况(执行次数)得到相关时间信息(代价边),使用子图抽取技术[23,24]得到相应子图,再使用相应判定过程即可.此外,还可以参考文献[7]给出的基于 Modechart 计算图的某些形式 RTL 公式的判定算法,以及如何利用计算图结构特点来缓解图中节点过多导致验证算法效率低下的方法.

3.2 一致性和完备性检查

对于 RTRSM*而言,规约一致性检查的重要性显而易见.对于完备性,有些未被说明的情况可能是因为对系统而言无意义而被故意省略,但也可能是遗漏了.所以,也应加以检测,以供分析员判断.

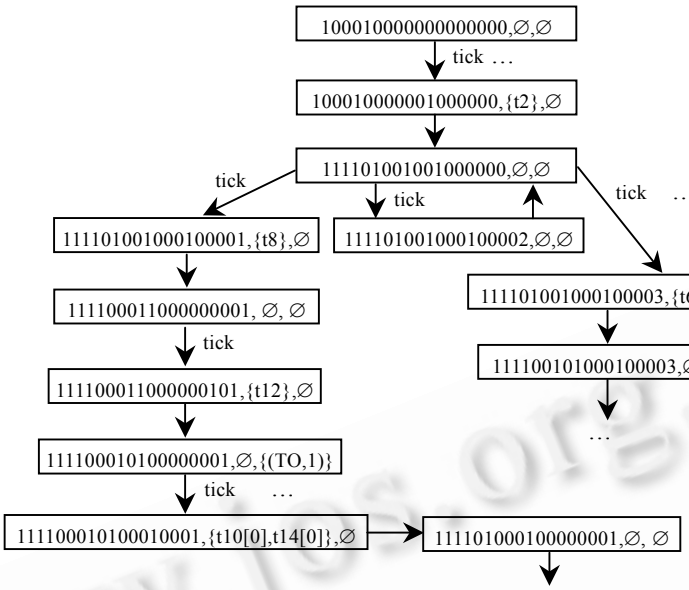

Fig.5 Part of the reachability graph of TCS

图 5 温控系统可达图(部分)

文献[25]指出基于状态的模型的一致性和完备性指的是:

- 每个状态必须有针对所有可能的输入事件的确定的行为(转换);
- 每个状态的所有迁出转换的卫士条件的逻辑或都必须永真,对于任何条件,都有一个可执行的转换;
- 每个状态的任意两个迁出转换的卫士条件的逻辑与都必须永假,对于每个可能的条件,每个状态只有一个可行的迁出转换.

对于 RTRSM*而言,还应考虑可同时执行的正交转换变量写冲突所导致的不一致.可参照文献[26]基于数据流和控制流的动态执行模型 DERTS 对操作写数据的处理,对具有相同写变量的正交转换:先判断其触发事件是否可能同时满足(考虑非事件),是,再判断其卫士条件的与是否永假,非永假则可能出现不一致.

对于基于状态的需求描述,也可基于系统状态可达图检查一致性和完备性.由第 3.1 节,在可达图构造过程中,当一个节点由一个全局转换有多个可能后继,包括一个局部状态有多个迁出转换可执行或同时执行的多个不同状态的迁出转换存在变量写冲突时,首先运用所规定的优先级原则进行处理,若仍有多个后继,则说明存在不一致.若一个节点的 q 中有事件对应的映射为真,但该节点的 *enabled* 为空,则说明没有定义对该情况的处理.该方法最大的问题在于状态空间爆炸问题.

3.3 两语言的PVS嵌入及规约的模拟执行

即使有抽象等方法,系统状态有穷的要求仍限制了模型检测的适用范围,而且,状态空间爆炸问题也限制了该方法的使用.定理证明作为另一重要的形式化验证方法,使用语言的抽象程度较高,可用于无穷状态系统,但受到语言表达能力和逻辑系统判定性问题的影响,且除了问题复杂性因素以外,手工验证难度大,易出错,也使其迫切需要具有较强推演证明功能和较好用户界面的自动或半自动工具的支持,但该类工具的开发极为艰巨而专业.一种代价较小的实现方法是将某语言(源语言)嵌入已有验证工具,从而利用该工具实现对源语言规约的验证.我们已将 RTRSM*和 RITL 通过语义编码嵌入基于高阶逻辑和相继式演算,具有丰富的类型理论,高效且较易用的形式化验证工具 PVS[12],从而可利用它来验证相应需求规约,同时,也对两语言本身的语义定义进行了检测和验证.所得到的 PVS 编码主要包括:数据类型定义,描述微步和宏步执行条件及结果的函数 NEXTStatus,以及关于状态变迁、初始条件、时间永不后退且必将前进等基本性质的公理等,要验证的性质则表示为需要验证的定理.同样,通过温控系统对以上工作,包括性质验证过程,予以说明和检验[21].

对于 ERS 而言,其需求规约的原型化检测要求:基于可执行的需求描述语言;具有模拟执行的形式化语义基础,使得模拟执行在视觉效果上可以预计,在语义基础上可以验证;提供环境模拟机制,处理好与外部环境的关系,使其能脱离实际物理环境.需求工程环境 SREE[20]为支持 RTRSM 的辅助需求定义和检测的集成工具环境,由需求规约的编辑、规则自动生成、规约检测和文档生成 4 大部分组成,其规约检查除静态的语法和语义检测,除了包括基于构图覆盖的需求检测以外,还支持所得到的系统模型的原型化验证.相对于其他用于需求工程的模拟执行工具[27],除了支持的语言不同以外,其主要特点一是实现方式,即由其模拟执行控制器(SC)解释执行 SRS(实际为规则的执行);二是外部环境的模拟方式,包括以交互或批处理的方式由用户控制外部环境的模拟和由专门的动画演示程序实现外部环境的模拟(半交互或动画程序全自动).前者给用户较大的主动权并显示丰富的执行信息;后者则可视性极强.可把 SREE 的规则自动生成器理解为可由 SC 解释执行的代码的自动生成器,基于 SRS,只需再提供输入、输出转换接口,将从实际环境中采集到的控制子系统所需的现场信息转换为 SC 所能接收的输入,并将 SC 产生的输出(处理结果)作为控制信息,转换为实际环境所能接收的控制命令,从而由 SRS 和 SC 可实现现场控制.也就是 SC 加上用户输入的由状态图和模板信息组成的需求实际上构成了 ERS 的原型.图 6 即为所实现的由状态图显示窗口和动画演示程序运行视图组成的机器人装配系统的动画演示模拟执行用户界面.要实现 RTRSM*的模拟执行,只需对 SC 和相应的动画演示程序稍加扩展即可.

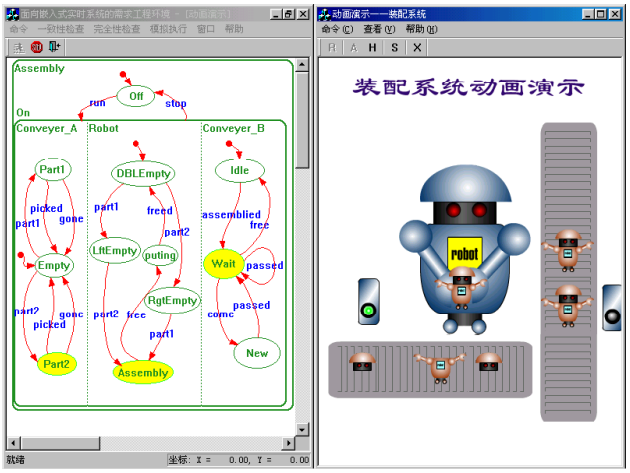

Fig.6 User interface of the assembly system's simulation with animation demo system

图 6 机器人装配系统的动画演示模拟执行用户界面

4 结束语

模板和规则的使用是 RTRSM*和 RTRSM 区别于其他相关工作的主要特点之一:模板保证系统内部数据的封闭性和可继承性,具有合成性;规则增强了模型的严密性,且是归约原型化检测的直接基础[20].具有转换有效期和事件预定机制的 HCA 既能较好支持复杂系统交互性的建模,在保持图形化优点的同时,又比大部分同样起源于 Statecharts 的建模语言具有更强的时间描述能力.逻辑 RITL 作为 RTRSM*的性质描述语言,能自然而全面地描述模型性质,包括实体和时间性质.作为适合 ERS 的需求描述符号系统,RTRSM*和 RITL 支持具有一定可读性和易用性、无二义性、结构化、可合成、可执行并具有严格形式语义的良好的需求规约.除了两语言的形式化定义以外,我们对相应的规约检测方法和技术也进行了研究,给出了状态可达图的构造算法及相应性质验证方法,实现了两语言的 PVS 语义嵌入,开发了支持规约模拟执行的辅助工具 SREE,并通过机器人自动装配系统和火电厂锅炉炉水自动防垢除垢控制系统等实例对该工具进行了检验.SREE 相对于最具有代表性的相关工作——基于 Statecharts,Activity Chart 和 Structure Chart 的商业工具 Statemate[9],除了归约的语法和静态语义检查以外,两者都支持规约的交互式和自动的模拟执行,但具体实现机制不同[21].Statemate 提供模拟控制语言(simulation control language)来预定义模拟输入序列,并能实现部分代码生成.此外,它与已有的模型检查技术相结合,并能基于所得到的设计模型自动生成测试用例[10],这些是 SREE 所未实现的.而 SREE 所提供的 COM 组件的调用机制[21]则是 Statemate 所不具备的,其所提供的模型执行信息可视性也较强.

除了完整的 RITL 公式的判定算法以外,我们将进一步开展的工作还包括所作形式化验证工作的工具实现,其他检测技术的设计、实现以及更多的大而复杂的应用实例检验等.

致谢 中国科学院软件研究所的王永吉研究员阅读了本文并提出了修改意见,在此我们表示感谢.

References:

[1] Harel D. Statecharts: A visual approach to complex systems. Science of Computer Programming, 1987,8(3):231~274.

[2] Heitmeyer C, Kirby J, Labaw B, Bharadwaj R. SCR*: A toolset for specifying and analyzing software requirements. In: Proc. of the 10th Int'l Conf. on Computer-Aided Verification. 1998. 526~531. http://citeseer.ist.psu.edu/heitmeyer98scr.html

[3] Alur R, Dill DL. A theory of timed automata. Theoretical Computer Science,1994,126(2):183~235.

[4] Henzinger TA, Manna, Z and Pnueli, A. Timed transition systems. In: Bakker JW, Huizing C, Roever WP, Rozenberg G, eds. Real-Time Theory in Practice. LNCS 600, Springer-Verlag, 1992. 226~251. http://ftp.cs.stanford.edu/cs/theory/luca/zmpapers.html

[5] Alur R, Henzinger TA. A really temporal logic. Journal of the ACM, 1994,41(1):181~204.

[6] Tang ZS, *et al*. Temporal Logic Programming and Software Engineering. Beijing: Science Press, 2002 (in Chinese).

[7] Stuart DA. Implementing a verifier for real-time systems. In: Proc. of the 11th IEEE Real-Time Systems Symp. 1990. IEEE Computer Society Press, 1990. 62~71. http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=128730

[8] Ostroff JS. Specifying and verifying real-time reactive systems in TTM/RTTL. Technical Report, CS-ETR-94-08, Toronto: Department of Computer Science, York University, 1994.

[9] Harel D, Lachover h, Naamad A, Pnueli A, Politi M, Sherman R, Trauring AS, Trakhtenbrot M. STATEMATE: A working environment for the development of complex reactive systems. IEEE Trans. on Software Engineering, 1990,16(4):403~414.

[10] Bohn J, Damm W, Klose J, Moik A, Wittke H. Modeling and validating train system applications using statemate and live sequence charts. In: Ehrig H, Krämer BJ, Ertas A, eds. Proc. of the Conf. on Integrated Design and Process Technology (IDPT 2002). Society for Design and Process Science, 2002.

[11] Holzmann GJ. The spin model checker. IEEE Trans. on Software Engineering, 1997,23(5):279~295.

[12] Shankar N. PVS: Combining specification, proof checking, and model checking. In: Srivas M, Camilleri A, eds. LNCS 1166, Springer-Verlag, 1996. 257~264. http://citeseer.ist.psu.edu/246180.html

[13] Kesten Y, Pnueli A. Timed and hybrid statecharts and their textual representation. In: Formal Techniques in Real-Time and Fault Tolerant Systems. LNCS 571, Springer-Verlag, 1991. 591~620. http://citeseer.ist.psu.edu/kesten92timed.html

[14] Douglass BP. Real-Time UML: Developing Efficient Objects for Embedded Systems. 2nd ed. Berkeley: Addison Wesley Longman, 1998.

[15] Selic B, Gullekson G, Ward PT. Real-Time Object-Oriented Modeling. New York: John Wiley & Sons, 1994.

[16] Cheng BHC, Campbell LA, McUmber WE, Stirewalt REK. Automatically detecting and visualizing errors in UML diagrams. Requirements Engineering Journal, 2002,7(4):264~287.

[17] Dong W, Wang J, Qi ZC. An approach of model checking UML Statecharts. Journal of Software, 2003,14(4):750~756 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/750.htm

[18] Litvak B, Tyszberowicz S, Yehudai A. Behavioral consistency validation of UML diagrams. In: Proc. of the 1st Int'l Conf. on Software Engineering and Formal Methods (SEFM 2003). Brisbane, 2003. http://csdl.computer.org/comp/proceedings/sefm/2003/1949/00/19490118abs.htm

[19] Alur R, Henzinger TA. Logics and models of real time: A survey. In: LNCS 600. Springer-Verlag, 1992. 74~106. http://citeseer.ist.psu.edu/alur92logics.html

[20] Shu FD, Wu GQ, Wang M. Embedded real-time software-oriented requirements engineering environment—SREE. Computer Science, 2002,29(4):4~8 (in Chinese with English abstract).

[21] Shu FD. Requirements specification and verification of embedded real-time software [Ph.D. Thesis]. Wuhan: Wuhan University, 2003 (in Chinese with English abstract).

[22] Heitmeyer CL, Jeffords RD, Jeffords RD. A benchmark for comparing different approaches for specifying and verifying real-time systems. In: Proc. of the 10th Int'l Workshop on Real-Time Operating Systems and Software. New York, 1993. http://chacs.nrl.navy.mil/publications/CHACS/1993/index1993.html

[23] Ostroff JS. Deciding properties of timed transition models. IEEE Trans. on Parallel and Distributed Systems, 1990,1(2):170~183.

[24] Campos S, Clarke EM, Grumberg O. Selective quantitative analysis and interval model checking: Verifying different facets of a system. Formal Methods in System Design, 2000,17(2):163~192.

[25] Heimdahl MPE, Leveson NG. Completeness and consistency in hierarchical state-based requirements. IEEE Trans. on Software Engineering, 1996,22(6):363~377.

[26] Wu GQ, Shu FD. Requirements specifications checking of embedded real-time software. Journal of Computer Science and Technology, 2002,17(1):56~63.

[27] Schmid R, Ryser J, Berner S, Glinz M, Reutemann R, Fahr E. A survey of simulation tools for requirements engineering. Technical Report, Zurich: Department of Information Technology, University of Zurich, 2000.

附中文参考文献:

[6] 唐稚松,等.时序逻辑程序设计与软件工程.北京:科学出版社,2002.

[17] 董威,王戟,齐治昌.UML Statecharts 的模型检验方法.软件学报,2003,14(4):750~756. http://www.jos.org.cn/1000-9825/14/750.htm

[20] 舒风笛,毋国庆,王敏.面向嵌入式实时软件系统需求工程环境——SREE.计算机科学,2002,29(4):4~8.

[21] 舒风笛.嵌入式实时软件系统的需求规约与验证[博士学位论文].武汉:武汉大学,2003.