软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(8):2815−2838 [doi: 10.13328/j.cnki.jos.006600]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向 CPS 时空约束的资源建模及其安全性验证方法*

陈小颖[1], 祝 义[1,2], 赵 宇[1], 王金永[2]

[1](江苏师范大学 计算机科学与技术学院, 江苏 徐州 221116)

[2](高安全系统的软件开发与验证技术工业和信息化部重点实验室(南京航空航天大学), 江苏 南京 210016)

通信作者: 祝义, E-mail: zhuy@jsnu.edu.cn

摘 要: 信息物理融合系统 CPS (cyber physical system)是在环境感知的基础上, 集合物理与计算的系统, 可以实现与环境的智能交互. CPS 信息物理空间的不断变化, 对 CPS 资源安全性造成一定的挑战. 因此, 如何研究这一类由时空变化而导致的 CPS 资源安全性问题成为关键. 针对该问题, 提出了面向 CPS 时空约束的资源建模及其安全性验证方法. 首先, 在 TCSP (timed communicating sequential process)的基础上扩展资源向量, 提出了时空资源通信顺序进程 DSR-TCSP (duration-space resource TCSP), 使其能够描述 CPS 拓扑环境下的资源; 其次, 从时空约束的资源安全性需求中获取时间安全需求, 通过 DSR-TCSP 的时间属性验证算法对时间安全需求进行验证; 再次, 将满足时间安全需求的模型转换为偶图与偶图反应, 并输入到偶图检验工具 BigMC 中, 验证其物理拓扑安全需求, 对没有通过验证的反例, 修改 DSR-TCSP 模型, 直至满足所提出的安全需求; 最后, 通过一个驾驶场景实例, 验证该方法的有效性.

关键词: 信息物理融合系统; 进程代数; 形式化验证; 时空约束; 资源安全性

中图法分类号: TP311

中文引用格式: 陈小颖, 祝义, 赵宇, 王金永. 面向 CPS 时空约束的资源建模及其安全性验证方法. 软件学报, 2022, 33(8): 2815−2838. http://www.jos.org.cn/1000-9825/6600.htm

英文引用格式: Chen XY, Zhu Y, Zhao Y, Wang JY. Modeling and Safety Verification Method for CPS Time and Topology Constrained Resources. Ruan Jian Xue Bao/Journal of Software, 2022, 33(8): 2815−2838 (in Chinese). http://www.jos.org.cn/1000-9825/6600.htm

Modeling and Safety Verification Method for CPS Time and Topology Constrained Resources

CHEN Xiao-Ying[1], ZHU Yi[1,2], ZHAO Yu[1], WANG Jin-Yong[2]

[1](School of Computer Science and Technology, Jiangsu Normal University, Xuzhou 221116, China)

[2](Key Laboratory for Safety-Critical Software Development and Verification (Nanjing University of Aeronautics and Astronautics), Ministry of Industry and Information Technology, Nanjing 210016, China)

Abstract: CPS (cyber physical system) combines physics and computation on the basis of environment perception and can realize intelligent interaction with the environment. However, the constant change of cyber physical space poses some challenges to the safety of CPS resources. Therefore, how to study this kind of CPS resource safety problems caused by topology and time changes becomes the key. This study proposes a CPS-oriented resource modeling and safety verification method to solve this problem. Firstly, on the basis of TCSP (timed communicating sequential process), resource vector is extended and DSR-TCSP (duration-space resource TCSP) is proposed, enable it to describe resources in the CPS topology. Secondly, the time safety requirements are obtained from the resource safety requirements of space and time constraints, and verified by the time verification algorithm of DSR-TCSP. Thirdly, the model meeting the

* 基金项目: 国家自然科学基金(62077029); 南京航空航天大学基本科研业务费科研基地创新基金(NJ2020022); 未来网络科研基金(FNSRFP-2021-YB-32); 徐州市应用基础研究计划(KC19004); 江苏省研究生科研创新计划(KYCX20_2380); 江苏省研究生科研创新计划(KYCX20_2384)

本文由“形式化方法与应用”专题特约编辑陈立前副教授、孙猛教授推荐.

收稿时间: 2021-09-05; 修改时间: 2021-10-14; 采用时间: 2022-01-10; jos 在线出版时间: 2022-01-28

time safety requirements is converted to the reaction of bigraphs and bigraphs reactive system, and the model is input into the bigraphs testing tool BigMC to verify its physical topology safety requirements. For the counter examples that do not pass the verification, the DSR-TCSP model is modified until the proposed safety requirements are met. Finally, a driving scenario is given to verify the effectiveness of the proposed method.

Key words: cyber physical system; process algebra; formal verification; space-time constraint; resource safety

信息物理融合系统 CPS (cyber physical system)是深度融合了计算、网络通信及控制设备的复杂系统, 使客观存在的事物具有自主计算的能力, 它通过计算进程和物理进程相互影响的反馈循环实现深度融合和实时交互来增加或扩展新的功能[1], 由于其人机交互以及环境的特殊性而产生一种特殊的资源: 时空约束下的资源, 例如: 停车场中的停车位、线路中的一段铁轨、系统中的一条数据、移动设备的一条消息等. 这一类特殊资源的安全性受到时间与物理拓扑空间的约束, 在该时间与空间的约束下, 若事件的执行不满足时空约束, 就会使资源进入不安全的状态, 影响着 CPS 的安全性.

在一些安全关键的 CPS 系统中, 如飞机控制系统、列车控制系统等, 安全性出现问题会产生严重后果. 作为信息物理融合系统的安全的影响之一, 信息物理空间下的资源安全性一直是 CPS 安全研究的热点问题. 时间与拓扑空间的不断变化, 会对资源安全性造成一定威胁, 甚至会使 CPS 产生严重的后果, 所以急需一种安全性验证的方法对 CPS 资源安全性进行保证. 因此, 如何验证在信息物理空间中时间与拓扑空间变化下 CPS 资源安全以保障 CPS 的安全性是当前面临的挑战.

近年来, 针对 CPS 安全性验证方面的研究也取得了许多成果. 文献[2]针对 CPS 的时空及非功能属性, 提出一种面向方面的时空 Petri 网建模方法以提高 CPS 的可靠性和可维护性. 文献[3]将领域环境模型组合到运行时验证过程中. 文献[4]研究时间与空间的一致性问题对 CPS 安全性的影响, 这些研究集中于非功能属性时间等方面的研究, 将时间等作为一种资源对系统进行验证. 文献[5]针对 CPS 具体任务进行资源调用时存在多种调度方案的问题, 提出了基于智能规划的 CPS 任务-虚拟资源调度机制, 其对虚拟资源的安全性调度问题展开研究, 对时间与空间考虑较少. 文献[6−8]对 CPS 中的能源进行管理以实现对 CPS 中的能耗估计. 文献[9]提出一个系统的解决方案, 以指定信息物理空间的安全策略, 确保信息对象和物理对象始终被安全处理, 该方法提出了拓扑空间对 CPS 安全性的影响, 但时间对 CPS 系统安全性的影响考虑较少. 文献[10]提出了一种在线社交网络时空访问控制模型及其可视化验证, 结合自主访问控制的优点, 使用形式语言描述基于时空和实时访问的访问控制规则. 与以上工作相比, 本文是对时空约束的资源这一类特殊资源的建模与验证, 对该资源随着时间与物理拓扑空间的变化进行建模与验证, 是对已有工作的有效补充,具有一定的研究意义.

通信顺序进程 CSP(Communicating Sequential Process)[11]是一种应用非常广泛的进程代数方法. 它能够很好解决进程同步、异步等问题.时间通信顺序进程 TCSP(timed communicating sequential process)[12]是在 CSP 上扩展了相应的时间因素, 能够适应现有 CPS 的开发, 因此也成为研究 CPS 的重要方法. 但是 TCSP 的空间的描述能力有限, 尤其是信息物理空间中的物理拓扑空间. 此外, TCSP 还缺少对资源的描述能力. 因此, 需要对 TCSP 进行扩展, 使其能够对物理拓扑空间以及信息物理空间中的资源进行描述, 进而可以对信息物理空间下 CPS 资源安全性进行验证.

本文针对信息物理空间中 CPS 资源安全性验证问题, 提出面向 CPS 时空约束的资源建模及其安全性验证方法. 技术路线如图 1 所示, 该方法首先在 TCSP 上扩展时空资源条件执行算子与时空资源条件中断算子, 提出时空资源通信顺序进程 DSR-TCSP, 使其具有描述 CPS 物理拓扑空间与资源的能力; 其次从时空约束的资源安全性需求中获取时间安全需求, 通过 DSR-TCSP 的时间属性验证算法对时间安全需求进行验证; 再次将满足时间安全需求的模型转换为偶图与偶图反应, 并输入偶图检验工具 BigMC 中, 将获取的物理拓扑安全需求也输入到 BigMC 进行验证, 对没有通过验证的反例, 修改 DSR-TCSP 模型, 直至满足所提出的安全需求; 最后得到满足时空约束下资源安全性需求的 DSR-TCSP 模型. 本文通过一个驾驶场景的实例来说明该方法的有效性.

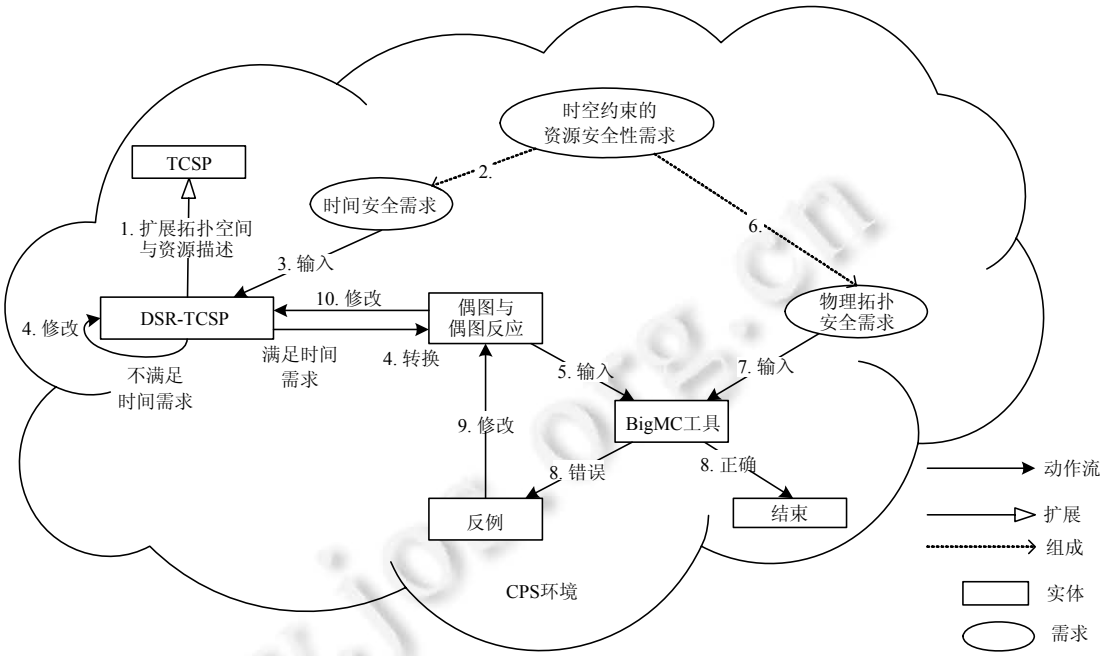


图 1 技术路线

1 预备知识

1.1 资源与资源安全性

我们关注的是时空约束下的资源, 它是一类特殊的资源, 例如停车场中的停车位、线路中的一段铁轨、系统中的一条数据、移动设备的一条消息等.

定义 1(资源安全性). 在 CPS 时间与空间约束下, 事件的执行不满足时空约束就会对这类资源的安全性造成威胁, 这一类情况统一称为时空约束下的资源安全性.

例如, 若要描述整栋大楼的物理拓扑模型, 则将整个大楼作为一个根节点, 将大楼中的房间作为其子节点, 从而形成一个描述该物理拓扑位置的树. 对于每个物理位置域集合 $\boldsymbol{POS}:=\{p_1,p_2,...,p_m \mid m\in \mathrm{N}^+\}$, 表示建筑分布中空间区域的集合. $p_i(i\in \mathrm{N}^+)$为某个具体的空间区域, 称物理位置域. 物理位置域的节点之间具有一定的包含关系. 信息实体表示相应的数据的信息. 信息空间中包含信息实体和信息实体所在的位置. 信息位置域集合 $\boldsymbol{CPOS}:=\{cp_1,cp_2,...,cp_n \mid n\in \mathrm{N}^+\}$. $cp_i(i\in \mathrm{N}^+)$为具体的信息空间. 如具体的记录 record 或者一个承载数据的软件系统 system.对于物理拓扑环境的建模, 我们使用层次化来描述, 而对运动的物理实体与信息实体的位置使用对实体的坐标位置捕捉, 进而向构造的物理层次拓扑模型映射, 以获得实体所在的物理位置域或信息位置域.

定义 2(物理位置域的包含关系). 若位置域在层次结构中, p_i 节点是 p_j 节点的父节点, 则 p_i 包含 p_j, 也可叫作 p_j 包含于 p_i, 写作 $p_j\subseteq p_i$, 表示为 $p_i(p_j)$.

定义 3(物理位置域与信息位置域的包含关系). 若信息域$\{cp_k|cp_k\in CPOS,k\in \mathrm{N}^+,k\leqslant m\}$在物理域$\{p_r|p_r\in POS, r\in \mathrm{N}^+,r\leqslant n\}$中, 则表示为 $p_r(cp_k)$.

1.2 偶图和偶图反应系统

偶图提供了位置变化的图形化标识, 其更直观表现了物理位置的变化. 接下来简单介绍偶图与偶图反应的基本概念以及偶图工具 BigMC.

(1) 偶图包含: 位置图与连接图. 位置图是一个森林, 其根节点为区域, 节点的嵌套关系可以通过位置图

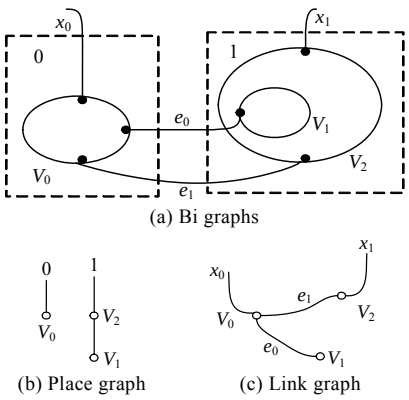


图 2 偶图分析

表示出来. 连接图是一个超图, 它的节点与位置图相同, 除节点外, 其还包括连接节点的边. 端口(port)是边与节点的连接. 如图2中, 图2(a)是偶图F, 图2(b)和(c)分别为F的位置图、连接图. 图2(a)中共有两个区域 (region), 用虚线框表示分别为0, 1. V_0, V_1, V_2表示节点标识 (node), V_1与V_2节点是嵌套关系.端口(port)在图中为黑色实心圆点.

定义 4. 偶图的位置图 $F^P=(V,ctrl,prnt):m\to n$.

m和n分别代表地点数与区域数. V为节点集合, $ctrl:=V\to K$, 表示V到K的映射, 控制节点K用于描述节点V的特性: 原子节点与复合节点, 前者可以有相应的反应规则, 后者不允许有反应规则. $prnt:=m\uplus V\to V\uplus n$.其中$\uplus$表示两个互不相交的集合的并(union), 箭头指向节点嵌套节点. 图2中位置图为F^P:0→2.

定义 5. 偶图的连接图 $F^L=(V,E,ctrl,link):X\to Y$.

X和Y分别为内部名与外部名, V为节点集合, E为连接边集合, $ctrl:=V\to K$同上. $link:=X\uplus Por\to E\uplus Y$ 表示节点上的端口和与链接连接的映射, 其中Por是端口集合, 图2中连接图为$F^L:\phi\to\{x_0x_1\}$.

定义 6. 偶图 $F=(V,E,ctrl,prnt,link):(m,X)\to(n,Y)$.

偶图由 $F^P=(V,ctrl,prnt):m\to n$ 和 $F^L=(V,E,ctrl,link):X\to Y$ 组成, 其中, $\langle m,X\rangle$称为内部界面(inner face), $\langle n,Y\rangle$称为外部界面(outer face). 图 2 中偶图为 $F^L:\langle 0,\phi\rangle\to\langle 2,\{x_0x_1\}\rangle$.

(2) 项语言

偶图的图形直观描述了物理位置的变化, 但是其很难被计算机理解, Milner等人提出了代数系统[13]. 表1为部分项语言(term language)表示. 项语言以代数的方式来推演系统完整性等性质.

表 1 项语言

Term language expression	Meaning
$R\|\|T$	Juxtaposition of roots
$R\|T$	Juxtaposition of nodes
$R\circ T$	Composition
$R.T$	Nesting(R contains T)
$/x.R$	R with outer name x replaced by an edge
x/y	Connection inner names y to outer name x

(3) 偶图反应系统(bigraphs reactive system, BRS)

偶图反应系统表示为$redex\to reactum$, 箭头前为反应物, 箭头后称为生成物. 它是一个动态的过程, 可以根据具体场景定义相应的反应规则, 从而可以改变当前偶图的结构, 此时, 将反应物偶图就根据当前的反应规则变成了生成物偶图. 如图3是一个反应规则, 左右两边分别为反应物和生成物, 该反应规则表示为 $C[x_0].(P)\,|\,D[x_1]\to C[x_0].(D[x_1]\,|\,P)$. 该反应规则表示带有连接x_1的对象D进入带有连接x_0的对象C中,在整个的变化中, 连接关系依然保持. 当偶图或者当前的部分偶图与反应规则中$redex$匹配时, 那么当前反应物就按照反应规则变成相应的生成物.

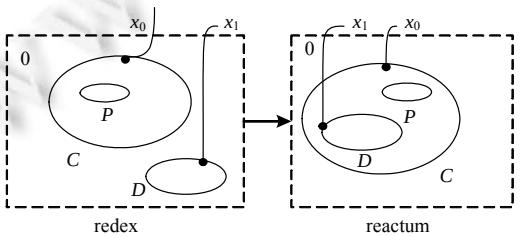


图 3 反应规则

(4) 偶图建模工具 BigMC

偶图与偶图反应系统有很多工具可以支持, 比如 BigRed[14]和 BigMC[15]等. BigMC 是一种模型检查器, 运行于 BRS 上. BRS 是由 Milner 等人开发的一种形式化工具, 强调局部性和连通性的正交概念. 其语言与项语言较为相近, 能够更好地对偶图进行建模. 用 BigMC 对物理空间下的资源安全属性进行模型检验, 将安全属性用偶图描述为 PER, 模型检验即判断偶图 B 是否与 PER 匹配: 如果匹配, 则满足安全属性; 否则不满足安全属性 PER, 存在违反状态, 工具将产生违反属性的反例路径.

偶图与偶图反应能够对物理拓扑空间与物理拓扑空间的动态变化进行有效描述, 但是偶图不支持 CPS 时间属性的验证. CPS 的复杂性要求对系统的同步、异步、并发等问题都需要考虑, 并且由时间与物理拓扑空间变化带来的资源安全性也是影响 CPS 安全性的重点问题. 因此, 我们在进程代数 TCSP 上扩展资源与物理拓扑空间描述, 提供更为系统的 CPS 安全性验证的形式化模型.

2 时空资源通信顺序进程 DSR-TCSP

2.1 时空资源通信顺序进程语法

定义 7. DSR-TCSP.时空资源通信顺序进程 DSR-TCSP 可以定义为

$$P ::= STOP \mid SKIP \mid WAIT\ t \mid a \xrightarrow{(r,object)} P \mid P;Q \mid P \square Q \mid P \sqcap Q \mid P \overset{d}{\triangleright} Q \mid f(P) \mid P \setminus A \mid P\ {}_A\|_B\ Q \mid P \,|||\, Q \mid \mu X \cdot f(X) \mid$$
$$Con >> P(Con ::= SPACE \mid TIME \mid RES \mid Con1 \wedge Con2 \mid Con1 \vee Con2 \mid true) \mid$$
$$P \blacktriangleright Fin\_Con(Fin\_Con ::= SPACE \mid TIME \mid RES \mid Con1 \wedge Con2 \mid Con1 \vee Con2 \mid false) \vdash Q$$

其中,

- $STOP$: 停止, 表示中断进程;
- $SKIP$: 跳过, 表示进程除终止不做任何动作;
- $WAIT\ t$: 等待, 表示进程 t 时间后就终止;
- $a \xrightarrow{\vec{r},object} P$: 前缀操作, 表示对 $object$ 执行完事件 a 后执行进程 P, 使资源向量 $\vec{r}$ 发生改变, $\vec{r} := \langle PT,(t,t_{wait}),res\rangle$. 其中, t 与 t_{wait} 分别是执行时间与等待时间, res 表示在物理拓扑为 PT, 时间(t,t_{wait})下的资源. res 可为空, 为空时可以省略;
- $P;Q$: 顺序复合, 表示按照进程 P 和进程 Q 顺序执行;
- $P \square Q$: 外部选择, 由环境控制选择执行 P 进程还是 Q 进程;
- $P \sqcap Q$: 内部选择, 表示 P、Q 哪个进程执行决定于进程的内部;
- $P \overset{d}{\triangleright} Q$: 超时, 表示时间 d 内, P、Q 之间无通信;
- $f(P)$: 换标, 表示 P 进程事件可根据函数 f 映射成另一个名字, 结构不变;
- $P \setminus A$: 集合隐藏, 表示集合 A 事件隐藏, 不显示;
- $P\ {}_A\|_B\ Q$: 同步并发, $A \cap B$ 为 P、Q 共有的事件集, P 与 Q 在 $A \cap B$ 并发, 除 $A \cap B$ 外事件交叉进行;
- $P \,|||\, Q$: 异步并发, 执行的是 P 或 Q 的事件;
- $\mu X \cdot F(X)$: $F(X)$是包含进程变量 X 的一个前缀表达式;
- $Con>>P(Con::=SPACE|TIME|RES|Con1\wedge Con2|Con1\vee Con2|\text{true})$称为时空资源条件执行算子, 在 Con 条件成立时, 执行进程 P. Con 条件变量包含 3 个部分: 物理位置模型 $SPACE$、时间模型 $TIME$ 以及资源模型 RES. $SPACE=F_{judge}(F_{ptp}(x,y,z),l)$是一个物理位置域判断函数, 是点到域的映射函数. $F_{ptp}(x,y,z)=l$ 函数输入三维坐标点(x,y,z), 输出该对象所在的物理位置域 l. 如 $F_{ptp}(3,4,6)=office$, 即物体当前位置坐标为(3,4,6), 所在区域为 $office$. $F_{judge}(F_{ptp}(x,y,z),l)$判断当前三维坐标位置所在区域是否为 l: 若是, 则返回 true, 否则返回 false. 如 $F_{judge}(F_{ptp}(3,4,6),office)$=true, 即当前对象所处的位置为(3,4,6), 并且位置是 $office$; 相应地, $F_{judge}(F_{ptp}(3,4,6),corridor)$=false. 物体所处的三维坐标点$(x,y,z)$可以根据所处区域向 CPS 的物理拓扑空间区域映射. 当没有位置约束时, $SPACE$ 默认为 true. $TIME= F_t(t_{current}\in(t_i,t_j))$是一个

谓语动词. $t_{current},t_i,t_j$ 是时间点, (t_i,t_j)表示从 t_i 到 t_j 且包含 t_i,t_j 时间点的时间段, *TIME* 判断当前时间点 $t_{current}$ 是否在(t_i,t_j)中:若 $t_{current}\in(t_i,t_j)$, *TIME*:=true; 否则, *TIME*:=false. 接下来, 将 *TIME* 简写为(t_i,t_j). RES 形式为 *res*≡*n*. *res* 为执行该进程的资源条件, 其中, *c* 是一个实数, ≡∈{≥,>,=,≤, <}. *Con* 也包括条件的组合, *Con*1∧*Con*2 表示条件 *Con*1 与条件 *Con*2 同时满足, 用"∧"连接. *Con*1∨ *Con*2 表示条件 *Con*1 与条件 *Con*2 任一条件满足即可, 用"∨"连接. 例如($F_{judge}(F_{ptp}$(3,4,6),*office*)∧(8,17)∧(*printer*>1))>>*P*, 即当前对象所处物理位置域是 *office*, 时间在 8 点到 17 点之间并且资源 *printer* 数量大于 1 才能执行进程 *P*, 否则不执行 *P*;

- *P*▶*Fin_Con*(*Fin_Con*::=*SPACE*|*TIME*|*RES*|*Con*1∧*Con*2|*Con*1∨*Con*2|false) ├*Q* 称为时空资源条件中断算子, 其中, *SPACE*, *TIME*, *RES* 与时空资源条件执行算子中的模型相同, 分别为进程 *P* 终止执行时所涉及的空间、时间与资源满足的条件, 默认为 false, 可省略. 如果条件 *Fin_Con* 满足, 则中断正在执行的进程 *P* 进而执行进程 *Q*. 与时空资源条件执行算子相同, 条件同时满足用"∧"连接条件, 条件至少一个满足用"∨"连接条件. 如 *P*▶($F_{judge}(F_{ptp}$(3,4,6)=*office*)∨(17,24)) ├*Q* 则表示当进程 *P* 执行时, 对象所在物理位置域为 *office* 或者当前时间在 17 点到 24 点之间时, 终止 *P* 进程, 执行 *Q* 进程;

TCSP 操作 *a*?*x* 与 *a*!*x* 为进程代数基本操作,分别表示接收与发送.其中,*a* 为通道 channel,*x* 为消息.

2.2 DSR-TCSP的时空资源迁移系统

定义 8. TCSP 的语义[16]描述为一个时间迁移系统 $TTS_{TCSP}=\langle NODES,\sum_T,\rightarrow\rangle$,进程表示为节点集合 *NODES*. $\sum_T$ 是带延迟时间的事件集,即 $\{(t_0,a_0),(t_1,a_1)...(t_n,a_n)\}$,→是迁移关系,是一个三元关系,$\rightarrow\subseteq NODES\times\sum_T\times NODES$, $N_1\xrightarrow{(t,a)}N_2$,表示 N_1 执行事件 *a*,延迟 *t* 个时间单元变成 N_2 代表的进程.

定义 9. 一个时空资源迁移系统 DSRTTS 为 $DSRTTS_{DSR\text{-}TCSP}=\langle NODES,\Sigma_{(T,PT,RES)},\rightarrow\rangle$.

- *NODES* 为节点集合,表示各个进程;
- $\Sigma_{(T,PT,RES)}$是带延迟时间、物理拓扑空间变化以及时空约束下资源变化操作的事件集, 即$\{(pt_0,res_0,t_0,a_0),(pt_1,res_1,t_1,a_1),\ldots,(pt_n,res_n,t_n,a_n)\}$, 且$\Sigma_T\subseteq\Sigma_{(T,PT,RES)}$, 当 $pt_n=\varepsilon\wedge res_n=\varepsilon$时, $\Sigma_T=\Sigma_{(T,PT,RES)}$;
- →是迁移关系, 是一个三元关系, $\rightarrow\subseteq NODES\times\Sigma_{(T,PT,RES)}\times NODES$.

$N_1\xrightarrow{(t,pt,res,a)}N_2$ 表示 N_1 执行的进程执行事件 *a*, 延迟 *t* 个时间单元, 物理拓扑为 *pt* 且时空约束的资源为 *res* 时, N_1 进程变为 N_2 进程.

定义 10. DSR-TCSP 语义可以描述为一个时空资源迁移系统:

$$DSRTTS_{DSR\text{-}TCSP}=\langle NODES,\Sigma_{(T,PT,RES)},\rightarrow\rangle.$$

2.3 DSR-TCSP的操作语义

进程间的复合运算参照原有的进程组合定义[12]. 接下来仅列出与资源相关算子的操作语义, 由于篇幅原因, 省略了对称语义.

定义 11. DSR-TCSP 的操作语义.

(1) 时空资源条件执行算子 *Con*>>*P*(*Con*::=*SPACE*|*TIME*|*RES*|*Con*1∧*Con*2|*Con*1∨*Con*2|true)的操作语义.

$$\frac{F_{judge}(F_{ptp}(x,y,z),l)=\text{false}\vee TIME=\text{false}\vee(res\equiv n)=\text{false}\vee(Con1\wedge Con2)=\text{false}\vee(Con1\vee Con2)=\text{false}}{s\xrightarrow{(t,pt,res,a)}s}$$

$$\frac{F_{judge}(F_{ptp}(x,y,z),l)=\text{true}\wedge TIME=\text{true}\wedge(res\equiv n)=\text{true}\wedge(Con1\wedge Con2)=\text{true}\wedge(Con1\vee Con2)=\text{true}}{s\xrightarrow{(t,pt',res',a)}s'}$$

只有当 $F_{judge}(F_{ptp}(x,y,z),l)$为 true, 即当前实体所在位置三维坐标处于 *l* 域内, *TIME* 与 *res*≡*n* 为 true 时, 状态 *s* 则执行事件 *a*, 延迟 *t* 个时间单元, 物理拓扑与时空约束的资源更改为 *pt*′和 *res*′变为 *s*′状态, 否则不发生状态的迁移.

(2) 时空资源条件中断算子 *P*▶*Fin_Con*(*Fin_Con*:=*SPACE*|*TIME*|*RES*|*Con*1∧*Con*2|*Con*1∨*Con*2|false) ├*Q* 的操作语义.

$$\frac{F_{judge}(F_{ptp}(x,y,z),l)=\text{false}\vee TIME=\text{false}\vee(res\equiv n)=\text{false}\vee(Con1\wedge Con2)=\text{false}\vee(Con1\vee Con2)=\text{false}}{s\xrightarrow{(t,pt,res,a)}s}$$

$$\frac{F_{judge}(F_{ptp}(x,y,z),l)=\text{true}\wedge TIME=\text{true}\wedge(res\equiv n)=\text{true}\wedge(Con1\wedge Con2)=\text{true}\wedge(Con1\vee Con2)=\text{true}}{s\xrightarrow{(t,pt',res',a)}s'}$$

当中断条件 *Fin_Con* 为 *true* 时, 中断当前进程, 发生状态 *s* 到 *s'*的迁移, 否则状态不发生改变.

(3) 前缀操作 $\dfrac{-}{(a\xrightarrow{(\vec{r},object)}P)\xrightarrow{(a,\vec{r},object)}P}$ 表示在任何情况下, $a\xrightarrow{(r,object)}P$ 对 *object* 执行事件 *a* 后都是相同的 $\vec{r}$ 资源变化, 变成进程 *P*.

(4) 顺序组合 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'}{P;Q\xrightarrow{(a,\vec{r},object)}P';Q}$ 表示若进程 *P* 对 *object* 执行事件 *a* 后成为 *P'*, 则进程 *P*;*Q* 执行对 *object* 执行事件 *a* 后变成 *P'*;*Q*.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Con=\text{true}}{Con>>P;Q\xrightarrow{(a,\vec{r},object)}P';Q}$ 与 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Con=\text{false}}{Con>>P;Q\xrightarrow{(a,\vec{r},object)}P;Q}$ 表示若进程 *P* 对 *object* 执行事件 *a* 后成为 *P'*, 并且 *Con* 为 true 时, 则进程 *Con*>>*P*;*Q* 执行对 *object* 执行事件 *a* 后变成 *P'*;*Q*; 否则是 *P*;*Q*.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Fin\_Con=\text{true}}{P\blacktriangleright Fin\_Con\vdash Q;R\xrightarrow{(a,\vec{r},object)}Q';R}$ 与 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Fin\_Con=\text{false}}{P\blacktriangleright Fin\_Con\vdash Q;R\xrightarrow{(a,\vec{r},object)}P';R}$ 表示若进程 *P* 对 *object* 执行事件 *a* 后成为 *P'*, 进程 *Q* 对 *object* 执行事件 *a* 后成为 *Q'*, 并且 *Fin_Con* 为 true 时, 则进程 *P*▶*Fin_Con* ⊢*Q*;*R* 对 *object* 执行事件 *a* 后变成 *Q'*;*R*; 否则是 *P'*;*R*.

(5) 外部选择 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'}{P\square Q\xrightarrow{(a,\vec{r},object)}P'}$ 表示若进程 *P* 对 *object* 执行事件 *a* 后成为 *P'*, 则进程 *P*□*Q* 对 *object* 执行事件 *a* 后成为 *P'*, 表示外部选择执行进程 *P*.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Con=\text{true}}{Con>>P\square Q\xrightarrow{(a,\vec{r},object)}P'}$ 与 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Con=\text{false}}{Con>>P\square Q\xrightarrow{(a,\vec{r},object)}P}$ 表示若进程 *P* 对 *object* 执行事件 *a* 后成为 *P'*, 并且 *Con* 为 true 时, 则进程 *Con*>>*P*□*Q* 对 *object* 执行事件 *a* 后成为 *P'*; 否则还是 *P*, 表示外部选择执行进程 *Con*>>*P*.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Fin\_Con=\text{true}}{P\blacktriangleright Fin\_Con\vdash Q\square R\xrightarrow{(a,\vec{r},object)}Q'}$ 与 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Fin\_Con=\text{false}}{P\blacktriangleright Fin\_Con\vdash Q\square R\xrightarrow{(a,\vec{r},object)}P'}$ 表示若进程 *P* 对 *object* 执行事件 *a* 后成为 *P'*, 进程 *Q* 对 *object* 执行事件 *a* 后成为 *Q'*, 并且 *Fin_Con* 为 true 时, 则进程 *P*▶*Fin_Con* ⊢*Q*□*R* 对 *object* 执行事件 *a* 后成为 *Q'*; 否则为 *P'*, 表示外部选择执行进程 *P*▶*Fin_Con* ⊢*Q*.

(6) 内部选择 $\dfrac{-}{P\sqcap Q\xrightarrow{(\tau,\vec{r},object)}P}$ 表示任何情况下, 系统初始时选择进程 *P*.

(7) 超时 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'}{P\overset{d}{\triangleright}Q\xrightarrow{(a,\vec{r},object)}P'\overset{d-t}{\triangleright}Q}$ [*t*≤*d*]表示如果进程 *P* 在 *t* 时间内演化变为 *P'*, 且 *t*≤*d*,那么 $P\overset{d}{\triangleright}Q$ 在 *t* 时间内演化为 $P'\overset{d-t}{\triangleright}Q$.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Con=\text{true}}{Con>>P\overset{d}{\triangleright}Q\xrightarrow{(a,\vec{r},object)}P'\overset{d-t}{\triangleright}Q}$ 与 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Con=\text{false}}{Con>>P\overset{d}{\triangleright}Q\xrightarrow{(a,\vec{r},object)}P\overset{d-t}{\triangleright}Q}$ [*t*≤*d*]表示如果进程 *P* 在 *t* 时间内演化变为 *P'*, *Con* 为 true, 且 *t*≤*d*, 那么 $Con>>P\overset{d}{\triangleright}Q$ 在 *t* 时间内演化为 $P'\overset{d-t}{\triangleright}Q$; 否则为 $P\overset{d-t}{\triangleright}Q$.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Fin\_Con=\text{false}}{P\blacktriangleright Fin\_Con\vdash Q\overset{d}{\triangleright}R\xrightarrow{(a,\vec{r},object)}P'\overset{d-t}{\triangleright}R}$ 与 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Fin\_Con=\text{true}}{P\blacktriangleright Fin\_Con\vdash Q\overset{d}{\triangleright}R\xrightarrow{(a,\vec{r},object)}Q'\overset{d-t}{\triangleright}R}$

[*t*≤*d*]表示如果进程 *P* 在 *t* 时间内演化变为 *P'*, 进程 *Q* 在 *t* 时间内演化变为 *Q'*, *Fin_Con* 为 false 且 *t*≤*d*, 那么

$P\blacktriangleright Fin\_Con\vdash Q\overset{d}{\triangleright}R$ 在 t 时间内演化为 $P'\overset{d-t}{\triangleright}R$; 否则为 $Q'\overset{d-t}{\triangleright}R$.

(8) 同步并发

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'}{P\ {}_A\|_B\ Q\xrightarrow{(a,\vec{r},object)}P'\ {}_A\|_B\ Q'}$ [$a\in A\cap B$]表示在 $a\in A\cap B$ 的情况下, 若进程 P 与进程 Q 执行同步事件 a 后分别成为进程 P'与进程 Q', 则进程 $P_A\|_BQ$ 对 *object* 执行同步事件 a 后变成 $P'_A\|_BQ'$, 资源按照 $\vec{r}$ 变化.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Con=\text{true}}{Con>>P\ {}_A\|_B\ Q\xrightarrow{(a,\vec{r},object)}P'\ {}_A\|_B\ Q'}$ 与 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Con=\text{false}}{Con>>P\ {}_A\|_B\ Q\xrightarrow{(a,\vec{r},object)}P\ {}_A\|_B\ Q'}$ 表示在 $a\in A\cap B$ 的情况下, 若进程 P 对 *object* 执行事件 a 后变成 P', 进程 Q 对 *object* 执行事件 a 后成为 Q', 并且 *Con* 为 true, 则进程 $Con>>P_A\|_BQ$ 对 *object* 执行事件 a 后变成 $P'_A\|_BQ'$; 否则为 $P_A\|_BQ'$, 资源按照 $\vec{r}$ 变化.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ R\xrightarrow{(a,\vec{r},object)}R'\ Fin\_Con=\text{false}}{P\blacktriangleright Fin\_Con\vdash Q\ {}_A\|_B\ R\xrightarrow{(a,\vec{r},object)}P'\ {}_A\|_B\ R'}$

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ R\xrightarrow{(a,\vec{r},object)}R'\ Fin\_Con=\text{true}}{P\blacktriangleright Fin\_Con\vdash Q\ {}_A\|_B\ R\xrightarrow{(a,\vec{r},object)}Q'\ {}_A\|_B\ R'}$ 表示在 $a\in A\cap B$ 的情况下, 若进程 P, Q, R 对 *object* 执行事件 a 后分别变成 P', Q', R', 并且 *Fin_Con* 为 false 时, 则进程 $P\blacktriangleright Fin\_Con\vdash Q_A\|_BR$ 对 *object* 执行事件 a 后变成 $P'_A\|_BR'$; 否则为 $Q'_A\|_BR'$.

(9) 异步并发 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'}{P\|\|Q\xrightarrow{(a,\vec{r},object)}P'\|\|Q}$ 表示若进程 P 执行事件 a 后成为 P', 进程 $P\|\|Q$ 对 *object* 执行事件 a 后变成 $P'\|\|Q$.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Con=\text{true}}{Con>>P\|\|Q\xrightarrow{(a,\vec{r},object)}P'\|\|Q}$ 与 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Con=\text{false}}{Con>>P\|\|Q\xrightarrow{(a,\vec{r},object)}P\|\|Q}$ 表示若进程 P 对 *object* 执行事件 a 后成为 P'并且 *Con* 为 true 时, 则进程 $Con>>P\|\|Q$ 对 *object* 执行事件 a 后变成 $P'\|\|Q$; 否则为 $P\|\|Q$.

$\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Fin\_Con=\text{false}}{P\blacktriangleright Fin\_Con\vdash Q\|\|R\xrightarrow{(a,\vec{r},object)}P'\|\|R}$ 与 $\dfrac{P\xrightarrow{(a,\vec{r},object)}P'\ Q\xrightarrow{(a,\vec{r},object)}Q'\ Fin\_Con=\text{true}}{P\blacktriangleright Fin\_Con\vdash Q\|\|R\xrightarrow{(a,\vec{r},object)}Q'\|\|R}$ 表示若进程P执行事件a后成为P', 进程Q执行事件a后成为Q', 并且*Fin_Con*为false时, 则进程$P\blacktriangleright Fin_Con\vdash Q\|\|R$ 对 *object* 执行事件 a 后变成 $P'\|\|R$; 否则为 $Q'\|\|R$.

2.4 精化关系

下面定义子语义模型, 然后给出 DSR-TCSP 包含 TCSP 语义的定理证明, 证明 TCSP 的语义模型是 DSR-TCSP 的子语义模型.

定义 12. 设通信顺序进程 P_A 能够经过精化关系得到通信顺序进程 P_B, P_A 与 P_B 分别属于 A、B 类语义 M_A 与 M_B,那么称语义 M_B 是语义 M_A 的子语义模型[17]. P_A 与 P_B 的精化模型如下:

$$\frac{P_B\ \text{sat}\ S_B\ \text{in}\ M_B}{P_A\ \text{sat}\ S_A\ \text{in}\ M_A}(P_B\sqsubseteq P_A).$$

S_A 是 P_A 的语义, S_B 是 P_B 的语义.

定义 13. DSR-TCSP 所有可接受的语言是时空资源迁移序列的集合.

定理 1. 时间通信顺序进程 TCSP 的语义模型是时空资源通信顺序进程 DSR-TCSP 的子语义模型.

证明: 令 P_{TCSP} 是一个时间通信顺序进程. 由定义 8 知 TCSP 语义是一个 $TTS_{TCSP}=\langle NODES,\Sigma_T,\rightarrow\rangle$,它可接受的语言为 L,根据 TCSP 构造一个 DSR-TCSP, 由定义 10 得知, $P_{DSR\text{-}TCSP}$ 是一个 $DSRTTS_{DSR\text{-}TCSP}=\langle NODES,\Sigma_{(T,PT,RES)},\rightarrow\rangle$. 设 $P_{DSR\text{-}TCSP}$ 接受的语言为 L'. 此时, 在 L 中任取一个时间迁移序列 $R=\langle(t_0,a_0),(t_1,a_1),\ldots,(t_{n-1},a_{n-1})\rangle$, 在 L' 中都有唯一一个 $R'=\langle(pt_0,c_0,res_0,t_0,a_0),(pt_1,c_1,res_1,t_1,a_1),\ldots,(pt_{n-1},c_{n-1},res_{n-1},t_{n-1},a_{n-1})\rangle$ 与之对应, 因此, $\Sigma_T\subseteq\Sigma_{(T,PT,RES)}$. 当 $pt_n=\varepsilon\wedge res_n=\varepsilon$时, $\Sigma_T=\Sigma_{(T,PT,RES)}$. 因此, $P_{DSR\text{-}TCSP}$ 到 P_{TCSP} 是精化关系. □

由定理 1 得知, DSR-TCSP 功能属性使用 TCSP 工具完成. 接下来对扩展的时空资源性质进行分析, 以检

测 CPS 的资源安全性.

2.5 时间属性验证

首先, 验证资源安全性受时间安全需求的影响. 时间因素影响着资源的安全性. 若在资源允许的时间范围外使用资源, 则会对资源安全性造成威胁. 接下来, 我们通过一个时间属性验证算法对资源安全性的时间需求进行验证(如图 4 所示).

时间属性的验证为验证条件中的 *TIME* 模型, 时间属性验证算法通过一个深度优先算法对时空资源状态迁移图进行遍历. 检验当前时间是否在时间的需求(t_i,t_j)时间段内: 若满足当前的时间需求, 返回 true; 否则返回 false.

```
abnormal:=∅; cur_path={N0}; totalt:=0; curr_t:=currentime
  repeat
    ln:=last node in cur_path;   //取当前路径的最后一个节点
    if successor nodes of last node have been visited   //删除已经访问的子节点
      then delete last node of cur_path;
        totalt=totalt-curr_t;   //删除最后一个节点时总时间减去相关的边的时间
    else
      begin
      bn:=take a unvisited successor node of ln;   //取一个未被访问 ln 的孩子节点 bn
      totalt=totalt+curr_t   //将当前路径与该孩子节点迁移边的时间加入 totalt
      if time constraint(ti,tj) exists, totalt<ti or totalt>tj
        then abnormal =abnormal∪{en};   //从源节点到当前节点 bn 之后的时间值
                                          不在(ti,tj)时间段时, 将异常节点记下来
      else
      cur_path=cur_path∪{bn};
    end
  until cur_path=∅;
  if abnormal=∅ then
    return true;
  else return false;
```

图 4　时间属性验证算法

2.6 死锁定位

若系统违反时间安全需求而造成的死锁问题, 需要对时间迁移系统进行修正, 直至其各个节点满足时间安全需求. 对于该系统的修正, 首先通过死锁定位算法 *getdeadlock*(·)对违反时间安全需求节点进行定位, 然后通过死锁修改算法进行修改(如图 5 所示).

```
deadlock:= ∅; cur_path={N0};
repeat
  ln:=last node in cur_path;   //取当前路径的最后一个节点
  if successor nodes of last node have been visited   //删除已经访问的子节点
    then delete last node of cur_path;
  else
    begin
      bn:=take a unvisited successor node of ln;   //取一个未被访问 ln 的孩子节点 bn
      if bn=null   //即该未访问节点不存在孩子节点, 存在死锁现象
      deadlock=deadlock∪{ln}   //将 ln 节点放入 deadlock 集合中
      else
        cur_path=cur_path∪{bn};
    end
  until cur_path=∅;
  if deadlock=∅ then
    return true;
  else return false;
```

图 5　死锁定位算法 *getdeadlock*(·)

死锁定位算法是采用深度优先算法对整个时空资源迁移图进行遍历, 找出死锁节点. 在整个时空资源状态迁移图中, 死锁节点为不包含孩子节点的节点. 该节点后续无法继续执行, 因此需要对死锁节点进行定位, 以方便后续对死锁的修改操作.

2.7 死锁修改

对当前定位到的死锁节点 *deadlock* 集合中的修改主要包含 3 个操作: 给死锁节点添加一条边 *ed*; 将该节点删除, 即不选择此节点的方案; 添加一个错误处理节点 *en* 和边 *ed*, 从而修改死锁(如图 6 所示).

通过时间需求检验与死锁定位和修改后的 DSR-TCSP 模型满足时间安全需求, 接下来进行模型转换, 以验证物理拓扑安全需求.

G=curent state space graph; //*G* 为当前的状态迁移图
deadlock=*getdeadlock*(·).*deadlock*; //从死锁定位算法 *getdeadlock*(·)中获得死锁集合 *deadlock*
repeat
  dn:=a node in *deadlock*; //取当前 *deadlock* 中的一个节点
  a=choose *a* to deal with the deadlock of *dn*; //选择处理死锁节点 *dn* 的方式
  switch(***a***):
  case 0: add an edge *ed* in graph *G*, *G*=*G*∪{*ed*}; **break**;
  case 1: delete the deadnode *dn* from graph *G*, *G*=*G*/{*dn*}; **break**;
  case 2: add an error-handling node *en* and an edge *ed* in graph *G*, *G*=*G*∪{*en*}∪{*ed*}; **break**;
  delete node *dn* from *deadlock*; //处理后的节点 *dn* 从集合中删除
end
until *deadlock*=∅; //将所有的死锁节点都处理
return *G*;

图 6 死锁修改算法

3 DSR-TCSP 物理拓扑对应的资源安全性验证

为了验证 CPS 系统在信息物理空间中资源的物理拓扑安全需求, 通过使用 AMMA (ATLAS model management architecture)[18]平台定义 ATL 转换规则, 将 DSR-TCSP 转换为偶图与偶图反应, 并使用偶图工具 BigMC 进行模型检测, 对物理拓扑环境下时空所对应的资源的安全性进行验证.

3.1 转换的一致性验证保证

在模型转换方面, 如何保证模型转换前后的一致性一直都是一个非常重要的问题[19]. 本文从宏观和微观两个角度说明语义的一致性. 从宏观上需要 DSR-TCSP 满足 Bigraphs 语义要求, Bigraphs 和 DSR-TCSP 两模型都是离散的, 所以宏观上看有一致的语义. 微观上, 要求语言 A 概念集在语言 B 中有对等语义的概念集, 反之亦然[20]. 因此 DSR-TCSP 向 Bigraphs 的转换之前, 需要两者概念集的对等.

DSR-TCSP 模型元素并不是都能映射到 Bigraphs, 我们是将 DSR-TCSP 中的物理拓扑空间与物理拓扑空间的变化等元素转换为 Bigraphs, 对 DSR-TCSP 元模型重构使其在 Bigraphs 元模型上找到对应元素. 图 7(a)和(b)为 DSR-TCSP 和 Bigraphs 的元模型.DSR-TCSP 的规约 Specification 对应于 Bigraphs 的项语言 TermLanguage; DSR-TCSP 的变量 VariableDeclarations 与参数 parameter 也对应于 Bigraphs 中的变量与参数; SpaceType 是在 TCSP 上扩展的物理拓扑空间, 对应于 Bigraphs 的节点 NodeType; 物理空间之间的关系 SpaceRelation 对应于 Bigraphs 中的节点的关系 NodeRelation; DSR-TCSP 元模型中的操作算子执行 Operations 对应于 Bigraphs 中的偶图反应 Reaction.

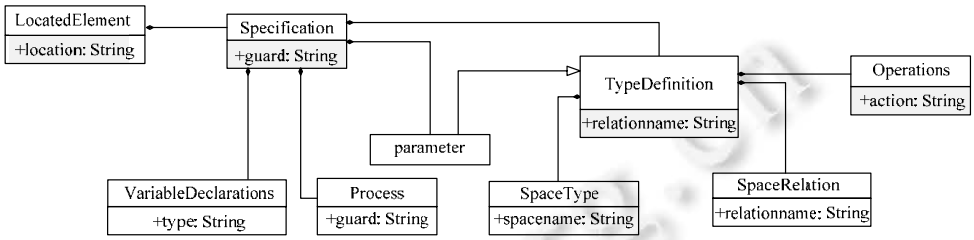
3.2 DSR-TCSP到偶图的转换规则

DSR-TCSP 到偶图的转换可以对应到转换过程如下.

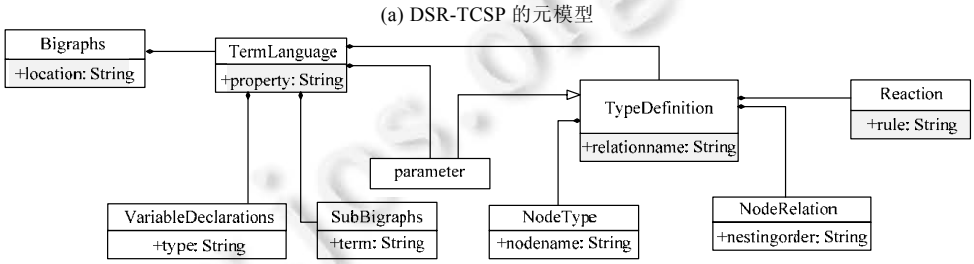
(1) *POS* 集合与 *CPOS* 集合中的物理位置转换成节点集合 *V*;
(2) 物理位置域的包含关系 $p_i(p_j)$与物理位置域和信息位置域的包含关系 $p_r(cp_k)$转换成节点的嵌套;
(3) 通信信道转换成相关的连接 link;
(4) 将 DSR-TCSP 中的 $a \xrightarrow{(r,object)} P$ 事件执行带来的物理拓扑空间以及时空约束的资源变化转换为偶图反应;
(5) 针对时空资源条件执行算子 $Con>>P(Con::=SPACE|TIME|RES|Con1\wedge Con2|Con1\vee Con2|\text{true})$与时空资源条件中断算子 $P\blacktriangleright Fin\_Con(Fin\_Con::=SPACE|TIME|RES|Con1\wedge Con2|Con1\vee Con2|\text{false}) \vdash Q$ 中的

SPACE 与 *RES* 转换成偶图反应规则的反应物(偶图).

(6) 由 DSR-TCSP 到偶图反应规则的映射, 根据具体的进程事件, 将事件集 ***A*** 中的事件 *a* 按照其物理拓扑位置的变化转换为具体的偶图反应规则.



(a) DSR-TCSP 的元模型



(b) Bigraphs 的元模型

图 7 DSR-TCSP 的元模型与 Bigraphs 的元模型

根据以上转换策略, 我们定义了具体转换规则.

转换规则 1. 物理位置到偶图的反应规则.

V: *POS*, *CPOS*⇒*V* //*POS* 位置资源集合与 *CPOS* 信息资源集合中的物理位置转换成节点集合 *V*

ctrl: *V*→*K* //节点到控制的映射, *K* 可以为 CPS 环境中的所有实体

prnt: $p_i(p_j)\Rightarrow p_j\rightarrow p_i$; //物理位置域的包含关系转换成节点间的嵌套关系

$p_r(cp_k)\Rightarrow cp_k\rightarrow p_r$ //物理位置域与信息位置域的包含关系转换成节点间的嵌套关系

link: *channel*⇒*link* //*link* 是进程之间的通信通道的连接关系

E: *link* 连接的边集

m=*r* //实际 CPS 场景中的地点数为 *r*

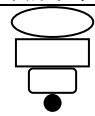
n=*k* //实际 CPS 场景中的区域数 *k*

X 为 CPS 实体端口所对应的内部名

Y 为 CPS 实体端口所对应的外部名

转换后的偶图则用符号表示为(见表 2).

表 2 物理位置的偶图符号表示

资源类型	节点特性	图形表示
主体资源	active	
位置资源	active	
信息资源	active	
端口	active	

3.3 ATL转换规则

完成模型转换, 首先需要创建相应的元模型, 然后用 ATL 虚拟机完成两模型实例模型的转换. 因此, 构建 DSR-TCSP 到 Bigraphs 的转换就是针对 DSR-TCSP 和 Bigraphs 的 KM3 元模型定义 ATL 规则, 该规则是在元模型层上定义上的.

图 8 中, DSR-TCSP model 是元模型, Bigraphs model 是目标模型. DSR-TCSP model 转换成其元模型

DSR-TCSP, Bigraphs model 转换成其元模型 Bigraphs. 都转换成其元元模型 KM3. Bridge DSR-TCSP to Bigraphs 是一个模型转换的实例.DSR-TCSP 和 Bigraphs 是元模型, DSR-TCSP 和 Bigraphs 通过 Ecore 创建, DSR-TCSP model 和 Bigraphs model 需分别转换为其元模型的实例. Bridge DSR-TCSP to Bigraphs 是由用户定义的模型转换模型, 描述了具体的从 DSR-TCSP 到 Bigraphs 转换的程序.

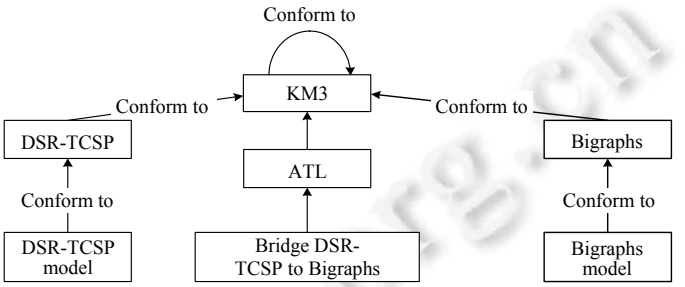


图 8 DSR-TCSP 到 Bigraphs 的 ATL 转换规则

因此, 一个完整的模型转换程序需要 4 部分: DSR-TCSP、Bigraphs、DSR-TCSP model 以及 Bridge DSR-TCSP to Bigraphs, Bigraphs model 是目标模型.

通过第 3.2 节的转换策略,定义从 DSR-TCSP 到 Bigraphs 的具体映射规则见表 3.

表 3 DSR-TCSP 到 Bigraphs 具体映射规则

Name	Description
Integer2IntegerType	Integer2IntegerType Match DSR-TCSP.IntegerType to Bigraphs.Integer
Real2RealType	Real2RealType Match DSR-TCSP.floatType to Bigraphs.Real
Boolean2BooleanType	Boolean2BooleanType Match DSR-TCSP.BooleanType to Bigraphs.Boolean
String2StringType	String2StringType Match DSR-TCSP.StringType to Bigraphs.String
space2node	space2node Match DSR-TCSP.space to Bigraphs.node
inclusion2nesting	inclusion2nesting Match DSR-TCSP.inclusion to Bigraphs.nesting
channel2link	channel2link Match DSR-TCSP.channel to Bigraphs.link
operation2reaction	operation2reaction Match DSR-TCSP.operation to Bigraphs.reaction
action2rule	action2rule Match DSR-TCSP.action to Bigraphs.rule

ATL 规则具体内容如下.

```
1.  -- @path DSRTCSPs=/DSR-TCSP2Bigraphs/DSR-TCSPs.ecore
2.  -- @path Bigraphs=/DSR-TCSP2Bigraphs/Bigraphs.ecore
3.  module DSRTCSP2Bigraphs; --Module Template
4.  create OUT: Bigraphs from IN: DSRTCSPs;
5.  rule pos2nodes {
6.      from s: DSRTCSPs!Pos(s.isPos(·))
7.      to t: Bigraphs!Nodes (
8.          n←s.pos
9.      )
10. }
11. rule cpos2nodes {
12.     from s: DSRTCSPs!Cpos(s.isCpos(·))
13.     to t: Bigraphs!Nodes (
14.         n←s.cpos
15.     )
16. }
```

```
17. helper context DSRTCSPs!Pos def: isPos(·): Boolean=
18.     if not self.transpos.oclIsUndefined(·) then
19.         true
20.     else
21.         false
22.     endif;
23. helper context DSRTCSPs!Cpos def: isCpos(·): Boolean=
24.     if not self.transcpos.oclIsUndefined(·) then
25.         true
26.     else
27.         false
28.     endif;
…
```

由于该转换是基于 AMMA 平台进行转换, 基于该平台的转换是相互的, 转换后的模型可以逆转换为原模型. 即: 相对于 DSR-TCSP 向偶图的转换, 逆转换的源模型为 Bigraphs model, 目标模型为 DSR-TCSP model. 通过该逆转换, 其语法一致性问题也可以得到保证.

4 实例分析

4.1 场景描述

驾驶场景与智能停车场都是典型的 CPS 系统, 图 9 以一个局部城市驾驶场景的物理部署图为例.

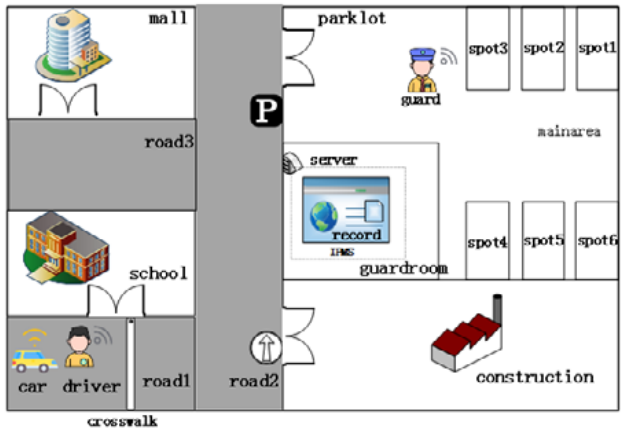


图 9 城市局部物理部署

图 9 为某城市的局部部署结构图. 灰色区域为道路, 共有 3 条道路 *road*1, *road*2 与 *road*3, 其中, *road*1 路段的学校门口有一条人行道(crosswalk). 空白区域为城市中的建筑, 为了方便描述, 本文列举了局部区域的 4 个区域资源: 学校(school)、商场(mall)、停车场(parklot)与施工地(construction). 道路上有两个路标, 分别为左转(leftsign)与停车(parksign).

本例中, 简化停车场中的停车位资源为(*spot*1,*spot*2,*spot*3,*spot*4,*spot*5,*spot*6), 设置了 6 个停车位, 停车场管理员房间(guardroom)中的服务器(server)上部署着智能停车场管理系统(intelligent parking management system, IPMS), 并且有来访记录(record)资源. 因此, 司机只有在管理员(guard)在场才能进入 guardroom, 不允许独自进入 guardroom, 以保证 record 资源的安全性. 停车场中的区域为主区域(mainarea), 停车场开放时间为

5 时至 20 时.

IPMS 的工作原理如图 10 所示. 主要分为 3 个模块: 数据采集模块(data collection model, DCM)、决策模块(design model, DM)和执行模块(enforcement model, EM). 数据采集模块为一些摄像头、雷达等传感器的数据采集与相关的数据预处理. 随后, 系统将处理的数据发送给相关的决策模块, 决策模块通过一系列的数据存储、数据计算并最终决策, 并将决策后的结果输入到执行模块进行相关动作的执行.

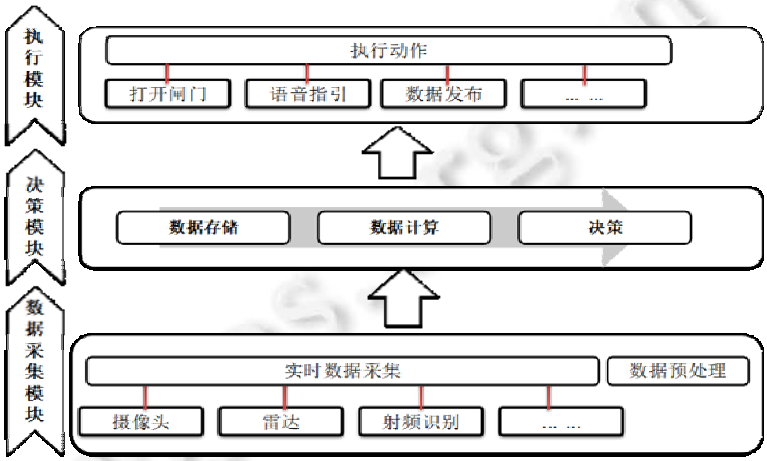


图 10 智能停车管理系统框架

到达停车场, 若停车场处于工作时间并且还有停车位, 则停车场打开闸门, 司机(driver)登录 IPMS 获取相关的语音指引等提示.

现有汽车 *car* 现行 *road*1, 经过学校再通过 *road*2, 最终到达停车场停车. 由司机目标选择而定. 当汽车执行这个目标时, 需要汽车的两个模块共同工作: 速度管理模块(speed management model, SMM)和方向管理模块(direction management model, DMM). SMM 与 4 个单元相连: 启动关闭单元(start stop unit, SSU)、速度通知单元(speed notification unit, SNU)、加速单元(acceleration unit, AU)和刹车单元(brake unit, BU). DMM 与两个单元相连: 方向盘(steering wheel, SW)与方向通知单元(direction notification unit, DNU). 文献[21]将人类信息处理总结为四阶段过程: 信息获取、信息分析、决策和行动选择、行动实施. 我们将该过程简化为 3 个模块: 信息获取(*GET*)、目标(*G*)、执行(*EXE*). 其中的目标 *G* 为通过人的处理得出最后目标的过程. 在当前场景中, 该司机 *driver* 要去 *parklot* 停放车辆并进入 *guardroom* 读取 *record*, 从当前位置出发, 选择:

$$road1 \rightarrow road2 \rightarrow \text{parklot} \rightarrow \text{guardroom}.$$

4.2 场景建模

(1) 我们对该场景进行建模为

$\boldsymbol{POS}:=\{p_{mall},p_{school},p_{crosswalk},p_{parklot},p_{construction},p_{road1},p_{road2},p_{road3},server,leftsign,parksign,car,driver,guard,mainarea,spot,guardroom\}$,

$\boldsymbol{CPOS}:=\{IPMS,DM,EM,DCM,record,GET,G,EXE,SMM,SSU,SNU,AU,BU,DMM,SW,DMU\}$.

- 该物理部署关系为

$PL(p_{mall},p_{school},p_{parklot}(mainarea(spot,guard,guardroom(server))),p_{construction},p_{road1}(p_{crosswalk},car,driver),p_{road2}(leftsign,parksign),p_{road3})$.

- 该信息空间部署关系为

$IPMS(DM,EM,DCM,record)$

$driver(GET,G,EXE)$

$car(SMM,SSU,SNU,AU,BU,DMM,SW,DMU)$

- 该场景的通信信道集合

channel:={*pk*,*lt*,*cw*,*bu*,*au*,*sn*,*ss*,*name*,*st*,*sn*,*ac*,*br*,*dm*,*log*,*read*,*cname*,*dm*,*sw*,*work*,*logg*,*gname*,*sp*1,*sp*2,*sp*3,*sp*4,*sp*5,*sp*6,*re*,*login*,*city*}

事件集合 ***A***:={*in*,*out*,*accelerate*,*brake*,*enter*,*exit*,*login*,*loginout*,*turn*,*read*}

DRIVER, *CAR* 和 *IPMS* 等定义如下.

$DRIVER_{initial}$ 为当前物理拓扑环境下的初始模型, 表示为多个进程的并发. 其中包含了 *DRIVER* 的 3 个模块 *GET*, *G* 与 *EXE* 的之间的交互. 例如: 在 *G* 模块产生相应的目标通过 *exe*1 通道发送给 *EXE*, 则相应的 *G* 进程中包含 *exe*1!→*STOP* 的并发, *EXE* 包含 *exe*1?→*STOP* 的并发, 以表示对目标 *goal* 的收发操作(如图 11 所示).

$$DRIVER_{initial}=GET||G||EXE|name?\rightarrow STOP||st?\rightarrow STOP||sn?\rightarrow STOP||ac?\rightarrow STOP||br?\rightarrow$$
$$STOP||dm?\rightarrow STOP||log?\rightarrow STOP||read?\rightarrow STOP||get1!\rightarrow STOP||get2!\rightarrow STOP||exe2?\rightarrow STOP$$
$$GET=get1?\rightarrow STOP||get2?\rightarrow STOP \quad G=get2!\rightarrow STOP||exe1!\rightarrow STOP \quad EXE=exe1?\rightarrow STOP||exe2!\rightarrow STOP$$

图 11 初始 *DRIVER* 的进程模型

CAR 进程包含了 *car* 的了两个模块、4 个单元的交互. 因此, *CAR* 进程是这几个单元的进程并发. 同时, 各个模块与单元中的交互是相应通道的收发操作组成的进程的并发(如图 12 所示).

$$CAR=SMM||SSU||SNU||AU||BU||DMM||DMU||SW$$
$$SMM=ss2?\rightarrow STOP||su2?\rightarrow STOP||au2?\rightarrow STOP||bu2?\rightarrow STOP \quad SSU=ss1?\rightarrow STOP||ss2!\rightarrow STOP$$
$$SNU=su1?\rightarrow STOP||su2!\rightarrow STOP \quad AU=au1?\rightarrow STOP||au2!\rightarrow STOP \quad BU=bu1?\rightarrow STOP||bu2!\rightarrow STOP$$
$$DMM=sw2?\rightarrow STOP||dmu2?\rightarrow STOP\ DMU=dmu1?\rightarrow STOP||dmu2!\rightarrow STOP \quad SW=sw1?\rightarrow STOP||sw2!\rightarrow STOP$$

图 12 *CAR* 的进程模型

IPMS 进程是 DM, EM, DCM 以及 *record* 发送动作的相关的进程并发. *DCM* 将采集的数据通过信道 *dm*1 发送, *DM* 通过信道 *dm*1 接收 *DCM* 发送的数据进行决策. 同样的, *DM* 通过信道 *dm*2 发送决策数据, *EM* 通过信道 *dm*2 接收决策并通过 *em* 信道发送执行命令(如图 13 所示).

$$IPMS=DM||EM||DCM||record!\rightarrow STOP$$
$$DCM=dm1!\rightarrow STOP \quad DM=dm1?\rightarrow STOP||dm2!\rightarrow STOP \quad EM=dm2?\rightarrow STOP||em!\rightarrow STOP$$

图 13 *IPMS* 的进程模型

该场景为 *driver* 先进入 *ROAD*2, 再进入 *parklot*, 进而进入 *guardroom* 读取 *record*. *DR_ENTER_ROAD*2 进程由一系列动作执行, *enter car*→*accelerate car*→*brake car*→*enter road*2. *DRIVER* 进程为 *DR_ENTER_ROAD*2 执行后, log*in IPMS*→*enter parklot*→*enter guardroom*→*read record*(如图 14 所示).

$$DR\_ENTER\_ROAD2 = \mu X \cdot (in \xrightarrow{(PTM,(0.3,0.2),car)} accelerate \xrightarrow{(PTM,(0.2,0.2),car)} brake \xrightarrow{(PTM,(0.1,0.1),car)} enter \xrightarrow{(PTM,(0.2,0.3),road2)} X)$$
$$DRIVER = \mu X \cdot (DR\_ENTER\_ROAD2;(F_{judge}(F_{ptp}(x,y,z),parklot) \wedge (5,20)) >> \log in \xrightarrow{(PTM,(0.2,0.2),IPMS)} (5,20) \wedge$$
$$(1 \leqslant spots \leqslant 6) >> enter \xrightarrow{(PTM,(0.2,0.3),parklot)} enter \xrightarrow{(PTM,(0.1,0.1),guardroom)} read \xrightarrow{(PTM,(0.1,0.1),record)} X)$$

图 14 *DRIVER* 的进程模型

GUARD 初始进程是通过信道 *gname* 输入 *guard* 姓名数据以及通过 log*g* 信道输出登录信息两个进程的并发. 在当前场景中, *guard enter guardroom*→log*in IPMS*→*exit guardroom*(如图 15 所示).

$$GUARD_{initial}=logg!\rightarrow STOP||gname?\rightarrow STOP$$
$$GUARD = \mu X \cdot (enter \xrightarrow{(PIM,(0.3,0.2),guardroom)} \log in \xrightarrow{(PIM,(0.2,0.2),IPMS)} exit \xrightarrow{(PIM,(0.1,0.1),guardroom)} X)$$

图 15 *GUARD* 的进程模型

MALL 与 *PARKLOT* 分别通过信道输出工作数据. *SPOT* 进程是 6 个停车位通过信道发送是否被使用数据的并发. 同理, *CROSSWALK*, *PARKSIGN*, *LEFTSIGN* 进程也分别由通道发送被使用数据(如图 16 所示).

$$MALL=work1!\rightarrow STOP \quad PARKLOT=work2!\rightarrow STOP$$
$$SPOT=sp1!\rightarrow STOP||sp2!\rightarrow STOP||sp3!\rightarrow STOP||sp4!\rightarrow STOP||sp5!\rightarrow STOP||sp6!\rightarrow STOP$$
$$CROSSWALK=cw!\rightarrow STOP \quad PARKSIGN=pk!\rightarrow STOP \quad LEFTSIGN=lt!\rightarrow STOP$$

图 16 其他进程模型

整个驾驶场景 *ADS* 是以上的所有进程在 ***A*** 事件集上的并发(如图 17 所示).

$$D1=DRIVER_{initial}||DRIVER$$
$$G1=GUARD_{initial}||GUARD$$
$$ADS = D1||_A G1||_A CAR||_A IPMS||_A GUARD||_A MALL||_A PARKLOT||_A SPOT||_A CROSSWALK||_A PARKSIGN||_A LEFTSIGN$$

图 17 *ADS* 进程模型

(2) 接下来, 按照第 2.3 节转换规则 1 对该模型进行转换, 则转换结果如下.

该具体过程中节点集合:

$\boldsymbol{V}$:={*mall*,*school*,*crosswalk*,*construction*,*parklot*,*road*1,*road*2,*road*3,*server*,*leftsign*,*parksign*,*car*,*driver*,*guard*, *mainarea*,*guardroom*,*IPMS*,*DM*,*EM*,*DCM*,*record*,*GET*,*G*,*EXE*,*SMM*,*SSU*,*SNU*,*AU*,*BU*,*DMM*,*SW*,*DMU*}

物理位置域的包含关系与物理位置域和信息位置域的包含关系转换成相应的嵌套关系, 对于上述模型中的信道集合 ***channel***, 则转换成偶图中的端口. 对于相同信道的收发过程, 映射为拓扑空间中的连接, 如 *DCM* 进程对应的 $dm1!\rightarrow STOP$ 与 *DM* 进程对应的 $dm1?\rightarrow STOP$ 是同一信道 *dm* 的收发过程, 则对应的信息资源节点 *DCM* 与 *DM* 会有相应的连接 *link*.

通过应用 DSR-TCSP 到 Bigraphs 模型转换方法, 将该场景的 DSR-TCSP 模型转换为 Bigraphs 模型, 这个模型转换其实是实例化 DSR-TCSP 到 Bigraphs 的 ATL 规则.该过程可通过 AMMA 平台实现. 图 18 给出了相应配置信息定义界面.

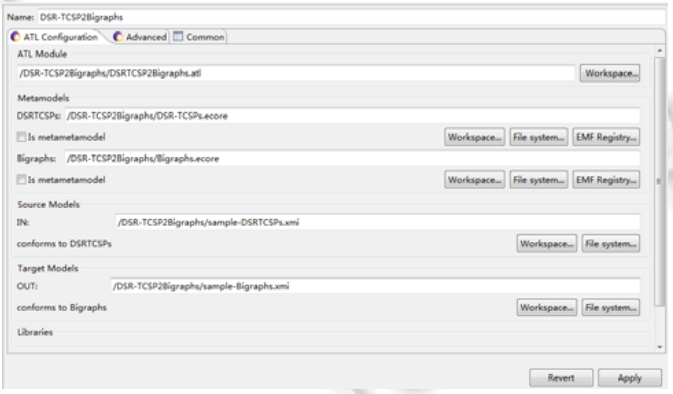


图 18 ATL 配置视图

工具转换后得到模型的项语言表达:

mall[*work*]|*shcool*|*construction*|*road*2.(*parksign*[*pk*]|*leftsign*[*lf*])|*road*3|*parklot*[*work*].*mainarea*.(*spot*[-,-,-,-,-,-,-]|*guard*[*jack*,-,-]|*guardroom*.(*server*.*IPMS*.(*DCM*[*e*2]|*DM*[*e*1,*e*2]|*EM*[*e*1,login]|*record*[*re*]))|*road*1.(*driver*[*e*5,*e*6,*tom*,-,-,-,-,-,-,-,-].(*GET*[*e*3,*e*5]|*G*[*e*3,*e*4]|*EXE*[*e*4,*e*6])|*car*[*c*1].(*SMM*[*e*7,*e*8,*e*9,*e*10]|*SSU*[*e*10,*ss*]|*SNU*[*e*9,*sn*]|*AU*[*e*8,*au*]|*BU*[*e*7,*bu*]|*DMM*[*e*11,*e*12]|*SW*[*e*11,*sw*]|*DMU*[*e*12,*dm*])|*crosswalk*[-]);

为了更清晰直观地表达物理拓扑模型, 我们将以上项语言输入工具 BigMC-GUI[22], 得到该场景的偶图如图 19 所示.

该过程的操作事件集合 ***event***={*in*,*out*,*accelerate*,*brake*,*enter*,*exit*,login,log*inout*,*turn*,*read*}.

下面通过转换规则, 将场景的事件集 ***event*** 的物理拓扑资源变化转换为相应的偶图反应规则. 对于不同的执行进程主体执行相同的事件对应不同的资源向量 ***r*** 的变化.

由于篇幅的原因, 以下转换规则只介绍了关于进入 *car* 事件 $in\xrightarrow{(r,car)}STOP$ 与停车位资源使用事件

$enter \xrightarrow{(r,parklot)} STOP$ 进程的映射规则, 具体的映射规则见附录部分.

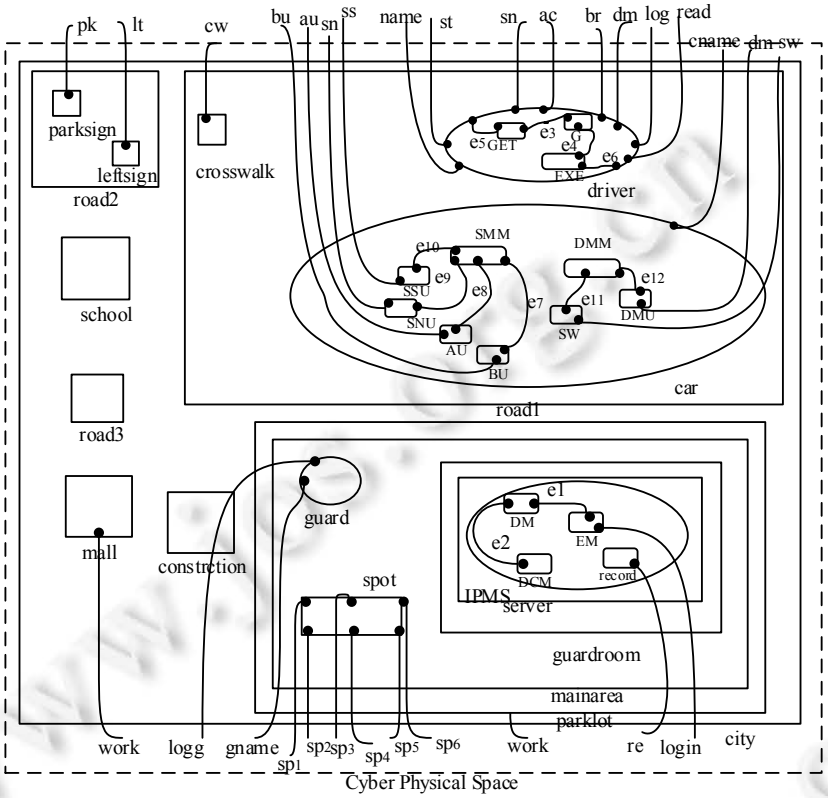


图 19 城市局部物理部署偶图

转换规则 2. 事件映射为相应的反应规则(见表 4).

表 4 事件映射的反应规则

进程事件执行	资源变化
$RULES1 := in \xrightarrow{(r,car)} STOP$	*driver*[*e5*,*e6*,*tom*,-,-,-,-,-,-,-].\$0|*car*[*c1*].\$1|\$2→ *car*[*c1*].(*driver*[*e5*,*e6*,*tom*,-,-,-,-,-,-,-].\$0|\$1)|\$2;
$RULES8 := F_{judge}(F_{ptp}(x,y,z),parklot) \wedge (5,20) \wedge (1 \leqslant spots \leqslant 6)$ >>($enter \xrightarrow{(r,parklot)} STOP$)	(1) **0 *spot* used(6 *spots left*)** *parklot*[*work*].*mainarea*.(\$0|*spot*[-,-,-,-,-,-])|\$1→ *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,-,-,-,-,-]|\$1; (2) ***1spot used*(5 *spots left*)** *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,-,-,-,-,-])|\$1→ *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,*used*,-,-,-,-]|\$1; (3) **2 *spots used*(4 *spots left*)** *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,*used*,-,-,-,-])|\$1→ *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,*used*,*used*,-,-,-]|\$1; (4) **3 *spots used*(3 *spots left*)** *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,*used*,*used*,-,-,-])|\$1→ *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,*used*,*used*,*used*,-,-]|\$1; (5) **4 *spots used*(2 *spots left*)** *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,*used*,*used*,*used*,-,-])|\$1→ *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,*used*,*used*,*used*,*used*,-]|\$1; (6) **5 *spots used*(1 *spot left*)** *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,*used*,*used*,*used*,*used*,-])|\$1→ *parklot*[*work*].*mainarea*.(\$0|*spot*[*used*,*used*,*used*,*used*,*used*,*used*]|\$1;

可以看到: 执行 $enter \xrightarrow{(r,parklot)} STOP$ 进程时, 由于 *parklot* 的 *spot* 资源条件为 $1 \leqslant spots \leqslant 6$, 因此只有当前剩余 *spot* 在 1 到 6 之间时才能执行进入 *parklot*. 当进入 *parklot* 停车后, 一个 *spot* 进入使用状态 *used*, 进而拓扑空间描述发生变化.

执行反应规则后, 物理部署空间与资源将发生变化, 如图 20 所示为执行 *RULES*1:*driver* 进入 *car* 后的物理部署图.

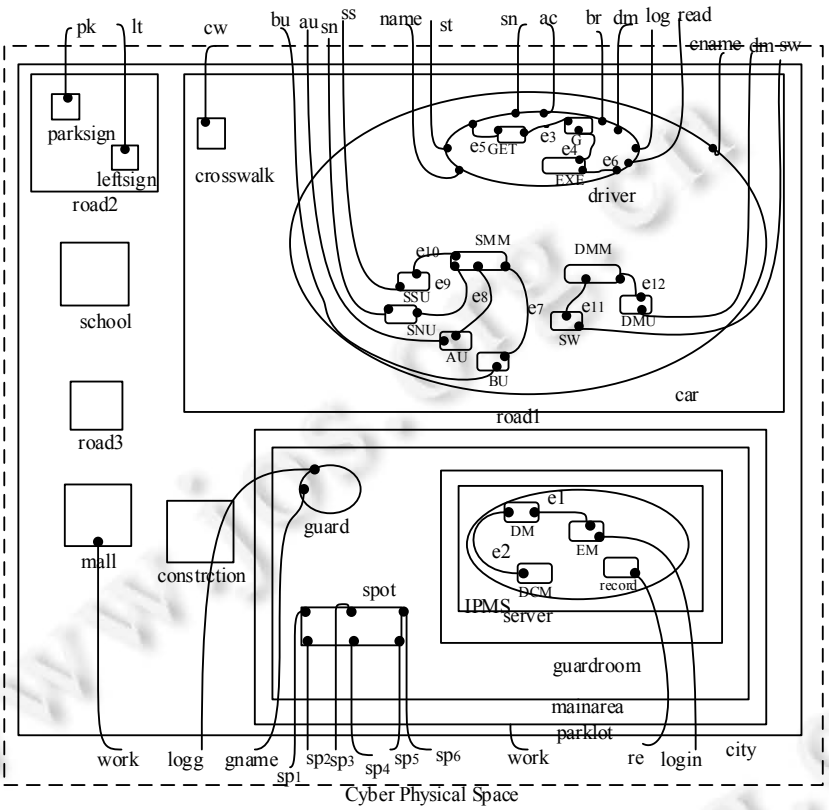


图 20 *driver* 进入 *car* 后的生成物模型

4.3 模型检验——性质验证与修改

(1) 时间属性的检验

针对第 4.2 节建模的 *ADS* 进程的符号迁移系统如图 21 所示, 通过第 2.5 节时间属性的验证算法对该过程的时间属性进行验证. 为了简化状态空间大小, 仅画出了 *DRIVER* 时空约束的资源状态迁移图.

对该过程执行第 2.5 节时间属性的验证算法, 当达到节点 N_4 时, 这个节点有时间约束, 为 *parklot* 的开放时间为(5,20). 当 *driver* 进入车辆 *car*(N_0)时刻为 18, 进行深度优先算法执行, 目前 *totalt* 时间为 2, 按照此执行时间可以刚好到达停车场, 即刚好满足 N_4 点的时间约束, 顺利登录 *IPMS* 进入 *parklot*; 否则不满足 N_4 的时间约束.

(2) 死锁状态定位与修改

对于以上的时空约束的资源状态迁移图中通过死锁定位算法进行死锁节点的定位, 定位该过程的死锁节点集合 *Deadlock*:={N_5}, 发现 N_5 节点在 *parklot* 地点停车而不进行其他动作执行. 由于 *parklot* 开放时间在(5,20), 因此在其他时间并不开放, 需要对死锁状态进行修改: 在该节点上添加相关边或删除该节点. 因此在 N_5 处, 可以删除 N_5 节点, 即寻找全天开放的停车场; 或者进行容错处理, 添加节点 N_{10} 的容错处理措施(例如路边停车位), 隔天再进停车场. 以下对于死锁节点修改选择寻找临时停车位, 修改后的时空资源约束状态迁移图如图 22 所示.

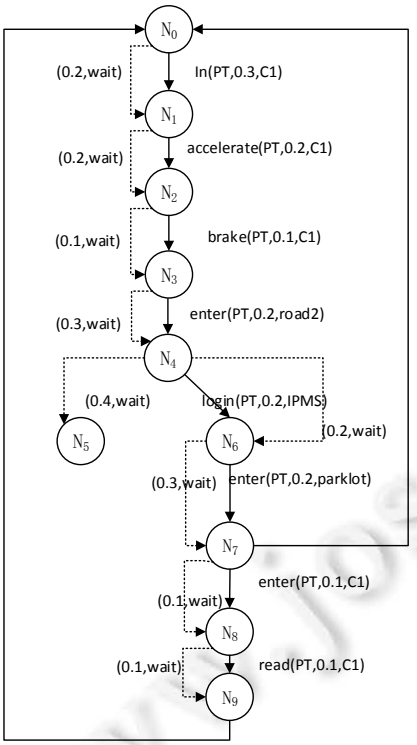


图 21 时空约束的资源状态迁移图

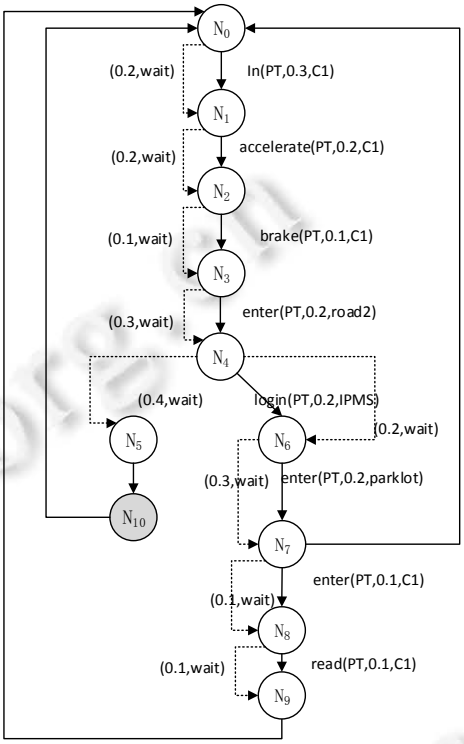


图 22 修改后时空约束的资源状态迁移图

那么该过程的 DSR-TCSP 模型需作相应修改:

DRIVER 进程中应该添加 $brake \blacktriangleright F_{judge}(F_{ptp}(x,y,z),parklot) \wedge (5,20) \vdash error$ 进行容错处理.

我们将修改后模型及规则以及以下 *property* 输入到 BigMC 工具后, 进行模型检测:

%*property CppResourceSecue* !*matches*

(*parklot*[*work*].*mainarea*.(*guard*[*jack*,-,-]|*guardroom*.(*driver*[*e*5,*e*6,*tom*,-,-,-,-,-,-,-,-,-].$1|$2)|$3)|$4)

%*property CppSpotsSecue* !*matches*

(*parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,*used*,*used*,*used*,*used*]|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-,-].$1)|$2)|$3).

第 1 条属性 *CppResourceSecue* 是 *driver* 不允许独自在 *guardroom* 中, 以保护 *guardroom* 中的资源安全.

第 2 条属性 *CppSpotsSecue* 是不允许在 *spot* 资源都占用的情况下, 允许 *car* 进入到停车场的主区域.

将初始状态与反应规则 *RULES* 输入 BigMC 中, 产生反例路径如图 23 所示.

图 23 反例路径图

该反例路径违反 CppResourceSecue 属性, 在找到反例 counterexample 后, 工具 BigMC 则找出从初始节点到该节点的所有节点, 对于该反例为路径为 5(root)→4(RULES5:driver enters road2)→3(RULES14:driver enters mainarea)→2(RULES12:guard enters guardroom)→1(RULES12:driver enters guardroom)→0(RULES13:guard exits guardroom). 因此, 我们应该规定, 当 guard 离开 guardroom 时, guardroom 中把不允许再有司机在房间. 在 DSR-TCSP 中将 GUARD 进行修改为

$$\text{GUARD}:=\mu X\bullet(enter\xrightarrow{(PIM,(0.3,0.2),guardroom)}\log in\xrightarrow{(PIM,(0.2,0.2),IPMS)}(driver<1)>>exit\xrightarrow{(PIM,(0.1,0.1),guardroom)}X).$$

同时对 RULES13 做相应的修改为:

$$\text{RULES13}:=((driver<1)>>exit\xrightarrow{(r,guardroom)}STOP):$$

parklot[*work*].*mainarea*.(*driver*[*e5*,*e6*,*tom*,-,-,-,-,-,-,-,-].$0|(*guardroom*.($1|*guard*[*jack*,*login*,-]))|$2)|$3→

parklot[*work*].*mainarea*.(*driver*[*e5*,*e6*,*tom*,-,-,-,-,-,-,-,-].$0|(*guard*[*jack*,-,-]|*guardroom*.$1|$2))|$3;

随着 *driver* 与 *guard* 的物理拓扑空间变化, 物理空间不满足 *guardroom* 中的资源安全性需求, 修改后的进程代数模型保证了 *guardroom* 中的资源安全性, 从而保证了 CPS 资源安全性.

5 相关工作比较

近年来对 CPS 安全性的建模与验证取得较大的进展. 表 5 中列出本文与关键的相关工作的比较.

表 5 DSR-TCSP 方法和其他方法的比较

FM	*T*	*P*	CS	PS	EM	RD	VE	RSV
Process Calculus[28]	√	×	√	√	√	√	√	×
Time Automaton[29]	√	×	√	×	√	×	√	×
ST-LUSTRE[33]	√	√	√	√	√	√	×	×
Biographical Reactive Systems[35]	√	√	×	√	√	×	√	×
Temporal Topograph[36]	√	√	√	√	√	×	√	×
TA-CPAC[37]	×	√	√	√	√	×	√	×
DSR-TCSP (our work)	√	√	√	√	√	√	√	√

Note: √: support; ×: no support; FM: formal tools; *T*: time; *P*: position; CS: cyber space; PS: physical space; EM: entities mobility; RD: resource description; VE: verification; RSV: resource safety verification

文献[23]提出了一种新的 CPS 设计范式, 通过具有不确定性分布的不确定性变量, 来建模要实施组件的成本和延迟时间的不确定性影响, 然后递归得出可靠性特征. 文献[24]提出了一种用于 CPS 系统在线验证的增量验证技术, 给出了一种区分待验证问题与已验证问题的方法. 文献[25]提出了一种基于领域模型语言工程的随机混合建模语言(SHML)支持随机建模 CPS 中的混成行为. 该研究更注重 CPS 的随机性以及系统各部分之间的交互. 而对 CPS 的非功能属性时间与空间因素的研究较少. 关于 CPS 的时间或空间性质建模与验证问题, 国内外已有相关研究. 文献[26]扩展 BPMN 以支持各种 CPS 概念和属性, 进而提出了一种一致性验证方法, 旨在验证与每个过程与时间相关的物理特性不会引起冲突. 文献[27–31]研究在时间变化下, CPS 各部分的安全交互. 这些传统的 CPS 建模与验证方法大多局限于时间域内的分析, 对于空间拓扑变化考虑较少, 因此可能会存在一些 CPS 时空下的资源安全性问题. 文献[32]提出一种面向安全关键型信息物理系统的新型时空语言 ST-LUSTRE, 以支持 CPS 的时空属性的统一建模, 并给出 ST-LUSTRE 在基于时间集的拓扑空间和自然数上的解释. 这部分工作同时考虑时间与空间, 将时间与空间进行统一研究. 文献[33]提出了一种方法论和技术框架, 该方法论和技术框架支持不断发展的网络物理空间的建模以及它们的时空特性的推理. CPS 的空间安全性不仅体现在信息空间的安全, 物理空间也是影响 CPS 安全性一个重要因素.

针对于物理空间的研究, 近些年来研究成果丰富. 文献[34]使用偶图反应系统来建模网络空间和物理空间及其动力学的拓扑结构的方法, 并通过模型检查执行推测性威胁分析, 以推断拓扑部署演变对安全性要求的满足情况. 文献[35]提出一种对空间场景快照的演化进行建模并验证时空模型的方法, 首先在拓扑空间中引入偶图来定义一种新颖的地形图, 从而研究对时空行为建模与分析的可表达性和可验证性. 文献[36]首先提出拓扑部署模型, 以捕获网络和物理世界的拓扑特征; 然后, 基于此模型提出了一种拓扑感知的网络物理

访问控制模型(TA-CPAC), 通过动态调整权限分配, 保证网络和物理世界的安全. 但是该研究重点在于访问控制策略的制定, 对于 CPS 中的资源安全性考虑的较少, 且没有对时间因素作重点研究. 文献[37]在基于角色的访问控制上扩展了时间属性, 研究时间影响下的访问控制模型. 文献[38]研究了在时空约束下的角色的访问控制. 这些访问控制仅研究了时间与空间因素, 但未针对于信息物理空间中时空所对应的资源进行重点研究, 不能保证信息物理空间下 CPS 时空约束下资源的安全性. 作为已有工作的延续工作, 本文中加入了物理空间拓扑属性, 对 CPS 中资源随物理拓扑空间与时间的变化进行建模、验证与修改, 从而能够保证 CPS 的时空资源的安全性, 实现可信 CPS.

与以上研究工作相比, 本文重点研究时空约束下的资源这类特殊资源的安全性, 如停车场中的停车位、线路中的一段铁轨、系统中的一条数据、移动设备的一条消息等的安全性. 这一类特殊资源受到时间与物理拓扑空间的影响, 其安全性问题影响着 CPS 的安全性. 本文首先在 TCSP 中引入物理拓扑空间与资源, 提出了 DSR-TCSP; 其次, 对 DSR-TCSP 的时间安全需求进行验证; 再次, 通过模型转换方法将 DSR-TCSP 转换为偶图与偶图反应, 并进行物理拓扑安全需求的模型检验, 进而对 DSR-TCSP 模型进行修改, 保证 CPS 时空约束下资源的安全性.

6 结束语

针对 CPS 时空约束下的资源安全性问题, 本文提出了面向 CPS 时空约束的资源建模及其安全性验证方法. 首先, 在 TCSP 中引入物理拓扑空间与资源, 提出了 DSR-TCSP; 其次, 通过 DSR-TCSP 的时间属性验证算法对时间安全需求进行验证; 再次, 将满足时间安全需求的模型转换为偶图与偶图反应, 并输入偶图检验工具 BigMC 中, 验证其物理拓扑安全需求, 对没有通过验证的反例, 修改 DSR-TCSP 模型, 使其满足所提出的安全需求; 最后, 得到满足时空约束下资源安全性需求的 DSR-TCSP 模型.

本文中考虑到时间因素的影响, 但对于时间 TIME 模型仅做简要时间段考虑, 在后续工作中会对 TIME 模型进一步研究. CPS 与环境交互, 较为复杂, 对于模型检测中出现的死锁等问题, 我们仅做了简要修改, 如何根据 CPS 的场景进行适应性的模型修改是值得研究的. 从 DSR-TCSP 到偶图的转换的正确性保证,本文做了对应元素转换的说明, 关于转换的语义一致性保证的形式化证明, 将在未来工作中展开研究. 此外,在 CPS 中, 服务是一个重要概念, 针对于 CPS 的系统建模过程, 如何将服务概念引入并保证在拓扑空间变化下的服务安全性, 也将会是下一步的研究重点.

References:

[1] He JF. Cyber-physical systems. Computer Society Newsletter, 2010, 6(1): 25−29 (in Chinese).

[2] Song ZH, Zhang GQ. Modeling of CPS based on aspect-oriented spatial-temporal Petri net. Computer Science, 2017, 44(7): 38−41, 73 (in Chinese with English abstract).

[3] Luo CX, Wang R, Guan Y, *et al*. CPS integrated modeling method for real-time data. Ruan Jian Xue Bao/Journal of Software, 2019, 30(7): 1966−1979 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5753.htm [doi: 10.13328/j.cnki.jos.005753]

[4] Chen XY, Zhu Y, Zhao Y, *et al*. Hybrid AADL modeling and model transformation for CPS time and space properties verification. Ruan Jian Xue Bao/Journal of Software, 2021, 32(6): 1779−1798 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6249.htm [doi: 10.13328/j.cnki.jos.006249]

[5] Xu JQ, Guo XJ, Wang JF, *et al*. Research on CPS resource service model and resource scheduling. Journal of Computer Science, 2018, 41(10): 2330−2343 (in Chinese with English abstract).

[6] Orumwense EF, AboAlEz KM. Energy management in a cloud-based cyber-physical system. IET Cyber-physical Systems: Theory and Applications, 2021, 6(2): 93−103.

[7] Apat HK, Bhaisare K, Sahoo B, *et al*. Energy efficient resource management in fog computing supported medical cyber-physical system. In: Proc. of the 2020 Int'l Conf. on Computer Science, Engineering and Applications. Piscataway: IEEE, 2020. 1−6.

[8] Zhu Y, Xiao FX, Zhou H, *et al*. A method of energy consumption modeling and analysis for embedded real-time system software is presented. Computer Research and Development, 2014, 51(4): 848−855 (in Chinese with English abstract).

[9] Cao Y, Huang ZQ, Kan SL, *et al*. Location-constrained access control model and verification methods. Computer Research and Development, 2018, 55(8): 1809−1825 (in Chinese with English abstract).

[10] Zhang L, Zhang Z, Zhao T. A novel spatio-temporal access control model for online social networks and visual verification. Int'l Journal of Cloud Applications and Computing, 2021, 11(2): 17−31.

[11] Hoare CAR. Communicating sequential processes. Communications of the ACM, 1978, 21(8): 666−677.

[12] Reed GM, Roscoe AW. A timed model for communicating sequential processes. In: Proc. of the Int'l Colloquium on Automata, Languages, and Programming. Berlin: Springer-Verlag, 1986. 314−323.

[13] Milner R. The Space and Motion of Communicating Agents. Cambridge: Cambridge University Press, 2009. 1−191.

[14] Faithfull AJ, Perrone G, Hildebrandt TT. Big red: A development environment for bigraphs. Electronic Communications of the Easst, 2013, 61: 1−10.

[15] Perrone G, Debois S, Hildebrandt TT. A verification environment for bigraphs. Innovations in Systems and Software Engineering, 2013, 9(2): 95−104.

[16] Schneider S. An operational semantics for timed CPS. Information and Computation, 1995, 116(2): 193−213.

[17] Fei Y, Zhong L, Jha NK. An energy-aware framework for dynamic software management in mobile computing systems. ACM Trans. on Embedded Computing Systems, 2008, 7(3): 1−31.

[18] Bézivin J, Jouault F, Rosenthal P, *et al*. Modeling in the large and modeling in the small. In: Proc. of the Model Driven Architecture. Berlin: Springer Verlag, 2004. 33−46.

[19] Zhu M, Li BX, Chen QQ, *et al*. CPS modeling and property verification based on differential dynamic logic. Acta Electronica Sinica, 2012, 40(6): 1126−1132 (in Chinese with English abstract).

[20] Caplat G, Sourrouille JL. Model mapping using formalism extensions. IEEE Software, 2005, 22(2): 44−51.

[21] Parasuraman R, Sheridan TB, Wickens CD. A model for types and levels of human interaction with automation. IEEE Trans. on Systems, Man, and Cybernetics—Part A: Systems and Humans, 2000, 30(3): 286−297.

[22] https://github.com/bigmc

[23] Jiang Y, Wang M, Jiao X, *et al*. Uncertainty theory based reliability-centric cyber-physical system design. In: Proc. of the 2019 Int'l Conf. on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data. Piscataway: IEEE, 2019. 208−215.

[24] Bu L, Xing SP, Ren XY, *et al*. Incremental online verification of dynamic cyber-physical system. In: Proc. of the 2019 Design, Automation & Test in Europe Conf. & Exhibition. Piscataway: IEEE, 2019. 782−787.

[25] Du D, Guo T, Wang Y. SHML: Stochastic hybrid modeling language for CPS behavior. In: Proc. of the 26th Asia-Pacific Software Engineering Conf. Piscataway: IEEE, 2019. 220−227.

[26] Graja I, Kallel S, Guermouche N, *et al*. Modelling and verifying time-aware processes for cyber-physical environments. IET Software, 2019, 13(1): 36−48.

[27] Zhang J, Zhu Y, Xiao F. Modelling and analysis of real-time and reliability for WSN-based CPS. Int'l Journal of Internet Protocol Technology, 2019, 12(2): 76−84.

[28] Su Q, Wang T, Chen TM, *et al*. CPS security modeling and validation based on time automaton. Information Security Research, 2017, 3(7): 601−609 (in Chinese with English abstract).

[29] Tariq MU, Florence J, Wolf M. Improving the safety and security of wide-area cyber–physical systems through a resource-aware, service-oriented development methodology. Proc. of the IEEE, 2017, 106(1): 144−159.

[30] Tran HD, Nguyen LV, Musau P, *et al*. Decentralized real-time safety verification for distributed cyber-physical systems. In: Proc. of the Int'l Conf. on Formal Techniques for Distributed Objects, Components, and Systems. Berlin: Springer- Verlag, 2019. 261−277.

[31] Sun CB, Cheng S, Yuan K, *et al*. Real time simulation platform of power cyber-physical system based on node mapping model. Power System Technology, 2019, 43(7): 2368−2375 (in Chinese with English abstract).

[32] Liu J, Wang JY, Li ZW, *et al*. ST-LUSTRE: A new space-time language for security- critical networked physical systems. Int'l Journal of Engineering, 2017, 13(8): 1219−1232.

[33] Tsigkanos C, Kehrer T, Ghezzi C. Modeling and verification of evolving cyber-physical spaces. In: Proc. of the 11th Joint Meeting on Foundations of Software Engineering. New York: ACM, 2017. 38−48.

[34] Tsigkanos C, Pasquale L, Ghezzi C, *et al*. On the interplay between cyber and physical spaces for adaptive security. IEEE Trans. on Dependable and Secure Computing, 2016, 15(3): 466−480.

[35] Li T, Chen X, Sun H, *et al*. Modeling and verification of spatio-temporal intelligent transportation systems. In: Proc. of the 2020 IEEE 19th Int'l Conf. on Trust, Security and Privacy in Computing and Communications. Piscataway: IEEE, 2020. 568−575.

[36] Cao Y, Huang Z, Kan S, *et al*. Specification and verification of a topology-aware access control model for cyber-physical space. Tsinghua Science and Technology, 2019, 24(5): 497–519. [doi: https://doi.org/10.26599/TST.2018.9010116]

[37] Mondal S, Sural S, Atluri V. Security analysis of GTRBAC and its variants using model checking. Computers & Security, 2011, 30(2–3): 128–147. [doi: https://doi.org/10.1016/j.cose.2010.09.002]

[38] Toahchoodee M, Ray I. On the formalization and analysis of a spatio-temporal role-based access control model. Journal of Computer Security, 2011, 19(3): 399–452. [doi: 10.3233/JCS-2010-0418]

附中文参考文献:

[1] 何积丰. 信息-物理融合系统. 中国计算机学会通讯, 2010, 6(1): 25–29.

[2] 宋振华, 张广泉. 基于 AOP 的时空 Petri 网的 CPS 建模. 计算机科学, 2017, 44(7): 38–41, 73.

[3] 罗晨霞, 王瑞, 关永, 等. 面向实时数据的 CPS 一体化建模方法. 软件学报, 2019, 30(7): 1966–1979. http://www.jos.org.cn/1000-9825/5753.htm [doi: 10.13328/j.cnki.jos.005753]

[4] 陈小颖, 祝义, 赵宇, 等. 面向 CPS 时空性质验证的混成 AADL 建模与模型转换方法. 软件学报, 2021, 32(6): 1779–1798. http://www.jos.org.cn/1000-9825/6249.htm [doi: 10.13328/j.cnki.jos.006249]

[5] 徐久强, 郭雪静, 王进法, 等. CPS 资源服务模型和资源调度研究. 计算机学报, 2018, 41(10): 2330–2343.

[8] 祝义, 肖芳雄, 周航, 等. 一种嵌入式实时系统软件能耗建模与分析的方法. 计算机研究与发展, 2014, 51(4): 848–855.

[9] 曹彦, 黄志球, 阚双龙, 等. 位置约束的访问控制模型及验证方法. 计算机研究与发展, 2018, 55(8): 1809–1825.

[19] 朱敏, 李必信, 陈乔乔, 等. 基于微分动态逻辑的 CPS 建模与属性验证. 电子学报, 2012, 40(6): 1126–1132.

[28] 苏琪, 王婷, 陈铁明, 等. 基于时间自动机的 CPS 安全建模和验证. 信息安全研究, 2017, 3(7): 601–609.

[31] 孙充勃, 成晟, 原凯, 等. 基于节点映射模型的电力信息物理系统实时仿真平台. 电网技术, 2019, 43(7): 2368–2377.

附录

进程事件执行	资源变化
$RULES1:=in \xrightarrow{(r,car)} STOP$	*driver*[*e5*,*e6*,*tom*,-,-,-,-,-,-,-].$0|*car*[*c1*].$1|$2→ *car*[*c1*].(*driver*[*e5*,*e6*,*tom*,-,-,-,-,-,-,-].$0|$1)|$2
$RULES2:=out \xrightarrow{(r,car)} STOP$	*C1*.(*driver*[*d*,*d*,*d*].$0|$1)|$2→*driver*[*x*,*y*,*z*].$0|*C1*.$1|$2
$RULES3:=accelerate \xrightarrow{(r,car)} STOP$	*car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-].$0|$1)|$2→ *car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,*a*,-,-,-,-,-].$0|$1)|$2
$RULES4:=\text{true}>>(brake \xrightarrow{(r,C1)} STOP)$	*car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-].$0|$1)|$2→ *car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,*b*,-,-,-,-].$0|$1)|$2
$RULES5:=(enter \xrightarrow{(r,road2)} STOP)$	(1) ***car enter road2*** *road2*.$0|*car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-,-].$1|$2)|$3→ *road2*.($0|*car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-,-].$1|$2))|$3; (2) ***driver enter road2*** *road2*.$0|*road1*.(*driver*[*e5*,*e6*,*tom*,-,-,-,-,-,-,-].$1|$2)|$3→ *road2*.($0|(*driver*[*e5*,*e6*,*tom*,-,-,-,-,-,-,-].$1))|*road1*.($2)|$3
$RULES6:=F_{judge}(F_{ptp}(x,y,z),leftsign)>>$ $(turn \xrightarrow{(r,C1)} STOP)$	*road2*.(*leftsign*[*lt*].(*car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-,-].$0)| *SMM*[*e7*,*e8*,*e9*,*e10*]|*SSU*[*e10*,*ss*]|*SNU*[*e9*,*sn*]|*AU*[*e8*,*au*]| *BU*[*e7*,*bu*]|*DMM*[*e11*,*e12*]|*SW*[*e11*,*sw*]|*DMU*[*e12*,*dm*])) → *road2*.(*leftsign*[*lt*].(*car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,*lt*,*lt*,-,-].$0)| *SMM*[*e7*,*e8*,*e9*,*e10*]|*SSU*[*e10*,*ss*]|*SNU*[*e9*,*sn*]|*AU*[*e8*,*au*]| *BU*[*e7*,*bu*]|*DMM*[*e11*,*e12*]|*SW*[*e11*,*lt*]|*DMU*[*e12*,*lt*]))
$RULES7:=F_{judge}(F_{ptp}(x,y,z),parklot)\wedge$ $(5,20)>>(login \xrightarrow{(r,IPMS)} STOP)$	(1) ***driver* login** *parklot*[*work*].*mainarea*.($0|*guardroom*.(*server*.*IPMS*.$1|$2))| *driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-,-].$3→ *parklot*[*work*].*mainarea*.($0|*guardroom*.(*server*.*IPMS*.$1|$2))| *driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,l,-].$3; (2) ***guard* login** *parklot*[*work*].*mainarea*.(*guardroom*.($0|*guard*[*jack*,-,-])|$1)|$2→ *parklot*[*work*].*mainarea*.(*guardroom*.($0|*guard*[*jack*,login,-])|$1)|$2
$RULES8:=F_{judge}(F_{ptp}(x,y,z),parklot)\wedge$ $(5,20)\wedge(1\leqslant spots\leqslant 6)>>$ $(enter \xrightarrow{(r,parklot)} STOP)$	(1) **0 *spot used*** *parklot*[*work*].*mainarea*.($0|*spot*[-,-,-,-,-,-])| *road2*.($1|*parksign*[*pk*].($2|*car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-,-].$3)))| $4→*parklot*[*work*].*mainarea*.($0|*spot*[*used*,-,-,-,-,-]| *car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-,-].$3))|*road2*.($1|*parksign*[*pk*].($2))|$4; (2) **1 *spot used*** *parklot*[*work*].*mainarea*.($0|*spot*[*used*,-,-,-,-,-])|*road2*.($1|*parksign*[*pk*].($2| *car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-,-].$3)))|$4→ *parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,-,-,-,-]| *car*[*c1*].(*driver*[*e5*,*e6*,*tom*,*s*,-,-,-,-,-,-,-].$3))|*road2*.($1|*parksign*[*pk*].($2))|$4;

附录(续)

进程事件执行	资源变化
$RULES8:=F_{judge}(F_{ptp}(x,y,z),parklot)\wedge (5,20)\wedge(1\leqslant spots\leqslant 6)>>(enter \xrightarrow{(r,parklot)} STOP)$	(3) **2 *spots used*** *parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,-,-,-,-])|*road*2.($1|*parksign*[*pk*].($2|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$3)))|$4→*parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,*used*,-,-,-]|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$3))|*road*2.($1|*parksign*[*pk*].($2))|$4; (4) **3 *spots used*** *parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,*used*,-,-,-])|*road*2.($1|*parksign*[*pk*].($2|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$3)))|$4→*parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,*used*,*used*,-,-]|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$3))|*road*2.($1|*parksign*[*pk*].($2))|$4; (5) **4 *spots used*** *parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,*used*,*used*,-,-])|*road*2.($1|*parksign*[*pk*].($2|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$3)))|$4→*parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,*used*,*used*,*used*,-]|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$3))|*road*2.($1|*parksign*[*pk*].($2))|$4; (6) **5 *spots used*** *parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,*used*,*used*,*used*,-])|*road*2.($1|*parksign*[*pk*].($2|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$3)))|$4→*parklot*[*work*].*mainarea*.($0|*spot*[*used*,*used*,*used*,*used*,*used*,*used*]|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$3))|*road*2.($1|*parksign*[*pk*].($2))|$4
$RULES9:=(enter \xrightarrow{(r,road3)} STOP)$	*road*3.$0|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$1|$2)|$3→ *road*3.($0|*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$1|$2))|$3
$RULES10:=F_{judge}(F_{ptp}(x,y,z),mall)\wedge (8,17)>>(enter \xrightarrow{(r,mall)} STOP)$	*road*3.(*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$1|$2)|$3)|*mall*[*work*]|$4→ *road*3.$3|*mall*[*work*].*car*[*c*1].(*driver*[*e*5,*e*6,*tom*,*s*,-,-,-,-,-,-,-].$1|$2)|$4
$RULES11:=F_{judge}(F_{ptp}(x,y,z),guardroom)\wedge (5,20)>>(read \xrightarrow{(r,record)} STOP)$	*parklot*[*work*].*mainarea*.((*guardroom*.(*guard*[*jack*,login,-]|*driver*[*e*5,*e*6,*tom*,-,-,-,-,-,-,-].$1|$2)|$3))|$4→ *parklot*[*work*].*mainarea*.((*guardroom*.(*guard*[*jack*,login,-]|*driver*[*e*5,*e*6,*tom*,-,-,-,-,-,-,login,*re*].$1|$2)|$3))|$4
$RULES12:=F_{judge}(F_{ptp}(x,y,z),guardroom)\wedge (5,20)>>(enter \xrightarrow{(r,guardroom)} STOP)$	(1) **guard enter guardroom** *parklot*[*work*].*mainarea*.(guard[*jack*,-,-]|*guardroom*.$0|$1)|$2→ *parklot*[*work*].*mainarea*.(*guardroom*.($0|*guard*[*jack*,-,-])|$1)|$2; (2) **driver enter guardroom** *parklot*[*work*].*mainarea*.((*driver*[*e*5,*e*6,*tom*,-,-,-,-,-,-,-,-].$1|*guardroom*.(*guard*[*jack*,*l*,-]|$2)|$3))|$4→ *parklot*[*work*].*mainarea*.((*guardroom*.(*guard*[*jack*,login,-]|*driver*[*e*5,*e*6,*tom*,-,-,-,-,-,-,-,-].$1|$2)|$3))|$4
$RULES13:=F_{judge}(F_{ptp}(x,y,z),guardroom)\wedge (5,20)>>(exit \xrightarrow{(r,guardroom)} STOP)$	*parklot*[*work*].*mainarea*.(*guardroom*.($0|*guard*[*jack*,login,-])|$1)|$2→ *parklot*[*work*].*mainarea*.(*guard*[*jack*,-,-]|*guardroom*.$0|$1)|$2
$RULES14:=(5,20)>>(enter \xrightarrow{(r,mainarea)} STOP)$	*road*2.($0|*driver*[*e*5,*e*6,*tom*,-,-,-,-,-,-,-,-].$1)|*parklot*[*work*].*mainarea*.$2|$3→ *road*2.$0|*parklot*[*work*].*mainarea*.(*driver*[*e*5,*e*6,*tom*,-,-,-,-,-,-,-,-].$1|$2)|$3;
$RULES15:=(exit \xrightarrow{(r,mainarea)} STOP)$	*parklot*[*work*].*mainarea*.(*guardroom*.($0|*guard*[*jack*,login,-])|$1)|$2→ *parklot*[*work*].*mainarea*.(*guard*[*jack*,-,-]|*guardroom*.$0|$1)|$2

陈小颖(1997－), 女, 硕士, CCF 学生会员, 主要研究领域为软件工程, 形式化方法, CPS 系统.

祝义(1976－), 男, 博士, 教授, CCF 高级会员, 主要研究领域为软件工程, 形式化方法, CPS 系统, 智能软件开发.

赵宇(1997－), 男, 硕士, CCF 学生会员, 主要研究领域为软件工程, 形式化方法, 软件缺陷预测, 机器学习.

王金永(1983－), 男, 博士生, CCF 学生会员, 主要研究领域为时空约束系统规约, 协同无人驾驶安全性分析, 形式化模型检测.