软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(2):427−446 [doi: 10.13328/j.cnki.jos.004562]

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

一种状态事件故障树的时间特性分析方法*

徐丙凤[1], 黄志球[1], 胡 军[1,2], 魏 欧[1], 李伟漳[1,3]

[1](南京航空航天大学 计算机科学与技术学院,江苏 南京 210016)

[2](计算机软件新技术国家重点实验室(南京大学),江苏 南京 210023)

[3](南京航空航天大学 航天学院,江苏 南京 210016)

通讯作者: 徐丙凤, E-mail: xubingfeng@nuaa.edu.cn, http://www.nuaa.edu.cn

摘 要: 状态事件故障树是一种适合于描述构件化嵌入式系统失效因果链的建模技术,其顶层事件描述失效发生的结果.对顶层事件发生的平均时间进行分析,是获得系统平均失效时间参数的一种有效方法,可为系统的安全性评估提供支持.由于状态事件故障树缺乏严格语义,使得必须先对其进行形式化描述才能进行定量分析.为此,提出了一种基于交互马尔可夫链的状态事件故障树时间特性分析方法.首先,精化交互马尔可夫链的交互动作,建立接口交互马尔可夫链模型,并基于该模型对状态事件故障树的构件和逻辑门进行形式语义描述;其次,通过并行组合构件与逻辑门的形式语义模型,得到整个状态事件故障树的形式语义模型,并在该过程中使用弱互模拟对状态空间进行约简;然后,基于状态事件故障树的形式语义给出顶层事件发生的平均时间计算方法;最后,给出飞机着陆雷达控制系统和喷淋防火系统的状态事件故障树时间特性分析的实例研究.为构件化系统失效时间特性的分析提供了一种新方法.

关键词: 状态事件故障树;交互马尔可夫链;平均时间分析;形式化方法

中图法分类号: TP311

中文引用格式: 徐丙凤,黄志球,胡军,魏欧,李伟漳.一种状态事件故障树的时间特性分析方法.软件学报,2015,26(2):427−446. http://www.jos.org.cn/1000-9825/4562.htm

英文引用格式: Xu BF, Huang ZQ, Hu J, Wei O, Li WW. Time property analysis method for state/event fault tree. Ruan Jian Xue Bao/Journal of Software, 2015,26(2):427−446 (in Chinese). http://www.jos.org.cn/1000-9825/4562.htm

Time Property Analysis Method for State/Event Fault Tree

XU Bing-Feng[1], HUANG Zhi-Qiu[1], HU Jun[1,2], WEI Ou[1], LI Wei-Wei[1,3]

[1](College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

[2](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

[3](College of Astronautics, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: State/Event fault tree (SEFT) is a modeling technique for describing the causal chains which lead to failure in component-based embedded systems, and the top event of SEFT describes the result of the failure. One important way for capturing the mean time parameter of system failure is to quantitatively analyze the mean time of the top event occurrence, which provides support for system safety evaluation. However, it is necessary to formally describe SEFT semantics in order to quantitatively analyze the time property. In this paper, a time property analysis method for SEFT based on interactive Markov chain (IMC) is presented. Firstly, interface interactive Markov chain (Interface-IMC) is proposed based on refining the interactive action of IMC. Secondly, semantics of components and logic gates in SEFT are formally described by Interface-IMC. Thirdly, the semantics of SEFT is obtained by composing all the Interface-IMCs generated in the above steps. During this process, weak bisimilarity technique is applied to reduce state space. Then, a quantitative time

* 基金项目: 国家自然科学基金(61272083, 61170043); 回国留学人员科研启动基金(SEM 2012); 中央高校基本科研业务费专项资金(CXZZ11_0218, NS2012129); 江苏省普通高校研究生科研创新计划(CXZZ11_0218)

收稿时间: 2013-04-21; 修改时间: 2013-07-09; 定稿时间: 2014-01-10

analysis method is presented based on the formal semantic model of SEFT. Finally, the time analysis processes for the SEFT of aircraft radar landing control system and sprinkler system are illustrated by the proposed method. The method provides a new solution for analyzing time properties of component-based system failure.

Key words: state/event fault tree; interactive Markov chain; mean time analysis; formal method

嵌入式实时系统已在航空航天、核工业、公共交通等安全关键领域中得到广泛应用,其系统失效将会导致财产的重大损失、环境的破坏甚至人员的伤亡[1].因此,对嵌入式实时系统中可能出现的危险失效状况进行安全相关时间特性(如危险失效的平均时间[2]、故障间隔时间[3]等)的建模与分析,已成为现代复杂嵌入式系统安全性保障的重要方法和研究热点.状态事件故障树(state/event fault tree,简称 SEFT)[4]是一种结合了故障树元素与状态机语义的系统行为安全性建模方法,其模型中的顶层事件表示系统的失效结果,并通过系统基本构件与不同类型逻辑门操作的组合来刻画系统失效结果发生的因果层次关系.SEFT 适合于对当前安全关键领域中所应用的大多数分布式构件化嵌入式系统的安全行为进行建模[5].

对 SEFT 顶层事件发生的时间进行定量分析,是获得系统失效时间参数的一种有效途径.但由于 SEFT 模型中尚缺乏严格语义,为分析其时间特性,需要给出其准确语义.目前,对 SEFT 进行严格的语义描述已有一些相关研究工作,如,文献[6,7]采用传统的 Petri Net 对 SEFT 进行语义描述,但 Petri Net 缺少描述 SEFT 构件之间消息交互的形式语义,使得对 SEFT 中的构件和逻辑门进行描述之后仍需要专业人员进行手动合并和修改才能够形成完整的分析模型;并且在形成 SEFT 完整语义模型的过程中,难以进行状态空间的约简.此外,由于 SEFT 具备构件化特征以及描述系统动态行为的能力,因此也很难使用诸如二叉决策图(binary decision diagrams,简称 BDD)[8]等之类的技术对其进行准确描述.本文工作给出了一种基于交互马尔可夫链(interactive Markov chain,简称 IMC)的 SEFT 时间特性分析方法,能够给出符合 SEFT 模型特征的形式语义模型并进行 SEFT 时间特性的自动定量分析.考虑到 SEFT 顶层事件发生的平均时间即该失效在系统中出现的平均时间,该时间特性是评估系统安全性的一项非常重要的指标[9],因此,本文主要针对 SEFT 中顶层事件发生的平均时间分析展开论述.

本文第 1 节对 SEFT 的建模元素进行详细描述,并给出一个 SEFT 的建模实例.第 2 节提出一个新的接口交互马尔可夫链模型(interface interactive Markov chain,简称 Interface-IMC).该模型将 IMC 中的交互动作精化为输入、输出动作,并给出并行组合以及弱互模拟的语义.第 3 节采用 Interface-IMC 对 SEFT 构件行为和逻辑门的语义进行建模.第 4 节通过并行组合方式得到 SEFT 的全局行为语义模型,在组合的过程中,采用弱互模拟对状态空间进行约简,并设计 SEFT 顶层事件发生的平均时间分析方法.第 5 节分别给出飞机着陆雷达控制系统和喷淋防火系统的 SEFT 建模以及时间特性分析的实例研究.第 6 节进行相关工作的比较.最后,对全文进行总结并阐述未来的工作.

1 状态事件故障树

本节概要描述了状态事件故障树的基本建模元素,并给出了一个飞机着陆雷达控制系统的状态事件故障树应用实例.

状态事件故障树(SEFT)[6]模型中包含了状态机元素和故障树元素,其在传统故障树的基础上进一步区分了系统行为状态和事件,且引入了显式的事件符号和因果边用于描述构件状态的转换以及构件之间失效的因果关系.在 SEFT 中,系统各构件通过多种类型的逻辑门进行连接,通过构件行为的描述以及构件之间的关系描述系统中失效发生的因果链.其基本建模元素如图 1 所示,包括:

(a) 构件(component):表示具有相对独立功能的软件或硬件实体,通过端口(port)与其他构件进行连接.

(b) 端口(port):表示 SEFT 中构件以及逻辑门之间的交互,包括事件的输入/输出端口、状态的输入/输出端口这 4 种类型.

(c) 逻辑门(gate):表示多个构件失效时相互之间影响的逻辑层次关系.其中,

(1) AND 门表示所有输入失效都发生时输出失效才会发生;

(2) OR 门表示所有输入当中有一个失效发生时输出失效就会发生;

(3) VOTING 门表示当前输入当中的某几个失效发生时输出失效发生;

(4) PAND 门表示当前输入以从左到右的顺序全部发生时输出失效才会发生.

(d) 事件(event):表示一种原子事件,即,无时间延迟的触发动作.

(e) 状态(state):表示构件中相关变量的抽象等价类集合.在每个时间点,系统应处于有限状态集合中的某个状态,并且在一定的时间区间内状态不变.如:飞机的高度是相关变量,该变量具有{too low, acceptable,too high}这 3 种状态,在每个时间点,系统处于这 3 个状态中的一个,并且在某个时间范围内都处于该状态.

(f) 边(edge):表示状态之间转换,包含两种类型:

(1) 一种是时序边,其停留时间根据指数分布随机改变;

(2) 另一种是因果边,即事件触发引起状态改变.

(a) (子)构件 (b) 端口(状态输入/事件输入/状态输出/事件输出) (c) 门(因果链的连接) (d) 事件 (e) 状态 (f) 时序边(前驱/后继关系) (g) 因果边(触发/卫式关系)

Fig.1 Main modeling elements of SEFT

图 1 SEFT 的主要建模元素

图 2(a)描述了一个构件内部的时序边,系统在 S_1 状态的停留时间服从参数为λ指数分布,之后转换到 S_2 状态.图 2(b)描述了系统中 C_1 和 C_2 两个构件之间的因果边,C_1 构件中事件 E_1 的发生触发构件 C_2 中 E_1 的发生.

(a) 概率事件 (b) 触发事件

Fig.2 Different ways of event occurrence

图 2 事件发生的不同方式

我们可以使用 SEFT 模型对安全关键系统中导致某个失效结果的原因建立起失效因果关系链.图 3 给出了一个飞机着陆雷达控制系统(aircraft radar landing control system,简称 ARLCS)的 SEFT 失效模型示例.该例中的失效结果为“飞机高度低于指定着陆高度时着陆轮却未放下”.该系统中主要包含 3 个构件,分别是着陆轮控制器(landing wheel controller)、雷达系统接口(radar interface)、雷达数据验证器(radar data validator):

- 着陆轮控制器用于控制着陆轮收起或者放下,当飞机高度小于 100m 或在地面时,着陆轮必须放下.
- 雷达接口负责提供当前飞机所处高度的数据.
- 雷达数据验证器用于监控雷达接口是否正常工作以及所提供的数据是否正确,当发现雷达接口故障时发出预警.

该 SEFT 模型通过两个 AND 门以自底向上的方式连接 3 个构件,描述了该失效结果发生的因果关系链.即,在飞机的着陆轮收起的情况下,当飞机的高度小于 100m 时,雷达接口出现故障且雷达数据验证器未能正常预警,出现着陆轮放不下的情形.其中,各构件的内部行为通过有限状态机进行描述,该有限状态机同时描述构件的动作与时间行为.如,雷达系统接口到达 failed 状态的时间服从参数为λ的指数分布.

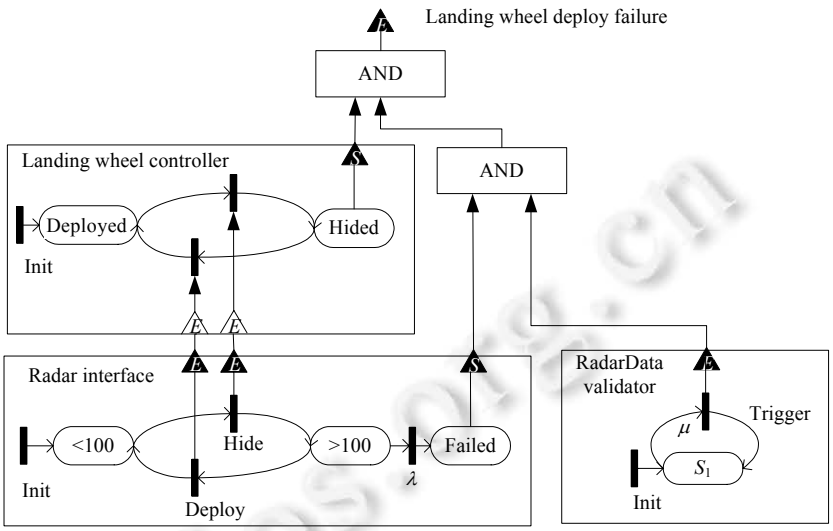


Fig.3 SEFT example of ARLCS

图 3 ARLCS 的 SEFT 示例

2 接口交互马尔可夫链

如前所述,SEFT 模型缺乏严格语义,难以直接对其进行分析.因此,本节建立了符合 SEFT 特征的形式语义模型,用于严格定义 SEFT 的语义.通过将交互马尔可夫链(IMC)[10,11]的交互动作语义精化为输入、输出动作,本节中提出了一种接口交互马尔可夫链(Interface-IMC)模型,定义了并行组合操作用于描述构件系统中的组合行为语义,并给出了用于对组合过程中的状态空间进行约简的弱互模拟操作定义.

2.1 Interface-IMC模型

考虑到 SEFT 构件内部采用有限状态机同时描述功能与时间特性,通过端口接收输入或进行输出与外部构件以及逻辑门进行交互,因而,SEFT 的构件和逻辑门可以看作是通过接收特定的输入信号或产生输出信号与环境进行交互.IMC 具有同时描述构件行为和时间特性的能力,并且提供了并行分析框架,适合描述 SEFT 中构件与逻辑门的行为特性.但 IMC 中缺少输入、输出动作的概念,难以准确描述 SEFT 中构件之间的交互特征.为了尽可能地使所建立的模型与 SEFT 中构件和逻辑门的行为语义相一致,本文对 IMC 中的交互动作进行精化,即,将 IMC 中的动作区分为输入、输出动作,并明确地标识出在当前状态上哪些输入动作是可接受的,而在当前状态没有标识的动作则视为不可接受的,从而建立起一个 Interface-IMC.

具体的 Interface-IMC 的形式定义如下:

定义 1. 一个接口交互马尔可夫链 P 是一个五元组$(S,s^0,A,\rightarrow,\rightarrow^M)$,其中,

- S 是状态集合.
- s^0 是初始状态.
- A 是动作集,其中,$A=(A^I,A^O,A^{\text{int}})$,$A^I$ 为输入动作集,A^O 为输出动作集,A^{int} 为内部动作集,记 $A^V=A^I\cup A^O$ 为 P 的可见动作集,A^{int} 为 P 的不可见动作集.
- $\rightarrow$是交互转换集合.通常将$(s,a,s')\in\rightarrow$记作 $s\xrightarrow{a}s'$.
- $\rightarrow^M\subseteq S\times\mathbb{R}_{>0}\times S$ 是马尔可夫转换集合.通常将$(s,\lambda,s')\in\rightarrow^M$ 记作 $s\xrightarrow{\lambda}{}^M s'$, 表示 s 到 s'转换发生的时间服从参数为λ的指数分布,λ称为转换率.

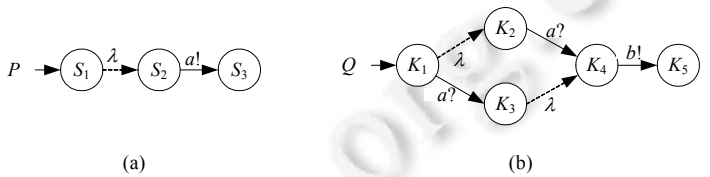
图 4 以图形化的方式描述了两个 Interface-IMC P 和 Q 的直观语义.P 和 Q 中存在两种类型的转换:

(1) 实线箭头表示采用动作进行标记的交互转换;

(2) 虚线箭头表示采用时间参数进行标记的马尔可夫转换.

由定义 1,图 4(a)中:Interface-IMC P 状态集为 $S=\{S_1,S_2,S_3\}$,初始状态为 S_1;动作集 A 中,$A^I=\varnothing$,$A^O=\{a\}$,$A^{\text{int}}=\varnothing$;交互转换集$\rightarrow=\{(S_2,a!,S_3)\}$;马尔可夫转换集$\rightarrow^M=\{(S_1,\lambda,S_2)\}$.

该 Interface-IMC 直观语义为:系统从初始状态 S_1 出发,经过一个停留时间 $t=T(\lambda)$转换到 S_2 状态,t 服从参数为λ的指数分布;然后,通过产生一个输出动作 a 转换到 S_3 状态.图 4(b)中的状态 K_2 则是通过接收输入动作 a 转换到 K_4 状态.



(a) (b)

Fig.4 Two examples of Interface-IMC

图 4 Interface-IMC 的两个示例

2.2 Interface-IMC的并行组合

并行组合操作让使用子构件的本地行为模型来构建构件系统的全局行为模型成为可能[12].在并行组合的过程中,两个 Interface-IMC 之间会存在交互同步,即,其中一个 Interface-IMC 输出动作 a,而另一个 Interface-IMC 正好需要接收该动作.不妨设有两个构件 P 和 Q 的 Interface-IMC 模型,记 P 和 Q 的共享动作集:

$$shared(P,Q)=(A_P^O\cap A_Q^I)\cup(A_P^I\cap A_Q^O).$$

即,一个 Interface-IMC 的输出动作是另一个 Interface-IMC 的输入动作,则两个 Interface-IMC 在执行该动作时需同步,且此相同动作的集合就是共享动作集 $shared(P,Q)$.如图 4 中,P 和 Q 的共享动作集为 $shared(P,Q)=\{a\}$. $P\|Q$ 可以描述如下:

(1) 如果动作不需要进行同步,则 P 和 Q 可以单独进行状态转换,即,如果 P 执行任何动作并转换为 P',则在并行语义中也存在相同的行为.即,$P\|Q$ 可以转换为 $P'\|Q$.

(2) 如果交互转换的动作需要进行同步,则 P 和 Q 需要同时执行该操作,即,$P\|Q$ 可以转换为 $P'\|Q'$.此时,当 P(或 Q)的输出动作和 Q(或 P)输入动作同步时,将产生一个并行组合 $P\|Q$ 中的内部动作.

Interface-IMC 的“并行组合”定义如下:

定义 2. P 和 Q 为两个 Interface-IMC,并行组合 $P\|Q$ 为

$$(S_P\times S_Q,(S_P^0,S_Q^0),((P\|Q)^I,(P\|Q)^O,(P\|Q)^{\text{int}}),\rightarrow_{P\|Q},\rightarrow_{P\|Q}^M),$$

其中,

- $(P\|Q)^I=(A_P^I\cup A_Q^I)\setminus(A_P^O\cup A_Q^O)$.
- $(P\|Q)^O=(A_P^O\cup A_Q^O)\setminus(A_P^I\cup A_Q^I)$.
- $(P\|Q)^{\text{int}}=A_P^{\text{int}}\cup A_Q^{\text{int}}\cup((A_P^I\cup A_Q^I)\cap(A_P^O\cup A_Q^O))$.
- $\rightarrow_{P\|Q}=\{(s,t)\xrightarrow{a}_{P\|Q}(s',t)\mid s\xrightarrow{a}_P s'\wedge a\in A_P\setminus A_Q\}\cup\{(s,t)\xrightarrow{a}_{P\|Q}(s,t')\mid t\xrightarrow{a}_Q t'\wedge a\in A_Q\setminus A_P\}\cup$
  $\{(s,t)\xrightarrow{a}_{P\|Q}(s',t')\mid s\xrightarrow{a}_P s'\wedge t\xrightarrow{a}_Q t'\wedge a\in shared(P,Q)\}$.
- $\rightarrow_{P\|Q}^M=\{(s,t)\xrightarrow{\lambda}^M(s',t)\mid s\xrightarrow{\lambda}_P^M s'\}\cup\{(s,t)\xrightarrow{\lambda}^M(s,t')\mid t\xrightarrow{\lambda}_Q^M t'\}$.

根据以上并行组合规则,图 4(a)中的 P 和图 4(b)中的 Q 并行组合结果为图 5(a)中的 $P\|Q$.

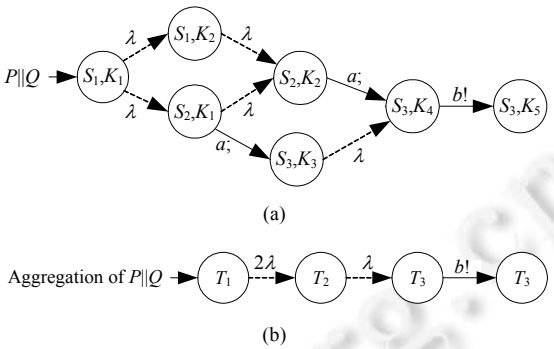


Fig.5 Composition and aggregation of Interface-IMC

图 5 Interface-IMC 的组合及聚合

2.3 Interface-IMC的状态等价

由于两个 Interface-IMC 并行组合后的状态空间大小与它们的状态空间数的乘积成正比,因此在多个 Interface-IMC 并行组合的过程中可能会出现状态空间爆炸的情况,需要采用合适的状态空间约简技术.本文采用 Interface-IMC 中状态等价的方法,通过将等价状态聚合,得到一个保持原模型行为语义但规模更小的模型.具体而言,我们采用类似 IMC 中弱互模拟[10]的概念进行 Interface-IMC 状态空间等价类的划分,然后再采用聚合操作进行状态空间的约简,即,将同一等价类中的状态合并为一个状态.与 IMC 不同的是,Interface-IMC 通过将 IMC 的交互动作精化为输入、输出动作,且明确地标识了状态上所允许的动作集,不允许的动作是不作标识的,这使得在并行组合过程中可以更有效地生成 Interface-IMC 模型的状态空间.在 Interface-IMC 中,两个状态视为弱互模拟等价,则要求所有这两个状态具有相同的可观察行为,且表现相同的性能特性.

设 P 是一个 Interface-IMC,$s,s'\in S$ 是 P 中的状态,若存在一个转换序列: $s\xrightarrow{a_1}s_1\xrightarrow{a_2}s_2\ldots\xrightarrow{a_n}s_n=s'$, $a_i\in A^{\text{int}}$,则记为 $s\Rightarrow s'$,并称$\Rightarrow$为弱转换关系.若存在状态 $s_1,s_2,a\in A^V$ 使得 $s\Rightarrow s_1\xrightarrow{a}s_2\Rightarrow s'$, 则记为 $s\overset{a}{\Rightarrow}s'$. 若 s 状态不存在内部转换或输出转换,则称该状态是稳态,即,无法通过瞬时转换到达另一状态.此外,从状态 s 到状态集 C 的累积转换率标识从状态 s 到达状态集 C 中所有状态的转换率之和,定义为

$$\Upsilon_M(s,C)=\sum\{|\,\lambda\,|\,s\xrightarrow{\lambda}s'\wedge s'\in C\,|\},$$

其中,{|…|}标记转换率的多个集合.集合 $C^{\text{int}}=\{s'|\exists s\in C.s'\Rightarrow s\}$ 包含所有通过弱转换($\Rightarrow$)到达 C 集合的状态.

Interface-IMC“弱互模拟”的定义如下:

定义 3. 设 $P=\langle S,s^0,A,\rightarrow,\rightarrow^M\rangle$是一个 Interface-IMC,$R$ 是 S 上的一个等价关系,当且仅当对于所有的$(s,t)\in R$, $a\in A$ 有以下条件成立时,R 是一个弱互模拟关系:

- $s\overset{a}{\Rightarrow}s'$,则 $t\overset{a}{\Rightarrow}t'$ 且$(s',t')\in R$;
- 对于 R 上的所有等价类 C,若 $s\Rightarrow s'$且 s'是稳态,则存在 $t'\in S$,使得 $t\Rightarrow t'$,t'是稳态且$\gamma_M(s',C^{\text{int}})=\gamma_M(t',C^{\text{int}})$.

若 P 和 Q 包含在某些弱互模拟关系当中,即,P 和 Q 是弱互模拟的,记作 $P\approx Q$.Interface-IMC 的弱互模拟定义对 IMC 弱互模拟定义进行了精化,将交互动作精化为输入、输出动作.对 Interface-IMC 进行弱互模拟等价类划分之后,可以通过聚合操作得到一个与原 Interface-IMC 弱互模拟等价但规模更小的模型,即,将相同等价类中的状态合并为同一状态.图 5(b)给出了针对图 5(a)中的 Interface-IMC 应用弱互模拟操作进行等价类划分并且进行状态空间聚合后的结果.可以看出:图 5(a)中的状态$(S_1,K_2),(S_2,K_1),(S_2,K_2),(S_3,K_3)$是弱互模拟等价的,这 4 个状态都是经过一个转换时间服从参数为λ的指数分布到达(S_3,K_4)状态,在图 5(b)中,这 4 个状态聚合为 T_2 状态.

定理 1. P_1和P_2是两个具有相同动作集的Interface-IMC,P_3是一个可以与P_1和P_2进行并行组合的Interface-IMC,则,

(1) $P_1 \approx P_2$,则 $P_1 || P_3 \approx P_2 || P_3$;

(2) $P_1 \approx P_2$,则 $P_3 || P_1 \approx P_3 || P_2$.

定理 1 的证明见附录.定理 1 表明,本文定义的弱互模拟针对并行组合操作是可替换的.该定理保证了可以利用弱互模拟化简之后的构件与其他构件进行并行组合,且最终的并行组合结果与不进行任何状态约简的并行组合结果是弱互模拟的.这样,在多个 Interface-IMC 的并行组合过程中,就可以采用弱互模拟操作来减少模型的规模.

3 SEFT 的形式化语义描述

对 SEFT 进行形式语义描述是分析其时间特性的基础,本节采用上一节中所建立的 Interface-IMC 模型对 SEFT 的构件与逻辑门进行严格的语义描述.

3.1 SEFT构件的语义

如第 1 节中所述,SEFT 中含有两种转换边:一种是时序转换边;一种是因果转换边.对于时序转换边,由于描述的是构件内部状态转换关系,可以采用 Interface-IMC 中的交互转换对其进行表示.对于随机延时的时序转换边,其表达的是系统经过一个指数时间的随机延迟转换到下一状态,可采用 Interface-IMC 中的马尔可夫转换进行表示;对于表示构件之间事件触发关系的因果转换边,存在触发事件与被触发事件之间的关系,可以使用 Interface-IMC 输入/输出动作的同步对因果转换边进行描述.

综上,采用 Interface-IMC 描述 SEFT 构件语义的方法如下:

(1) 构件的状态采用 Interface-IMC 的状态描述;

(2) 构件的事件采用 Interface-IMC 的动作描述;

(3) 构件内部状态通过时序边相连的情形,可采用 Interface-IMC 中的交互转换来表示;

(4) 构件之间的因果边采用 Interface-IMC 之间输出与输入动作发送与接收的同步,对动作触发进行描述;

(5) 状态和事件端口所表示的消息交互,可用 Interface-IMC 中消息的输入、输出类型对其进行描述.

考虑到本文的主要工作是对 SEFT 进行时间特性分析,因此在描述时序边语义时,对于不参与其他构件交互的构件内部状态的时序边上的信息,只保留对应的时间参数,而不需要保留事件信息.此外,因果边采用 Interface-IMC 之间的输出与输入动作的发送与接收进行描述,即,两个构件对于同一个动作同步,一个构件发送消息,另一个构件需要接收相同的消息.如图 2(b)中的因果边的语义可用 Interface-IMC 描述为:构件 C_1 行为表示为 $C_1.S_1 \xrightarrow{\lambda}{}^{M} C_1.S_2, C_1.S_2 \xrightarrow{E_1!} C_1.S_3$, C_2 行为表示为 $C_2.S_1 \xrightarrow{E_1?} C_2.S_2$, 通过 E_1 动作的同步,就可以描述构件事件发生的因果关系,即,构件 C_1 发送 E_1 消息,构件 C_2 接收到 E_1 消息后,则从 S_1 状态转换到 S_2 状态.

3.2 SEFT逻辑门的语义

下面使用 Interface-IMC 对 SEFT 逻辑门进行语义描述.SEFT 逻辑门上所有的状态接口输入或者事件接口输入都来自构件或其他逻辑门的输出动作,下面给出每一种逻辑门的语义定义,其中,每一种逻辑门的语义(记为 $[\![...]\!]_{ELT}$)是一个函数,以若干动作作为输入,输出为相应的 Interface-IMC:

(1) AND 门

图 6(a)给出了两输入 AND 门(AND,2)的语义,即,函数 $[\![(\text{AND},2)]\!]_{ELT} : A^3 \to$ Interface-IMC, 以 AND 门的输出以及两个输入信号作为参数.当 AND 门接收到两个输入信号时才会触发;

(2) OR 门

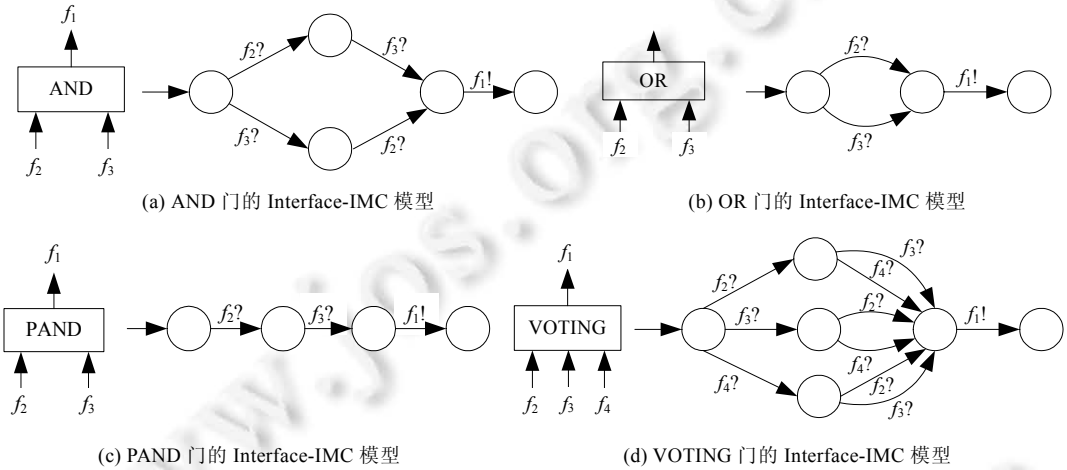
图 6(b)给出了两输入 OR 门(OR,2)的语义,即,函数 $[\![(\text{OR},2)]\!]_{ELT} : A^3 \to$ Interface-IMC, 以 OR 门的输出和两个输入信号为参数.当 OR 门接收到其中一个输入信号时,OR 门被触发;

(3) PAND 门

图 6(c)给出了两输入 PAND 门(PAND,2)的语义,即,函数 $[\![(\text{PAND},2)]\!]_{ELT}: A^3 \to$ Interface-IMC, 以 PAND 门的输出和两个输入作为参数.当 PAND 门接收到以从左到右的顺序发生的两个输入信号时被触发;

(4) VOTING 门

图 6(d)给出了 2/3 VOTING 门(VOTING,3,2)的语义,即,函数 $[\![(\text{VOTING},3,2)]\!]_{ELT}: A^4 \to$ Interface-IMC, 以 VOTING 门的输出和 3 个输入信号为参数.当 VOTING 门的 3 个输入信号中的两个两个输入信号被触发时, VOTING 门被触发.



(a) AND 门的 Interface-IMC 模型 (b) OR 门的 Interface-IMC 模型

(c) PAND 门的 Interface-IMC 模型 (d) VOTING 门的 Interface-IMC 模型

Fig.6 Semantic model of SEFT logic gate based on Interface-IMC

图 6 基于 Interface-IMC 的 SEFT 逻辑门语义模型

在得到 SEFT 构件与逻辑门的 Interface-IMC 语义模型之后,我们就可以通过并行组合 SEFT 中的构件和逻辑门的语义模型得到整个 SEFT 的行为语义模型,并基于此进行时间特性分析.

4 SEFT 的时间特性分析

本节介绍 SEFT 的时间特性分析方法.这里假定 SEFT 模型是合法模型(注:判断 SEFT 是否合法的具体步骤见文献[6]),并且 SEFT 各构件中的失效参数都已通过统计计算方法获得.以下首先给出 Interface-IMC 的并行组合状态空间约简方法的说明及相应算法,然后给出了 SEFT 顶层事件发生的平均时间的详细分析方法及相应的原型分析工具架构.

4.1 接口交互马尔可夫链的约简

Interface-IMC 并行组合过程中可能会出现状态空间爆炸的情形,下面对本文所采用的状态空间约简技术进行详细说明.根据第 2.3 节中的定理 1 可知,Interface-IMC 的弱互模拟针对并行组合具有可替换性,因此可以使用聚合之后的构件代替原构件继续与其他构件进行并行组合操作.因此在 Interface-IMC 组合过程中,每进行两个 Interface-IMC 的组合后,都可采用 Interface-IMC 的弱互模拟技术对所生成的 Interface-IMC 进行状态空间约简操作,约简状态空间之后再进行下一次组合.这样,就可以在组合的过程中减少所生成的状态.具体而言,进行状态空间约简分为两个步骤:

- 首先,划分组合模型中的状态空间的等价类;
- 其次,根据所划分的等价类对模型进行聚合操作.

其中,划分组合模型中的状态空间的等价类是该步骤的关键,下面着重对这部分进行介绍.

目前已经存在一些采用弱互模拟操作对模型进行约简的算法.比如,文献[13]提出了一种针对 IMC 计算最

小模型的算法,但是只能计算无环模型,而系统行为模型中往往存在环,所以难以直接采用这种方法.文献[14]的CADP(construction and analysis of distributed processes)工具集中的 BCG_min 可用于对标记转移系统(label transition system,简称 LTS)的相关模型进行约简,但难以直接将该方法应用到接口交互马尔可夫链模型中进行状态空间的约简.文献[10]给出了基于 IMC 的弱互模拟等价类划分方法,能够适用于计算存在环的 IMC 模型.而 Interface-IMC 将 IMC 的交互动作精化为输入、输出动作,所以在对 Interface-IMC 的状态空间进行弱互模拟等价类划分的过程中,与 IMC 中稳态[10]的定义不同的是,Interface-IMC 的稳态是指不存在输出转换和内部转换的状态.此外,采用交互动作对 Interface-IMC 的状态空间进行等价类划分时,需严格区分动作的输入、输出类型.

Interface-IMC 的弱互模拟等价类划分算法如算法 1 所示.下面首先给出算法中出现的一些公式和符号的相应说明.

若 Interface-IMC 中的 P 状态经过一系列的内部动作演化可以到达稳定状态则记为 $P\searrow^{\tau}P'$, 即, $P\searrow^{\tau}P'$ 当且仅当 $P\overset{\tau}{\Rightarrow}P'$ 且 $P'\overset{\tau}{\not\rightarrow}\wedge P'\overset{a!}{\not\rightarrow}$, $a!$为任意输出动作;否则,记为 $P\not\searrow^{\tau}P'$.

在算法中将用到函数γ_O:$S\times Act\times 2^S\mapsto$ {true,false},$\gamma_O(P,a,C)$为真当且仅当 P 可以通过 $\overset{a}{\Rightarrow}$ 演化到状态集 C 中,其中,a 可以是输入动作或输出动作.

该算法的思想是:首先,根据 Interface-IMC 状态空间中所有状态是否为稳态对状态空间进行等价类划分;再分别针对每个等价类利用动作集 Act 对状态空间进行等价类划分;此外,对于稳态等价类中的状态,进一步根据时间参数特性对状态进行划分.如此迭代,直到所有的动作都已用于进行状态空间的划分,最后得到一个划分成若干个等价类的状态空间.

算法 1. Computing weak bisimilarity classes.

$IRefine(Part,a,C):=\left(\bigcup_{X\in Part}\left(\bigcup_{v\in\{\text{true,false}\}}\{\{P\in X\mid\gamma_O(P,a,C)=v\}\}\right)\right)-\{\phi\}$;

$IM\_Refine(Part,C):=\left(\bigcup_{X\in Part}(M\_Spread(X,C)\cup\{Rest(X,C)\})\right)-\{\phi\}$;

$M\_Spread(X,C):=\bigcup_{v\in\mathbb{R}^+}\{Enrich(X,\{P\in X\}\mid P\overset{\tau}{\not\rightarrow}\text{ and }\gamma_M(P,C)=v)\}$;

$Enrich(X,Y):=\{P\in X\mid p\searrow^{\tau}P'\text{ implies }P'\in Y\}$;

$Rest(X,C)=X-\{P\in Y\mid Y\in M\_Spread(X,C)\}$.

Input: Interface-IMC $Interface\text{-}IMC_R=(S,s^0,A,\rightarrow,\rightarrow^M)$.

Output: $TC\_Part\cup TD\_Part$. //Weak bisimilarity classes of $Interface\text{-}IMC_R$

Function: Computing weak bisimilarity classes of $Interface\text{-}IMC_R$.

Substitute all internal actions of Interface-IMC with τ;

$TC\_Part:=\{\{P'\in S\mid P'\searrow^{\tau}\}\}-\{\phi\}$;

$TD\_Part:=\{\{P'\in S\mid P'\not\searrow^{\tau}\}\}-\{\phi\}$;

$Spl:=Act\times\{TC\_Part\cup TD\_Part\}$.

repeat

choose (a,C) in Spl;

$Old:=TC\_Part\cup TD\_Part$;

$TC\_Part:=IRefine\{TC\_Part,a,C\}$; //partition TC_Part by action a

$TD\_Part:=IRefine\{TD\_Part,a,C\}$; //partition TD_Part by action a

$TC\_Part:=IM\_Refine\{TC\_Part,C\}$; //partition TC_Part by Markov transition

$New:=(TD\_Part\cup TC\_Part)-Old$;

$Spl:=(Spl-\{(a,C)\})\cup(Act\times New)$; //update Spl

until Spl is empty;

return $TC\_Part\cup TD\_Part$.

算法主要分为两个部分:

第 1 部分是初始化操作,计算能够通过内部动作到达稳态的所有状态集 *TC_Part* 以及通过内部动作无法到达稳态的所有状态集 *TD_Part*,根据该计算结果修改 *Spl*,即,$Act\times\{TC\_Part\cup TD\_Part\}$.

第 2 部分是算法的主体部分,先从 *Spl* 中取出一个动作,使用该分类动作对 *TC_Part* 和 *TD_Part* 划分等价类;针对 *TC_Part* 进一步采用时间参数特性对其状态进行等价类划分,且根据新增等价类添加相应的分类动作到 *Spl* 中,当 *Spl* 为空时循环结束.

在整个算法的处理流程中,算法的第 1 部分是初始化操作,根据文献[15]的分析计算可知,其时间复杂度为 $O(n^{2.376})$.第 2 部分在最坏情况下需执行 n 次(即,每个划分中只含有一个状态),根据文献[10,16]可知,该过程的时间复杂度为 $O(n\times m)$,其中,n 为 Interface-IMC 的状态数,m 为马尔可夫转换数和经过内部传递闭包计算之后的交互转换数之和,m 最大值为 $n\times(n-1)$.因此,该算法的时间复杂度为 $O(n^3)$.

在得到了状态空间的等价类划分之后,我们就可以使用聚合方法对 Interface-IMC 模型中的等价类进行约简操作,即,分别对模型中的每个等价类进行处理,将处于同一个等价类的状态合并为一个状态,并构造 Interface-IMC 模型中其他状态到该新状态的概率转换以及动作转换.经过聚合操作之后,可以得到一个状态空间更小且与原来的模型弱互模拟的 Interface-IMC.例如,针对图 5(a)中的 Interface-IMC,可以使用该算法步骤进行等价类的划分,得到的等价类为$\{\{(S_1,K_1)\},\{(S_1,K_2),\ (S_2,K_1),(S_3,K_3)\},\{(S_2,K_2),(S_3,K_4)\},\{(S_3,K_5)\}\}$,进行聚合操作之后得到如图 5(b)所示的 Interface-IMC.

4.2 SEFT时间特性计算

采用 Interface-IMC 精确描述 SEFT 的语义模型之后,SEFT 的顶层事件在 Interface-IMC 中表示为一个输出动作,因此,SEFT 顶层事件发生的平均时间即为相应的 Interface-IMC 模型中顶层事件对应的输出动作发生时间的期望值.由 Interface-IMC 交互动作的定义可知,输出动作是由构件自身控制的,通常可认为输出动作的发生不产生时间延迟,因此,对该输出动作发生的时间期望值进行计算可进一步转换为计算该输出动作的出发状态到达时间期望值.在本文中,为了方便 SEFT 平均时间的计算,采用了一个 IMC 的分析工具 IMCA(interactive Markov chains analyzer)[17],可用于计算交互马尔可夫链(IMC)中从初始状态到达目标状态时间的最大和最小期望值.

如前文所述,为了准确地描述 SEFT 构件之间的消息交互,Interface-IMC 将 IMC 的交互动作精化为输入和输出动作,可以表达构件以及逻辑门之间的并行组合语义,并得到 SEFT 的完整语义模型.考虑到对 SEFT 顶层事件的平均时间的计算而言,在 Interface-IMC 中动作的输入、输出类型其实并不影响时间期望值的计算结果,因此,在 Interface-IMC 中计算到达特定状态的时间期望值时,可将输入、输出动作都处理为一类相同的 IMC 的交互动作.这样就可以很方便地使用 IMCA 来计算 Interface-IMC 的平均时间特性.

此外,从 SEFT 构件和逻辑门的形式语义模型的并行组合中可以得到,除了描述 SEFT 顶层事件语义的输出动作未形成内部动作以外,其他所有构件和逻辑门的输入动作都经过同步成为内部动作,经过弱互模拟操作后被约简.因此,在最终描述 SEFT 完整语义的 Interface-IMC 中,事实上只存在若干个马尔可夫转换和一个输出动作转换.令描述 SEFT 完整语义的 Interface-IMC $P=(S,s^0,A,\rightarrow,\rightarrow^M)$,SEFT 顶层事件所对应的输出动作的出发状态为 s'.在 IMCA 中,其接受的输入文件中包含了马尔可夫转换与交互转换以及所分析的初始状态 INITIALS 和目标状态 GOALS.我们可以 s^0 为初始状态 INITIALS,以 s' 为目标状态 GOALS,根据 P 中的转换关系构造 IMCA 可接受模型文件.IMCA 计算出的 INITIALS 到达 GOALS 时间的最大和最小期望值结果可以对应到 SEFT 中,即为顶层事件发生的最大和最小平均时间,也就是在系统中各构件的失效参数已知的情况下,系统中特定失效发生的最大和最小平均时间.这个结果可以为评估系统的安全性提供参考.对于安全关键系统而言,我们通常会采用最小平均时间作为评估系统安全性的标准.

综上所述,可对本文所提出的 SEFT 时间特性分析方法的实现模型进行概括,具体的处理流程如图 7 所示:

- 首先,对 SEFT 进行形式语义描述,创建所有逻辑门和构件的 Interface-IMC 模型;
- 其次,针对所得到的 Interface-IMC 集合进行并行组合操作,得到 SEFT 的完整语义模型,并且在组合的

过程中利用弱互模拟操作进行状态空间的约简;

- 最后,对 SEFT 的完整语义模型采用时间特性分析工具 IMCA(interactive Markov chains analyzer)[18]对顶层事件发生时间的期望值进行计算.

由于该处理过程中每个步骤都是可终止的,因此本文的方法是可终止的.

在 CSFATA 中,上述主要模块的功能是基于 Eclipse 平台开发的,已经完成其核心功能的设计与实现,目前正在进行与相关工具的集成和更多的实例研究.

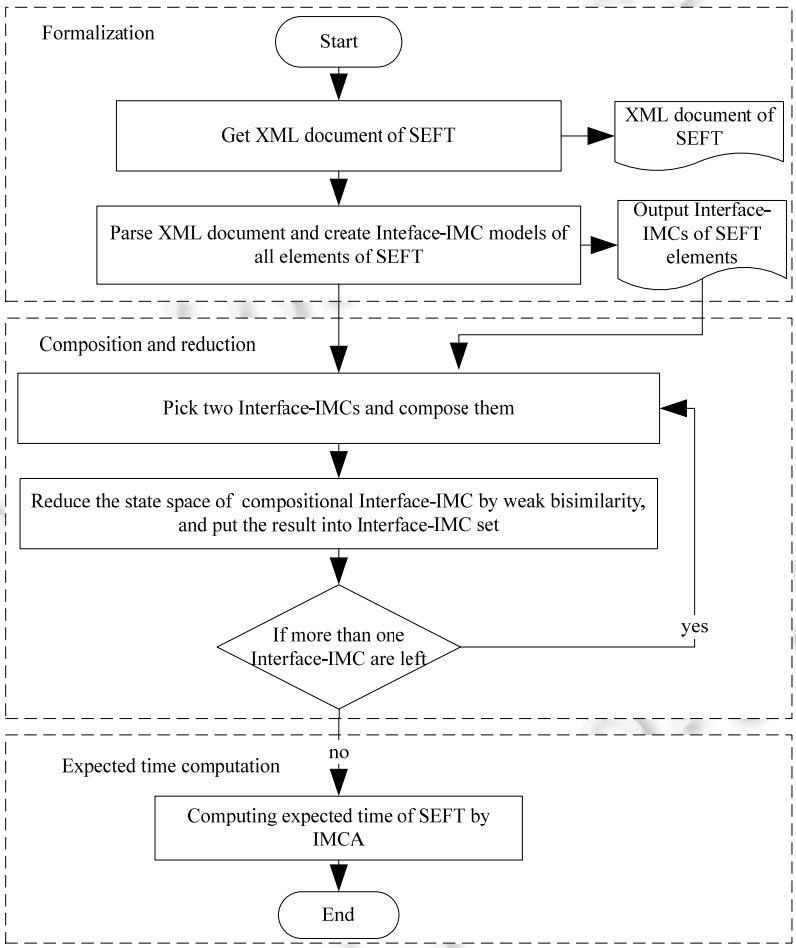


Fig.7 Implementation flow of SEFT time property analysis

图 7 SEFT 时间特性分析的实现流程

在上述工作的基础上,我们设计了一个相应的系统失效时间建模与分析原型工具(critical system failure time modeling & analyzer,简称 CSFATA).该工具的系统架构如图 8 所示,包括 4 个主要的子模块:

- 用户接口模块(user interface)用于与用户进行交互,接受用户的输入操作;
- Interface-IMC 模型建立模块(Interface-IMC generator)用于对 SEFT 的 XML 文件进行解析,并创建 Interface-IMC 模型;
- 核心算法处理模块(kernel algorithm processor)用于完成并行组合以及状态空间的约简,并利用已有的分析工具 IMCA 对最终的接口交互马尔可夫链模型进行时间期望值计算;
- 结果显示模块(result displayer)用于对计算结果进行输出显示.

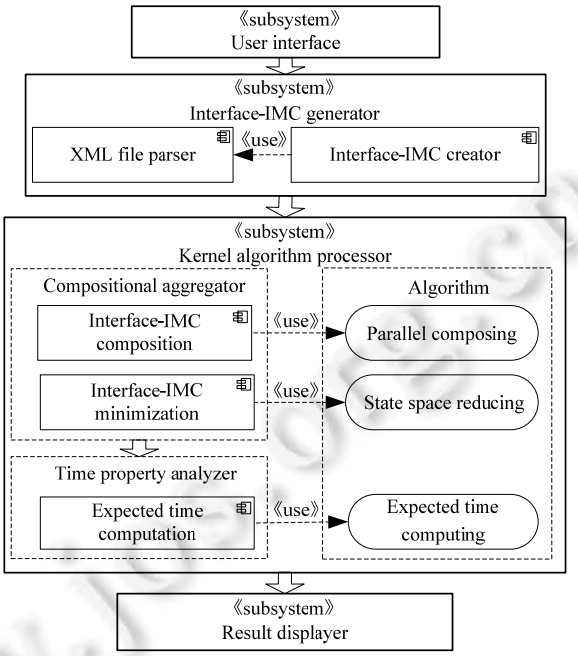


Fig.8 System architecture of CSFATA

图 8 CSFATA 的系统架构

5 实例研究

本节分别采用飞机着陆雷达控制系统和喷淋防火系统的状态事件故障树实例,根据所提出的方法对其顶层事件发生的最大和最小平均时间进行分析.为表述清晰,本文给出了完整的执行过程描述.

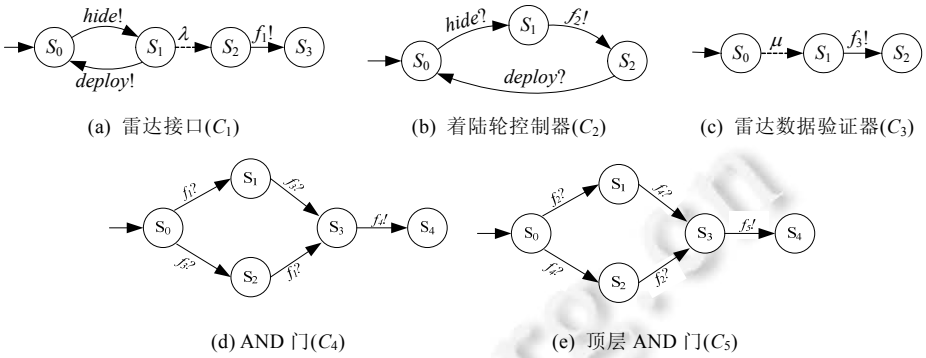
5.1 飞机着陆雷达控制系统SEFT实例

针对第 1 节中图 3 给出的飞机着陆雷达控制系统(ARLCS)的状态事件故障树模型,使用本文所提出的方法对其顶层事件发生的最大和最小平均时间进行分析.

首先,对于图 3 中的 SEFT,采用 Interface-IMC 给出其严格语义.该 SEFT 中一共包含 3 个构件与两个逻辑门,采用 Interface-IMC 精确描述 3 个构件与两个逻辑门的语义,如图 9 所示.本文根据 SEFT 中的构件与逻辑门的层次对得到的 Interface-IMC 进行编号.可以看出,对该 SEFT 的顶层事件发生的平均时间的分析就转换成计算到达 C_5 中 S_3 状态时间的期望值.

在得到 SEFT 中构件和逻辑门的 Interface-IMC 形式语义模型之后,通过对这些 Interface-IMCs 进行并行组合操作得到 SEFT 的精确语义模型.该实例中组合的顺序为$((C_1||C_2)||(C_3||C_4))||C_5$,每进行一步组合,都采用弱互模拟对组合产生的 Interface-IMC 状态空间进行约简,并将约简的结果再进行下一次组合.以此类推,形成最终的组合 Interface-IMC.

整个并行组合以及约简过程如图 10 所示.由于在该实例中 f_5!是顶层逻辑门的输出,SEFT 中的顶层事件发生时间的期望值计算即计算图 10(h)中 S_0 状态到达 S_3 状态的时间期望值.

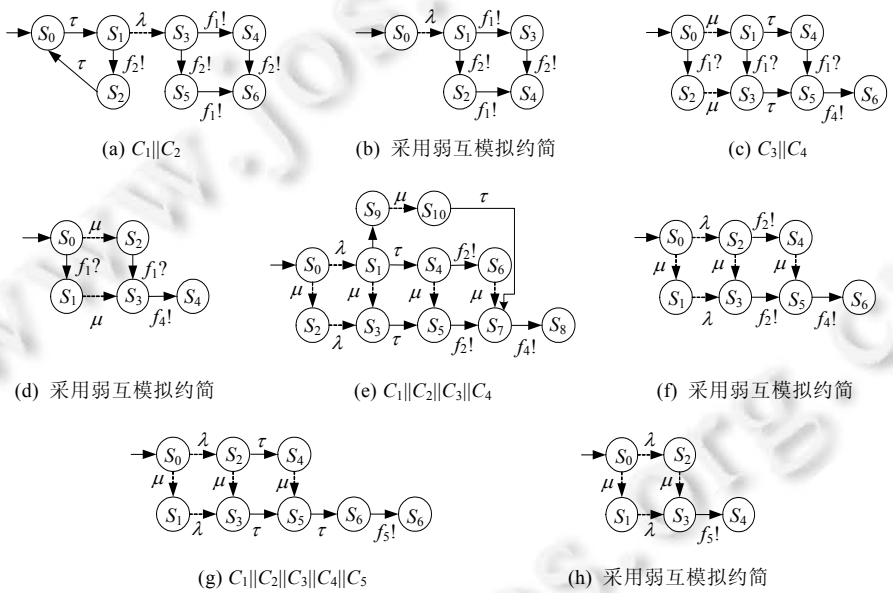


(a) 雷达接口(C_1) (b) 着陆轮控制器(C_2) (c) 雷达数据验证器(C_3)

(d) AND 门(C_4) (e) 顶层 AND 门(C_5)

Fig.9 Semantic models of SEFT components and logic gates in example

图 9 实例中 SEFT 构件与逻辑门的语义模型



(a) $C_1||C_2$ (b) 采用弱互模拟约简 (c) $C_3||C_4$

(d) 采用弱互模拟约简 (e) $C_1||C_2||C_3||C_4$ (f) 采用弱互模拟约简

(g) $C_1||C_2||C_3||C_4||C_5$ (h) 采用弱互模拟约简

Fig.10 Composition and aggregation process of the aircraft radar landing control system

图 10 飞机着陆雷达控制系统组合聚合过程

下面利用 IMCA 对图 10(h)进行时间期望值的计算.

图 10(h)的 IMCA 输入文件 ARLCS.imc 见表 1:初始状态为 S_0,目标状态为 S_3,并通过状态与转换描述了整个 Interface-IMC.这里,各构件的失效都满足指数分布,具体的失效率参照文献[19]中提供的数据.λ和μ的取值分别为 0.002 和 0.001,假设时间单位为小时(h).该实例关注对 SEFT 中顶层事件发生的最大和最小平均时间的分析,即,在 SEFT 的 Interface-IMC 语义模型中,从初始状态到目标状态时间的最大和最小期望值.使用 IMCA 对本文实例进行计算.通过计算,该状态事件故障树中顶层事件发生时间的最大和最小期望值相同,都为 1 166h.即,在该系统中各构件发生故障时间参数已知的情形下,顶层事件发生的最大和最小平均时间都为 1 166h.其物理含义为:飞机着陆雷达控制系统“飞机高度低于特定值时着陆轮未放下”失效发生的最大和最小平均时间均为 1 166h,且该时间值越大,系统能够正常工作的时间就越长.该参数可为评估飞机着陆雷达控制系统的安全性提供参考.

Table 1 IMCA input file for the example

表 1 实例的 IMCA 输入文件

ARLCS.imc
#INITIALS
S_0
#GOALS
S_3
#TRANSITIONS
S_0 S_1 0.001
S_0 S_2 0.002
S_1 S_3 0.002
S_2 S_3 0.001
S_3 S_4 f_5

5.2 喷淋防火系统SEFT实例

图 11 给出了一个喷淋防火系统的 SEFT 失效模型示例.该例中的失效结果为“喷淋防火系统失效”.该系统主要包含 3 个构件,分别是感烟传感器(smoke detector)、感温传感器(heat detector)、喷淋控制器(sprinkler controller).当感烟传感器或者感温传感器被触发时即进行火灾报警,然后喷淋控制器进行喷淋灭火.该 SEFT 模型通过一个 AND 门和一个 OR 门以自底向上的方式连接 3 个构件,描述了该失效结果发生的因果关系链.即,当感烟传感器和感温传感器都失效或者喷淋控制器失效时,造成喷淋防火系统失效.系统中各构件的失效都满足指数分布,具体的失效率参照文献[20]中提供的数据,即,感烟传感器的失效率λ_1=0.002,感温传感器的失效率为λ_2=0.004,喷淋控制器的失效率为λ_3=0.000001,这里假设时间单位为小时(h).

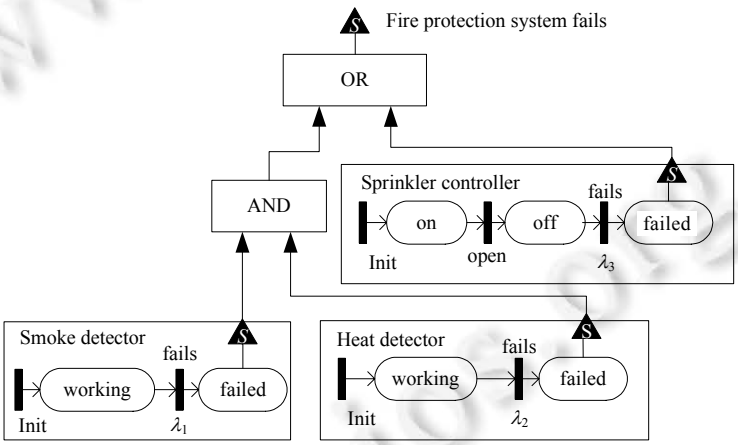
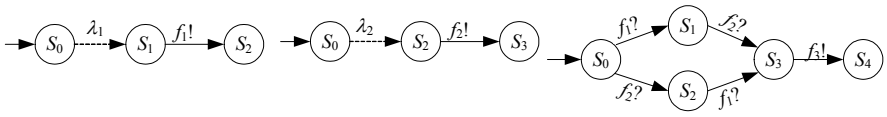


Fig.11 SEFT of sprinkler system

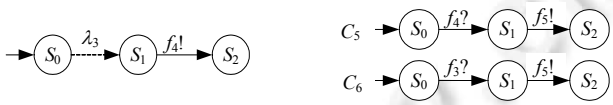
图 11 喷淋防火系统 SEFT

首先,针对图 11 中的 SEFT 采用 Interface-IMC 给出其严格语义.该 SEFT 一共包含 3 个构件和两个逻辑门,采用 Interface-IMC 描述 3 个构件与两个逻辑门的语义如图 12 所示.其中,由 OR 门的语义可以看出,任意一个输入的发生都能触发输出的发生.因此,这里将 OR 门描述为如图 12(e)所示的两部分,分别与对应的构件进行并行组合.本文根据 SEFT 中逻辑门的层次对得到的 Interface-IMC 进行编号.可以看出,对该 SEFT 的顶层事件发生的平均时间的分析,就转换成计算图 12 中从初始状态到达 C_5 和 C_6 中 S_1 状态时间的期望值.在得到 SEFT 中构件和逻辑门的 Interface-IMC 的形式语义模型之后,通过对这些 Interface-IMCs 进行并行组合聚合操作,得到 SEFT 的形式语义模型.由于 OR 门的逻辑语义描述分为两部分,因此并行组合也分为两部分,组合顺序根据模型的特征给定,分别为$((C_1||C_3)||C_4)||C_5$ 和 $C_4||C_6$.组合聚合过程如图 13 所示,由于 OR 门的存在,最终得到的 Interface-IMC 模型分别如图 13(f)和图 13(h)所示.由于在该实例中f_5!是顶层逻辑门的输出,计算 SEFT 中的顶层

事件发生时间的期望值即计算图 13(f)中 S_0 状态到达 S_3 或者图 13(h)中 S_0 状态到达 S_1 状态的时间期望值.



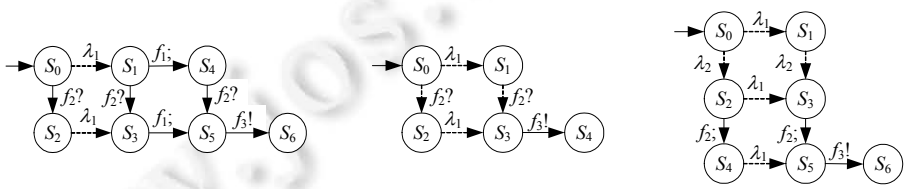
(a) 感烟传感器(C_1) (b) 感烟传感器(C_2) (c) AND 门(C_3)



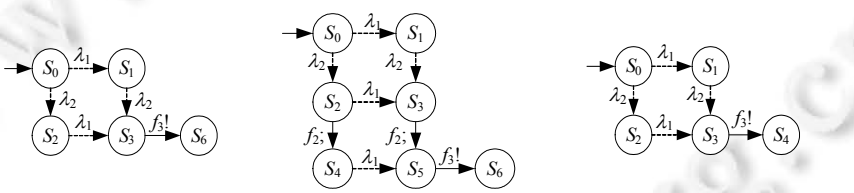
(d) 喷淋控制器(C_4) (e) OR 门(C_5,C_6)

Fig.12 Semantic models of sprinkler system SEFT components and logic gates

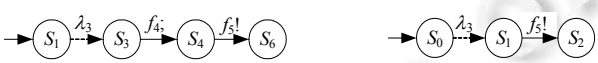
图 12 喷淋灭火系统 SEFT 构件和逻辑门的语义模型



(a) $C_1||C_3$ (b) 采用弱互模拟约简 (c) $(C_1||C_3)||C_2$



(d) 采用弱互模拟约简 (e) $((C_1||C_3)||C_2)||C_5$ (f) 采用弱互模拟约简



(g) $C_4||C_6$ (h) 采用弱互模拟约简

Fig.13 Composition and aggregation process of the sprinkler system

图 13 喷淋防火系统组合聚合过程

下面针对这些模型采用 IMCA 进行时间特性分析.图 13(f)和图 13(h)的 IMCA 输入文件 sprinkler1.imc 和 sprinkler2.imc 见表 2.

Table 2 IMCA input file for the sprinkler system example

表 2 喷淋防火实例的 IMCA 输入文件

Sprinkler1.imc	Sprinkler2.imc
#INITIALS	#INITIALS
S_0	S_0
#GOALS	#GOALS
S_3	S_1
#TRANSITIONS	#TRANSITIONS
S_0 S_1 0.002	S_0 S_1 0.000001
S_0 S_2 0.004	S_1 S_2 f_5
S_1 S_3 0.004	
S_2 S_3 0.002	
S_3 S_4 f_5	

该实例关注对顶层事件发生的最大和最小平均时间的分析,即,在 SEFT 的 Interface-IMC 语义模型中,从初始状态到目标状态时间的最大和最小期望值.使用 IMCA 对该实例进行计算.通过计算,该状态事件故障树中顶层事件发生时间的最大和最小期望值分别为 1 000 000h 和 583h.即,在该系统中各构件发生故障时间参数已知的情形下,顶层事件发生时间的最大和最小平均时间分别为 1 000 000h 和 583h.其物理含义为:喷淋防火系统"喷淋防火系统失效"发生的最大和最小平均时间分别为 1 000 000h 和 583h.值得注意的是,从上述例子中也可以看出:由于原始数据是指数分布的参数,因此,当原始数据越大,其平均时间就越小.

6 相关工作

在嵌入式实时系统开发的过程中获得与系统安全相关的时间特性,是对系统进行安全性评估的依据.目前,在系统失效因果链描述方面,故障树(fault tree,简称 FT)[21]已经在工业界广泛应用.但 FT 无法表达动态系统中的时间依赖以及序列依赖的行为,已有一系列工作对其建模能力进行扩展以支持动态系统以及实时系统的建模,主要包括:动态故障树(dynamic fault tree,简称 DFT)[22]在故障树的基础上扩展了动态门用于建模动作序列之间的依赖关系;时序故障树(temporal fault tree,简称 TFT)基于连续逻辑[23]、区间时序逻辑[24]等时序逻辑描述系统中的时序关系,并针对 TFT 进行定性分析,考虑可达性、可满足性等;此外,文献[25]给出了优先与门(PAND)的定义并处理序列依赖的动态行为;文献[26]提出一些新的表达失效之间时间特性的逻辑门,用于表达原因和结果之间的定量时间关系.以上方法都可以表达系统中的失效发生的因果关系链,但是仍缺少描述构件化系统中建模事件顺序以及状态依赖的能力.SEFT[4]将传统故障树元素与状态机元素结合起来,既能表达危害场景发生的因果关系链,又能描述动态系统行为中的时间依赖和序列依赖属性,适用于对构件化嵌入式系统失效因果链进行建模分析.

由于 FT,DFT 以及 SEFT 都缺乏形式语义,难以直接对它们进行分析,通常采用具有精确语义的形式模型严格描述其语义再进行分析.文献[27]采用二叉决策图描述 FT 的语义,在精确定义语义的基础上计算 FT 中顶层事件发生的概率;文献[28]通过利用广义随机 Petri Net 对 FT 进行精确语义描述,并分析其概率特性;文献[29]采用马尔可夫链模型对 DFT 进行精确的语义描述,并给出定量分析框架.而对于 SEFT 的定量分析,文献[6]采用 Petri Net 对其进行形式语义描述,并在此基础上进行顶层事件发生的概率特性分析.但由于 Petri Net 缺乏描述构件之间消息交互的语义,难以精确描述构件之间的并行组合,因此在使用 Petri Net 对 SEFT 进行形式语义描述之后,需要手动对模型进行修改以及合并才能生成完整的 SEFT 语义模型,且未提供状态空间约简方法,难以有效地自动生成完整的语义模型,因此难以基于该语义模型对状态事件故障树的时间特性进行分析.

在对系统的失效发生的时间特性分析方面,文献[30]利用具有时间依赖的故障树(fault tree with time dependencies,简称 FTTD)对失效发生的时间进行定量分析.文献[31]结合 FTTD 和 UML(unified modeling language)状态机图分析从原因的出现到最终失效发生之间所经过的时间间隔.文献[32]针对安全关键实时嵌入式系统中的基于失效依赖的最坏执行时间进行了详细的分析.同时,还存在一些工作对安全关键嵌入式软件的时间特性进行建模分析.文献[33]利用实时接口自动机描述实时系统的构件交互行为模型,并且对给定的实时规约进行验证.文献[34]中给出了一个实例研究,说明如何将一个简单的 UML 顺序图转换成一系列时间自动机,然后使用时间自动机的相关验证工具进行验证.文献[35]对基于场景的规约进行了时间扩展,用于表示实时系统的时间属性,并生成对应的时间自动机,以评估其表达能力.目前,针对 SEFT 时间特性进行分析的相关工作较少.

因此,本文考虑采用合适的形式模型精确描述 SEFT 的语义并支持其时间特性分析.SEFT 中顶层事件表示危害的发生,利用逻辑门连接构件表达危害发生的因果关系.构件内部行为采用有限状态机表达并同时描述功能与随机特性,通过端口与外部进行交互.因此,SEFT 中的构件和逻辑门可以看作是通过接收特定的输入信号或产生输出信号与环境进行交互.SEFT 构件内部的这种同时描述功能和随机时间的特征可以采用 IMC 进行描述,但由于 IMC 缺乏输入、输出动作的概念,难以有效地描述 SEFT 中构件之间的消息的发送与接收.所以,本文将 IMC 的交互动作精化为输入、输出动作,用于更精确地描述构件之间的交互行为.这样可以满足 SEFT 中同

时描述功能和随机时间特性以及构件之间交互的需求,因此可以采用 Interface-IMC 严格描述 SEFT 的语义,以便支持后续的时间特性分析.

与已有的工作相比,本文根据 SEFT 的特点设计接口交互马尔可夫链对其语义进行描述,并设计相应的时间特性分析方法,为安全关键嵌入式实时系统的安全性评估提供支持.

7 总结与未来的工作

本文针对状态事件故障树的时间特性分析提出了一种新的解决方法,主要工作包括:对交互马尔可夫链的交互动作语义进行了精化,建立接口交互马尔可夫链模型,并定义了并行组合用于描述构件的组合行为语义,以及弱互模拟操作用于约简状态空间;采用接口交互马尔可夫链对 SEFT 构件与逻辑门的语义进行精确描述,通过这些接口交互马尔可夫链的并行组合得到整个 SEFT 的语义模型,并在组合的过程中利用弱互模拟技术对状态空间进行约简;得到 SEFT 的严格语义模型之后,采用已有的时间特性分析工具 IMCA 对 SEFT 中顶层事件发生时间的最大和最小期望值进行计算;最后,以两个应用实例说明了本文方法的可行性.

下一步工作是构建更多复杂的系统实例,并且在接口交互马尔可夫链组合过程中,研究更为有效地减少状态空间的方法.在进行接口交互马尔可夫链并行组合的过程中发现,选择不同的组合顺序对所生成的状态空间的规模有较大的影响,因此,在未来的工作中将考虑如何得到最优的组合顺序.此外,还将进行基于接口交互马尔可夫链的 SEFT 概率特性分析以及系统的其他时间特性的分析.

References:

[1] Daskaya I, Huhn M, Milius S. Formal safety analysis in industrial practice. In: Proc. of the 16th Int'l Workshop on. Formal Methods for Industrial Critical Systems (FMICS 2011). LNCS 6959, Berlin: Springer-Verlag, 2011. 68−84. [doi: 10.1007/978-3-642-24431-5_7]

[2] Bukowski JV. Defining mean time-to failure in a particular failure-state for multi-failure-state systems. IEEE Trans. on Reliability, 2001,50(2):221−228. [doi: 10.1109/24.963132]

[3] Magott J, Skrobanek P. Timing analysis of safety properties using fault trees with time dependencies and timed state-charts. Reliability Engineering & System Safety, 2012,97(1):14−26. [doi: 10.1016/j.ress.2011.09.004]

[4] Kaiser B, Gramlich C, Forster M. State/Event fault trees—A safety analysis model for software-controlled systems. Reliability Engineering & System Safety, 2007,92(11):1521−1537. [doi: 10.1016/j.ress.2006.10.010]

[5] Elmqvist J, Nadjm-Tehrani S. Safety-Oriented design of component assemblies using safety interfaces. Electronic Notes in Theoretical Computer Science, 2007,182(29):57−72. [doi: 10.1016/j.entcs.2006.09.031]

[6] Kaiser B. State/Event trees: A safety and reliability analysis technique for software controlled systems [Ph.D. Thesis]. Kaiserslautern: Universität Kaiserslautern, 2007.

[7] Grunske L, Kaiser B, Papadopoulos Y. Model-Driven safety evaluation with state-event-based component failure annotations. In: Proc. of the 8th Int'l Symp. on Component-Based Software Engineering (CBSE 2005). LNCS 3489, Berlin: Springer-Verlag, 2005. 33−48. [doi: 10.1007/11424529_3]

[8] Bryant RE. Graph-Based algorithms for Boolean function manipulation. IEEE Trans. on Computers, 1986,100(8):677−691. [doi: 10.1109/TC.1986.1676819]

[9] National Aeronautics and Space Administration. NASA Software Safety Guidebook, NASA-GB-8719.13, 2004.

[10] Hersmans H. Interactive Markov Chains. Berlin: Springer-Verlag, 2002.

[11] Zhang L, Neuhäußer M. Model checking interactive Markov chains. In: Proc. of the 16th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2010). LNCS 6015, Berlin: Springer-Verlag, 2010. 53−68. [doi: 10.1007/978-3-642-12002-2_5]

[12] De Alfaro L, Henzinger T. Interface automata. ACM SIGSOFT Software Engineering Notes, 2001,26(5):108−120. [doi: 10.1145/503209.503226]

[13] Crouzen P, Hermanns H, Zhang L. On the minimisation of acyclic models. In: Proc. of the Concurrency Theory 2008 (CONCUR 2008). LNCS 5201, Berlin: Springer-Verlag, 2008. 295−309. [doi: 10.1007/978-3-540-85361-9_25]

[14] Garavel H, Mateescu R, Lang F, Serwe W. CADP 2006: A toolbox for the construction and analysis of distributed processes. In: Proc. of the 19th Int'l Conf. on Computer Aided Verification (CAV 2007). LNCS 4590, Berlin: Springer-Verlag, 2007. 158−163. [doi: 10.1007/978-3-540-73368-3_18]

[15] Coppersmith D, Winograd S. Matrix multiplication via arithmetic progressions. In: Proc. of the 19th ACM Symp. on Theory of Computing. New York: Academic Press, 1990. 251−280. [doi: 10.1145/28395.28396]

[16] Bouali A. Weak and branching bisimulation in FCTOOL. Technical Report, 1575, Valbonne Cedex: INRIA Sophia Antipolis, 1992.

[17] Guck D, Han TT, Katoen JP, Neuhäußer MR. Quantitative timed analysis of interactive Markov chains. In: Proc. of the 4th NASA Formal Methods Symp. (NFM 2012). LNCS 7226, Berlin: Springer-Verlag, 2012. 8−23. [doi: 10.1007/978-3-642-28891- 3_4]

[18] Guck D. Interactive Markov chains analyzer. http://www-i2.informatik.rwth-aachen.de/imca/imc_analyzer.pdf

[19] Stamatelatos M, Vesely W, Dugan J, Fragola J, Minarick J. Fault Tree Handbook with Aerospace Applications. Washington: NASA, 2002.

[20] Maksimovic T. Data on reliability of sprinkler systems. 2011. http://dspace.vgtu.lt/bitstream/1/775/1/9_Maksimovic_S2.pdf

[21] Ortmeier F, Schellhorn G. Formal fault tree analysis-practical experiences. Electronic Notes in Theoretical Computer Science, 2007, 185(13):139−151. [doi: 10.1016/j.entcs.2007.05.034]

[22] Čepin M, Mavko B. A dynamic fault tree. Reliability Engineering & System Safety, 2002,75(1):83−91. [doi: 10.1016/S0951-8320(01)00121-1]

[23] Hansen KM, Ravn AP, Stavridou V. From safety analysis to software requirements. IEEE Trans. on Software Engineering, 1998, 24(7):573−584. [doi: 10.1109/32.708570]

[24] Schellhorn G, Thums A, Reif W. Formal fault tree semantics. In: Proc. of the 6th Biennial World Conf. on Integrated Design and Process Technology (IDPT 2002). Pasadena: Society for Design and Process Science, 2002. 1−8.

[25] Walker M, Papadopoulos Y. Qualitative temporal analysis: Towards a full implementation of the fault tree handbook. Control Engineering Practice, 2009,17(10):1115−1125. [doi: 10.1016/j.conengprac.2008.10.003]

[26] Palshikar GK. Temporal fault trees. Information and Software Technology, 2002,44(3):137−150. [doi: 10.1016/S0950-5849(01)00223-3]

[27] Reay KA, Andrews JD. A fault tree analysis strategy using binary decision diagrams. Reliability Engineering & System Safety, 2002,78(1):45−56. [doi: 10.1016/S0951-8320(02)00107-2]

[28] Bobbio A, Franceschinis G, Gaeta R, Portinale L. Exploiting Petri nets to support fault tree based dependability analysis. In: Proc. of the 8th Int'l Workshop on Petri Net and Performance Models (PNPM'99). IEEE Computer Society Press, 1999. 146−155. [doi: 10.1109/PNPM.1999.796561]

[29] Boudali H, Crouzen P, Stoelinga M. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. IEEE Trans. on Dependable and Secure Computing, 2010,7(2):128−143. [doi: 10.1109/TDSC.2009.45]

[30] Magott J, Skrobanek P. Method of time Petri net analysis for analysis of fault trees with time dependencies. Computers and Digital Techniques, 2002,149(6):257−271. [doi: 10.1049/ip-cdt:20020804]

[31] Merle G, Roussel JM, Lesage JJ, Bobbio A. Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events. IEEE Trans. on Reliability, 2010,59(1):250−261. [doi: 10.1109/TR.2009.2035793]

[32] Höfig K, Domis D. Failure-Dependent execution time analysis. In: Proc. of the Joint ACM SIGSOFT Conf.—QoSA and ACM SIGSOFT Symp.—ISARCS on Quality of Software Architectures—QoSA and Architecting Critical Systems—ISARCS (QoSA-ISARCS 2011). New York: ACM Press, 2011. 115−122. [doi: 10.1145/2000259.2000279]

[33] Hu J, Yu XF, Zhang Y, Li XD, Zheng GL. Scenario-Based consistency verification of component-based Real-time system designs. Ruan Jian Xue Bao/Journal of Software, 2006,17(1):48−58 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/48.htm [doi: 10.1360/jos170048]

[34] Firley T, Huhn M, Diethers K, Gehrke T, Goltz V. Timed sequence diagrams and tool-based analysis—A case study. In: France R, Rumpe B, eds. Proc. of the 2nd Int'l Conf. on UML (UML'99). LNCS 1723, Berlin: Springer-Verlag, 1999. 645−660. [doi: 10.1007/3-540-46852-8_45]

[35] Zhang PC, Li B X, Li WR. Syntax and semantics of timed property sequence chart. Ruan Jian Xue Bao/Journal of Software, 2010, 21(11):2572−2767 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3711.htm [doi: 10.3724/SP.J.1001.2010.03711]

附中文参考文献:

[33] 胡军,于笑丰,张岩,李宣东,郑国梁.基于场景构件式实时软件设计的一致性检验.软件学报,2006,17(1):48−58. http://www.jos.org.cn/1000-9825/17/48.htm [doi: 10.1360/jos170048]

[35] 张鹏程,李必信,李雯睿.时间属性序列图:语法和语义.软件学报,2010,21(11):2572−2767. http://www.jos.org.cn/1000-9825/3711.htm [doi: 10.3724/SP.J.1001.2010.03711]

附录:定理证明

定理 1. P_1 和 P_2 是两个具有相同动作集的 Interface-IMC,P_3 是一个可以与 P_1 和 P_2 进行并行组合的 Interface-IMC,则:

(1) $P_1 \approx P_2$,则 $P_1 \| P_3 \approx P_2 \| P_3$;

(2) $P_1 \approx P_2$,则 $P_3 \| P_1 \approx P_3 \| P_2$.

证明:首先证明定理 1 中的情形(1).为了证明 $P_1 \approx P_2$ 则 $P_1 \| P_3 \approx P_2 \| P_3$,即证明 $P_1 \approx P_2$ 的前提下,$P_1 \| P_3$ 与 $P_2 \| P_3$ 满足定义 3 所定义的弱互模拟关系(即,满足定义 3 中弱互模拟定义的两个条件).

- 首先证明定理 1 中的情形(1)满足定义 3 中的第 1 个条件.

P_1 和 P_2 是两个具有相同动作集的 Interface-IMCs,令 P 是它们的并集且 $x \approx_P y$,也就是说,存在一个 P 上的弱互模拟关系 R,使得 $(x,y) \in R$.定义关系 R' 如下:

$$R' = \{(x \| z, y \| z) \mid (x,y) \in R \wedge z \in S_{P_3}\}.$$

由于 R 是 P 上的等价关系,显然,R' 是 $P \| P_3$ 上的等价关系.

令 $(s\|u, t\|u) \in R'$,(s,t) 在 R 中.对于所有的 $a \in A$,令 $s \| u \overset{a}{\Rightarrow} s' \| u'$, 则存在 $s'' \| u''$ 和 $s''' \| u'''$,使得 $s \Rightarrow s''$,$u \Rightarrow u''$,$s'' \| u'' \xrightarrow{a} s''' \| u'''$,$s'' \Rightarrow s'$,$u''' \Rightarrow u'$.根据并行组合的定义可知,$s'' \| u'' \xrightarrow{a} s''' \| u'''$ 蕴含下列条件成立:

- $s'' \xrightarrow{a} s''' \wedge u''' = u'' \wedge a \in A(P) \wedge a \notin A(P_3)$ 或
- $u'' \xrightarrow{a} u''' \wedge s''' = s'' \wedge a \in A(P_3) \wedge a \notin A(P)$ 或
- $s'' \xrightarrow{a} s''' \wedge u'' \xrightarrow{a} u''' \wedge a \in shared(P, P_3)$.

也即,

- $s \overset{a}{\Rightarrow} s' \wedge u \Rightarrow u' \wedge a \in A(P) \wedge a \notin A(P_3)$ 或
- $u \overset{a}{\Rightarrow} u' \wedge s \Rightarrow s' \wedge a \in A(P_3) \wedge a \notin A(P)$ 或
- $s \overset{a}{\Rightarrow} s' \wedge u \overset{a}{\Rightarrow} u' \wedge a \in shared(P, P_3)$.

由于 $(s,t) \in R$,则存在一个 t',$(s',t') \in R$,使得:

- $t \overset{a}{\Rightarrow} t' \wedge u \Rightarrow u' \wedge a \in A(P) \wedge a \notin A(P_3)$ 或
- $u \overset{a}{\Rightarrow} u' \wedge t \Rightarrow t' \wedge a \in A(P_3) \wedge a \notin A(P)$ 或
- $t \overset{a}{\Rightarrow} t' \wedge u \overset{a}{\Rightarrow} u' \wedge a \in shared(P, P_3)$.

从并行组合的定义可知,$t \| u \overset{a}{\Rightarrow} t' \| u'$; 从 R' 的定义可知,$(s'\|u', t'\|u')$ 在 R' 中.

从以上可以看出:定义 3 的第 1 个条件对于 R' 也是成立的.

- 再证明定理 1 中的(1)满足定义 3 中的第 2 个条件.

从 R' 的定义中可以推导出它具有以下等价类:

$$S / R' = \{\{x \| y \mid x \in C\} \mid C \in S_P / R, y \in S_{P_3}\}.$$

由于 $(s,s') \in R$ 当且仅当 $\forall u \in S_{P_3}.(s \| u, s' \| u) \in R'$ 对于等价类 $[s\|u]_{R'}$:$[s\|u]_{R'} = \{s'\|u \mid s' \in [s]_R\}$,现令 $(s\|u, t\|u) \in R'$,$s\|u \Rightarrow s'\|u'$,$s'\|u'$ 是稳态,C 是 R' 的等价类且 $C \neq [s'\|u']_{R'}$.根据并行组合的定义可知,这表示 $s \Rightarrow s'$,$u \Rightarrow u'$ 且 s' 和 u' 是稳态.R' 的每个等价类可从 R 等价类和 S_{P_3} 中的一个状态导出.

令 D 是 R 的一个等价类,y 为该状态,因此,$C = \{s\|y \mid s \in D\}$ 且 $u' = y$ 蕴含 $D \neq [s']_R$.

注意到,如果 $u' \neq y$,则 $s'\|u'$ 不在 C 中.C 的后向闭包为

$$C^{\text{int}}=\{x'||y'|\exists x||y\in C.x'||y'\Rightarrow x||y\}=\{x'||y'|\exists x\in D.x'\Rightarrow x\wedge y'\Rightarrow y\}.$$

现在计算从 $s'||u'$ 到 C^{int} 的累计转换率:

$$\begin{aligned}\gamma_M(s'||u',C^{\text{int}})&=\sum\{|\lambda|\exists x\in D.s'\xrightarrow{\lambda}{}^{M}x'\wedge x'\Rightarrow x\wedge u'\Rightarrow y|\}+\sum\{|\lambda|\exists x\in D.u'\xrightarrow{\lambda}{}^{M}y'\wedge s'\Rightarrow x\wedge y'\Rightarrow y|\}\\&=\sum\{|\lambda|\exists x\in D.s'\xrightarrow{\lambda}{}^{M}x'\wedge x'\Rightarrow x\wedge u'=y|\}+\sum\{|\lambda|\exists x\in D.u'\xrightarrow{\lambda}{}^{M}y'\wedge s'=x\wedge y'\Rightarrow y|\}\end{aligned}$$

(由于s'和u'是稳态)

$$=\begin{cases}\gamma_M(s',D^{\text{int}})+\gamma_M(u',\{y\}^{\text{int}}), & \text{if } u'=y\wedge s'\in D\\ \gamma_M(s',D^{\text{int}}), & \text{if } u'=y\wedge s'\notin D\\ \gamma_M(u',\{y\}^{\text{int}}), & \text{if } u'\neq y\wedge s'\in D\\ 0, & \text{if } u'\neq y\wedge s'\notin D\end{cases}$$

$$=\begin{cases}\gamma_M(s',D^{\text{int}}), & \text{if } u'=y\wedge s'\notin D\\ \gamma_M(u',\{y\}^{\text{int}}), & \text{if } u'\neq y\wedge s'\in D\text{(因为}u'=y\text{蕴含}D\neq[s']_R).\\ 0, & \text{if } u'\neq y\wedge s'\notin D\end{cases}$$

由$(s,t)\in R$可知:$s\Rightarrow s'$且s'是稳态蕴含存在一个t',使得$t\Rightarrow t'$,t'是稳态,且$(s',t')\in R$且对于所有的R的等价类除了$[s']_R$之外有$\gamma_M(s',D^{\text{int}})=\gamma_M(t',D^{\text{int}})$.显然,$s'$和$t'$在$R$的相同等价类中.现在可以得到以下结论:

$$\gamma_M(s'||u',C^{\text{int}})=\begin{cases}\gamma_M(s',D^{\text{int}}), & \text{if } u'=y\wedge s'\notin D\\ \gamma_M(u',\{y\}^{\text{int}}), & \text{if } u'\neq y\wedge s'\in D\\ 0, & \text{if } u'\neq y\wedge s'\notin D\end{cases}$$

$$=\begin{cases}\gamma_M(t',D^{\text{int}}), & \text{if } u'=y\wedge t'\notin D\\ \gamma_M(u',\{y\}^{\text{int}}), & \text{if } u'\neq y\wedge t'\in D\\ 0, & \text{if } u'\neq y\wedge t'\notin D\end{cases}$$

$$=\gamma_M(t'||u',C^{\text{int}})\text{(因为}s'\notin D\text{蕴含}D\neq[s']_R).$$

因此,定义3中的第2个条件对于R'也是成立的.对于任何状态$z\in S_{P_3}$可知:如果$(x,y)\in R$,则$(x||z,y||z)\in R'$.因为R'是弱互模拟关系,则$x||z\approx y||z$得证.对于定理1中的情形(2),可以用类似的方法进行证明,因为并行组合具有完全的对称性.通过以上证明可以得出,本文定义的弱互模拟对于并行组合是同余的. □

徐丙凤(1986－),女,安徽安庆人,博士,讲师,CCF 会员,主要研究领域为软件验证,软件安全性分析.

黄志球(1965－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为软件工程,形式化方法,服务计算.

胡军(1973－),男,博士,副教授,CCF 会员,主要研究领域为软件验证,航空嵌入式系统设计,模型驱动工程.

魏欧(1974－),男,博士,副教授,CCF 会员,主要研究领域为软件验证,模型检测.

李伟湋(1981－),女,博士生,CCF 会员,主要研究领域为复杂软件可靠性分析,可靠性数据分析.