计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(12):163−171 [doi: 10.15888/j.cnki.csa.008210]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于使用场景的时间扩展 EFSM 建模与完整性验证①

刘卓媛, 尤　枫, 赵瑞莲, 尚　颖

(北京化工大学 信息科学与技术学院, 北京 100029)
通讯作者: 尚　颖, E-mail: Shangy@mail.buct.edu.cn

摘　要: 为了提高嵌入式实时系统软件的质量和可靠性, 采用基于模型的软件测试方法是最有效的途径之一. 但是, 由于该类软件具有实时特性且十分复杂, 一般的模型通常缺乏对其实时特性以及软件行为的描述, 且需要丰富的专业领域知识才能将其建立的较为精确完整, 导致建模的难度和成本增加, 难以保证测试的充分性和有效性. 使用场景是用户与软件之间的交互实例, 详细描述了软件的系统行为而不关注其内部的复杂结构. 因此, 为了降低建模的难度, 本文基于使用场景的规范化表示 Scene 来构建模型, 并使用时间扩展 EFSM 模型来描述该类软件的实时特性; 为了保证所建模型的完整性, 本文设计了模型完整性评估准则, 通过验证模型迁移中约束条件的完整性来确定模型是否完整的表征了系统的行为; 针对不完整的模型, 根据约束条件设计了待补全迁移生成策略生成待补全迁移, 并通过动态模拟模型的可行迁移序列执行过程将其补全到模型中, 以提高模型的完整性. 最后, 本文针对 4 个嵌入式实时系统软件构建时间扩展 EFSM 模型并进行了一系列的实验. 实验表明, 本文提出的方法不仅有效的构建了模型, 而且能够将生成的待补全迁移有效补全到模型中, 进一步提高了模型的完整性.

关键词: 使用场景; 时间约束; 时间扩展 EFSM; 模型完整性验证

引用格式: 刘卓媛,尤枫,赵瑞莲,尚颖.基于使用场景的时间扩展 EFSM 建模与完整性验证.计算机系统应用,2021,30(12):163–171. http://www.c-s-a.org.cn/1003-3254/8210.html

Time-Extended EFSM Modeling and Integrity Verification Based on Usage Scenarios

LIU Zhuo-Yuan, YOU Feng, ZHAO Rui-Lian, SHANG Ying

(School of Information Science and Engineering, Beijing University of Chemical Technology University, Beijing 100029, China)

Abstract: The model-based software testing method is one of the most effective ways to improve the quality and reliability of embedded real-time system software. However, general models usually lack the description of their real-time characteristics and software behavior for the reason that this type of software is complicated with the real-time characteristics. As a result, a wealth of professional domain knowledge is required to build them more accurately and completely. This gives modeling a rise in difficulty and cost. It is thus difficult to guarantee the adequacy and effectiveness of the test. The usage scenario is an example of the interaction between the user and the software, which describes the system behavior of the software in detail without paying attention to its internal complex structure. Therefore, to reduce the difficulty of modeling, this study builds the model based on the standardized representation of the usage scenario and uses the time-extended EFSM model to describe the real-time characteristics of this type of software; to ensure the integrity of the model, this study designs the evaluation criteria for model integrity to determine whether the model fully characterizes the behavior of the system by verifying the integrity of the constraints in the model transitions;

① 基金项目: 国家自然科学基金 (62077003, 6187202)
Foundation item: National Natural Science Foundation of China (62077003, 6187202)
收稿时间: 2021-02-22; 修改时间: 2021-03-19; 采用时间: 2021-04-06

for the incomplete model, a to-be-completed transition generation strategy is designed according to the constraints to generate the to-be-completed transition, and it is completed into the model through the execution process of the feasible transition sequence of the dynamic simulation model to enhance model integrity; finally, this study builds a time-extended EFSM model for four pieces ofembedded real-time system software and carries out a series of experiments. Experiments show that the method proposed in this study can not only build the model but also complement the generated to-be-completed transitions to the model, further improving model integrity.

Key words: usage scenarios; time constraints; time-extended EFSM; model integrity verification

1 引言

当前, 随着嵌入式实时软件在高科技领域 (尤其是在航空, 航天和现代武器制造等许多关键领域) 的日益增长的应用, 人们更加关注嵌入式实时操作系统的可靠性[1]. 软件测试是保证软件质量和可靠性的主要手段, 针对实时嵌入式软件有着不同的测试方法, 其中基于模型的软件测试技术由于测试效率高, 并且能够发现其他测试技术难以发现的故障[2], 常被应用于该类软件的测试.

实时嵌入式软件不仅具有实时性特征, 而且往往全部或部分表现出基于状态的行为[3], 因此针对该类软件的建模需支持事件驱动、状态转移描述、复杂动态交互行为和严格的时间限制等领域特征[4]. 近年来, 在实时嵌入式软件的测试中最常用的模型有 FSM、UML、EFSM 等. 其中, EFSM 模型由于能同时描述被测软件的控制流和数据流信息, 更精确地描述软件系统的动态行为[5], 常被用于软件测试中. 然而, 目前针对该类软件的基于模型的软件测试方法仍然存在以下问题: (1) 不能完全满足该类软件在建模及测试中对实时性的要求; (2) 软件通常非常复杂, 而目前大多数研究都是通过分析需求或功能类的文档去构建模型, 通常需要丰富的专业领域知识, 才能构建相对准确和完整的模型, 从而导致建模工作的成本和难度都比较高.

为了解决以上两大问题, 在基于模型的软件测试的研究中, 时间自动机[6,7] 在 FSM 模型上为迁移添加了时钟约束, 为状态添加了不变式约束, 以此来描述实时嵌入式软件的时间特征. 但是, 该模型缺乏对嵌入式软件行为信息的描述. Yin 等人基于 EFSM模型提出了 RT-EFSM (Real-Time Extended Finite State Machine model)[8,9] 模型扩展了 EFSM 模型中状态转换过程中对时间特征的描述, 能够有效表征嵌入式软件的实时特性, 但是在建模时需要对功能和需求文档深入分析, 建模成本较高.

而文献 [10] 从用户的角度描述软件的系统行为, 通过记录用户使用软件过程中与软件交互的场景实例, 来描述软件的主要功能、使用方式以及边界条件等主要信息[11], 因此基于使用场景构建模型既可以相对细致的描述软件的行为特征, 又不需要考虑软件内部的逻辑结构和特性, 很大程度降低了建模的难度. 在 Wang 等人的工作中[12] 利用动态分析获取用户与 Web 应用程序之间的交互信息, 将其表征为用户行为轨迹 (trace) 并映射到 EFSM 模型中, 有效实现了面向 Web 应用程序的 EFSM 模型的构建. 因此本文将基于使用场景构建实时嵌入式软件的 RT-EFSM 模型.

但是, 场景难以覆盖系统部件实例之间的所有交互[13], 特别是需要复杂的交互才能探究的软件系统行为. 因此, 基于使用场景构建的模型大多是不完整的. 为了验证模型的完整性, 严亚伟等人[14] 主要针对系统的数据完整性进行分析和验证. 针对数据完整性的定量评估问题, 提出使用概率计算树逻辑 (PCTL) 和马尔可夫决策过程 (MDP) 分别对完整性进行形式化的定义和评估, 实现了对完整性的定量验证, 为系统开发中的完整性需求提供支持. 文献 [15] 提出事件及其执行条件可以代表 Web 应用程序的行为, 依据事件和 JS 分支覆盖为 Web 应用程序 EFSM 模型定义了完整性准则, 对模型的完整性进行评估, 并提出了面向 Web 应用的 EFSM 模型自动补全方法, 提高了模型的完整性.

综上, 本文将场景表征为形式化的 *Scene*, 并从中识别出构建 RT-EFSM 模型的状态和迁移, 利用状态和迁移的等价性有效合并状态和迁移并确定迁移中变量条件的上下界, 从而构建 RT-EFSM 模型. 而对于实时嵌入式软件, 在同一触发事件的不同条件下, 会导致系统的状态、参数以及后续操作发生不同的变化. 因此该类软件的动态行为涉及用户使用该软件时触发的事

件、触发事件时的条件、触发事件后发生改变的系统状态、参数及后续操作. 由于事件和相应的执行条件共同决定了系统到达的状态和后续操作, 因此它们代表了该类软件的系统行为, 从而事件及条件的完整性决定了系统行为模型的完整性. 由于 RT-EFSM 模型中的迁移条件包括时钟条件和非时钟条件, 因此, 本文主要根据时钟变量条件及非时钟变量条件来设计模型完整性准则来验证模型的完整性, 并利用时钟变量条件在时间轴[9] 上的连续性以及非时钟变量条件的对立分支条件生成待补全迁移, 然后在模型上搜索可行的迁移序列以遍历它们, 通过执行生成的迁移序列, 将待补全迁移添加到模型中, 从而提高模型的完整性.

本文的主要贡献概括如下:

(1) 本文基于使用场景 Scene 自动化构建 RT-EFSM 模型;

(2) 本文定义了时钟变量和非时钟变量完整性准则来评估嵌入式软件的 RT-EFSM 模型的完整性;

(3) 针对 RT-EFSM 模型, 提出了模型的待补全迁移生成策略生成待补全迁移, 并利用模型补全方法将其添加到模型中;

(4) 本文以 4 个实时嵌入式系统为实验对象进行一系列实验来验证所提方法的有效性.

2 基于使用场景的时间扩展 EFSM 构建与完整性验证方法

2.1 方法整体框架

本文基于实时嵌入式软件的使用场景对软件的动态行为进行建模, 并对模型进行完整性评估及补全, 以保证模型的完整性及后续测试的有效性. 基于使用场景的时间扩展 EFSM 构建与完整性验证的方法框架如图 1 所示, 主要包括: (1) 基于使用场景构建 RT-EFSM 模型; (2) 设计完整性评估准则并对 RT-EFSM 模型进行完整性验证; (3) 模型补全. 对于不完整的模型, 设计待补全迁移生成策略生成待补全迁移, 并使用模型补全方法将待补全的迁移添加到模型中, 从而提高模型的完整性.

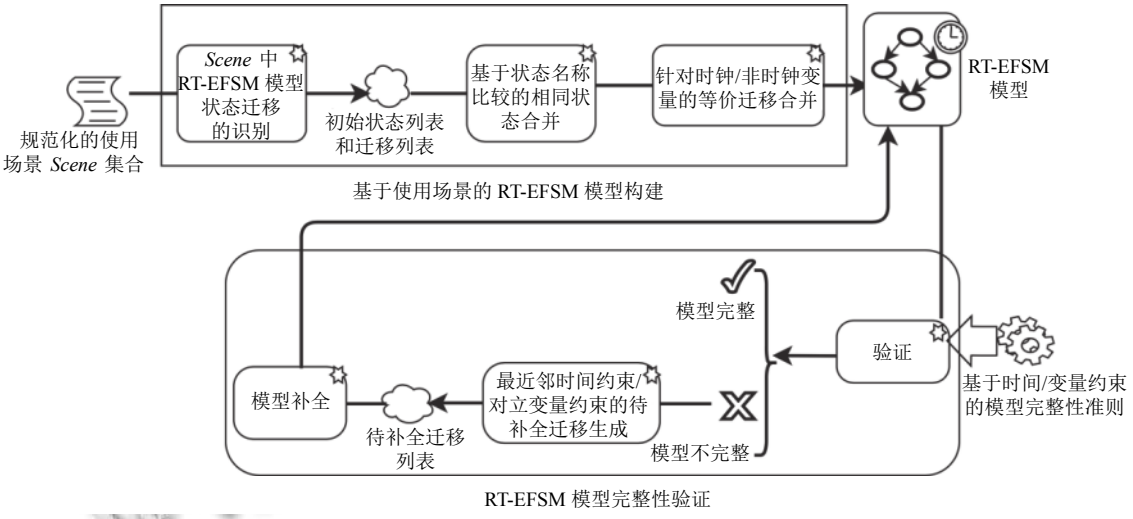

图 1 基于使用场景的 RT-EFSM 构建与完整性验证方法框架

2.2 基于使用场景的 RT-EFSM 模型构建

(1) 使用场景的规范化表示

使用场景主要记录了用户与系统进行交互时所执行的一系列行为活动, 是用例的一个实例, 反映了系统的性能和运行方式[10]. 使用场景一般用自然语言进行描述, 为了自动化地构建 RT-EFSM 模型, 需要对使用场景进行规范化表示, 本文将其定义为 *Scene*.

定义 1. 使用场景 (*Scene*): 一条 *Scene* 可以表示为 $Scene=<IA_0,\cdots, IA_j,\cdots, IA_{n-1}>$, 代表用户与实时嵌入式系统软件进行的一系列交互.

EScene 中的任一交互 IA_j 表示为<*Source*, *Event*, *Cond*[*Vcond*, *Ccond*], *Action*, *Target*>, 其中 *Source* 和 *Target* 分别表示系统软件在事件触发前后所处的状态, 用一个二元组<*Label*, *StateName*>表示, *Label* 表示状态标号, *StateName* 表示状态名称; *Event* 表示用户执行的操作, 即触发事件; *Cond* 表示触发事件时系统软件所

满足的条件, *Vcond* 和 *Ccond* 分别表示事件发生时所满足的变量约束条件和时间约束条件, *Action* 表示系统软件在事件触发后执行的响应及引起参数更改的后续操作.

(2) *Scene* 中的 RT-EFSM 状态与迁移识别

为了构建 RT-EFSM 模型, 需从规范化的 *Scene* 集合中识别模型的状态和迁移. 根据 RT-EFSM 模型的定义[9], 模型的状态集合可表示为 $S=\{s_0, \cdots, s_j, \cdots, s_n\}$, 其中的每一个状态 $s_j=<Label, StateName>$, *Label* 和 *StateName* 分别表示状态的标号和名称; 迁移集合 $T=\{t_0, \cdots, t_j, \cdots, t_n\}$表示允许模型从一个状态迁移到另一个状态的交互信息集合, 集合中的任一迁移表示为 $t_j=<Head(t_j), Event(t_j), Cond(t_j), Action(t_j), Tail(t_j)>$. 因此, *Scene* 中的任一交互 IA_j 可视为模型中的迁移 t_j, IA_j 中的 *Source* 与 *Target* 可分别对应 t_j 的 $Head(t_j)$ 与 $Tail(t_j)$, $<Event,Cond[Vcond,Ccond],Action>$对应于 t_j 中的$<Event(t_j),Cond(t_j),Action(t_j)>$. 从而当遍历完 *Scene* 集合中的每一条 *Scene* 时, 可获得 RT-EFSM 模型的状态列表 *S* 和迁移列表 *T*.

(3) 基于使用场景的 RT-EFSM 模型构建

根据上节分析, 根据 *Scene* 集合可获取 RT-EFSM 模型的状态列表 *S* 和迁移列表 *T*. 但是由于用户与系统可能存在多次重复的交互, 所以收集的 *Scene* 集合中的 *Scene* 之间可能存在相同的交互信息, 从而导致获取的状态列表和迁移列表中可能会包含相同的状态或迁移. 因此, 合并状态和迁移是构建 RT-EFSM 模型的关键.

根据对 *Scene* 集合的进一步分析, 由于嵌入式软件的实时性, 用户的某些操作必须在特定的时间内完成, 但是每一条 *Scene* 记录的只是在某一时间点用户与系统软件的交互信息. 因此, 在一定的时间范围内, 用户触发相同的事件后系统软件的状态变化和相应的响应是相同的; 而超过了某一时间点时, 即使触发相同的事件, 却会导致系统软件的状态和响应不同. 而这种实时性表征为迁移 t_j 中的时钟变量等式的不同取值. 因此迁移的合并过程中需要根据时钟变量的值集确定时间约束, 即迁移的合并可能涉及时钟约束的派生.

同样, 对于系统软件有次数限制的操作, 用户与系统软件进行交互的次数最终表征为模型迁移 t_j 中计数器变量[16] 等式的不同取值, 因此迁移的合并过程中需要根据计数器变量的值集确定计数器变量约束, 即迁移的合并可能涉及非时钟变量约束的派生.

为了有效合并迁移列表 *T* 并确定迁移的时钟约束和非时钟约束, 本文提出了等价迁移和弱等价迁移概念, 其定义如下:

定义 2. 等价迁移: 给定迁移 $t_1=<Head(t_1), lbl(t_1), Tail(t_1)>$和 $t_2=<Head(t_2), lbl(t_2), Tail(t_2)>$, 其中标签 $lbl(t)=<$Event$(t), Cond(t), Action(t)>$, $Cond(t)=[V_C,T_C]$, 当且仅当 $Head(t_1)=Head(t_2)$, $lb(t_1)=lb(t_2)$, 且 $Tail(t_1)=Tail(t_2)$ 时, t_1 与 t_2 等价.

定义 3. 弱等价迁移: 给定迁移 $t_1=<Head(t_1), lbl(t_1), Tail(t_1)>$和 $t_2=<Head(t_2), lbl(t_2), Tail(t_2)>$, 其中标签 $lbl(t)=<Event(t), Cond(t), Action(t)>$, $Cond(t)=[V_C,T_C]$, 当且仅当 $Head(t_1)=Head(t_2)$, $lbl(t_1)\sim lbl(t_2)$, 且 $Tail(t_1)=Tail(t_2)$ 时, t_1 与 t_2 弱等价. 其中 $lbl(t_1)\sim lbl(t_2)$ 表示标签信息中除 $Cond(t_1)$ 中的时钟变量和计数器变量与 $Cond(t_2)$ 中对应的同一变量的取值不同外, 其余各个组成部分均相等.

根据等价迁移和弱等价迁移的定义, 遍历迁移列表 *T*, 识别出其中的等价迁移及弱等价迁移, 并对这两种迁移进行合并. 在弱等价迁移的合并过程中, 能够派生出迁移的时间约束及计数器变量约束. 其派生过程如下: 遍历由多个互为弱等价迁移组成的弱等价迁移列表, 抽取其中的时钟变量和计数器变量及其取值, 获得<*var*, *valueSet*>对, 其中 *var* 代表某一变量, *valueSet* 代表该变量的值集. 对于变量 *var* 确定其值集 *valueSet* 的最小边界值 MIN 和最大边界值 MAX, 从而派生出变量 *var* 满足的约束为: MIN≤*var*≤MAX. 当确定时间约束及计数器变量约束后, 将弱等价迁移列表中的每一条迁移 t_j 上的时钟变量及计数器变量等式修改为派生的时间约束和计数器变量约束, 从而将弱等价迁移列表变为等价迁移列表, 实现了弱等价迁移的合并.

为了实现基于使用场景构建 RT-EFSM 模型, 设计了算法 1, 从使用场景中识别模型的状态和迁移, 并合并等价状态和迁移, 最终得到 RT-EFSM 模型.

算法 1. RT-EFSM 模型构建算法

1) 将使用场景表示为规范化的 *Scene*, 获得场景的 *Scene* 集合;

2) 识别 *Scene* 集合中 RT-EFSM 的状态及迁移, 获得其状态列表 *S* 及迁移列表 *T*;

3) 采用距离公式 (本文采用编辑距离) 对状态列表 *S* 中的任意两个状态的状态名称进行字符串比较, 判断两个状态节点是否为同一状态. 若相同, 则删除状态标号较大的状态, 获得无相同状态的状态集合 *S'*, 并修订迁移列表中的相应状态, 获得新的迁移列表 *T'*;

4) 遍历迁移列表 T'中的迁移, 将迁移列表 T'中当前迁移 t 与列表中后面的迁移的各个组成部分一一比较, 如果存在互为弱等价或互为等价的迁移, 则将它们加入到迁移列表 T_1'中, 如果不存在, 则将添加到 T''中;

5) 合并迁移列表 T_1'中的迁移将其变为等价迁移列表 T_1'', 保留 T_1''中的其中一条迁移添加到迁移 T'', 实现等价迁移的合并, 返回第 4) 步.

6) 最终获得状态集合 S'以及迁移集合 T'', 即 RT-EFSM 模型.

2.3 基于使用场景构建的 RT-EFSM 模型完整性验证

(1) 模型完整性准则

由于使用场景 *Scene* 难以涵盖应用系统的所有行为, 因此很难保证基于 *Scene* 构建的 RT-EFSM 的模型的完整性. 由于事件及其执行条件的完整性决定了系统行为模型的完整性, 因此,本文基于 RT-EFSM 模型迁移 t 上的条件 $Cond(t)$ 中的时间约束以及变量约束条件考虑模型的完整性, 提出了模型完整性评估准则.

定义 4. 时间约束对立迁移: 给定迁移 t_1=<$Head(t_1)$, $lbl(t_1)$, $Tail(t_1)$>, t_2=<$Head(t_2)$, $lbl(t_2)$, $Tail(t_2)$>, 其中标签 $lbl(t)$=<$Event(t)$, $Cond(t)$, $Action(t)$>, $Cond(t)$=[V_C, T_C], 如果 $Head(t_1)$=$Head(t_2)$, $Tail(t_1)$!=$Tail(t_2)$, $lb1(t)$ 中的触发事件 $Event(t)$ 相同, 且 $Cond(t_1)$与 $Cond(t_2)$ 中的同一时钟变量 c 的时间约束不同, 那么称 t_1 和 t_2 为时钟变量 c 的时间约束对立迁移.

根据如上定义, 互为时间约束对立迁移的多个迁移组成时钟变量 c 的时间约束对立迁移集 TR_c.

定义 5. 时区[9]: 设 RT-EFSM 模型中实时嵌入式软件中状态转换时钟集合为 L, 则 L 的取值范围为 [0, +∞). 分布在 L 时间轴上的时间点 L_1, L_2,⋯, L_k (L_1<L_2 <⋯<L_k) 将时钟集合 L 划分为 k+1 个子区域, 其中每一个子区域称为一个时区.

根据如上定义以及图 2 的时区划分可知, 对于时间约束 L_1<ω_1<L_2 和 L_2≤ω_2<L_3, 分布在时钟集合 L 时间轴上的时区是连续的.

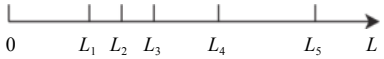

图 2 时区划分

定义 6. 变量约束对立迁移: 给定迁移 t_1=<$Head(t_1)$, $lbl(t_1)$, $Tail(t_1)$>, t_2=<$Head(t_2)$,$lbl(t_2)$,$Tail(t_2)$>, 其中标签 $lbl(t)$=<$Event(t)$,$Cond(t)$,$Action(t)$>, $Cond(t)$=[V_C,T_C], 如果 $Head(t_1)$=$Head(t_2)$, $Tail(t_1)$!=$Tail(t_2)$, $lb1(t)$ 中的触发事件 $Event(t)$ 相同, 且 $Cond(t_1)$与 $Cond(t_2)$ 中的同一非时钟变量 v 的约束相反, 那么称 t_1 和 t_2 为变量 v 的变量约束对立迁移.

根据上述定义, 本文提出的时间约束完整性准则以及变量约束完整性准则定义如下:

定义 7. 时间约束完整性准则: 假设 RT-EFSM 模型上的迁移集合为 T={t_1, t_2,⋯,t_j,⋯,t_n}(n>0), 对于任意一条迁移 t_j, 如果 t_j 上的任意时钟变量 c 存在时间约束对立迁移集 TR_c, 并且 TR_c 中 c 的所有时间约束分布在时钟集合 L 时间轴上的时区是连续的, 则称该模型满足时间约束完整性准则.

定义 8. 变量约束完整性准则: 假设 RT-EFSM 模型上的迁移集合为 T={ t_1, t_2,⋯,t_j,⋯,t_n}(n>0), 对于任意一条迁移 t_j, 如果 t_j 上的任意非时钟变量 v 存在变量约束对立迁移, 则称该模型满足变量约束完整性准则.

根据以上两种准则, 通过对已构建的 RT-EFSM 模型的迁移进行静态分析并进行完整性验证. 通过静态分析获得造成模型不完整的时间约束或变量约束及其所在的迁移, 根据这些信息对模型进行补全.

(2) 待补全迁移生成

对于不完整的模型, 其时间约束的时区分布不连续或者变量约束的对立约束不存在, 即模型没有表征在不连续的时区范围内或变量约束的对立约束下软件的动态行为. 因此, 本文提出了最近邻时间约束迁移生成策略以及对立变量约束迁移生成策略, 生成待补全迁移并将其补充到 RT-EFSM 模型中.

① 最近邻时间约束迁移生成策略

假设造成 RT-EFSM 模型不完整的时间约束为 p_0≤c≤p_1, 其所在的迁移为 t, 则可以根据时钟变量 c 的时间约束对立迁移集 TR 获得该变量的时间约束在时钟集合 L 时间轴上的时区分布, 从而得到与迁移 t 的时间约束分布最近邻的时区及其所在迁移 $[p_2,p_3](t_1)$, $[p_4, p_5](t_2)$(其中 p_3<p_0, p_1<p_4), 由于模型缺少对处于不连续时区 $(p_3, p_0)\cup(p_1, p_4)$ 中的时间点的软件行为进行表征. 因此待补全迁移表征在当前时区的最近邻时区表示的时间约束下, 触发相同的事件并满足相同的变量约束条件时, 系统可能的动态行为. 因此设待补全迁移的时间约束为 p_3<c<p_0, p_1<c<p_4, 构造的待补全迁移如式 (1) 所示:

$$\begin{cases} t_1' = < Head(t_1), Event(t_1), Cond(t_1)[Vc, p_3 < c < p_0], \\ \quad Action(t_1') = \text{null}, Tail(t_1') > \\ t_1' = < Head(t), Event(t), Cond(t)[Vc, p_3 < c < p_0], \\ \quad Action(t_1') = \text{null}, Tail(t_1') > \\ t_2' = < Head(t), Event(t), Cond(t)[Vc, p_1 < c < p_4], \\ \quad Action(t_2') = \text{null}, Tail(t_2') > \\ t_2' = < Head(t_2), Event(t_2), Cond(t_2)[Vc, p_1 < c < p_4], \\ \quad Action(t_2') = \text{null}, Tail(t_2') > \end{cases} \tag{1}$$

② 对立变量约束迁移生成策略

假设造成 RT-EFSM 模型不完整的非时钟变量 v 的约束为 V_C, 其所在的迁移为 t, 构造的待补全迁移表征为当该迁移的变量约束与迁移 t 的变量约束 V_C 相反时, 在相同的源状态下, 触发相同的事件并满足相同的时钟约束条件, 系统可能的动态行为. 因此构造的待补全的迁移如式 (2) 所示:

$$t' = < Head(t), Event(t), Cond(t)[\neg Vc, Tc], \quad Action(t') = \text{null}, Tail(t') > \tag{2}$$

根据这两个迁移补全策略, 生成了待补全的迁移列表 T_f. 为了提高模型的完整性, 需要将其添加到 RT-EFSM 模型中.

(3) RT-EFSM 模型补全

为了将生成的待补全迁移列表 T_f 中的迁移添加到模型中, 需要识别其后续状态和操作[15], 并通过动态执行可行迁移序列来遍历待补全迁移. 即模型补全方法如下: 对于 T_f= $\{t_0,\cdots, t_j,\cdots,t_n\}$ (n>0) 中的每一条迁移 t_j, 找到该迁移的源节点在原模型中的位置, 以该迁移作为初始迁移, 通过前向搜索生成可行迁移序列[16], 并利用 GA 算法找到触发该序列的输入数据[17], 使迁移序列与迁移数据共同构成可执行路径. 并将可执行路径作为具体的测试实例, 对其进行动态执行, 从执行结果中识别待补全迁移触发的状态和后续操作.

如果触发状态与所建立的 RT-EFSM 模型中某一状态相同, 即它们的状态名称相同, 则将该状态作为补全迁移的目标状态. 否则, 触发状态为新状态, 则将此状态添加到 RT-EFSM 模型的状态集合 S 中, 并将补全迁移的目标状态设置为新状态.

综上, 模型完整性验证与补全算法如算法 2 所示.

算法 2. RT-EFSM 模型完整性验证与补全算法

1) 遍历模型的迁移及迁移上的时钟约束和变量约束, 获取模型迁移上的时钟约束对立迁移 *TR*; 获取模型迁移上未找到变量约束对立迁移的迁移集 *CR*.

2) 根据 *TR* 与 *CR* 以及待补全迁移生成策略生成待补全迁移集 T_f.

3) 对于 T_f 中的每一条迁移 t_j, 判断是否能够在 RT-EFSM 模型中找到一条可执行迁移路径来触发 t_j, 如果能找到, 则将该迁移补全到模型中, 否则将该待补全迁移 t_j 放置在 T_f 的最后, 考虑另一个待补全迁移.

4) 如果 T_f 中所有待补全迁移拥有其相应的可执行迁移序列, 或者达到了时间预算, 那么模型补全就终止了.

3 实验分析

3.1 实验设计

(1) 研究问题

为了验证本文所提方法的有效性, 实验部分主要解决以下两个主要问题:

研究问题 1: 基于使用场景构建 RT-EFSM 模型的方法是否有效? 该方法的时间效率如何?

研究问题 2: 本文提出的待补全迁移生成策略是否有效?

(2) 实验对象

本文将 4 个实时嵌入式系统作为实验对象, 包括 ATM 机、三层楼的综合电梯系统、简易飞行安全系统及飞机发动机控制软件. 表 1 为 4 个嵌入式系统的使用场景的详细信息, 包括实时系统名称、使用场景总数、*Scene* 列表中包含的交互总数、状态总数、事件总数.

表 1 嵌入式系统的使用场景信息

系统名称	使用场景总数	交互总数	状态总数	事件总数
ATM	41	157	198	6
三层楼的综合电梯系统	18	100	118	7
简易飞行安全系统	38	261	297	14
飞机发动机控制软件	50	267	317	12

(3) 评估指标

① 由于用户与实时嵌入式系统的交互是通过事件实现的, 事件及其执行条件导致系统的不同响应. 而使用场景记录了用户与系统进行交互时所执行的一系列行为活动, 是用例的一个实例. 可以看出, 软件的系统行为与事件及使用场景密切相关. 因此, 为了验证建模方法的有效性, 即生成的模型是否将使用场景 *Scene* 集合中包含的信息完整表征出来, 该信息不仅包括每一次用户与系统软件进行交互的信息, 还包括使用场景所表示的可行执行序列. 因此, 本文提出了事件覆盖率 (Event Coverage, *EC*) 和场景覆盖率 (Scene Coverage, *SC*) 两个度量指标来衡量本文提出的模型构建方法的有效性.

事件覆盖率 (Event Coverage, *EC*), *EC* 衡量 RT-EFSM 模型中出现的事件占 *Scene* 集合中的所有事件的比例:

$$EC = \frac{|RT.Event|}{|S.scene|} \times 100\% \quad (3)$$

其中, |*RT.Event*|表示生成的 RT-EFSM 模型中包含的事件数, |*S.scene*|表示在 *Scene* 集合中的总事件数.

场景覆盖率 (Scene Coverage, *SC*), *SC* 表示 *Scene* 集合中能够被基于 RT-EFSM 模型生成的可执行测试路径覆盖的 *Scene* 条数占总 *Scene* 条数的比例:

$$SC = \frac{|RT.scene|}{|S.scene|} \times 100\% \quad (4)$$

其中, |*RT.scene*|表示被基于 RT-EFSM 模型生成的测试路径覆盖的 *Scene* 条数, |*S.scene*|表示在 *Scene* 集合中的总 *Scene* 条数.

② 本文的模型补全方法认为, 待补全迁移成功添加到模型中的标准是通过对模型的动态执行生成可行迁移序列来遍历待补全迁移. 即在模型中能够找到一条从初始节点开始到待补全迁移的可执行序列并生成可执行测试路径. 因此, 为了评估待补全迁移生成策略的有效性, 即根据该策略生成的待补全迁移能够成功添加到模型中, 本文提出了待补全迁移可行率 (for Complete Transition Feasibility, *fCTF*) 的评估指标.

待补全迁移可行率 (for Complete Transition Feasibility, *fCTF*) 表示待补全迁移列表中能够被在原 RT-EFSM 模型中动态模拟执行生成的可执行测试路径触发的待补全迁移条数占待补全迁移总数的比例:

$$fCTF = \frac{|Executable\ Paths\ Number|}{|forComplete\ Transition\ Number|} \times 100\% \quad (5)$$

其中, |*Executable Paths Number*|表示能够被基于 RT-EFSM 模型生成的可执行测试路径触发的待补全迁移条数, |*forComplete Transition Number*|表示待补全迁移列表中的待补全迁移总数.

3.2 实验结果与分析

对研究问题 1, 根据 4 个实时嵌入式系统的 *Scene* 集合构建其 RT-EFSM 模型. 如图 3 所示是根据 ATM 系统的使用场景 *Scene* 集合构建的 RT-EFSM 模型. 该模型中的节点代表模型的状态, 其中同心圆节点代表模型的初始节点, 边代表模型的迁移, 由模型的一个状态节点指向另一个节点. 例如图中的 S0 代表模型的初始节点, T1 表示模型的一个迁移.

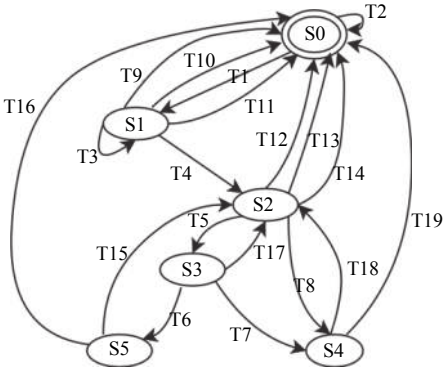

图 3 ATM 系统的 RT-EFSM 模型

表 2 显示了 ATM 系统的 RT-EFSM 模型的迁移信息, 包括迁移的源节点 Head(*t*), 目标节点 Tail(*t*),触发事件 Event(*t*), 迁移转换条件 Cond(*t*), 以及系统响应及操作 Action(*t*). 通过本文的模型构建方法, 可以将从初始 *Scene* 集合中包含了 198 个状态和 157 个迁移, 在合并相同的状态和迁移之后, 状态和迁移分别减少为 5 和 19, 其中的一个状态表示 ATM 系统的一个功能页面, 例如 S0 表示“输入密码页面”, 实现了状态、迁移的有效合并.

表 2 ATM 系统 RT-EFSM 模型迁移信息

迁移	Head(*t*)	Tail(*t*)	Event(*t*)	Cond(*t*)	Action(*t*)
T1	S0	S1	Card(pin,B,x)	(x≥1) & (x≤2)	Write("Enter Pin");N=0;x=0
T2	S0	S0	Card(pin,B,x)	x= =3	Write("Response timeout"); return_card();x=0;y=0
T3	S1	S1	Enter_pin(p)	(p!=pin) &(N≥0) &(N≤2) &(y≥10) &(y≤49)	Write("Wrong Pin,ReEnter"); N=N+1;x=0;y=0
T4	S1	S2	Enter_pin(p)	(p= =pin) &(N≥0) &(N≤2) &(y≥10) &(y≤49)	Write("SelectWithdrawal/GetBalance/Cancel");N=N+1;x=0;y=0
T5	S2	S3	Withdrawal()	(y≥9) &(y≤31)	Write("Enter amount");x=0;y=0
…	…	…	…	…	…
T19	S4	S0	Return()	(x≥1) & (x≤2)	return_card();x=0;y=0

如表 3 所示统计了使用本文提出的建模方法构建的 4 个系统的 RT-EFSM 模型的相关信息及时间效率, 包括系统名称、模型的状态数量、迁移数量、事件覆盖率 (*EC*)、场景覆盖率 (*SC*) 以及建模的时间开销.

表 3 模型构建的结果及时间效率分析

系统名称	状态数量	迁移数量	*EC* (%)	*SC* (%)	建模时间开销(s)
ATM	6	19	100	100	0.4621
三层楼的综合电梯系统	5	25	100	100	0.4839
简易飞行安全系统	5	34	100	100	0.4960
飞机发动机控制软件	12	33	100	100	0.5070

结合表 1 和表 3 可以看出, 本文的建模方法可以有效缩减模型的状态和迁移数, 例如三层楼的综合电梯系统的 *Scene* 集合中包含的状态数和迁移数分别为 118 和 100, 而所建的 RT-EFSM 模型中状态和迁移数减少到 5 和 25; 简易飞行安全系统的 *Scene* 集合中包含的状态数和迁移数分别为 297 和 261, 所建模型中的状态和迁移分别减少到 5 和 34. 除了状态及迁移的数量外, 我们利用两个衡量指标, 即事件覆盖率 (*EC*) 和场景覆盖率 (*SC*) 来度量各模型的覆盖情况, 以验证模型构建方法的有效性. 如表 3 的实验结果所示, 利用 *Scene* 构建的模型的 *EC* 和 *SC* 均为 100%. 即本文提出的建模方法可以保留 *Scene* 集合中的所有事件和场景表示的执行序列信息. 此外, 针对 4 个实时嵌入式系统, 模型构建的最大时间开销是 0.5070 s, 时间成本是可以接受的.

综上, 本文提出的模型构建方法可以有效合并 *Scene* 集合中的相同状态和迁移, 同时保证了 RT-EFSM 模型对实时嵌入式软件动态行为的覆盖且时间开销较小. 因此, 基于使用场景构建 RT-EFSM 的方法可以有效建立实时嵌入式软件的行为模型.

对于研究问题 2, 为了评价本文提出的待补全生成策略的有效性, 本文统计了根据待补全迁移生成策略生成的待补全迁移数量、能够被基于 RT-EFSM 模型生成的可执行测试路径触发的待补全迁移条数 (EPN), 以及待补全迁移可行率 (*fCTF*) 和不可行率 (*nfCTF*), 其中不可行率 $nfCTF = 1 - fCTF$.

由表 4 所示可以看出, 根据本文提出的待补全迁移生成策略可以生成待补全迁移, 例如该策略为 ATM 系统生成的待补全迁移数为 16, 能够被基于模型生成的可执行测试路径触发的待补全迁移条数为 15. 为三层楼的综合电梯系统生成了 10 个待补全迁移, 其中有 7 条待补全迁移能够被基于模型生成的可执行测试路径触发. 另一方面, 通过 *fCTF* 和 *nfCTF* 两个指标评估待补全迁移生成策略的有效性. 例如, 简易飞行安全系统的待补全迁移可行率为 86.67%, 不可行迁移可行率为 13.33%. 飞机发动机控制软件的待补全迁移可行率为 78.57%, 不可行迁移可行率为 21.43%. 其中 *fCTF* 最低为 70.00%, *nfCTF* 最高为 30.00%, 并且各个实时系统的待补全迁移可行率都远远高于其不可行率, 从而说明本文提出的待补全迁移策略能够生成待补全的迁移, 并且生成的待补全迁移的可行率较高, 即待补全迁移能够有效补全到模型中.

表 4 待补全迁移策略的有效性分析

系统名称	待补全迁移数量	EPN	*fCTF* (%)	*nfCTF* (%)
ATM	16	15	93.75	6.25
三层楼的综合电梯系统	10	7	70.00	30.00
简易飞行安全系统	15	13	86.67	13.33
飞机发动机控制软件	14	11	78.57	21.43

4 结论与展望

本文提出了一种基于使用场景自动构建 RT-EFSM 模型的方法, 设计了模型完整性验证的准则, 并通过该准则验证模型的完整性. 针对不完整的模型, 利用待补全迁移生成策略生成待补全的迁移, 并利用模型补全算法将其填补到模型中. 本文完成了 4 个实时系统模型的构建与完整性验证, 实验表明本文提出的自动建模方法可以将用户使用场景集中的冗余状态和迁移进行有效的合并, 花费的时间开销较小. 并且根据待补全迁移策略生成的待补全迁移能够有效补全到模型中, 从而提高了模型的完整性. 在后续工作中, 将扩大实验规模, 并进一步优化补全模型的方法和策略, 从而提高本文方法的有效性.

参考文献

1 Yin YF, Liu B, Zhong DM, *et al*. On modeling approach for embedded real-time software simulation testing. Journal of Systems Engineering and Electronics, 2009, 20(2): 420–426.

2 杨朝红, 宫云战, 肖庆, 等. 基于模型的软件测试. 北京化工大学学报, 2007, 34(S1): 85–88. [doi: 10.3969/j.issn.1671-4628.2007.z1.021]

3 Yin YF, Liu B, Su D. Research on formal verification technique for aircraft safety-critical software. Journal of

Computers, 2010, 5(8): 1152–1159.
4 Yin YF, Liu B, Ni HY. A survey on the formal testing techniques for real-time embedded software. The 2nd International Conference on Information Science and Engineering. Hangzhou: IEEE, 2010. 6426–6429. [doi: 10.1109/ICISE.2010.5691486]
5 潘雄, 郝帅, 苑政国, 等. 基于无关变量分离的 EFSM 测试数据进化生成. 北京航空航天大学学报, 2019, 45(5): 919–929. [doi: 10.13700/j.bh.1001-5965.2018.0531]
6 Saeedloei N, Kluźniak F. From scenarios to timed automata. Proceedings of the 20th Brazilian Symposium. Recife: Springer, 2017. 33–51.
7 Okano K, Yang P, Ogata S, *et al*. Deriving of time constants in timed automata for hazard transition sequences for STAMP/STPA. Procedia Computer Science, 2020, 176: 1392–1401. [doi: 10.1016/j.procs.2020.09.149]
8 Yin YF, Li Z, Liu B. Real-time embedded software test case generation based on time-extended EFSM: A case study. 2020 Wase International Conference on Information Engineering. Beidai: IEEE, 2010. 272–275.
9 Yin YF, Liu B, Ni HY. Real-time embedded software testing method based on extended finite state machine. Journal of Systems Engineering and Electronics, 2012, 23(2): 276–285. [doi: 10.1109/JSEE.2012.00035]
10 Lin LJ, Lu L. Automatic conversion from application scenarios to state diagrams. 2014 IEEE International Conference on Progress in Informatics and Computing. Shanghai: IEEE, 2014. 446–450. [doi: 10.1109/PIC.2014.6972375]
11 林丽洁. 基于 OCL 的应用场景到状态图的自动转换 [硕士学位论文]. 济南: 山东大学, 2015.
12 Wang WW, Guo JX, Li Z, *et al*. EFSM-oriented minimal traces set generation approach for web applications. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). Tokyo: IEEE, 2018. 12–21. [doi: 10.1109/COMPSAC.2018.00011]
13 张晓春. 基于用户场景的需求引出及其行为建模技术研究 [硕士学位论文]. 金华: 浙江师范大学, 2007. [doi: 10.7666/d.y1237629]
14 严亚伟, 周雁舟, 惠文涛. 模型检测在完整性形式化验证中的应用研究. 计算机工程与应用, 2017, 53(4): 59–63, 134. [doi: 10.3778/j.issn.1002-8331.1507-0081]
15 Zhao RL, Chen C, Wang WW, *et al*. Automatic model completion for web applications. Proceedings of the 20th International Conference. Helsinki: Springer, 2020. 207–227.
16 程俊, 李征, 赵瑞莲. 基于 EFSM 模型的不可行迁移路径判定. 内蒙古大学学报 (自然科学版), 2011, 42(5): 498–504.
17 Zhao RL, Harman M, Li Z. Empirical study on the efficiency of search based test generation for EFSM models. 2010 3rd International Conference on Software Testing, Verification, and Validation Workshops. Paris: IEEE, 2010. 222–231. [doi: 10.1109/ICSTW.2010.44]