基于有色 Petri 网的 SysML 序列图的分析与验证

王松锋，熊选东，张亮忠，付建丹
（解放军信息工程大学 电子技术学院，郑州 450004）

摘　要：针对 SysML 序列图本身缺乏分析和验证手段的问题，提出了一种序列图到有色 Petri 网的转换方法：定义了将序列图的常用操作转换为等价有色 Petri 网的转换规则，重点是把序列图的常用结构如可选结构、条件结构、并行结构以及循环结构等映射为有色 Petri 网。这当中既包含结构元素，如库所、变迁、输入/输出弧，又包含逻辑元素，如全局声明中的颜色集和变量、颜色集与库所、弧表达式以及初始标志。应用这些规则可以将序列图转换为有色 Petri 网模型，进而对其进行仿真分析，并可通过有色 Petri 网工具验证模型的无死锁性、可达性、有界性和活性。最后通过数字证书更新的实例分析了映射前后两种模型的语义，验证了映射的正确性。

关键词：系统建模语言；有色 Petri 网；序列图；建模；模型转换
中图分类号：TP301　　**文献标志码**：A　　**文章编号**：1001-3695(2012)09-3341-07
doi:10.3969/j.issn.1001-3695.2012.09.037

Analysis and verification of SysML sequence diagrams based on colored Petri net

WANG Song-feng, XIONG Xuan-dong, ZHANG Liang-zhong, FU Jian-dan
(*College of Electronic Technology, the PLA University of Information Engineering, Zhengzhou* 450004, *China*)

Abstract: Aiming at the problems of SysML sequence diagrams lack of analysis and verification methods, this paper presented a method for conversing sequence diagrams to the colored Petri net, defining the equivalent conversion rules for converting sequence diagrams of the common operations into a colored Petri net, mainly focusing on mapping sequence diagram's common structure such as optional structures, alternate structures, parallel structure and loop structure into colored Petri nets. They not only contained structure elements, such as place, transition, input and output arcs, but also contained the logic elements, such as the global declaration of the color sets and variables, color sets and places, and the initial marking, arc expression. Using these transformation rules, the method could transform sequence diagrams into colored Petri nets, and made its simulation and analysis. Besides of this, it could verify the characters of the model, such as absence-deadlocks, boundness, liveness, etc. Finally, the examples of digital certificates updated analysed the semantic of the model before and after mapping, verified the correctness of the mapping.

Key words: SysML; colored Petri net(CPN); sequence diagram; modeling; model transformation

0　引言

系统建模语言(systems modeling language, SysML)是 2006 年由对象管理组织(Object Management Group, OMG)和国际系统工程学会(International Council on Systems Engineering, INCOSE)提出的系统体系结构设计的标准建模语言，是统一建模语言 UML 在系统工程应用领域的延伸和扩展，可用于由软硬件、数据和人综合而成的复杂系统的集成体系结构说明、分析和设计[1]。

SysML 针对系统工程领域中系统设计与建模的特点，提供了可视化、图形化的系统建模支持，得到了 IBM、I-Logi、Sparx 等众多公司的支持，广泛应用于复杂系统建模。SysML 和 UML 一样，为了保持描述的清晰易懂，在给出自身的语义说明时，采用了半形式化的描述方法，使用自然语言描述约束和详细语义，力求实现形式严格和易于理解之间的平衡。因此，SysML 本身缺乏分析和验证的手段，只能进行静态建模，不能进行动态仿真。也就是说，对于 SysML 描述的图形，目前尚缺乏严密有效的验证和分析方法，同时也难以在模型实现之前进行仿真。针对这一问题，可采取的办法是将序列图转换为形式化描述的模型，然后进行分析、验证。

本文给出了 SysML 序列图的有色 Petri 网建模方法，将非形式化的 SysML 序列图转换为具有精确语义且可执行的有色 Petri 网模型，在非形式化的图形表示和形式化图形定义之间建立映射关系。主要目的就是采用 SysML 建模的同时，转换为有色 Petri 网并利用有色 Petri 网的分析技术对 SysML 模型的动态行为进行验证，既避免了形式化方法对数学基础要求较高、直接使用起来困难的局限性，又弥补了 SysML 缺少模型分析、无法进行分析和仿真等缺点。

1　国内外研究现状

在 SysML 和 UML 图形的形式化和验证方面，目前主要有以下几种方法：a)基于模型方法 Z 的转换[2,3]，将状态机图中各种状态模式转换为 Z 语言的模式运算来表达；b)基于逻辑方法的转换[4,5]，将状态机图转换到通用高阶逻辑形式规范语言 PVS 或者使用实时动作逻辑 RAL 对状态机图进行形式化描述与精化，进而进行验证；c)基于自动机的转换[6,7]，将状态机

收稿日期：2012-01-16；**修回日期**：2012-03-12

作者简介：王松锋(1986-)，男，硕士，主要研究方向为系统工程、Petri 网(375574648@qq.com)；熊选东(1965-)，男，研究员，主要研究方向为系统工程及计算机应用；张亮忠(1985-)，男，硕士，主要研究方向为系统工程；付建丹(1986-)，男，硕士，主要研究方向为系统工程.

图的操作映射到一个特定形式的自动机,使用基于自动机理论的模型检验方法来验证状态机图的线性时态逻辑性质;d)基于验证工具语言的转换[8],将时序图转换为模型检验器 SPIN 的输入语言 Promela 后,使用 SPIN 来验证系统设计模型是否满足某些关键性质需求;e)基于 Petri 网的转换[9~12],将状态机图映射为高级 Petri 网,利用 Petri 网的理论知识对所得到的网模型进行分析和验证。

面对如此多的形式化方法,经过分析其特点之后,选择有色 Petri 网作为描述执行模型的语言。有色 Petri 网(CPN)[13]是由丹麦的 Jensen 于 1981 年在 Petri 网[14]基础上定义的一种具有层次性的高级 Petri 网。它有机地结合了数据结构和层次分解,能同时用于验证系统功能和评估系统的性能,并支持自动或交互仿真,同时也是一种形式化、数学化的建模语言,具有良好定义的语法和语义,能自然地描述并发、同步、资源冲突等系统特性,可以分析系统的有界性、活性、可达性、冲突、死锁等结构性质。近年来,Petri 网和 CPN 在 UML 和 SysML 模型的形式化建模与分析方面开始得到较多研究,涵盖了一般 Petri 网、时间 Petri 网、面向对象 Petri 网和有色 Petri 网等多种类型。

对于序列图的 Petri 网的转换,国内外不同学者进行了研究。Jeng 等人[15]提出了一种把扩展 UML 序列图转换为库所/变迁网的方法。扩展的 UML 序列图增加了 UML 的描述能力,可以描述同步、冲突、选择等,但是该方法没有进行模型性质的形式化验证。Cunha 等人[16]在 Jeng 等人工作的基础上,通过模型检测器 SMV 验证了转换后模型的无死锁性、可达性、安全性和活性等性质,但是该验证方法涉及线性时态逻辑 CTL,增加了验证的难度。Ameedeen 等人[17,18]提出了一种把序列图转换为自由选择 Petri 网的规则。它首先把序列图分成几个片段,每个片段转换为 Petri 网的一个部分;然后把这些部分组合成一个完整的 Petri 网模型,本文中采用了相似的方法。但是,有一些片段的映射在文献[18,19]中没有提到,如描述消息之间相互关系的映射。Li 等人[19]提出了一种把序列图转换到对象 Petri 网的方法,但是它只关注组合片段的几个操作;Fernandes 等人[20]提出了一种把 UML 用例图和序列图描述的规范转换为一个有色 Petri 网模型的方法,转换后的模型可以使用成熟的 Petri 网工具分析,但是它没有提供一些重要元素的映射方法,如状态不变量。以上所有的工作并没有关注序列图上的时间信息,并且所有的目标 Petri 网不支持描述时间信息。Andrade 等人[21]提出了一种把序列图转换为时间 Petri 网的方法,但是异步消息的转换规则不可用,且没有涉及与数据有关的操作。Yao 等人[22]提出了把 UML 序列图转换到扩展有色 Petri 网(extended colored Petri net,ECPN)的方法,ECPN 模型形式化地描述了每个对象的变迁及其之间的交互。Eshuis[23]提出了一个两种层次的转换方法,可以把活动图转换为模型检测器 SMV 的输入语言。第一种转换为需求级转换,假设和环境可以完美地同步,对于输入可以很快地响应,该转换定义了一个状态机,从而可以进行有效的验证;第二种转换为应用级转换,该转换定义状态机,但是不能进行有效的验证。通过使用输入队列,该方法更加实际。本文的方法不是状态机,而是序列图。Amorim 等人在文献[24]中提出了一种方法,该方法把 LSC(live sequence charts)作为规范语言,LSC 定义了活性(如必须发生的事件),因此该方法可以区分必须发生的行为和可能发生的行为。由 LSC 语义开始,他们将时间 Petri 网用于模型的形式化。在本方法中,可以验证活性、安全性和可达性。

在国内,叶丽君[25]提出了序列图至一般 Petri 网模型的映射算法,针对消息传递和高级控制结构(可选结构、条件结构、并行结构以及循环结构)分别设计了映射算法,并验证了算法的正确性,有效解决了校验结果回溯分析的技术难题。本文采用的是有色 Petri 网。孔莹莹等人[26]提出了一种 UML 的形式化建模方法,对 UML2.0 序列图中 opt 等操作符给出了对应 CPN 图形的转换规则,实现了用 CPN 模型描述 UML 2.0 的用例图与序列图的目的。但是其并没有对转换后模型的库所变迁含义进行说明,也没有进行正确性验证和对转换后的有色 Petri 网模型进行形式化验证。但是,这些研究为应用有色 Petri 网对 SysML 构建的模型进行分析和验证提供了重要基础。

2 动态行为的可执行验证过程

将序列图转换为有色 Petri 网之后,就可以利用有色 Petri 网的性质对动态行为实施可执行验证。

在校验动态行为的正确性时,首先根据校验的目的选择模型所需满足的性质,只有当模型表现出一组所需要的性质时,才认为模型是正确的。一旦将这些性质放到模拟的系统的上下文中来解释,就可以使校验人员识别出该模型具有该应用领域特有的那些功能,以及还缺少哪些功能。其次,选择 Petri 网的分析方法验证模型是否满足这些性质。Petri 网支持多种不同开销和精度的验证技术,如模拟法、结构分析、列举法等。模拟法通过在所建立的网系统模型上授权变迁发生,模拟 Petri 网的运行,它能够揭示缺陷,并为最终用户提供重要的信息反馈,因此它支持对定义的验证。结构分析用于研究网系统的行为特征和结构之间的关系,依赖于状态变化方程,与 Petri 网的拓扑有关,主要分析与起始标志无关的特性。列举法是基于可达图验证,该方法用于描述 Petri 网的可达标志集,它既与 Petri 网的结构有关,也与 Petri 网的初始标志有关。通过分析 Petri 网的可达图,可了解 Petri 网的许多重要性质,如有界性、安全性、守恒性、可达性、覆盖性、死锁和活性等。动态行为的可执行验证过程如图 1 所示。

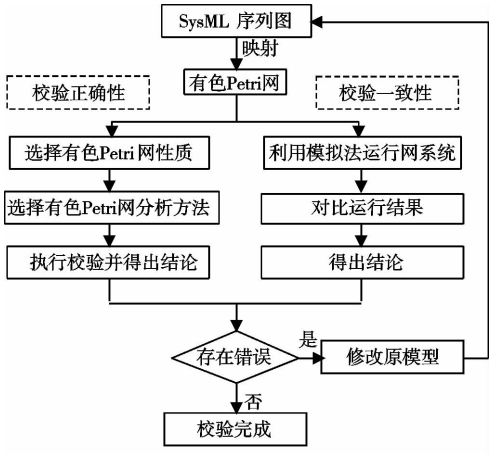


图1 动态行为的可执行验证过程

在校验动态行为的一致性时,通过模拟法对比序列图所对应的有色 Petri 网模型在功能、行为上的表现是否一致,通过列举法解决模拟法不能保证模型不会出现期望之外的行为问题。

经过校验之后,如果存在问题,通过回溯原模型找出其问题所在并进行修改,然后再对修改后的序列图重新建立有色 Petri 网可执行模型,对其实施校验,直到不再存在问题为止。

3　SysML 序列图到有色 Petri 网的转换规则

在 SysML 行为建模机制中,序列图可以用来描述对象之间消息传送的时间顺序,并根据控制流指定一连串的交互,控制流是通过对象生命线之间发送和接收消息定义的。在 SysML 中,时序图由对象(object)、生命线(lifeline)、消息(message)以及条件、循环、并行等条件逻辑组成。SysML 序列图重用了 UML 序列图。

序列图将交互关系表示为一个二维图,纵向是时间轴,时间沿竖线向下延伸;横向轴代表了块定义图中各个独立的对象,这些对象的活动用生命线表示。当对象存在时,生命线用一条纵向虚线表示;当对象的过程处于激活状态时,生命线是一条双道线。通常,把控制操作符表示为时序图上的一个矩形区域,其左上角有一个写在小五边形内的文字标签,用来表明控制操作符的类型。操作符作用于穿过它的生命线,这是操作符的主体。如果一条生命线并不在某个控制操作符的覆盖范围之内,那么这条生命线可能在操作符的顶部中断,然后在其底部重新开始。

常用的控制类型有:顺序执行,标签为 seq;条件执行,标签为 alt;可选执行,标签为 opt;并行执行,标签为 par; 循环(迭代),标签为 loop。将以上各种结构化控制操作符下的结构分别称之为可选结构、条件结构、并行结构以及循环结构。

基于有色 Petri 网的 SysML 序列图的验证,关键就是 SysML 序列图到有色 Petri 网的转换过程,只有完成这个转换过程,得到序列图的有色 Petri 网模型后,才能对模型进行分析和验证。

本章针对 SysML 序列图中一些常用操作,如可选结构、条件结构、顺序结构、并行结构以及循环结构,首先解释了操作本身的含义,然后使用非形式化的方式描述了把时序图的常用操作转换为等价的有色 Petri 网的转换规则,重点是把 UML 序列图的元素与有色 Petri 网元素之间建立一个明确的映射,这当中既包含结构元素,如库所、变迁、输入输出弧,又包含逻辑元素,如全局声明中的颜色集和变量、颜色集与库所、弧表达式以及识别函数之间的关联关系,以及转换后的有色 Petri 网的初始标志,并通过分析映射前后两种模型的语义,验证了映射的正确性。

序列图在转换为有色 Petri 网时,除了要刻画消息传递的逻辑关系,还要体现消息的一些属性。由于消息传递的复杂性,下面说明如何在有色 Petri 网模型中定义库所的颜色集、弧表达式和变迁的守卫函数。

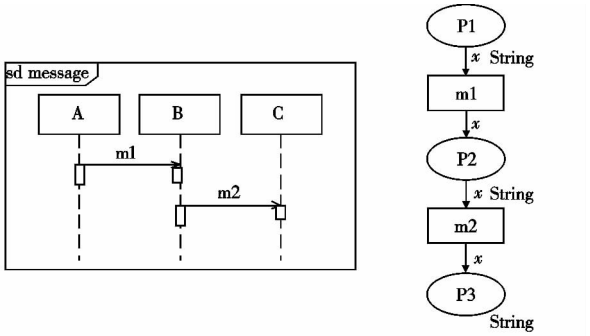
a)颜色集。颜色集表示数据类型。如果操作的调用没有参数也没有返回值,则进入库所的颜色集可定义为统一的类型。由于传递的是消息,所以将颜色集定义为字符串 string 型,然后以字符串的形式说明操作实现的功能。如果操作中有条件判断,则操作的进入库所的颜色集定义为 Bool 型;如果传入多个参数,则定义为复合颜色集。返回库所的颜色集则根据返回值的类型确定。通常将计数器的颜色集定义为 Int 类型。

b)弧表达式。弧表达式表明在弧上传递参数的数据类型以及传递方式。通常根据其输入库所的颜色集与输出库所的颜色集来注释弧表达式。

c)守卫函数。变迁的守卫函数是 CPN ML 语言的布尔表达式,由它来评价真或假。在一个守卫函数加进去之前,注释的默认文本是[]。根据输入弧注释(arc inscription)变量(variables)的值,守卫函数也可以限制输出弧注释变量的值。

规则 1　消息传递

图 2(a)中的序列图表示消息的简单交互。有三条生命线,其中有两条消息 m1 和 m2。转换后的 CPN 模型如图 2(b)所示。其中库所定义为字符串类型,弧注释 x 为字符串型变量。序列图中的每个消息的传递转换为一个变迁,也即消息的执行通过 CPN 模型中对应变迁的发生来表示。其中库所表示消息执行时对应消息的改变。序列图中的守卫条件通过变迁的守卫条件来表示。如果在一条生命线的相同点上有多个消息,使用单独的变迁表示所有的消息。

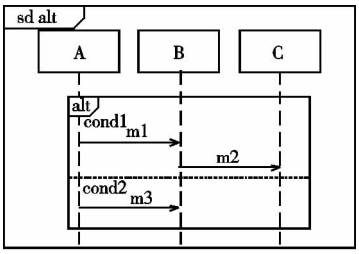


(a)序列图中简单的消息传递　　(b)转换后的CPN模型

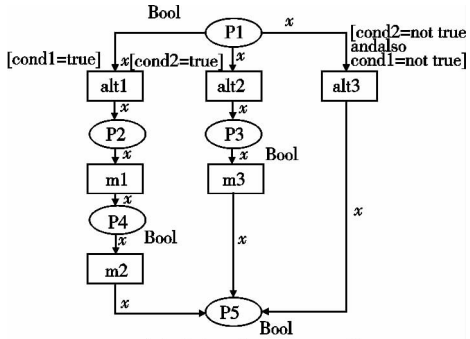
图2　消息传递及CPN模型

规则 2　条件执行

标签为 alt。控制操作符的主体用水平虚线分割成几个分区,每个分区表示一个条件分支并有一个守卫条件。如果一个分区的守卫条件为真,就执行这个分区,但是最多只能执行一个分区;如果有多于一个守卫条件为真,那么选择哪个分区是不确定的,而且每次执行的选择可能不同。如果所有的守卫条件都不为真,那么控制将跨过这个控制操作符而继续执行。其中的一个分区可以用特殊的守卫条件[else],如果其他所有区域的守卫条件都为假,那么执行该分区。



(a)序列图中的条件执行



(b)条件执行alt转换后的CPN模型

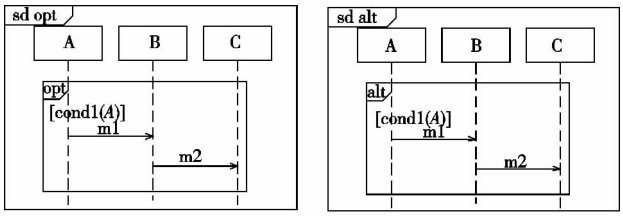
图3　条件执行与CPN模型

图 3(a)中的序列图转换为图 3(b)中的 CPN。其中库所定义为布尔类型,弧注释 x 为布尔型变量。序列图中的每一个操作转换为 CPN 中的一个序列分支。所有的序列分支有一个共同的输入库所 P1,且以共同的输出库所 P5 结束。变迁 alt1 表示条件 cond1 为真时执行左边的分支;变迁 alt2 表示条件

cond2 为真时执行右边的分支;变迁 alt3 表示 cond1 和 cond2 都不为真的情况。由图 3(b)可知,可通过简单地增加分支来表示不同的条件。

规则 3 可选执行

标签为 opt。类似于编程语言中的 if 语句,如果进入操作符时守卫条件成立,那么该控制操作符的主体就会得到执行。守卫条件是一个用方括号括起来的布尔表达式,它可能出现在主体内部任何一条生命线的顶端,它可以引用该对象的属性。可选执行可以看做只有一个操作的条件执行。相似地,应用到条件执行上的转换规则也可以应用到可选执行上。图 4 给出了序列图中的可选执行和选择结构。



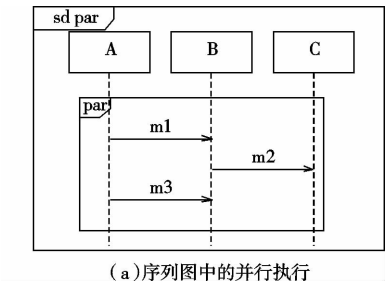
(a)序列图中的可选执行 (b)序列图中的选择结构

图4 可选执行与选择结构

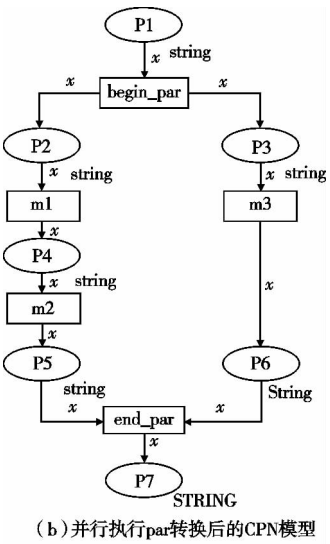
规则 4 并行执行

标签为 par。操作符 par 表示在这个片段的交互是并行执行的,可能包含两个或更多并发执行的子片段或调用。用水平虚线把序列图中的控制操作符的主体分割为几个分区,每个分区表示一个并行(并发)计算。通常情况下,不同分区覆盖不同的生命线。当进入控制操作符时并发地执行所有的分区。每个分区内的消息是顺序执行的,但是并行分区中的消息的相对次序则是任意的。如果不同的计算之间有交互存在,那么就不能用这种操作符。然而,现实世界中大量存在这种可分解为独立、并行活动的情况,因此这是一个很有用的操作符。

图 5 给出了序列图中的并行执行及 CPN 模型。



(a)序列图中的并行执行



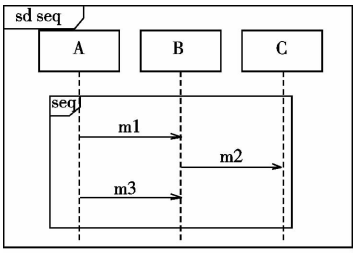
(b)并行执行par转换后的CPN模型

图5 并行执行及CPN模型

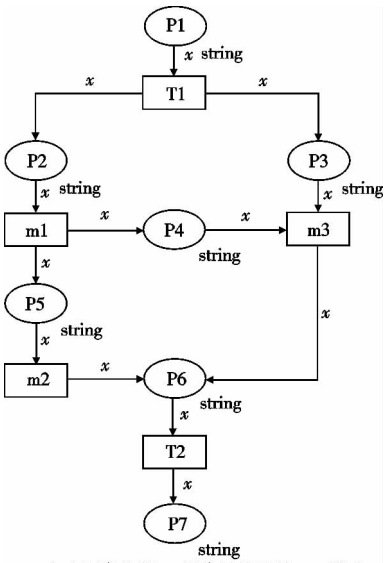
图 5(a)中的序列图转换为图 5(b)的 CPN 模型。其中库所定义为字符串类型,弧注释 x 为字符串型变量。并行执行转换的关键在于如何进入并发,图 5(b)中通过变迁“begin par”产生两个分支,每个分支代表一个操作,每个分支里的第一个库所将随着变迁“begin par”的发生各自有一个托肯。此时,完成了每个分支的变迁的分叉。在所有的分支执行完毕后,变迁“end par”才能发生。这样的分叉与合并结构就可以实现系统的并行处理,而且可以很简单地添加分支来实现多个进程的并行处理。

规则 5 顺序执行

使用“seq”表示操作行为之间顺序执行。在不同生命线上的不同操作可能以任何顺序发生。在相同生命线上的不同操作必须保证第一个操作在第二个操作之前发生。图 6(a)是一个有顺序操作的例子,消息 m1 和 m3 在相同的生命线上,在消息 m1 操作完成后,消息 m3 发生。因此,消息 m1 必须在消息 m3 之前发生。图 6(b)为图 6(a)转换的 CPN 模型。其中库所定义为字符串类型,弧注释 x 为字符串型变量。图 6(b)和图 5(b)相比,在变迁 m1 和 m3 添加了一个库所。这样,变迁 m3 要想发生,根据变迁发生规则,其前继库所里必须有托肯,这样变迁 m1 必须先发生,然后变迁 m3 才能发生。



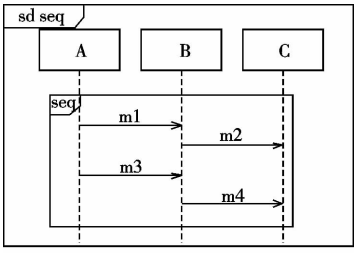
(a)序列图中的顺序执行



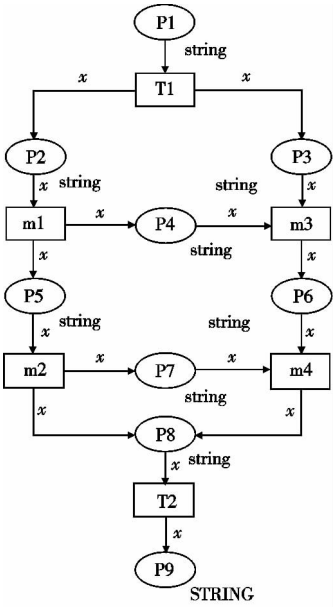
(b)顺序执行seq操作转换后的CPN模型

图6 顺序执行及CPN模型

图 7(a)是另外一个 seq 的示例;对应的 CPN 模型如图 7(b)所示。



(a)序列图中的顺序执行



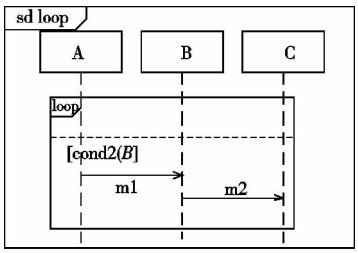
(b) 顺序执行seq操作转换后的CPN模型

图7　顺序执行与CPN模型

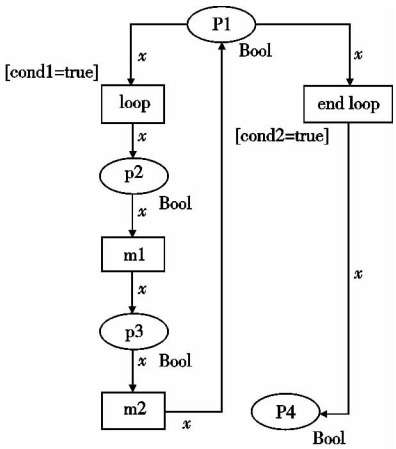
规则 6　循环执行

循环执行标签为 loop。在主体内的某个生命线的顶端给出一个守卫条件。只要在每次迭代之前守卫条件成立,那么循环的主体就会重复地执行。一旦在主体顶部中的守卫条件为假,控制就会跳过该控制操作符。

图 8(b)给出了图 8(a)所示循环操作的 CPN 模型。其中库所定义为布尔类型,弧注释 x 为布尔型变量。变迁"loop"和"end loop"有相同的输入库所。当循环的守卫条件为真时,则变迁"loop"使能;当循环的守卫条件为假时,则变迁"end loop"使能,其输出库所表示循环结束。



(a)序列图中的循环执行



(b) 循环执行loop操作转换后的CPN模型

图8　循环执行与CPN模型

规则 7　重用

操作符 ref 表示引用在别处定义的一个交互过程,利用 ref 操作符可以通过交互的分裂、重用和收集更方便地设计图形。在顺序图中可以通过引用 ref 交互片断重用在别处已定义好的交互过程,类似编程语言中的抽象与重用。如果用例图中一个用例包含另一个用例来完成自己的行为时,可以在顺序图中使用 ref 来描述这种关系,ref 片断的转换可以由一个表示此用例的"替代变迁"来实现。

4　转换规则正确性验证

本文提出的转换规则,在转换时严格根据消息执行映射,一般时序图模型中所包含的要素(对象、消息、消息的发送与接收)都能在其对应的有色 Petri 网模型中得到体现;而映射之后的有色 Petri 网模型中的库所与变迁在相应的时序图中也有其原型含义;库所与变迁的连接也严格按照时序图中的语义执行,确保了模型映射规则的正确性。基于消息的模型映射有利于验证映射前后语义的一致性以及回溯分析映射前后的两种模型,同时,使用该方法进行映射具有可扩展的优点。例如,针对带有条件限制的消息,可在表示发送该消息变迁的守卫函数中加入一个条件,当条件成立时,该变迁才可执行;针对考虑时间因素的时序图,可在对应的变迁上加上时间区间,就可以转换为时间有色 Petri 网。

目前,要判断 Petri 网模型是否与原模型保持一致,只能依赖于人工的方法对两种模型的元素进行一一比较[27]。下面以 PKI 系统中证书更新的序列图为例,说明上述映射中并行结构转换规则的正确性。

4.1　数字证书更新序列图

公钥基础设施(PKI)为大规模、分布式开放网络环境下的信息安全问题提供了行之有效的解决办法,PKI 的核心技术就是围绕数字证书的申请、颁发、使用、更新和撤销等整个生命周期展开的[28]。因此,证书申请、撤销、更新和证书查询这四个用例是 PKI 系统的主要用例,其中证书申请、撤销、更新是系统的核心用例。图 9 使用序列图对证书更新进行建模,可以清晰地描述终端实体、注册机构、认证机构和证书及 CRL 库之间的交互,其描述的交互过程如下:

a)终端实体生成证书更新请求,向注册机构请求更新。

b)注册机构接受证书更新请求,进行审核,向认证机构转发证书更新请求,请求更新。

c)认证机构签发新的用户证书,向资料库发布证书,并同时发布证书撤销列表。

d)如果证书更新成功,认证机构向注册机构返回用户证书,否则返回错误信息。

e)如果证书更新成功,注册机构向终端实体返回用户证书,否则返回错误信息。

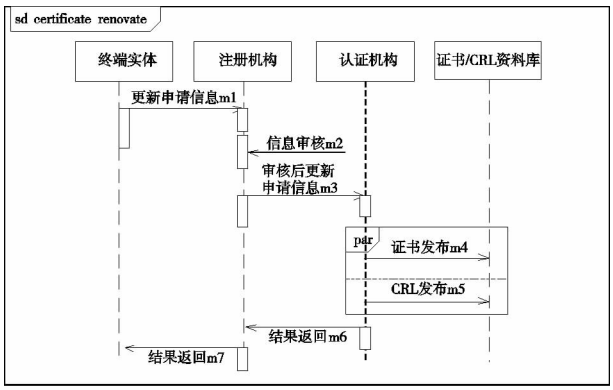


图9　证书更新序列图

4.2　Petri 网模型的转换

首先定义库所的颜色集和弧表达式:

colset STRING = string;

var n: STRING;

然后根据前面定义的转换规则,把证书更新序列图转换为有色 Petri 网模型,如图 10 所示。

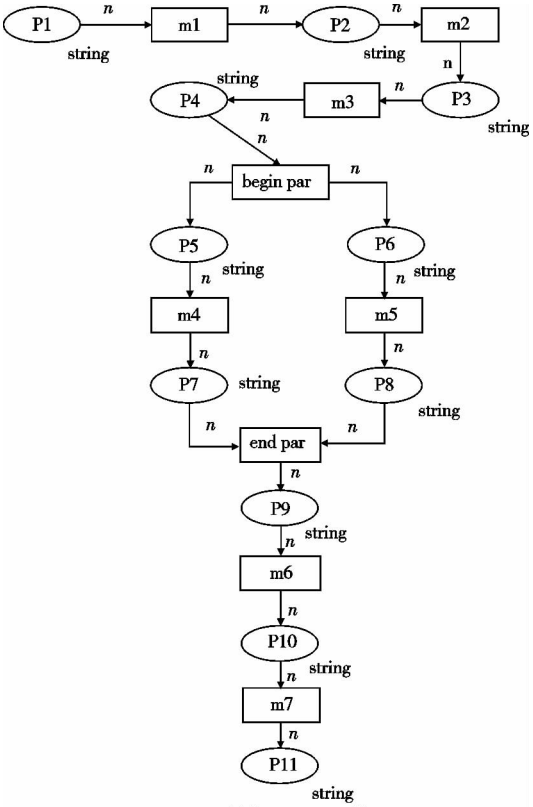


图10 转换后的CPN模型

其中,库所 P1 为更新申请信息的初始状态;P2 为待审核的更新信息;P3 为审核后的更新信息;P4 为待更新的审核信息;P5 为证书发布前的证书状态;P6 为 CRL 发布区的 CRL 状态;P7 为证书;P8 为证书;P9 为证书;P10 为返回的用户证书;P11 为返回用户证书。

变迁 m1 为发送证书更新请求;m2 为更新信息审核;m3 为发送审核后的更新信息;begin par 为进入并行结构;m4 为证书发布;m5 为 CRL 发布;m6 为认证机构向注册机构返回用户证书;m7 为注册机构向终端实体返回用户证书。

4.3 CPN 模型分析

本例中,研究对象是数字证书更新,主要考虑模型可达的状态、是否存在死锁、模型能否结束、系统是否有界等问题。因此,这里主要分析 Petri 网的可达性、无死锁性、活性、和有界性。

本例中,假定有色 Petri 网的库所 P1 里有一个托肯,然后使用通用有色 Petri 网仿真工具 CPN Tools[29] 构建 CPN 模型。当模型建立成功且没有语义错误后,就可以进入状态空间工具了,然后依次运行“计算状态空间”和“计算强连接部件图”工具,运行成功,说明已经产生了证书生命周期模型的状态空间。在产生状态空间后,通过“状态空间报告工具”,可以产生一个状态空间报告文件,下面列出了报告的部分内容:

状态空间报告
Statistics
state space
nodes:11
arcs: 11
secs: 0
status:full
boundedness properties
best Integer bounds upper lower
A′P10 1 1 0
A′P1 1 1 0
A′P111 1 0
A′P2 1 1 0
A′P3 1 1 0
A′P4 1 1 0
A′P5 1 1 0
A′P6 1 1 0
A′P7 1 1 0
A′P8 1 1 0
A′P9 1 1 0
home markings:[11]
liveness properties
dead markings:[11]
dead transition instances:none
live transition instances:none

从状态空间报告可以看出,CPN 的状态空间共有 11 个节点,11 条有向弧。由 Best integer bounds 可知每个库所可有的最大和最小托肯数,最大的数量即为库所的界,可看到最大的界为 1,所以该 CPN 是有界的。家态是指从所有可达标志都能够到达的标志,由 home markings 为标志 11 可知,模型家态为标志 11,这表明证书更新最终能够结束。由 dead transition instances 为标志 11 可知,网中的变迁运行到标志 11 后,运行结束。从报告中可以看出,在标志 11 更新结束。死标志是指没有发生变迁的标志,由 dead markings 为 none 可知,所有的变迁都可以发生,都有潜在的发生权。

在产生状态空间后,通过“显示指定节点工具”“显示前继节点工具”和“显示后继节点工具”,可以得到 CPN 模型的状态空间如图 11 所示。

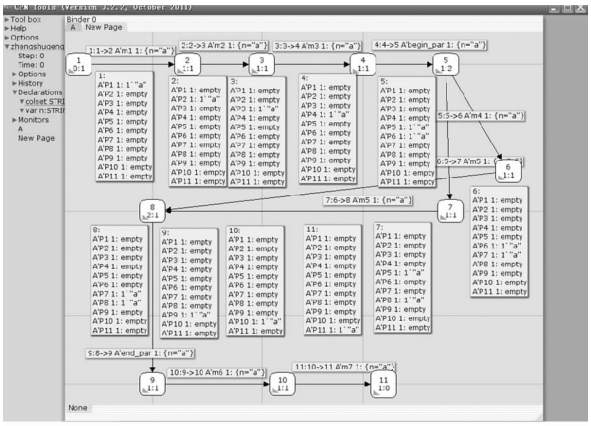


图11 可达标志图

可达图给出了由初始标志开始的所有可达的状态,因此可以保证其可达性。在图 11 中,大矩形依次表示标志 M1 到 M11,每个标志都代表了证书更新时各对象所处的状态;小矩形表示引起标志变化的变迁和该变迁发生时绑定的变量。在大矩形里的 $A'P_i(i=1\cdots11)$ 表示 A 页面的库所 P1…P11;empty 表示库所分量为 0;有限变迁序列 $M_1T_1M_2T_2\cdots T_{11}M_{11}$ 表示证书更新的整个过程。

通过回溯分析可知,各对象的每一个功能也都能在可达图中找到其对象的状态。因此可以说明模型能够正确体现各对象的功能。从图 11 中也可以看出,任意库所中的托肯数是有限的,因而是有界的,同时也是安全的。对于图中的任意变迁 t,对任何可达标志 $M\in[M0>$,总有从 M 可以达到的标志 $M'\in[M0>$,使得 $M'[t>$,因此,可以说该模型是活的,肯定不存

在死锁。这可以保证概念模型中各对象的交互不存在死锁。随着模型的运行,模型最终可以到达一个稳定的状态。模型的可达性、有界性和活性与状态空间报告得到的结果一致。

综合以上分析,模型正确体现了各对象的功能以及对象之间的交互关系,模型能够按照要求实现正确传输,而且有色 Petri 网模型与 SysML 模型之间也保持了动态上的一致性,模型中也没有出现不需要的对象或者不希望出现的状态,保证了证书更新序列图模型的完整性、正确性、一致性。

5　结束语

系统建模语言 SysML 可用于由软硬件、数据和人综合而成的复杂系统的集成体系结构说明、分析和设计,具有可视化的特性,且容易理解、易于表达;Petri 网是一种图形化、数学化的形式化建模语言,具有多种定性和定量分析方法,便于对模型进行仿真与分析。本文给出了 SysML 序列图到等价有色 Petri 网模型的转换规则,可以将 SysML 序列图转换为有色 Petri 网模型,并利用有色 Petri 网的分析技术对转换后的模型进行仿真与分析。此外,此方法具有可扩展的优点。例如,针对带有条件限制的消息,可在表示发送该消息的变迁的守卫函数中加入一个条件,当条件成立时,该变迁才可执行;针对考虑时间因素的时序图,可在对应的变迁上加上时间区间,就可以转换为时间有色 Petri 网。

本文的工作只是通过实例应用验证了转换规则。下一步研究如何使用基于 SysML 的工具绘制序列图,并通过自动转换规则得到其 CPN 模型,还要提出一些规则对 CPN 模型进行化简,以便于分析,避免状态空间爆炸问题。

参考文献:

[1] OMG. SysML version 1.2 [EB/OL]. (2010-06-01). http://www.omg.org/spec/SysML/1.2.

[2] 李桂,苏一丹. UML 状态图的形式化[J]. 广西大学学报,2003,28(4):318-321.

[3] SHROFF M, FRANCE R B. Towards a formalization of UML class structures in Z [C]//Proc of the 21st Annual International Computer Software and Applications Conference. Washington DC: IEEE Computer Society, 1997:646-651.

[4] 赖志明,尤晋元. 从 UML 状态图到 PVS 规范的自动转换、验证[J]. 电子学报,2002,30(S1):2122-2125.

[5] 罗蜜,张为群. 结合形式化方法的 UML 系统开发[J]. 西安师范大学学报,2003,28(2):203-208.

[6] 董威,王戟,齐志昌. UML state charts 的模型检验方法[J]. 软件学报,2003,14(4):750-756.

[7] 蒋慧,吴礼发,陈卫卫. UML programming guide 设计核心技术[M]. 北京:北京希望电子出版社,2001.

[8] 王璐珍,董威,陈火旺. UML 顺序图的自动验证[J]. 计算机工程与应用,2003,39(29):80-83.

[9] BOCCALATTE A, GIGLIO D, PAOLUCCI M. A CASE tool for information system project and development [C]//Proc IEEE International Conference on Systems, Man, and Cybernetice. 1999:1042-1047.

[10] BONDAVALLI A, MAJZIK I, MURA I. Automatic dependability analysis for supporting design decisions in UML [C]//Proc of the 4th IEEE International Symposium on High-Assurance Systems Engineering. 1999:64-71.

[11] BORDBAR B, GIACOMINI L, HOLDING D J. UML and Petri nets for design and analysis of distributed system [C]//Proc of IEEE International Conference on Contrel Applications. 2000:610-615.

[12] SAIDHANA J, SHATZ S M. UML diagrams to object Petri net models: an approach for modeling and analysis [C]//Proc of International Conference on Software Engineering and Knowledge Engineering. 2000:103-110.

[13] JENSEN K, KRISTENSEN L M, WELLS L. Colored Petri nets and CPN tools for modeling and validation of concurrent systems [J]. International Journal on Software Tools for Technology Transfer, 2007, 9(3):213-254.

[14] MURATA T. Petri nets: properties, analysis and applications[J]. Proceedings of the IEEE, 1989, 77(4):541-580.

[15] JENG M D, LU Wei-zhao. Extension of UML and its conversion to Petri nets for semiconductor manufacturing modeling[C]//Proc of IEEE International Conference on Robotics and Automation. 2002: 3175-3180.

[16] CUNHA E, CUSTODIO M, ROCHA H, *et al*. Formal verification of UML sequence diagrams in the embedded systems context[C]//Proc of Brazilian Symposium on Computing System Engineering. 2011:39-45.

[17] AMEEDEEN M A, BORDBAR B. A model driven approach to represent sequence diagrams as free choice Petri nets [C]//Proc of the 12th International IEEE Enterprise Distributed Object Computing Conference. Washington DC: IEEE Computer Society, 2008:213-221.

[18] AMEEDEEN M A, BORDBAR B, ANANE R, *et al*. A model driven approach to the analysis of timeliness properties[C]//Lecture Notes in Computer Science, vol 5562. Berlin: Springer, 2009:221-236.

[19] LI Guang-yu, YAO Shu-zhen. Research on mapping algorithm of UML sequence diagrams to object Petri nets[C]//Proc of WRI Global Congress on Intelligent Systems. Washington DC: IEEE Computer Society, 2009:285-289.

[20] FERNANDES J M, TJELL S, JORGENSEN J B, *et al*. Designing tool support for translating use cases and UML 2.0 sequence diagrams into a colored Petri net[C]//Proc of the 6th International Workshop on Scenarios and State Machines. Washington DC: IEEE Computer Society, 2007:2-11.

[21] ANDRADE E, MACIEL P, CALLOU G, *et al*. Mapping UML sequence diagram to time Petri net for requirement validation of embedded real-time systems with energy constraints[C]//Proc of ACM Symposium on Applied Computing. New York: ACM Press, 2009:377-381.

[22] YAO Shu-zhen, SHATZ S M. Consistency checking of UML dynamic models based on Petri net techniques[C]//Proc of the 15th International Conference on Computing. 2006:289-297.

[23] ESHUIS B. Symbolic model checking of UML activity diagrams[J]. ACM Trans on Software Engineering Methodology, 2006, 15(1):1-38.

[24] AMORIM L, BARRETO R, MACIEL P, *et al*. A methodology for software synthesis of embedded real-time systems based on TPN and LSC [C]//Lecture Notes in Computer Science, vol 3820. Berlin: Springer, 2005:50-62.

[25] 叶丽君. 基于 UML 描述的概念模型校验技术研究[D]. 郑州:解放军信息工程大学,2008.

[26] 孔莹莹,蒲海涛,隋瑞升. 基于 CPN 的 UML 2.0 形式化建模[J]. 青岛大学学报:工程技术版,2011,26(1):33-37.

[27] PETTIT Ⅳ R G, GOMAA H. Modeling behavioral patterns of concurrent software architectures using Petri nets [C]//Proc of the 4th Working IEEE/IFIP Conference on Software Architecture. Washington DC: IEEE Computer Society, 2004:57-66.

[28] 荆继武,林璟锵,冯登国. PKI 技术[M]. 北京:科学出版社, 2008:103-132.

[29] CPN Group. CPN Tools[EB/OL]. (2012-06). http://cpntools.org.