

基于 Token 选取的 OpenStack 单一平面网络建模方法

李 华^{1,2} 邢 熠¹ 张玉荣¹

(内蒙古大学计算机学院 呼和浩特 010021)¹ (内蒙古大学网络信息中心 呼和浩特 010021)²

摘 要 CPN 形式化建模适合为包含大量并发、通信、同步共享行为的软硬件系统建立形式模型,并完成系统功能和性能等方面的行为分析。在传统的 CPN 建模中,token 的选取采用穷举法,由此造成生成的 token 数量较多、CPN 模型生成的状态空间相当庞大甚至状态空间爆炸等问题。针对上述问题,提出将符号执行与 CPN 建模相结合,并在 CPN 模型的执行过程中采用一种基于 token 选取的方法,进而得到 CPN 模型的状态可达图。通过对 OpenStack 云平台支持创建的单一平面网络进行 CPN 建模,针对传统方法和所提方法生成的状态空间,分析了其规模的变化,验证了所提方法的有效性。

关键词 CPN, 符号执行, token 选择, OpenStack, 单一平面网络

中图分类号 TP393.1 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.11.012

Modeling OpenStack Single Plane Network Based on Token Selection

LI Hua^{1,2} XING Yi¹ ZHANG Yu-rong¹

(School of Computer Science, Inner Mongolia University, Hohhot 010021, China)¹

(Network & Information Center, Inner Mongolia University, Hohhot 010021, China)²

Abstract CPN suits for modelling the hardware and the software systems, which contain a large number behaviors of concurrencies, communication, synchronous sharing, and analyzing system function and performance. In CPN modelling, the traditional exhaustive method of selecting token is adopted, and the large number of token could cause that the generated state space is quite huge so as to bring reachability state explosion. On account of the above problem, this paper combined symbolic execution with CPN modelling, used selection method during the execution of the CPN model, and then obtained the reachability state space of the CPN model. Furthermore, a single plane network created by an OpenStack cloud platform was used as an example to be modeled, and the state space and reachability state generated by the traditional method and the new method were compared. The results show that the proposed method is effective.

Keywords CPN, Symbolic execution, Token selection, OpenStack, Single plane network

1 介绍

Petri 网^[1]被广泛应用于很多领域,是一种适合于分布式、并发、异步系统建模与分析的形式化方法^[2]。Coloured Petri Nets(CPN)^[3]是 Petri 网的一种高级形式,被广泛应用于很多领域,是一种适合于分布式、并发、异步系统建模与分析的形式化方法,可以为各种软件系统进行建模。CPN tools^[4]能动态运行所建模型,生成状态空间,并验证模型的活性、可达性及安全性等。相比 Petri 网,CPN 在协议建模的描述中有其自身的优势^[5]:

(1)增加了颜色集,使得 CPN 具有更强的描述能力。

(2)在 CPN 中,建模引入了层次的概念,使得建模工作可以详疏得当,更容易理解。

(3)有 CPN tools 的支持,在建模之后至少可以利用工具对模型的活性、可达性等性质进行自动检验,自动生成可达图。

上述优点极大地方便了基于 CPN 的建模、测试与验证工作。

在 CPN 的建模中,token 扮演很重要的角色,在 CPN 的执行过程及可达图的建立过程中起着重要的作用。但是目前对这方面的研究较少,使得基于 CPN 建模时对 token 的取值是一个困惑建模者的问题。

OpenStack^[6]是由 Rackspace 公司和 NASA(美国航空航天局)合作研发的一款开源云计算平台,并于 2010 年贡献到开源社区,它是一个完全开放源代码的项目,任何企业及个人都可以参与到该项目的开发和测试中,并贡献代码。目前,最新的 OpenStack 版本是 Juno,它是 OpenStack 项目创立以来的第 10 个版本,并且它支持创建的网络结构有 3 种,分别是单一平面网络结构、多平面网络结构和混合平面私有网络结构^[7]。

本文对 token 的选取和执行进行了深入讨论,并对 OpenStack 进行了 CPN 建模研究,利用所给出的方法控制了 OpenStack 单一平面网络的状态空间的规模,减少了因状态空间爆炸带来的问题。

本文第 2 节分析了研究现状,介绍了当前关于 CPN 形式

收稿日期:2015-07-01 返修日期:2016-02-24 本文受国家自然科学基金项目(61163011,61262082)资助。

李 华(1964—),女,博士,教授,主要研究领域为网络与测试、形式化方法,E-mail: cslhwa@imu.edu.cn;邢 熠(1971—),男,博士,讲师,主要研究领域为形式化建模、软件测试;张玉荣(1988—),女,硕士生,主要研究领域为软件测试。

化建模中状态图的规模问题及符号执行的相关研究工作;第3节分析了 token 的取值方法,实现了一种将符号执行和 CPN 建模相结合的 token 选取方法,并对其进行分析;第4节将 Openstack 构建的单一平面网络作为实例,对其进行传统 CPN 建模和基于 token 选取的 CPN 建模,并对结果进行分析,验证了本文提出方法的有效性;最后总结全文。

2 研究现状

CPN 是一种对并发、通信和同步等系统建模和验证的一种图形化语言;同时,CPN 也是一种将 Petri 网和一种高级编程语言结合的离散事件的建模语言。CPN 模型可由 CPN tools 来创建、编辑、模拟和分析。CPN 建模方法应用广泛,已经用于 Web 应用系统建模^[8]、网络协议建模^[9]、系统建模^[10]和软件建模^[11]等。CPN 模型可以进一步按照需要添加多个颜色集进行相应的扩展,如时间 CPN 模型等^[12]。文献[13]针对状态空间规模较大的协议,将属性模型与协议模型做投影运算来生成测试序列,有效地减少或避免了状态空间爆炸问题。由于移动自组织网络中的响应式路由协议 DYMO (Dynamic MANET On-Demand) 的路由操作复杂,并且网络拓扑动态性改变,使得构建协议规约模型极具挑战性。因此, J. Billington 等采用 CPN 形式化建模方法对路由协议的规约模型进行建模,其中弧上的函数表示 DYMO 的路由算法。最后与直观上的层次建模方法相比,该方法有效地缩减了状态空间^[14]。J. M Fernandes 提出了反应系统的需求工程的建模方法,根据交互图和用例图等来构建一个 CPN 模型,用以验证正在开发的系统的功能需求,并以自动气体泵控制器为例,根据控制器、与控制器交互的物理实体以及用户操作系统构建 CPN 模型^[15]。Jonathan Billington 采用 CPN 建模方法对 CNP(Contract Net Protocol)进行建模,此模型在不影响协议运行的前提下忽略了代理和进程之间具体的通信细节,通过对模型的分析,说明了文中 CPN 建模方法的有效性^[16]。针对操作系统的安全访问策略,文献[17]通过采用 CPN 形式化建模方法对微内核操作系统 Minix3 进行建模、测试及分析,通过有目的地针对系统调用等属性单独建模的方法,有效地控制了 CPN 模型的规模。文献[18]采用 CPN 建模方法对并发软件系统进行形式化建模,提出了一种基于迹等价的模型化简算法,该算法适用于 CPN 模型的状态空间很大的情况,能减小模型的规模。文献[19]使用 CPN 形式化建模方法对软件定义网络的南向接口的核心协议 OpenFlow 进行了建模,通过对欲测属性进行建模,控制了 CPN 模型的规模。

在采用 CPN 进行形式化建模时,token 的选取及其执行时的点火顺序是很重要的步骤,直接关系到形成的可达图的规模,进而影响问题分析的效率。如果 CPN 模型中 token 数量太多,会生成大量的冗余状态,降低测试人员的工作效率;token 数量太少,会导致描述的系统状态不完全。

符号执行的基本思想是用抽象符号的形式来表示程序源代码中变量的值,模拟程序的运行,适用于路径敏感的程序^[20]。符号执行不需要指定具体的输入数据,根据分析程序中变量之间的约束关系得到可执行路径,减少测试路径的冗余度。符号执行最初被应用到源程序的测试及测试数据的自动生成、C 语言实现的密码协议的安全性检验及软件验证方面。近年来,随着半形式化方法和形式化方法的发展,越来越多的研究将符号执行的优势与形式化方法结合起来,通过将

符号执行应用到控制流图、状态图、有限状态机等实例,证明了其可行性。符号执行还被应用到软件验证方面,作为基于 CEGAR(反例指导的抽象精化)的模型检验的一种选择,而符号执行中一些无边界的循环导致产生许多冗余的符号状态,为了提高执行效率,减少对冗余符号状态的执行, Joxan Jaffar 针对此问题提出了一种扩展,此扩展的主要特点是为了避免探索冗余的符号状态而进行抽象化执行,在早期就可以发现一些不可达路径,并将其从符号执行的路径中剔除^[21]。

本文将符号执行和形式化 CPN 结合起来,给出一种 token 的选取方法,并通过以云数据中心的核心软件 OpenStack 单一平面网络建模为例,对提出的方法进行验证。

3 符号执行与 CPN 建模相结合

在 CPN 建模及执行中,若初始运行时 CPN 中的 token 数目太少,则无法保证模型的活性、可达性等基本属性;反之,若初始时的 token 数目过多,会导致状态空间过大而不利于对状态空间的分析。需要考虑 token 的具体取值,同时在模型中,初始出现的 token 的数目对 CPN 模型生成的状态空间大小有着至关重要的作用。若在选取时不进行细化分析,则可能导致运行 CPN tools 时不能完全反应系统的执行过程^[22]。

符号执行在测试过程中是一种减少测试数据的有效方法,常被用于生成高覆盖度、低冗余度的测试用例,因此本文考虑把符号执行应用到 CPN 的建模执行中,获得最小 token 子集。

3.1 Token 取值分析

Petri 网的执行是由它的 token 的数量和分布来控制的。如果某一库所中存在 token,那么它就有可能控制 Petri 网的执行。可达标识表示 Petri 网的每个库所中的 token 的个数及其在 Petri 网上的分布。在 CPN 建模中,由于采用了多类型及多值的 token,使得 CPN 的抽象程度比一般的 Petri 网高,导致它的执行不容易被控制。在 token 具体类型的定义及 CPN 执行过程中,各个 token 的取值及其移出顺序直接关系到状态空间的规模,进而影响到所有基于 CPN 可达图研究问题的复杂性,因此需要深入研究。

3.1.1 问题分析

(1)Token 取值方面的考虑

1)简单颜色集:可以采用等价类加边界法等方法获取 token 的取值,通过 CPN tool 的执行,获取模型的可达图。

2)复合颜色集:首先采用等价类加边界法等方法获取复合颜色集中的简单颜色集变量,然后将复合颜色集中的简单颜色集变量进行组合得到 token 的取值。

(2)多个 token 开始点火时的移出顺序

1)逐个移出:多个 token 的点火顺序是依次的,并且前一个跑完,再点火后一个。

若 token 集中任意的两个 token 是相互独立的,即在 CPN 模型中任意的一个 token 的执行都不会对其它 token 的执行结果产生影响,那么 token 的点火可以逐个进行。

2)随机移出:多个 token 的点火顺序是随机的。

对于大多数的 CPN 模型,其 token 的点火都是采用随机移出的机制,即对 token 之间的关系没有限制。

3.1.2 解决方法

针对 token 的不当取值对 CPN 模型执行带来的问题,给

出将符号执行与 CPN 建模结合的解决方法,该方法的主要思想是采用符号作为 token 的某一字段的取值。例如,在某一协议的 CPN 模型中,token 中存在一个版本字段 type,则引入 F 和 T 分别表示 type 的两种取值。其中 $type=F$ 表示错误的版本; $type=T$ 表示正确的版本。采用这种方式将 token 中所有字段的可能取值使用符号表示,即将各个字段的取值进行了抽象,而不需要对各个字段的所有取值进行穷举(对于某些字段来说,其所有取值是无法通过穷举获得的)。

当 token 中各个字段的可能取值采用符号表示时,这个 token 中各个字段取值的个数也就确定了,因此可选 token 的数目也确定了。下面具体讨论符号执行引入 CPN 模型后,如何确定 token 数目。

3.2 初始时 token 数目的确定

当采用 CPN Tools 工具对系统进行 CPN 建模后,必须明确 token 如何选取,因为 token 的选取直接决定了 CPN 模型分析的困难程度,若 token 个数太少,则无法保证模型的活性、可达性等基本属性;token 个数太多又会使得状态空间变得太大,甚至状态空间爆炸。下面通过概率论中的乘法原理方法来确定 token 的数目。

本文利用“乘法原理”确定 token 的数目,需要分为 n 步完成,假如第一步有 m_1 种方法,第二步有 m_2 种方法, ..., 第 n 步有 m_n 方法,那么完成这件事共有 $N=m_1 * m_2 * \dots * m_n$ 种不同的方法。将其引入到符号代表的 token 中,一个 token 有 n 个字段,第 1 个字段有 m_1 种取值,第 2 个字段有 m_2 种取值, ..., 第 n 个字段有 m_n 种取值,则 token 集的取值有 $N=m_1 * m_2 * \dots * m_n$ 种。

假设用一个多元组 $p=(p_1, p_2, p_3, \dots, p_m)$ 表示一个 token, p_i 表示 token 中某个字段, p_i 可能的取值有 n 个(为简单起见,假设 n 为 p_i 取值的最大个数),记为 $(X_{i1}, X_{i2}, \dots, X_{in})$,根据数据包的字段的个数和可能的取值构造一棵高为 $n+1$ 的状态空间树,如图 1 所示。

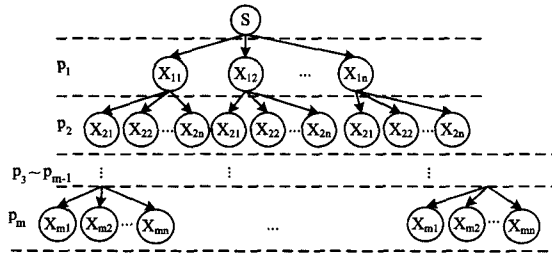


图 1 Token 的状态空间树

从第二层节点到每个叶节点的路径上节点序列的组合称为一个 token 取值,所有 token 取值的集合称为 token 集,即 token 的数目。

采用“乘法原理”所生成的 token 集是所有可能出现的 token 的全集,但是这些 token 可能存在冗余,即两个或多个 token 集中的 token 代表的是同一类型的消息。因此需要对 token 全集进行化简、去冗余,从而使得所选择的 token 集中的每个 token 所代表的都是不同类型的消息,此集合称为“token 最小子集”。

对 token 最小子集的获取采用“分支界限法”。其具体步骤是:假设数据包有 n 个字段 $n_1, n_2, \dots, n_n$,其中 n_1 有 m_{n1} 种取值, n_2 有 m_{n2} 种取值, ..., n_n 有 m_{nn} 种取值,则根据数据包的字段数和每个字段的可能取值构造一棵状态空间树 S ,然后对该状态空间树 S 进行化简,得到 S' ,将 token 的选取过程转

变为在状态空间树 S' 中搜索叶子节点的可达路径。

本文提出的状态空间树化简方法为属性划分法,将 token 包含的各个字段看成是该 token 的各个属性,并将这些属性分为关键属性和非关键属性。其中,关键属性指该属性字段是不可缺省的,并且其正确取值为受约束的特定值;非关键属性指该属性字段是可缺省的,并且其正确的取值是没有被约束的值。当构造状态空间树时,将关键属性置于上层,将非关键属性置于关键属性的下层。

假设有关键属性 A_i ,当其取值为错误值时,不论 A_{i+1} 等其它字段的取值如何,该 token 均代表关键属性 A_i 的值错误;当关键属性 A_i 的取值为正确值时,继续考虑 A_{i+1} 等其它字段的属性。假设有关键属性 B_j ,不论其取值正确与否,对属性 B_j 的每一个取值都需进行考虑。

为了更好地理解上述方法,下面以一个例子来介绍 token 集的选择方式。

假设数据包 B 有 3 个字段 $b1, b2$ 和 $b3$,其中 $b1, b2$ 是关键属性, $b3$ 是非关键属性。并且所有字段的错误取值均用 F 表示,假设 $b1$ 有 3 种取值(L 为低版本, H 为高版本, F 为错误的版本), $b2$ 有两种取值(T 为正确取值, F 为错误取值), $b3$ 有两种取值(T 为正确值, F 为错误值),其 token 子集的求解过程如下。

(1) 构建状态空间树 S ,如图 2(a)所示。

(2) 通过使用本文提出的属性划分法对状态空间树 S 进行化简得到 S' ,搜索 S' ,求得其叶子节点的可达路径。

对图 2(a)进行分析可知, $b1$ 为关键属性,当 $b1$ 取 F 字段时,无论 $b2$ 和 $b3$ 取值何值,此数据包所代表的含义均是 $b1$ 字段错误,因此可以用一个节点 T 来代替 $b2$ 字段的 T 和 F 取值。对状态空间树 S 采用自上而下、自左向右的方式合并节点,得到化简后的状态空间树 S' (见图 2(b)),算法描述如图 3 所示。

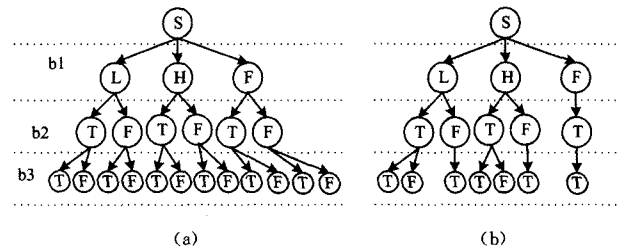


图 2 状态空间图

simplificationTree (S) { // 对状态空间树化简,并输出化简后的 token 集

if (节点为关键属性并且取值为 F) {

记录该节点;

将其下的每一层节点合并,且取值为 T ,同时记录这些节点;

}

if (T 没有子节点) {

记录树的根节点;

输出该序列;

}

记录树的根节点;

simplificationTree (T .child)

}

图 3 状态空间树化简算法

(3) 对 S' 的叶子进行搜索,得到从根节点的下层节点到叶子节点的路径序列。

采用状态空间树化简算法,得到化简后的状态空间树对应的 7 个序列: LTT, LTF, LFT, HTT, HTF, HFT, FTT, 即 token 集中的 7 个 token 元素。

在通过上述方法得到的 token 集中,任意两个 token 之间都是相互独立的,在 CPN 模型中,这些 token 可逐个执行,也可随机执行。本文采用逐个执行的方案。

从上述例子可以看出,关键属性的取值比非关键属性更重要。但是确定为关键属性还是非关键属性依赖于建模人员的具体需求。

4 示例

当对某一系统进行 CPN 建模时,token 的选取采用穷举的方法,会造成 CPN 模型中 token 的数量非常多,使得 CPN 模型生成的状态空间相当庞大,甚至爆炸。针对上述问题,本文提出将符号执行的思想应用于 token 的选取,并给出了具体的应用方法。下面以 CPN 对 OpenStack 云平台支持创建的单一平面网络建模为例,说明本文所提方法的可用性。

单一平面网络模型是 OpenStack 所支持构建的 3 个网络模型中最简单的一个,其结构如图 4 所示。

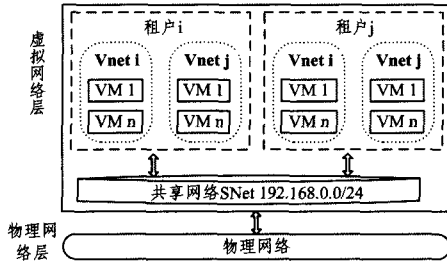


图 4 单一平面网络结构

4.1 单一平面网络模型的 token 选取

由图 4 可知,在单一平面网络中,所有的租户都处于同一个共享网络,因此在单一平面的 CPN 模型中,token 的选取可以不考虑租户是否在同一个共享网络这个约束条件。

在单一平面网络中,虚拟机资源之间的相互访问的行为约束由 4 部分构成,分别为租户(Tenant)、虚拟网络(VN)、网络地址(CN)和共享网络(PuN)。其中,租户、虚拟网络和网络地址这 3 个属性是关键属性,共享网络是非关键属性,这些属性的不同取值组合将构成一个 token 集,如表 1 所列。

表 1 单一平面网络中各行为约束取值的含义

网络行为 约束名	不同的符号取值所代表的含义		关键 属性
Tenant	Y: 相互访问的任意两个资源具有相同租户	N: 相互访问的任意两个资源具有不同租户	是
VN	Y: 相互访问的任意两个资源具有相同虚拟网络	N: 相互访问的任意两个资源具有不同虚拟网络	是
CN	Y: 相互访问的任意两个资源具有相同的网络地址	N: 相互访问的任意两个资源具有不同的网络地址	是
PuN	Y: 租户所在网络是已知共享网络	N: 租户所在网络是未知共享网络	否

由 3.2 节获取 token 集的方法可知,单一平面网络的 token 可取值有 YYYYY, YYYN, YNYN, YNYY 和 NYYYY 5 种。

根据单一平面网络的结构和行为约束,模型中的颜色集如图 5 所示。

▼ colset TENANTXVNXCNXPuNXDES = product TENANT * VN * CN * PuN * DES;

▼ var TE, VN, CN, PuN, DES; STRING;

图 5 单一平面网络行为定义的颜色集

其中, TENANT(租户)表示单一平面网络中,资源间通信的约束关系; VN 表示单一平面网络中,资源间通信的约束属性虚拟网络(VN); CN 表示单一平面网络中,资源间通信的约束属性资源是否在同一个子网中(CN); PuN 表示单一平面网络中,资源间通信的约束属性资源所在子网是共享网络(PuN); DES 表示单一平面网络中,资源间通信的行为描述;例如,当它们不在同一个租户进行资源间通信时,返回的消息 DES 将会是“Error Different Tenant”。

4.2 单一平面网络的 CPN 建模

对单一平面网络进行建模,即将单一平面网络看成黑盒,此时假设单一平面网络已经被 SDN 控制器成功构建,并且所有相关的设备运行都是正常的。采用将符号执行引入 CPN 建模的方法对单一平面网络进行建模,所建模型分别为:顶层模型(见图 6)、单一平面网络子层模型(见图 7)。

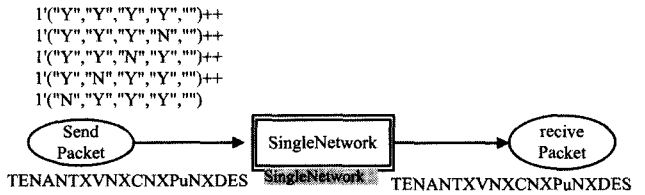


图 6 单一平面网络 Top 层模型

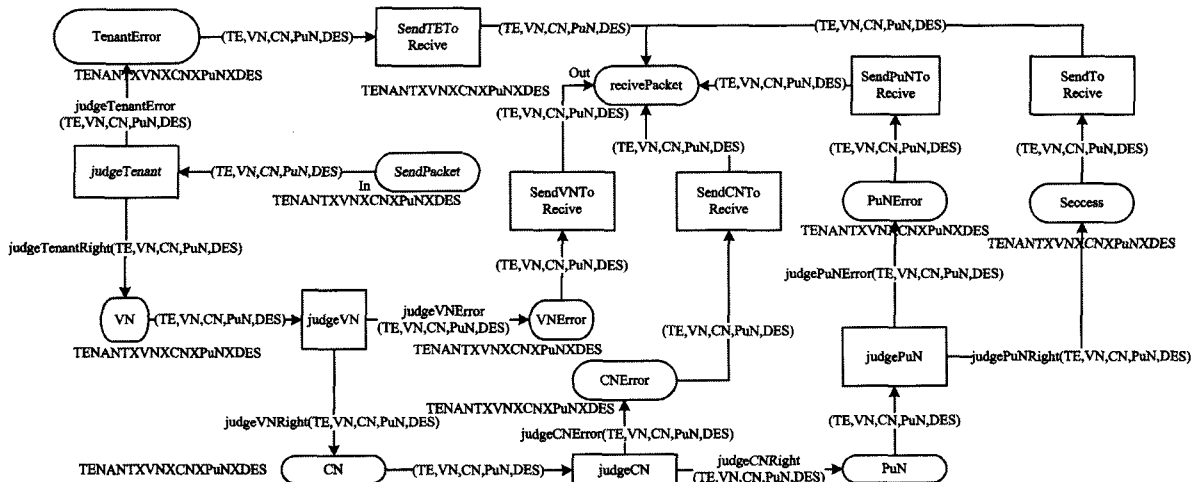


图 7 单一平面网络子层模型

采用 CPN Tools 工具对单一平面网络所建模型生成的状态空间报告如图 8 所示。

Statistics

State Space
Nodes: 2160
Arcs: 8388
Secs: 2
Status: Full
Sec Graph
Nodes: 2160
Arcs: 8388
Secs: 0
Boundedness Properties

图 8 单一平面网络 CPN 模型状态空间报告

从状态空间报告可知,单一平面网络的 CPN 模型的状态空间包含 2160 个状态节点及 8388 条弧。在 CPN 模型中,如果不对 token 的执行加以限制,则其生成的状态空间中将在很多的冗余序列,需要人工筛选这些序列以去除冗余。

通过本文提出的 token 选取方法,生成的 token 可看作是相对独立的,即每一个 token 的执行不会对其它的 token 执行产生干扰。因此,在 CPN 模型中可以通过控制 token 的执行顺序,使得某一时刻只有一类 token 在模型中流动,改进后的单一平面网络顶层模型如图 9 所示。在该模型中,加入了一个 Handle 子层用于控制 token 的执行顺序,如图 10 所示。

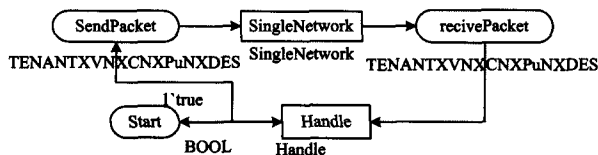


图 9 改进后的单一平面网络 Top 层

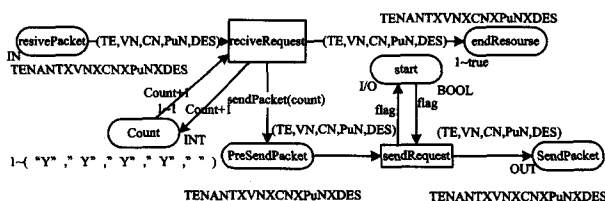


图 10 控制 token 执行的 Handle 子层

采用 CPN Tools 仿真工具对改进后的单一平面网络所建的模型生成状态空间,如图 11 所示。

Statistics

State Space
Nodes: 30
Arcs: 29
Secs: 0
Status: Full
Sec Graph
Nodes: 30
Arcs: 29
Secs: 0
Boundedness Properties

图 11 改进后单一平面网络 CPN 模型生成的状态空间报告

当采用传统的 token 选取方法(即穷举法)时,在单一平面网络的 CPN 模型中,token 的数量有 $2^4 = 16$ 个,采用本文提出的方法时其 token 数量只有 5 个。在 CPN 模型点火时,如果不采用本文提出的点火方法,CPN 模型生成的状态空间将包含 2160 个状态节点和 8388 条弧,而采用本文提出的点火方法时其状态空间中只有 30 个状态节点和 29 条弧。通过上述数量比较不难看出本文提出的方法的优点。

结束语 符号执行在建模中有两种应用:1)在建模过程中,可以使用代数中的抽象符号所代表的变量来控制数据流的走向;2)token 的选取,对于 CPN 模型来说,token 的选取至关重要。若选取的 token 过少,导致生成的状态空间不完整,从而生成的测试用例也不完整;若 token 选取过多,则会生成较多冗余,导致状态空间爆炸问题。本文提出将符号执行与 CPN 建模相结合,给出了一种 token 选取的新方法,并且以 OpenStack 云平台支持创建的单一平面网络建模为例,验证了符号执行和 CPN 建模相结合的可用性和有效性。

在下一步的工作中将进一步研究 token 的随机移出得到的可达图的形态。

参考文献

- [1] Peterson J L. Petri net theory and the modeling of systems[M]. Prentice-Hall PTR, 1981
- [2] Samanta B, Sarkar B. Application of Petri nets for systems modeling and analysis[J]. OPSEARCH, 2012, 49(4): 334-347
- [3] Jensen K. Coloured Petri nets; basic concepts, analysis methods and practical use[M]. Springer Science & Business Media, 1997
- [4] <http://cpntools.org>
- [5] Aalst W M P V D. Petri-net-based workflow management software[C]//Proceedings of the NFS Workshop on Workflow and Process Automation in Information Systems. IEEE Computer Society, 1996: 114-118
- [6] <http://www.openstack.org>
- [7] http://www.ibm.com/developerworks/cn/cloud/library/1402_chenhy_openstacknetwork
- [8] Li Jing-xia, Hou Zi-feng. Application of Colored Petri Net in Model for Web Service Composition[J]. Application Research of Computers, 2006(1): 22-27 (in Chinese)
- [9] 李景霞, 侯紫峰. 基于颜色 Petri 网的 Web 服务组合建模及应用[J]. 计算机应用研究, 2006(1): 22-27
- [10] Ning Liang, Zhang Zhi-hong. Modeling and analysis of SPIN protocol based on colored Petri net[J]. Computer Engineering and Design, 2007, 28(14): 3391-3393 (in Chinese)
- [11] 宁亮, 张志鸿. 基于有色 Petri 网的 SPIN 协议建模与分析[J]. 计算机工程与设计, 2007, 28(14): 3391-3393
- [12] Xiao Bing, Qu Tan. Applying Colored Petri Net in System Modeling and Simulation[J]. Computer Engineering, 2001, 27(1): 30-32 (in Chinese)
- [13] 肖兵, 瞿坦. 着色 Petri 网及其在系统建模与仿真中的应用[J]. 计算机工程, 2001, 27(1): 30-32
- [14] Chen Yong, Hu Xiao-hui, Dang Jian-wu. Research on SPN² Software Modeling Method Based on Stochastic Petri Nets[J]. Computer Engineering, 2010, 36(17): 230-233 (in Chinese)
- [15] 陈永, 胡晓辉, 党建武. 基于随机 Petri 网的 SPN² 软件建模方法研究[J]. 计算机工程, 2010, 36(17): 230-233

- [4] Huang Bin, Gao Cheng-hui, Chen Liang. Partner selection with fuzzy completion time and fuzzy due date in a virtual enterprise [J]. Systems Engineering-Theory & Practice, 2010, 30(6): 1085-1091(in Chinese)
黄彬, 高诚辉, 陈亮. 模糊完工时间和模糊交货期下的虚拟企业伙伴选择[J]. 系统工程理论与实践, 2010, 30(6): 1085-1091
- [5] Long Hao, Wang Hao. Task-resource assignment graph based heuristic for partner selection in virtual enterprises [J]. Application Research of Computers, 2013, 30(12): 3564-3567(in Chinese)
龙浩, 汪浩. 基于任务-资源分配图的虚拟企业伙伴优选算法[J]. 计算机应用研究, 2013, 30(12): 3564-3567
- [6] Huang Bin, Gao Cheng-hui, Chen Liang. Research on Virtual Enterprises Task Scheduling Based on Timed Place Petri Net [J]. Journal of Engineering Graphics, 2011, 32(1): 148-153(in Chinese)
黄彬, 高诚辉, 陈亮. 基于时延库所 Petri 网的动态联盟任务调度研究[J]. 工程图学学报, 2011, 32(1): 148-153
- [7] Ghoul R H, Benjelloul A, Kechida S. A Scheduling Algorithm Based on Petri Nets and Simulated Annealing [J]. American Journal of Applied Sciences, 2007, 4(5): 269-273
- [8] Ghoul R H, Sihem, Kechida S, Tebbikh H. Scheduling of complex manufacturing systems with Petri nets and genetic algorithms; a case on plastic injection moulds[J]. International Journal of Advanced Manufacturing Technology, 2013, 69(9-12): 2773-2786
- [9] Chinnusamy T R, Karthikeyan T, Krishnan M. Flexible Manufacturing System Scheduling Using Hybrid Petri Nets with Dispatching Rules [J]. International Journal of Mechanical & Mechatronics Engineering, 2014, 14(6): 62-71
- [10] Su Guo-jun, Wang Jin, Tian Li-guo. The FMS optimal scheduling based on Petri net model [J]. Systems Engineering-Theory & Practice, 2014, 34(10): 2716-2721(in Chinese)
苏国军, 汪晋, 田立国. 基于 Petri 网模型的柔性制造系统优化调度[J]. 系统工程理论与实践, 2014, 34(10): 2716-2721
- [11] Hu He-suan, Zhou Meng-chu, Li Zhi-wu. Liveness enforcing supervision of video streaming systems using nonsequential Petri nets[J]. IEEE Transactions on Multimedia, 2009, 11(8): 1457-1465
- [12] Shao Zhi-fang, Liu Zhong-ying. Summarize on Petri Net and optimal algorithm integrated for scheduling of FMS[J]. Computer Engineering and Applications, 2007, 43(8): 197-201(in Chinese)
邵志芳, 刘仲英. Petri 网与优化算法结合求解 FMS 调度研究综述[J]. 计算机工程与应用, 2007, 43(8): 197-201
- [13] Wan Jun, Zhao Bu-hui. Petri nets decoupling control for linear-time-invariant systems [J]. Control Theory & Applications, 2014, 31(9): 1213-1220(in Chinese)
万军, 赵不贿. 线性定常系统的 Petri 网解耦控制[J]. 控制理论与应用, 2014, 31(9): 1213-1220
- [14] Wan Jun, Zhao Bu-hui, Lu Ji-yuan. Research and Development of Cyber Net System Modeling Tool[J]. Computer Science, 2014, 41(7): 97-101(in Chinese)
万军, 赵不贿, 陆继远. 自控网系统建模工具的研究与开发[J]. 计算机科学, 2014, 41(7): 97-101

(上接第 70 页)

- [12] Boukreda D, Maamri R, Aknine S. Modeling and Analysis of Reliable Contract Net Protocol Using Timed Colored Petri Nets [C]//2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). IEEE, 2013, 2: 17-24
- [13] Sun Tao, Ye Xin-ming, Liu Jing, et al. CPN based protocol testing sequence generating method[J]. Journal of PLA University of Science and Technology (Natural Science Edition), 2012, 13(2): 165-170(in Chinese)
孙涛, 叶新铭, 刘靖, 等. 一种基于 CPN 的协议测试序列生成方法[J]. 解放军理工大学学报(自然科学版), 2012, 13(2): 165-170
- [14] Billington J, Yuan C. On modelling and analysing the dynamic MANET on-demand (DYMO) routing protocol [M]. Springer Berlin Heidelberg, 2009
- [15] Fernandes J M, Tjell S, Jorgensen J B. Requirements Engineering for Reactive Systems with Coloured Petri Nets; the Gas Pump Controller Example[C]//Proc. of the 8th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools(CPN Workshop 2007). 2007: 207-222
- [16] Billington J, Gupta A K. Effectiveness of coloured Petri nets for modelling and analysing the contract net protocol[M]//Eighth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools. 2007
- [17] Yue Jing-yuan. Property Based CPN Modelling and Testing of Micro-Kernel System Call[D]. Inner Mongolia University, 2012(in Chinese)
岳婧媛. 基于属性的微内核系统调用 CPN 建模和测试[D]. 内蒙古大学, 2012
- [18] Sun Tao. Research on Testing Method for Parallel Software Based on Colored Petri Nets[D]. Inner Mongolia University, 2012(in Chinese)
孙涛. 基于 CP-nets 模型的并行软件测试方法的研究[D]. 内蒙古大学, 2012
- [19] He Nan. OpenFlow Modeling and Interaction Properties Testing Based on CPN[D]. Inner Mongolia University, 2013(in Chinese)
贺楠. 基于 CPN 的 OpenFlow 建模及交互属性测试[D]. 内蒙古大学, 2013
- [20] Zhao Hui-juan, Sun Wen-hui. Research and Application of Symbolic Execution Based Unit Testing[J]. Computer Applications and Software, 2012, 29(6): 182-185(in Chinese)
赵慧娟, 孙文辉. 基于符号执行的单元测试研究与应用[J]. 计算机应用与软件, 2012, 29(6): 182-185
- [21] Samanta B, Sarkar B. Application of Petri nets for systems modeling and analysis[J]. Opsearch, 2012, 49(4): 334-347
- [22] Dong Lu-lu. Research on the Extension Property Modelling of OpenFlow Protocol and ITS Test Cases Generation[D]. Inner Mongolia University, 2014(in Chinese)
董路路. OpenFlow 协议可扩展性建模方法研究及测试例生成[D]. 内蒙古大学, 2014