高技术通讯 2013 年 第23 卷 第11 期:1139 ~1147

doi:10.3772/j.issn.1002-0470.2013.11.006

复杂网络软件的着色 Petri 网层次建模及模型集成确认方法①

刘 靖② 叶新铭 周建涛

(内蒙古大学计算机学院 呼和浩特 010021)

摘 要 为准确描述复杂网络软件多交互、高并发等行为特征,以提高复杂网络软件设计质量和效率,提出了一种基于着色 Petri 网(CP-net)的复杂网络软件层次化建模及模型集成确认方法。给出了复杂数据抽象、并发行为控制、同类实体建模等层次建模关键技术,以及功能单元覆盖划分、模型等价抽象等模型集成确认关键技术的解决方案,并以典型网络软件系统为例分析了上述方法的可用性和有效性。从目前鲜有对特定形式模型论述模型正确性确认方法的现状看,这种融合多种 CP-net 模型分析技术的模型集成确认方法能够有效确保层次模型正确描述网络软件多交互、高并发的复杂功能行为,为软件验证、一致性测试等重要软件分析过程的高效实施提供描述准确且规模可控的基础形式模型。

关键词 着色 Petri 网(CP-net),网络软件,模型确认,并发控制,模型检验

0 引 言

复杂网络软件的交互行为多、并发程度高,通常表现为通信报文类型多、行为控制逻辑复杂、通信实体执行单一功能所需的交互次数多、并发行为执行频繁等,这些特性极易导致软件设计缺陷和运行低效。形式验证和一致性测试作为广泛应用的两种软件分析确认技术,在处理规模更庞大、逻辑更复杂的网络软件系统时面临新挑战[1-3]。为有效提高分析复杂网络软件的效率和准确度,必须为其构建规模可控且大小适度的形式模型,并充分确认该模型正确描述了软件的各种预期功能行为。然而,软件验证与一致性测试过程虽然都依赖于软件的某种形式模型,但确认模型描述正确却并非其直接目的,且这些技术通常将模型能够正确描述软件行为作为基础前提[4,5]。相关研究中,虽然模型确认问题的重要性和研究价值均得以充分论述,但尚缺乏面向具体应用特性、研究一类具体的软件形式模型的构建与确认方法。因此,针对复杂网络软件多交互、高并发的特征,研究可行、有效的软件建模与模型确认方法,对增强复杂软件设计质量和效率具有重要的基础性意义和实用价值。本文提出了一种有效的解决方案——融合多种着色 Petri 网(colored Petri nets, CP-net)[6]模型分析技术的模型集成确认方法。本研究以 CP-net 作为描述软件系统的形式模型,充分利用其在可视化与层次化建模、复杂数据描述、并发行为描述、动态可执行等方面的优势,基于不同的模型抽象层次对软件整体流程和细节行为分别加以描述,尤其对软件并发执行进行精确建模,以实现软件功能描述的详细程度与模型大小间的有效权衡与控制。此外,集成 CP-net 支持的多种模型分析技术[6],即充分利用动态模拟、行为属性分析、模型检验等技术的协同执行,确认软件 CP-net 模型不但正确描述了系统基本功能流程,而且通过验证模型并发执行结果对软件关键功能需求的可满足性,还能够确认该模型正确描述了软件的并发行为。

1 相关工作

软件形式建模与验证的研究[4,7,8]大都基于一种形式模型来完成软件功能验证及分析,并以该模型能够正确描述软件行为作为前提,因此,如何确认该模型正确描述了软件功能行为并不是这类文献需

① 国家自然科学基金(61262017, 61262082),973 计划(2012CB315802),内蒙古自然科学基金重点项目(20080404Zd20)和内蒙古大学高层次引进人才基金资助项目。

② 男,1981 年生,博士生;研究方向:可信网络软件技术,Petri 网理论与应用;联系人,E-mail:liujing@imu.edu.cn

(收稿日期:2012-11-22)

要解决的问题。而本文以复杂网络软件为应用对象,研究其系统建模与模型确认方法,以确保软件模型正确描述了网络软件多交互、高并发的复杂功能行为,为软件验证等过程的高效实施提供描述准确且规模可控的基础形式模型。

在模型确认方法的相关研究中,Drusinsky 等[9]给出了形式化验证与确认(formal verification and validation,FV&V)过程的具体涵义,并比较了定理证明、模型检验、运行时验证及自动测试等三种主流 FV&V 技术的有效性和开销。该文献将“确认”(validation)定义为:确保构建了正确的系统,即构建的系统能够满足具体的预期目标。以该定义为基础,本文将“模型确认”定义为:确保构建了正确的软件形式模型,即构建的模型能够满足软件预期的功能需求和行为特征。该文献虽多次提及形式模型在相关 FV&V 技术上的重要性,但未给出具体的模型构建与确认方法。Heimdahl[10,11]通过分析当前被普遍认可与采用的基于模型驱动的软件开发过程,明确阐述了构建规模适度的软件系统模型并确认模型正确有效,对软件研发过程是至关重要的,但也未针对某种形式模型给出具体的模型构建与确认方法。还有一类研究[12-14]针对计算机仿真模型(simulation model)讨论了将模拟、形式验证及测试等技术融合于仿真模型确认过程的可行性与应用方式,本文针对复杂网络软件的模型确认方法研究,正是借鉴了文献[12-14]这种应用不同技术进行集成确认的研究思路。Koopman 等[15]讨论了基于模型的测试方法的一个问题:模型准确度对测试结论的影响,并以一种语言模型为例说明如何利用系统测试来检查该模型中存在错误或缺陷,而本文则关注如何确认描述软件系统功能行为的模型是否正确。

综上,模型确认问题的重要性和研究价值在相关文献中均得以充分论述,但尚缺乏面向具体应用特性、研究一类具体的软件形式模型的构建与确认方法,而这正是本文将论述并实际解决的问题。此外,CP-net 建模分析的主流工具 CPN Tools[16]能够有效支撑本文所提出的集成确认方法的实施,且已有大量研究和工程实践使用 CP-net 对复杂软硬件系统进行建模,并对系统功能、性能、安全性等方面进行分析改进[17-19],从中,CP-net 和 CPN Tools 的可用性与有效性也得到广泛验证与认可。

2 CP-nets 模型的层次建模方法

2.1 层次建模方法总体框架

针对复杂网络软件交互行为多、并发程度高的特点,在基本层次 CP-net 模块定义[6]的基础上,将描述该类型软件的层次 CP-net 模型定义如下。

定义 1 复杂网络软件的层次 CP-net 模型

一个复杂网络软件的层次 CP-net 模型(hierarchical CP-net for complicated network software, HCP4NS)可定义为五元组 HCP4NS = (S_N, S_L, S_C, S_T, SM_H),其中:

(1)S_N 为网络拓扑层模块;S_L 为实体逻辑层的有穷模块集。S_C 为通信交互层的有穷模块集;S_T 为功能事物层的有穷模块集;令 $S = \{S_N \cup S_L \cup S_C \cup S_T\}$, $\forall s \in S$,则 s 可以是一个层次的或一个非层次的基本 CP-net 模块。

(2)$SM_H: T_{sub} \rightarrow S$ 为模块层次关联函数,其中 T_{sub} 为替代变迁集,而 SM_H 描述了高层替代变迁与底层对应 CP-net 模块的关联规则,具体定义为:若 $t_1 \in T_{sub}$ 且 $t_1 \in S_N$,则 $SM_H(t_1) \in S_L$;若 $t_2 \in T_{sub}$ 且 $t_2 \in S_L$,则 $SM_H(t_2) \in S_C$;若 $t_3 \in T_{sub}$ 且 $t_3 \in S_C$,则 $SM_H(t_3) \in S_T$;若 $t_4 \in T_{sub}$ 且 $t_4 \in S_T$,则 $SM_H(t_4) \in S_T$。

HCP4NS 定义中还涉及到模块端口关联和位置融合集的定义与层次 CP-net 模块[6]相同。

为一个具体的复杂网络软件构建层次 CP-net 模型时,通常采用自顶向下的层次建模方式:网络拓扑层描述参与软件运行的实体数量及实体间的网络拓扑连接;实体逻辑层描述实体节点的功能逻辑,作为实体功能描述的最高级抽象,要清晰给出实体主体功能行为之间的因果关联;通信交互层描述通信报文的生成、发送、接收和解析等环节,重点描述这些环节的逻辑关联及针对不同报文的区分处理;功能事务层对软件行为的细节功能进行建模,如关键数据结构维护、数据获取与计算等,需要注意两个问题:不对公共事务重复建模,消除模型冗余;防止过于细节的建模导致软件模型状态空间膨胀。软件设计中关键算法的建模大都集中在功能事务层,通常对算法的核心逻辑以及关键数据处理进行描述。

上述层次化建模方法充分利用了 CP-net 模型的层次化构建技术[20],将软件复杂的交互行为建模在实体逻辑、通信交互和功能事务等不同层次上,一方面清晰描述软件功能在总体逻辑、报文交互、处理细节等不同层面的特征,支持模型描述的详细程度

与模型规模大小间的调整;另一方面,便于将软件细节功能集中在功能事务层中进行描述,并可重用其中的公共行为模块,易于模型的局部更新和改进,降低建模复杂度。

上述建模方法对复杂数据、并发行为和同类实体的建模,是保障模型准确度与规模大小的重要环节,建模不当则很容易引发软件模型的状态空间爆炸。因此,下文将重点论述这三个建模关键环节的技术解决方案。

2.2　复杂数据抽象

本文采用基于分级抽象的数据描述方式,首先提取复杂数据中的关键构成因素,形成基本字段,并利用 CPN Tools 提供的基本数据类型(整型、布尔型、字符串、枚举型等)进行描述;之后,依据复杂数据中基本字段间的构成及关联特性,选择一种适合的组合数据类型(叉乘型、记录型、链表型等)对其进行组织和描述,充分利用组合数据类型所支持的操作函数(元素提取、数据比较、数据拼接等)实现复杂数据的高效计算与更新。例如针对频繁访问的关键数据结构,分别使用记录型和链表型组织并描述其单项元素和整体结构,利用记录型和链表型的数据操作函数完成高效的数据存取和更新。

2.3　并发行为控制

Petri 网模型本身对系统并发和同步具备较强的描述能力,能够真实反映行为并发特征,而本文基于 CP-net 模型的层次建模方法在充分继承了这种优势的前提下,可以对软件不同程度的并发行为进行有针对性的处理:将能够以串行化方式执行的并发功能调整为顺序执行,以减少并发状态数;而将软件真正的并发功能建模于实体逻辑层或通信交互层,以便于在模型确认过程中对这类并发行为实施有效的正确性验证。具体而言,在网络实体执行流程中,可能存在互不相关的两个功能行为,例如根据接收报文的数据信息一方面要更新关键数据结构,而同时需要发送新的请求报文,这两种彼此独立的行为可以建模为并发执行,也可以建模为顺序执行,执行效果相同,但建模为并发执行所产生的状态空间要远大于建模为顺序执行所产生的状态空间,因为前者会造成大量后续细节功能的交叠执行,导致状态空间膨胀且对软件行为分析没有任何意义。因此,本文通过引入一种控制位置,使这两种行为对应的变迁能够依序点火,实现非真实并发行为的强制顺序化,可以在不损害软件正常功能的前提下有效削减软件模型的状态空间。但对于软件真正的并发功能行为,需要对其进行正常描述,并利用第 3 节给出的模型确认方法验证并发执行效果的正确性。

2.4　同类实体建模

网络软件中通常存在参与软件流程的多个相同实体,例如同时与服务器进行交互的多个客户端,其功能和行为完全相同,如果为每一个实体都构建独立的 CP-net 模块,虽然描述简单直观,但容易导致模型规模过于庞大,并且在新增或修改同类实体时扩展性很差。因此,本文将多个同类实体的功能行为建模在同一个 CP-net 模块上,通过参数化实体身份标识实现个体区分。首先,对不同个体实现参数化的身份标识描述,利用索引(index)类型标识不同实体,其索引序号即为个体的唯一身份标识;其次,在数据报文描述中增加报文发送者和接收者字段,用于区分同时出现在该实体模块中,但隶属不同个体的数据报文;最后,在关键数据结构中增加实体标识维度,便于对不同个体的关键数据进行统一存储和维护。

基于 CP-net 模型的网络软件分层建模方法能够切实在不同抽象层次上对软件交互流程、并发执行和细节功能进行精确描述与权衡控制,形成规模适度的软件层次 CP-net 模型,以促进后续模型确认过程的有效实施。第 4 节将具体阐述如何利用上述层次建模方法及关键技术为数字内容点对点分发软件系统构建准确描述软件功能且规模可控、大小适度的 HCP4NS 模型。

3　CP-nets 模型的集成确认方法

3.1　模型集成确认方法总体框架

通过前述方法构建的软件层次 CP-net 模型的规模通常较大,难以生成完全状态空间,使得通常直接基于完全状态空间的软件行为正确性分析方法不可实施。因此,本节给出一种融合动态模拟执行、行为属性分析、模型检验等技术的软件 CP-net 模型集成确认方法(见图 1)。

一方面,将软件执行中每一个相对独立的基本功能流程定义为一个功能单元,利用不同的初始标识赋值[6],并调整流程结束对应变迁的约束条件,从而实现从已构建的完整层次 CP-net 模型中提取不同的执行路径构成不同功能单元。对每一个功能单元都执行动态模拟和行为属性分析,以确认完整层次 CP-net 模型能够正确描述软件系统所有的基本功能。

另一方面,不同功能单元以顺序或并发方式联合执行,可以描述更加复杂的软件交互功能与并发行为,但由于很难通过枚举并动态模拟所有并发执行效果来确认软件并发执行效果,故本文将从软件需求被满足的角度确认软件模型对软件并发行为的描述是正确的。首先,基于并发结构等价的模型抽象方法,在不改变软件真实并发行为描述的前提下,将软件完整层次 CP-net 模型中已确认正确的细节功能进行适度抽象,形成软件抽象层次 CP-net 模型,缩减模型规模;之后,将软件关键的功能需求描述为时序逻辑公式,称为并发属性。基于抽象层次模型执行这些属性的模型检验,若其能够被满足,则依据模型检验基于完整状态空间执行的事实,可以判定在软件并发行为的所有可能执行情况下,软件关键的功能需求即并发属性都会被满足,从而说明软件模型对并发行为的描述是正确的。与一般验证过程所使用的模型检验方法相比,上述模型检验过程在选取待验证属性时,更侧重于选择那些能够反映软件真实并发行为执行效果的功能需求属性。

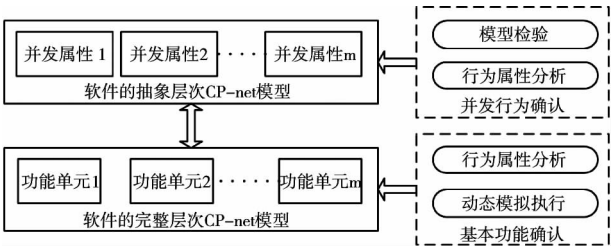

图 1 CP-net 模型集成确认方法的总体框架图

作为上述模型集成确认方法的核心环节,功能单元划分方法的合理性与模型等价抽象方法的可靠性对模型集成确认方法的有效实施是至关重要的。因此,下文重点论述这两方面技术的解决方案。

3.2 功能单元的覆盖划分

功能单元的合理划分需要充分覆盖软件所有可能的交互行为,且易于对功能单元进行动态模拟和行为属性分析。本文参考 Petri 网模型的测试覆盖标准[21,22],并结合功能单元划分的具体需求,提出如下功能单元划分标准:(1)完全覆盖模型中每个变迁的所有分支执行;(2)功能单元的执行能够生成完全状态空间,即功能单元内部不会出现不可控的并发交互行为,能够执行模型属性分析。依照此标准,划分功能单元的具体过程如下:

步骤 1:分解软件功能为若干子功能执行,通过初始标识赋值模拟各子功能的执行过程,并记录过程中所有被点火变迁的分支执行情况。

步骤 2:对各子功能进行行为属性分析,若其执行可以生成完全状态空间,则定义该子功能执行为一个功能单元,执行步骤 3;否则,重新执行步骤 1。

步骤 3:考察各变迁被点火时的分支执行情况:若该变迁未曾被点火,则针对该变迁描述的功能重新执行步骤 1;若某个变迁曾被点火,但存在未曾执行的分支(通常表现为输出弧表达式中存在未曾执行过的分支语句),则首先调整与该变迁相关的功能单元的初始标识赋值,使其覆盖没有执行的分支,并作为新的功能单元,对其重新执行步骤 2;否则同样针对该变迁描述的功能重新执行步骤 1。

对功能单元的行为属性分析是模型确认过程的重点之一,通过分析各功能单元在执行过程中所对应的标识有界性、终止标识、死锁变迁、活变迁等模型基本行为属性是否正常,可以确认软件完整层次 CP-net 模型是否正确描述了软件所有功能行为。

3.3 基于并发结构等价的模型抽象

本文利用 CPN Tools 提供的 ASKCTL 模型检验技术[16]来确认软件并发执行效果是正确的。基于并发结构等价实现模型抽象的具体算法过程如下:

步骤 1:在完整层次 CP-net 模型中,若 $\forall s \in \{S_N \cup S_L \cup S_C\}$,则保留 s。

步骤 2:若 $\forall s \in \{S_T\}$,且 s 中没有真正的并发行为,则 $\exists t \in T_{sub}, t \in S_C$,且 $SM_H(t) = s$,将替代变迁 t 修改为普通变迁,并保持其输入弧与输出弧的分支情况,实现细节功能抽象。

步骤 3:若 $\forall s \in \{S_T\}$,且 s 中存在真正并发行为,则保留 s,合并其中具有逻辑关联、但无分支处理的若干顺序执行的变迁元素,以适度抽象非并发行为功能,简化模型结构。

步骤 4:若多个功能事务层上的 CP-net 模块共享同一个关键数据结构(即采用位置融合集进行数据结构描述),则将该位置融合集建模于对应的通信交互层模块中,并建立其与相应行为变迁的逻辑关联,以保障软件功能的正确执行。

步骤 5:关键算法的描述模块通常不涉及实体间的并发交互,故采用步骤 2 进行模型抽象(可通过在关联位置指定不同的标识来模拟算法执行)。

上述并发结构等价的模型抽象方法中,步骤 2 和 3 是核心步骤,可多次迭代执行,且经过抽象完成后得到的软件抽象层次 CP-net 模型与软件原来的完整层次 CP-net 模型对软件并发行为的描述是等同有效的。首先,对替换为普通变迁的替代变迁而

言,其对应的功能事务层模块已经通过功能单元确认过程确认了其对软件功能行为的正确描述,可以使用普通变迁抽象表示这类操作能够正确执行,并且这类变迁中并无并发行为的相关描述。其次,真正的并发行为描述所对应的并发结构均在上述抽象方法中予以保留,包括带有并发关系的行为变迁,以及涉及并发操作的关键数据结构等。因此,经过并发结构等价的模型抽象后,软件模型规模大为缩减,能够支撑基于 ASKCTL 模型检验技术[16]的并发行为确认过程的有效实施。

4　实例分析:数字内容点对点分发软件

数字内容点对点分发软件系统由作者所在课题组研发,按照 BitTorrent 协议[23]实现数字内容资源在互联网中的快速分发和高效共享。在多个用户节点上可分别安装部署软件的一个执行实体,每个节点从其他节点下载文件的同时也为别人提供文件上传,这样软件实体间的通信交互更为复杂,且行为并发度高。下文针对该软件系统的复杂特性,构建规模适度且大小可控的层次 CP-net 模型,并确认该模型准确描述了软件复杂交互功能和并发行为。作为实际应用实例,也同时验证了本文基于 CP-net 模型的软件层次建模与模型集成确认方法的可行性与有效性。

4.1　系统功能描述及建模约束

数字内容点对点分发软件系统包含三种软件执行实体节点:下载者(leecher)、种子节点(seed)和索引服务器(tracker)。软件系统将共享的文件切分为等长的片段(piece)进行分发,一个节点可以同时从其他节点下载不同的文件片段。本文的建模分析场景为 2 个下载者和 1 个种子节点间共享单一文件的 3 个片段(已涵盖软件所有主要功能)。

下载者节点首先与索引服务器交互,获取可用节点列表(可为该下载者提供文件片段的节点,可以是其他下载者或者种子节点)。下载者随机选择其中部分节点握手建立连接,并交换位图(bitmap,用于记录节点已有片段)获知该节点已有哪些文件片段。下载者向已建立连接的各节点请求不同的文件片段。本文对基本阻塞算法、最少优先和随机第一片等两种基本片段选择算法[23]进行建模。

4.2　软件的 CP-net 层次建模

如图 2 所示,该软件系统完整的 HCP4NS 模型包括 17 个无重复的 CP-net 模块,分别描述了整体网络拓扑,以及三种不同节点描述在不同抽象层次上的逻辑流程、交互行为和功能细节。

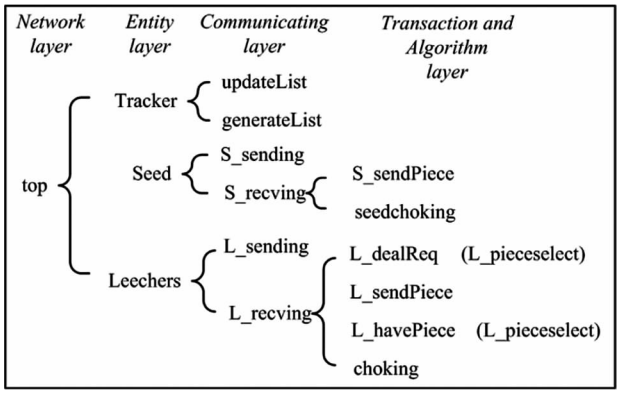

图 2　数字内容点对点分发软件完整 CP-net 模型

图 3 给出关键数据建模描述。其中,下载者标识采用 index 类型(colset PEERID),可实例化为 pid(0)、pid(1)等,便于同类节点的统一描述。数据报文(colset PAKCET)由报文内容及收发两方实体等字段叉乘连接,而报文内容(colset MSG)采用 union 类型统一组织,其元素为不同类型报文的具体描述,如 COMM_MSG 类型用于描述控制报文内容,TRANS_MSG 类型用于描述实际文件片段的请求与传输报文。BMSET 数据结构用于存储下载者自身已请求和已存储的文件片段信息,采用 list 类型描述整体结构及 record 类型描述单项记录。

```
colset INFOHASH = int with 1..file_no;
colset PEERID = index pid with 1..peer_no;
colset BITMAP = list INT with 0..bit_no;
colset TRANS_MSG = product PPTYPE * INFOHASH * BITMAP * UPRATE;
colset COMM_MSG = product PPTYPE * INFOHASH;
colset MSG = union TRANSMSG: TRANS_MSG + COMMMSG: COMM_MSG;
colset PACKET = product MSG * PEERID * PEERID;
colset BMENTRY = record file:INFOHASH * peer:PEERID * bitmaps: BITMAP * reqpics:BITMAP;
colset BMSET = list BMENTRY with 0..bmentry_no;
```

图 3　软件 CP-net 模型的数据建模

图 4 给出软件网络拓扑层模块(top),描述系统总体运行架构。不同下载者节点采用同一个替代变迁 *Leecher* 进行统一建模。图 5 给出的下载者实体逻辑层模块对应于图 4 中的 *Leecher* 替代变迁,用于进一步细化描述下载者的主要行为流程,包括请求

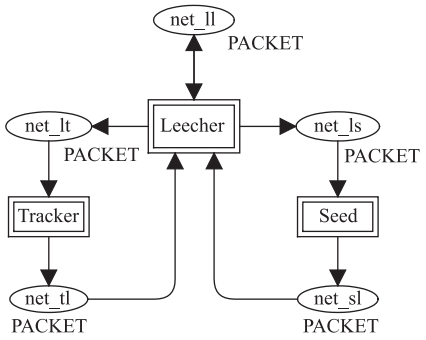

图 4 网络拓扑层模块

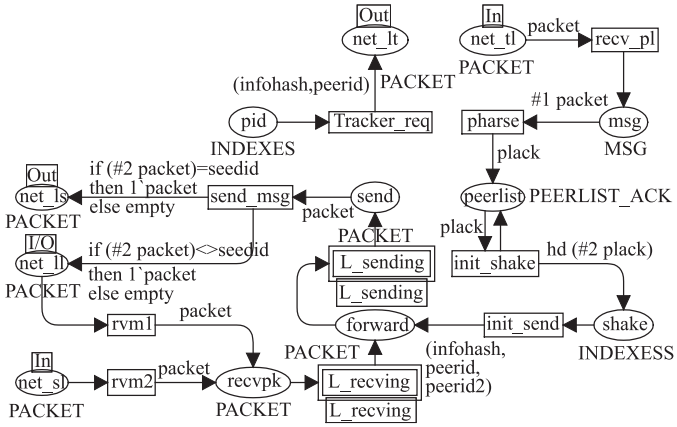

图 5 下载者的实体逻辑层模块

可用节点列表及解析、与种子节点或其他下载者间请求和收发文件片段等。同样,对应图 5 中的 *L_recving* 替代变迁,图 6 所示的通信交互层的模块实例描述了下载者节点接收到不同类型报文时更为细节的处理流程。当下载节点获得包含至少两个可用节点的 PEERLIST 列表时,会并发的与这些节点连接并请求不同片段,继而导致针对不同片段的请求、发送、存储等后续交互行为之间在同一个节点内的并发执行。如果将这些行为通过强制其顺序执行来消除并发特征,有违软件运行事实。因此,在正常描述这类并发行为的基础上,通过模型集成确认方法来验证并发行为描述的准确性。

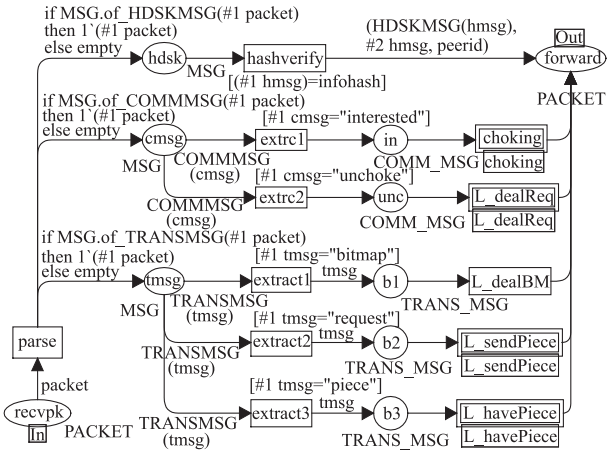

图 6 下载者交互报文处理流程的通信交互层模块

图 7 示例了下载者节点接收到片段后,更新 BMSET 结构并请求新片段的过程,对应于图 6 中的 *L_havePiece* 替代变迁。功能事务层建模过程中,并发行为的控制是最为重要的建模环节。如本文第 2.3 节所述,需要将能够以串行化方式执行的并发功能调整为顺次执行,以减少并发状态数。例如,变迁 *send* 描述新生成片段请求报文的发送行为,而变迁 *add* 与 *del* 联合描述了 BMSET 结构的更新行为。这两种行为相互独立,建模为并发执行时可达图中会存在大量因细节行为交叠所产生的并发冗余状态,而其最终执行效果与顺序执行相同,即所有并发执行路径都将汇集到同一个标识。因此,在变迁 *send* 与变迁 *del* 之间增加一个控制位置 *n*,使变迁 *del* 点火执行后变迁 *send* 才能被点火执行,即更新 BMSET 结构完成后再进行新生成片段请求报文的发送,实现了非真实并发行为的强制顺序执行,而并不破坏软件功能的正常执行。关键算法层的建模需要与功能事务层类似,因篇幅所限不再给出阻塞算法与片段选择算法的具体 CP-net 模块。

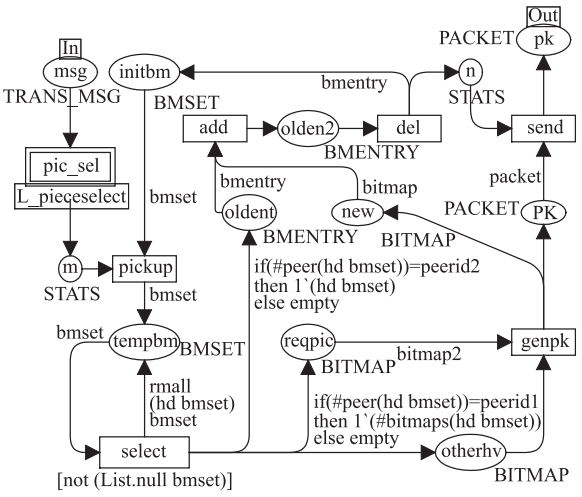

图 7 下载者请求新片段的功能事务层模块

至此,依据本文第 2 节提出的层次建模方法,构建了数字内容点对点分发软件的完整 HCP4NS 模

型,在不同抽象层次上对软件交互流程、并发执行和细节功能进行权衡控制与精确描述,使软件模型规模可控且大小适度,利于下文模型集成确认过程的实施。

4.3　软件 CP-net 模型的集成确认

根据前述功能单元划分标准和提取方法,可生成表 1 所示的 5 个功能单元,其中下载者 L1 是新加入的,下载者 L2 已有片段 p1 和 p2。

表 1　系统的功能单元描述

编号	功能描述	状态数	变迁分支覆盖率	死变迁/活变迁
a	L1 从索引服务器请求并接收可用节点列表	17	10%	无
b	L1 从 L2 随机下载一个文件片段	135	62%	
c	L1 从种子节点下载最少文件片段	157	94%	
d	L1 从 L2 下载了一个错误的文件片段	130	96%	
e	L2 执行阻塞算法	19	100%	

对上述每个功能单元进行多次动态模拟,旨在去除模型冗余部分、修正模型错误并适度抽象模型的细节部分。此外,对每个功能单元执行行为属性进行了分析,分析结论见表 1:各功能单元均可生成状态数较小的完全状态空间;变迁分支覆盖率列给出了该功能单元与之前所有功能单元的变迁分支覆盖率累加后所占模型所有变迁分支数目的比例,可看出 b 和 c 是主体功能单元,且功能单元划分已覆盖软件系统的全部行为(覆盖率达到 100%);产生的终止标识均对应于一种软件运行正常终止的场景,且无死锁变迁和活变迁等异常变迁,即系统不存在异常终止或无法终止等错误行为。至此,我们确认了该完整层次 CP-net 模型对软件系统基本功能行为的描述是准确有效的。

下面考虑并发行为的确认问题。功能单元 a 执行后,若 b 和 c 并发执行,则只能生成部分状态空间,无法通过验证关键功能需求属性是否满足来确认软件模型对并发执行的正确描述。我们根据 3.3 节提出的模型抽象方法,构建了与软件完整层次 CP-net 模型并发结构等价的抽象层次 CP-net 模型。具体而言,保留了原网络拓扑层、实体逻辑层和通信交互层的共 10 个模块,对含有替代变迁的 *S_recving* 和 *L_recving* 及算法模块进行了基于并发结构等价的模型抽象。作为示例,图 8 给出了与图 6 并发结构等价的抽象 CP-net 模块。原模块中的 4 个替代变迁均已替换为普通变迁,因其对应的功能行为已经过有效确认。

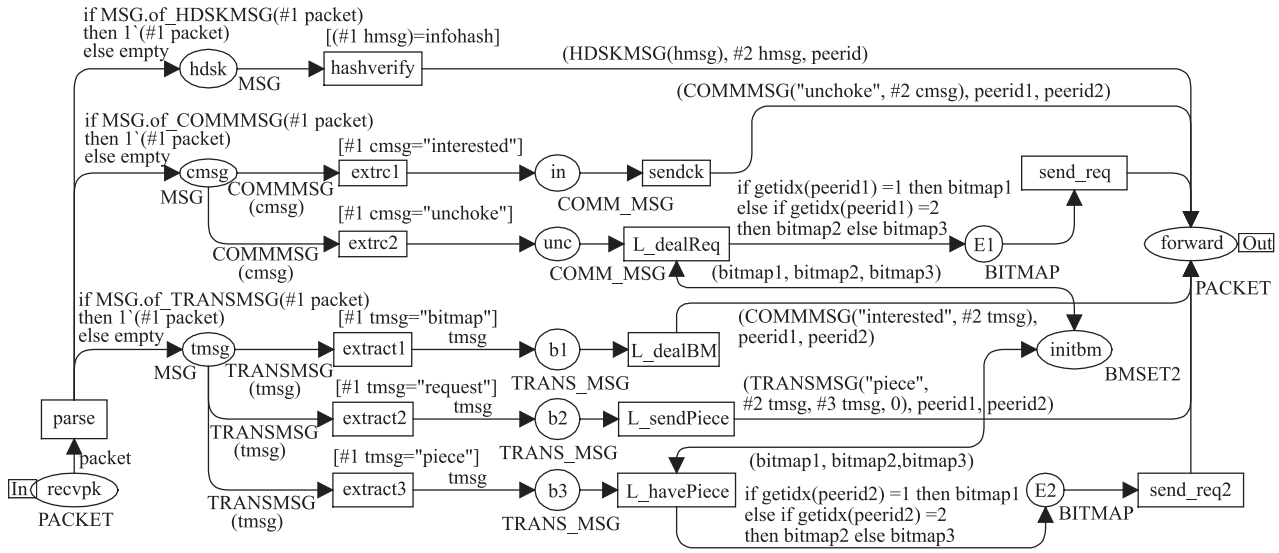

图 8　下载者交互报文处理流程的通信交互层抽象模型

该抽象层次模型可生成完全状态空间(9989 个状态节点和 19274 条状态变迁),且没有异常变迁和异常死标识。参照典型的属性描述模式[24],将在任何并发执行下都要被满足的系统关键功能需求描述为若干并发属性,表 2 给出了其中 4 个关键属性的 ASKCTL 公式描述。图示属性的验证结果均为 true,

表示其在抽象模型中成功通过验证,说明软件抽象模型所有可能的并发执行并不会违背这些属性所描述的软件关键需求。那么,当所有预期的软件功能需求属性都被验证成功后,便可确认软件抽象及完整模型对软件并发行为的等同描述是正确的。

表 2 并发属性 ASKCTL 描述及其模型检验结果

编号	ASKCTL 描述	说明	结论
a	Existence_Formula = MODAL(POS(AF("RequestSeed",IsReqfromSeed)));	L1 一定会从种子节点下载片段	true
b	Absence_Formula = POS(NOT(MODAL(AF("PieceRecv",IsMultiPieceRecv))));	L1 不会下载两个相同的片段	true
c	Precedence_Formula = INV(OR(MODAL(AF("Unchoke",IsUnchoke)),POS(NOT(MODAL(AF("ReceivePiece",IsRecvPiece))))));	L1 从 L2 下载片段是因为 L1 从 L2 接收过 unchoking 报文	true
d	Response_Formula = INV(OR(NOT(MODAL(AF("PieceRequest",IsPicReq))),EV(MODAL(AF("PieceHave",IsSendPiece)))));	只要 L1 向 L2 请求已有片段,L2 一定会发送该片段给 L1	true

通过上述实例应用可以看出,功能单元覆盖划分方法和基于并发结构等价的模型抽象方法能够有效克服因模型状态空间过大所导致的模型分析困难,充分保证了模型集成确认方法的可行性与有效性。

5 结 论

针对复杂网络软件交互行为多、并发程度高的行为特征,提出了一种基于 CP-net 的软件层次建模与模型集成确认方法,在确认软件完整 HCP4NS 模型正确描述了软件基本交互功能的基础上,通过验证模型并发执行结果对软件关键功能需求的可满足性,进一步确认了软件的完整和抽象层次 CP-net 模型都正确描述了软件的并发行为。结合数字内容点对点分发软件的实际建模工作,阐释了该层次建模及模型集成确认方法的可用性和有效性。

本文提出的基于 CP-net 的复杂网络软件层次建模与模型集成确认方法的主要优势在于:首先,从目前鲜有针对特定形式模型论述其模型确认方法的研究现状看,我们提出的融合多种 CP-net 模型分析技术的模型集成确认方法是一种可行且有效的解决方案,能够保证软件的 HCP4NS 模型正确描述了软件多交互、高并发的复杂功能行为。其次,经过层次化构建,以及集成确认后的软件 HCP4NS 模型,能够同时对软件的关键功能细节和高层并发行为进行准确描述,为软件验证、一致性测试等其他软件形式化分析过程的实施提供了切实可用的基础形式模型。

参考文献

[1] Woodcock J, Larsen P G, Bicarregui J, et al. Formal methods: practice and experience. *ACM Computing Surveys*, 2009, 41(4): 19:1-19:36

[2] 陈火旺,王戟,董威. 高可信软件工程技术. 电子学报, 2003, 31(12A): 1933-1938

[3] 梅宏,王千祥,张路等. 软件分析技术进展. 计算机学报, 2009, 32(9): 1697-1710

[4] Jhala R, Majumdar R. Software model checking. *ACM Computing Surveys*, 2009, 41(4): 21:1-21:54

[5] Tretmans J. Model based testing with labelled transition systems. *Formal Methods and Testing*, LNCS 4949, 2008, 1-38

[6] Jensen K, Kristensen L M. Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Berlin: Springer, 2009. 95-188

[7] Kupferman O, Vardi M Y. Model checking of safety properties. In: Proceedings of the 11th International Conference on Computer-Aided Verification, Trento, Italy, 1999. 172-183

[8] 蒋屹新,林闯,曲扬等. 基于 Petri 网的模型检测研究. 软件学报, 2004, 15(9): 1265-1276

[9] Drusinsky D, Michael J B, Shing M T. A visual tradeoff space for formal verification and validation techniques. *IEEE System Journal*, 2008, 2(4): 513-519

[10] Heimdahl M P. A case for specification validation. *Verified Software: Theories, Tools, Experiments*, LNCS 4171, 2008, 392-402

[11] Heimdahl M P. Let's Not Forget Validation. http://wenku.baidu.com/view/77137b0790c69ec3d5bb75b4.html: Baidu, 2011

[12] Leye S, Himmelspach J, Uhrmacher A M. A discussion on experimental model validation. In: Proceedings of the 11th International Conference on Computer Modeling and Simulation, Cambridge, UK, 2009. 161-167

[13] Balci O. Verification, validation, and certification of mod-

eling and simulation applications. In: Proceedings of the 2003 Winter Simulation Conference, New Orleans, USA, 2003. 150-158

[14] Sargent R G. Verification and validation of simulation models. In: Proceedings of the 2008 Winter Simulation Conference, Miami, USA, 2008. 157-169

[15] Koopman P, Achten P and Plasmeijer R. Testing and validating the quality of specifications. In: Proceedings of the 2008 IEEE International Conference on Software Testing Verification and Validation Workshop, Washington, DC, USA, 2008. 41-52

[16] CPN Tools. http://cpntools. org: AIS group, Eindhoven University of Technology, 2012

[17] 门鹏,段振华. 着色 Petri 网模型检测工具的扩展及其在 Web 服务组合中的应用. 计算机研究与发展,2009, 46(8):1294-1303

[18] Billington J, Yuan C. On Modelling and Analyzing the Dynamic MANET On-Demand (DYMO) Routing Protocol. *Transactions on Petri Nets and Other Models of Concurrency III*, LNCS 5800, 2009, 98-126

[19] Permpoontanalarp Y, Sornkhom P. A new colored petri net methodology for security analysis of cryptographic protocols. In: Proceedings of the 10th Workshop on Practical Use of Coloured Petri Nets and the CPN Tools, Aarhus, Denmark, 2009. 81 ~ 100

[20] Jensen K, Kristensen L M, Wells L. Coloured petri nets and CPN tools for modeling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 2007, 9(3-4):213-254

[21] Zhu H, He X D. A methodology of testing high-level petri nets. *Information and Software Technology*, 2002, 44(8): 473-489

[22] Farooq U, Lam C P, Li H. Towards automated test sequence generation. In: Proceedings of the 19th Australian Conference on Software Engineering, Perth, Australia, 2008. 441-450

[23] Cohen B. Incentives build robustness in BitTorrent. In: Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley, USA, 2003. 1-5

[24] Matthew B D, George S A, James C. Patterns in property specifications for finite-state verification. In: Proceedings of the 21st International Conference on Software Engineering, Los Angeles, USA, 1999. 411-420

Colored Petri nets based hierarchical modeling and integrated model validation approach for complicated network software

Liu Jing, Ye Xinming, Zhou Jiantao

(College of Computer Science, Inner Mongolia University, Hohhot 010021)

Abstract

An approach for hierarchical modeling and integrated model validation of complicated network software based on colored Petri nets (CP-net) is proposed to precisely describe the software's complicated functionalities and concurrent behaviors to improve the software's design and efficiency. Certain significant techniques for modeling, such as complex data abstraction, concurrency control and homogeneous entities modeling, are presented in detail, and the resolutions of function units generation and concurrent-equivalent model abstraction used for model validation are given. Besides, the proposed approach was applied to a specific network system as a representative to illustrate its usability and effectiveness. As there is few specific model validation methods discussed in literatures, this study contributes to a novel CP-net based integrated model validation approach with better feasibility. Validated software CP-net hierarchical models can specify complicated functionality and concurrent behaviors precisely for complex network software systems, and furthermore, they can be well used as fundamental formal models to promote the effectiveness and efficiency for the software verification or conformance testing technologies.

Key words: colored Petri nets (CP-net), network software, model validation, concurrency control, model checking