软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(4):804−818 [doi: 10.13328/j.cnki.jos.005191]

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

基于通信 Petri 网的异步通信程序验证模型*

杨启哲, 李国强

(上海交通大学 软件学院,上海 200240)

通信作者: 李国强, E-mail: li.g@sjtu.edu.cn

摘 要: 由于多栈的模型图灵等价,因此,通用的异步通信程序模型的验证问题不可判定.为此,基于Petri网,提出了一个新的模型通信——通信 Petri 网,对异步通信程序进行刻画.通过对输入通信进行 k-型限制以及对每个栈进行基于正则语言泵引理的抽象,通过将这样限制下的模型编码到数据 Petri 网,证明了限制下的新模型可覆盖性可判定.

关键词: 异步通信程序;通信 Petri 网;可覆盖性;程序验证;k-型

中图法分类号: TP311

中文引用格式: 杨启哲,李国强.基于通信 Petri 网的异步通信程序验证模型.软件学报,2017,28(4):804−818. http://www.jos.org.cn/1000-9825/5191.htm

英文引用格式: Yang QZ, Li GQ. Model on asynchronous communication program verification based on communicating Petri nets. Ruan Jian Xue Bao/Journal of Software, 2017,28(4):804−818 (in Chinese). http://www.jos.org.cn/1000-9825/5191.htm

Model on Asynchronous Communication Program Verification Based on Communicating Petri Nets

YANG Qi-Zhe, LI Guo-Qiang

(School of Software, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: Since multi-stack models are generally Turing-complete, verification problems on general models for asynchronous communication programs are undecidable. This paper proposes a new model based Petri net, named communication Petri nets (C-Petri nets) to model asynchronous communication programs. Applying the k-shaped restriction on the input communications and the abstraction on each stack based on popping lemma of regular languages, the coverability problem of the restricted C-Petri net is decidable by encoding the model to data Petri nets.

Key words: asynchronous communication program; communication Petri net; coverability; program verification; k-shaped

随着大规模复杂多核系统的广泛使用,并发程序得到了越来越多的应用.但并发程序不同于顺序程序具有通用的程序验证模型(下推系统),即使将所有数据域抽象成有限域,并发程序所对应的多栈模型也图灵等价,其上最简单的验证,如可达性不可判定[1].异步程序是并发程序中最常用的管理并发交互的一种方式,该种程序不需要等待函数调用返回结果,可以直接继续执行,直到程序需要用到调用的返回结果时再去等待函数的调用结果;同步程序则相反,一个同步调用要求程序必须等待至函数调用完成返回结果之后,程序才能继续执行下去.异步通信程序在异步调用之时,允许不同线程之间通过中间存储方式进行通信.这样的程序语言包括 Erlang, Scala 等,其程序行为更加复杂,难以进行建模和验证.

在异步程序验证模型的研究上,Sen 和 Viswanathan 提出了多重集下推系统[2],只允许在栈空的时候发生通信,并证明了该模型的可覆盖性可判定.这样的模型对于异步通信程序而言过于受到限制,因为一般来说,异步通信程序中的通信发生都不是在栈空的时候.Kochems 和 Ong 提出了异步偏序通信下推系统[3],并且对其进行

* 基金项目: 国家自然科学基金(61672340, 61472238, 91318301)
Foundation item: National Natural Science Foundation of China (61672340, 61472238, 91318301)
收稿时间: 2016-05-27; 修改时间: 2016-09-08, 2016-11-26; 采用时间: 2017-01-04; jos 在线出版时间: 2017-01-24
CNKI 网络优先出版: 2017-02-20 13:51:08, http://www.cnki.net/kcms/detail/11.2560.TP.20170220.1351.008.html

k-型限制.该限制将栈与通信绑定在一起,并且限定模型中栈上输入通信的数量不能超过 *k*.在该模型中,栈不再作为递归的标识,因而很难对于程序进行直观建模,并且对每个线程递归的深度限制得也比较严重.同时,这种编码会引起栈元素呈指数级扩张,使得模型很难找到可用于实现的高效算法.

本文基于 Petri 网,提出了通信 Petri 网.在通信 Petri 网里,一个线程被描述成在库所(place)里扩展的令牌(token),其中一个令牌是一个二维向量的形式,而令牌在通信 Petri 网的轨迹则是一个进程的行为表示.由于整个信箱存放的信息是无序的,因此消息的传递则是通过一个全局的向量来完成.同时,每个进程会反映出它所进行的通信和栈的情况.

为了保证可判定性,通信 Petri 网对于通信和栈的问题进行了分别的限制.对于通信,通信 Petri 网将 *k*-型限制转变为限制每个线程所能接受信息的最大数量为 *k*,也就是一个令牌的第 2 维向量的值不会超过 *k*;对于栈的部分,我们根据正则语言的汞引理,将栈抽象为一种等价关系.在上述两种方法的限制和抽象下,通信 Petri 网可以编码到数据 Petri 网,从而其覆盖性是可判定的.该方法在程序建模和验证中由于保持栈对于递归的刻画,因而更为直观和清晰,也更利于实现.

我们用一个例子来说明如何使用通信 Petri 网来验证异步通信程序.图 1 和图 2 给出了一个用 Erlang 实现的简单的老师自动网上给学生分配一个作业的系统,这个程序使用有限个信道来实现通信.

```
1 main()->setup_network().
2       reassign().
3
4 setup_network()->
5  spawn(student)
6  case (*) of
7   true ->setup_network();
8   false ->
9    spawn(start_resource(init)).
10    toResource!isReady
11    receive toTeacher:
12      ready ->()
13    end;
14  end, toStudents!homework.
15
16 reassign()->
17  receive toTeacher:
18   redist(Homework)->toStudents!Homework;
19   answer(Answer)->print(Answer);
```

```
21 start_resource(S)->
22  toTeacher! Ready.
23  resource(S)
24
25 resource(S)->
26  receive toResource:
27   lock_req->
28    toStudent!locked.
29    resource_locked(S)
30  end.
31
32 resource_locked(S)->
33  receive toResource:
34   unlock_req->resource(S);
35   getState->
36    toStudents!state(S);
37    resource_locked(S);
38   update(X)->resource_locked(
39  end
```

Fig.1 An example of program (I)

图 1 一个程序的例子(I)

假设作业需要学生们合力完成,并且共享一个资源,每个学生只需完成自己要做的那一部分.程序一开始打开网络并且打开资源,将有一个老师分配作业的系统,他将所得到的作业重新分配给学生并且打印学生给出的答案.学生在收到一个作业后将会对作业逆行分析,取出自己需要做的部分,将剩余的部分交还给老师再进行分配.如此循环.在显示的程序里,具体处理作业的程序被省略了.需要注意的是:由于解决作业的资源只有一个,而学生在处理作业时需要使用这个资源,因此会产生一系列有关是否互斥的问题.

注意到:在这样的程序里,尽管在每个函数里都有不停的函数调用,也会有着任意多的发送信息,但是对于任何一个线程,其接受消息的次数是有限的.也就是说,在异步模式下,可能停住的次数不是无限多次.并且在程序中栈的形式是没有记忆能力的,也就是说,这样的程序无法严格地分辨重复调用的不同.比如,*reassign*()重复调用自己,但程序并不会分辨 *reassign*()调用了自己 2 次或是 3 次这样的区别.

在这个例子中,程序里一共有 3 个信道:toResource,toStudents 和 toTeacher,发送的信息由于返回的答案和分配的作业是有限多个的,不妨假设共 *k* 种,因此可以用一个 3×*k* 维向量来模拟,第 *k*×*i*+*j* 维对应第(*i*+1)个信道里信息 *j* 的数量.比如:当对 toResource 发送第 1 种信息的时候,需要在第 1 维加 1.同时,从这一接受信息的时候开始则需要对第 1 维减 1.将每一行程序和调用这一行程序的函数编码设为一个状态,考虑程序中的 *reassign*()函数,

在整个程序中,它会被主程序以及自己调用,因此在编码至通信 Petri 网时,*reassign*()函数的一行会被表示成两个库所:一个代表主程序调用的 *reassign*()里的语句,另一个则代表自己调用自己里的语句.一个令牌则代表一个线程,程序的执行可以用相应的转移来进行描述.同样,考虑在 *reassign*()函数,不妨仅考虑是自己调用自己的时候,当收到重新分配作业的信息时,一个令牌会从 *reassign*()里对应 receive 语句的库所转移至 toStudent!Homework 的语句,同时将向量中表示对应信道对应信息的维数的值减 1.而如果产生 spawn 操作,令牌则会通过一个转移产生一个额外的令牌,新的令牌就将用来模拟 spawn 出来的线程的运行,那么这个程序便可以被一个通信 Petri 网编码,程序的运行可以被限制下的通信 Petri 网进行模拟.

```
1 student()->
2  receive toStudents:
3  Homework->
4   Answer=do_homework(Homework)
5   toTeacher!Answer
6  end, student()
7
8 do_homework(Homework)->
9  case decompose(Homework) of
10   (Homework1,Homework2)->
11    Answer=do_homework(Homework1);
12    toTeacher!Reassign(Homework2);
13    return Answer;
14   (Homework1,Answer1)->
15    Answer=Get_Answer(do_homework(Homework1,Answer1))
16    return Answer;
17  end.
18 decompose(Homework)->
19  lock(toResource)
20  toResource!getState.
21  receive toStudents:
22   state(State)->
23    (Result,newState)=decompose_homework(Homework,State)
24  end.
25  toResource!update(newState);
26  unlock(toResource).
27  return Result.
28
29 lock(C)->
30  C!lock_req.
31  receive toStudents:
32   locked->()
33  end.
34 unlock(C)->C!lock_req.
```

Fig.2 An example of program (II)

图 2 一个程序的例子(II)

本文第 1 节介绍文中用到的一些基础知识,包括基础记号、Petri 网的概念和数据 Petri 网的概念.第 2 节介绍本文提出的模型通信 Petri 网的语法和语义.第 3 节形式化介绍异步通信程序如何编码至通信 Petri 网的模型上,并且形式化地给出通信 Petri 网上的程序验证的安全性问题.第 4 节给出在 k-型限制下的通信 Petri 网上可覆盖性问题的可判定性证明.第 5 节介绍关于该研究的一些相关工作.第 6 节总结全文.

1 预备知识

本节阐述全文用到的一些基础记号、Petri 网的概念以及数据 Petri 网的概念.

1.1 基础记号

令 N 是自然数集合,$[d]$代表$\{1,2,\ldots,d\}$的自然数集合.记 N^d 表示所有 d 维向量的集合,$\forall a\in N^d$,记 $a[i]$表示向量 a 的第 i 维的值,$a[i]\in N$.记(n,k)为组合数,其中,如果 $k>n$,则$(n,k)=0$.令 U 是一个集合,U^*表示 U 里元素组成的字符串.此外,令 $M[U]$表示由 U 产生的一个多重集合(multiset),对于集合 U 里的一个元素 u_i,定义函数$\Pi_{M[U]}(u_i)$,表示在该多重集合内 u_i 出现的次数(parikh image).特别地,当多重集合不需要特别说明时,简记函数为$\Pi_M(u_i)$.此外,对于两个多重集合 $M[U_1]$和 $M'[U_2]$,$M[U_1]+M'[U_2]$为两个多重集合的不相交并.

对于两个多重集合 $M[U_1]$和 $M'[U_2]$,$M[U_1]\geqslant M'[U_2]$当且仅当对于$\forall a\in U_2$ 满足 $\Pi_{M[U_1]}(a)\geqslant\Pi_{M'[U_2]}(a)$.同样地,对于两个元组$(p_1,p_2,\ldots,p_k)$和 $(p'_1,p'_2,\ldots,p'_k)$,定义大小关系为 $(p_1,p_2,\ldots,p_k)\geqslant(p'_1,p'_2,\ldots,p'_k)$ 当且仅当对于$\forall i\in[k]$,有 $p_i\geqslant p'_i$.这样就扩展了在元组上序的大小关系.

令$\alpha\in A^*$为 A^*上的一个元素,称α是 A 上元素产生的一个字.令 $m\in(N^k)^*$,则 $m[i](j)$表示 m 里第 i 个 k 维向量的第 j 位上的值.特别地,如果 m 中每个向量的第 j 维都相同,则简写为 $m(j)$.同时,不妨简单令 $m(j)$的操作会对任意的 $i\in N$,$m[i](j)$都会进行相应的改变.

1.2 Petri网

Petri 网[4]是由卡尔·亚当·佩特所发明的一种适合描述异步的并发的模型,有如下定义.

定义 1.2. 一个 Petri 网是一个元组 $PN=(P,T,F)$,其中,(1) P 是有限的库所集合 $P=\{p_1,p_2,\ldots,p_k\}$;(2) T 是有限的令牌集合 $T=\{t_1,t_2,\ldots,t_m\}$;(3) $F\subset(P\times T)\cup(T\times P)$是一段有向弧.

一个 Petri 网的标记是一个 k 维向量 $m\in N^k$,其中,$m[i]$表示当前状态下第 i 个库所里令牌的数量;又,Petri 网里的一条转移 t 可以用两个 k 维向量 f 和 h 来进行表示,其中,f 表示消耗令牌的数量,h 则表示产生令牌的数量.那么,一个标记 m 可以经过转移 f 到 m',当且仅当 $m'=m-f+n$.

1.3 数据Petri网

数据 Petri 网[5,6]是在 Petri 网上扩展的一个模型.一个数据 Petri 网中存在一个全序的数据集合,而数据 Petri 网中的每个令牌都被映射到了其中一个数据上.数据 Petri 网有如下的定义.

定义 1.3. 一个 k 维的数据 Petri 网是一个元组 $PDN=(P,T,F,H)$,其中,

(1) P 是有限的库所集合 $P=\{p_1,p_2,\ldots,p_k\}$;

(2) T 是有限的转移标记集合 $T=\{t_1,t_2,\ldots,t_m\}$;

(3) $F=\bigcup_{t\in T}F_t, H=\bigcup_{t\in T}H_t$,其中,$F_t,H_t\in(N^k)^*$;并且对于每个 $t\in T,F_t,H_t$ 都具有相同的长度 n;并且 $F_t(i),H_t(i)\in N^k$,其中,$i\in[n]$.

一个数据 Petri 网的标记是由$(D\times N^k)^*$的字组成的.在数据 Petri 网的任何一个状态,数据 Petri 网只有有限多个令牌,因此,对应的数据都只有有限多个.同时,因为整个数据集是全序的,不妨设为 $d_1,d_2,\ldots,d_n$,并且有 $d_1<d_2<\ldots<d_n$.一个数据 Petri 网的标记是$(d_1,v_1)(d_2,v_2)\ldots(d_n,v_n)$,其中,$v_i$ 是一个 k 维向量,用来表示对应数据为 d_i 的令牌在各个库所的数量.注意到,只需要关注数据之间的大小关系,因此可以再一步简化数据 Petri 网的标记为 $v_1v_2\ldots v_n$.令 $m=v_1v_2\ldots v_n$ 是数据 Petri 网的一个标记,则 m 的一个 0 拓展是 $m'=u_0v_1u_1v_2\ldots u_{n-1}v_nu_n$,其中,$u_i\in(N^k)^*$是一串 N^k 上的字.

令 m 和 m'是数据 Petri 网的两个标记,t 是数据 Petri 网的一个转移,并且长度为 n.如果有下列条件满足:

1) 存在 m 的一个拓展 $u_0v_1u_1v_2\ldots u_{n-1}v_nu_n$,其中,$u_i\in(N^k)^*$;

2) $v_i\geqslant F_t(i)$;

3) 存在 m'的一个 0 拓展 $u_0v_1'u_1v_2'\ldots u_{n-1}v_n'u_n$,使得 $v_i'=v_i-F_t(i)+H_t(i)$,

则称标记 m 可以触发转移 t 到 m',并记为 $m\to_t m'$.同样地,如果标记 m 经过一连串的转移序列 $t_1t_2\ldots t_n$,使其到达标记 m',便记其为 $m\to_* m'$.

问题 1.1(数据 Petri 网的可覆盖性问题). 给定一个数据 Petri 网 $PDN=(P,T,F,H)$,一个初始标记 $m_{ini}=v_1v_2\ldots v_n$ 和标记 $m'=v_1'v_2'\ldots v_n'$,则数据 Petri 网的可覆盖性问题是指是否存在一系列的转移序列 $t_1t_2\ldots t_n$,使得 $m_{ini}\to_* m\geqslant m'$.

对于该问题的研究,在文献[6]中已经有了详细的结论,这里将使用其中的结论来证明通信 Petri 网的可覆盖性的相关问题.

定理 1.1(文献[6],Theorem4.1.a). 数据 Petri 网的可覆盖性问题和终止性问题是可判定的.

2 通信 Petri 网

通信 Petri 网的定义是在普通 Petri 网的基础上增加了相互通信的能力,同时抽象了栈的概念.它通过赋予每个库所一个二维向量,用以记录收发信息的信息.可以认为:在通信 Petri 网中,每个令牌有一个二维向量的标示符,而这个标示符的全集是一个 N^2 全集.有如下的形式化定义.

定义 2.1. 一个通信 Petri 网是一个元组(tuple):$CPN=(P,T,H,In,Out,f,\alpha)$,其中,

(1) P 是有限的库所集合 $P=\{p_1,p_2,\ldots,p_k\}$;

(2) T 是有限的令牌集合 $T=\{t_1,t_2,\ldots,t_m\}$;

(3) f 是一个 $T\to N^2$ 的映射函数;

(4) $In=\bigcup_{t\in T} In_t$,其中,$In_t\in(P\times N)^*$;

(5) $Out=\bigcup_{t\in T} Out_t$,其中,$Out_t\in(P\times N)^*$;

(6) $H=\bigcup_{t\in T}(In_t,t,n,Out_t)$, 其中 $n\in[N_0]$;

(7) $\alpha\in N^{N_0}$ 是一个 N_0 维的向量.

对于 T 中的每个转移 t,定义了 In_t 与 Out_t 两个集合用以表示令牌的消耗与产生.特别地,集合 T 只是给出了转移的标示符,通信 Petri 网的一条转移是(In_t,t,n,Out_t)这样的形式,其中,n 表示可能会对全局向量α有相应的操作.与普通 Petri 网不同,通信 Petri 网里定义了一个有关转移的函数 f,将在通信 Petri 网里的转移区分成了若干不同的方式.

一个通信 Petri 网的标记(c,α)是一个库所与对应令牌的多重集合的元组串和全局向量的二元串.

$$(c,\alpha)=((p_1,M_1[N^2])(p_2,M_2[N^2])\dots(p_k,M_k[N^2]),\alpha).$$

在这个模型中,每当有一个转移 t 被触发时,会产生一些令牌的消耗和一些令牌的产生,同时也会产生一个新的标记 $(c',\alpha')=((p'_1,M'_1[N^2])(p'_2,M'_2[N^2])...(p'_k,M'_k[N^2]),\alpha')$,并将其记为$(c,\alpha)\to_t(c',\alpha')$,对于不是单个的转移触发,记$(c,\alpha)\to_*(c',\alpha')$,当且仅当存在一连串的转移序列 $t_1t_2\dots t_n$,使得:

$$(c,\alpha)\to_{t_1}(c_1,\alpha_1)\to_{t_2}(c_2,\alpha_2)\to_{t_3}\dots\to_{t_{n-1}}(c_{n-1},\alpha_{n-1})\to_{t_n}(c',\alpha').$$

在通信 Petri 网中,有一个映射转移 t 到一个二维向量上的函数 f,函数将转移分成了 5 类.

- 有一类转移与普通的 Petri 网相同,表示令牌在库所之间的转移;
- 有两类则与全局向量α有关:其中一类需要对α开始做加法,即会使α的某些分量增加,见第 1 种规则;另一类需要对α做减法,即会使α的某些分量减少,见第 2 种规则;且整个模型中,要保持α始终大于等于 0;
- 还有两类转移,对参与转移消耗的令牌产生一些形式上的改变.

形式化地,对于一个转移 t,如果:

1) 若 $f(t)=(a,0)$,其中,$a>0$ 并且 $In_t,Out_t\subseteq P\times N$,则有:

$$\frac{(In_t=(p_i,m),t,n,Out_t=(p_j,m))\in H}{((p_1,M_1[E_1])...(p_k,M_k[E_k]),\alpha)\to((p_1,M'_1[E'_1])...(p_k,M'_k[E'_k]),\alpha')},$$

其中,

(1) 如果 $k\neq i,j$,则 $M'_k[E'_k]=M_k[E_k]$;

(2) 如果 $k=i$,则 $M'_k[E'_k]=M_k[E_k]-M[E_{k1}]$,其中,$M[E_{k1}]\subseteq M_k[E_k]$,$M[E_{k1}]$是 $M_k[E_k]$的一个子集,满足 $E_{k1}\subseteq E_k$ 并且$|M[E_{k1}]|=m$;

(3) 如果 $k=j$,则 $M'_k[E'_k]=M_k[E_k]+M[E_{k1}]$;

(4) 如果 $k=n$,则当$\alpha'[k]=\alpha[k]+m\times f(t)[1]$;$k\neq n$ 时,$\alpha'[k]=\alpha[k]$.

2) 若 $f(t)=(0,a)$,其中,$a>0$ 并且 $In_t,Out_t\subseteq P\times N$,则有:

$$\frac{(In_t=(p_i,m),t,n,Out_t=(p_j,m))\in H\wedge\alpha[n]-m\,\&\,f(t)[2]\geqslant 0}{((p_1,M_1[E_1])...(p_k,M_k[E_k]),\alpha)\to((p_1,M'_1[E'_1])...(p_k,M'_k[E'_k]),\alpha')},$$

其中,

(1) 如果 $k\neq i,j$,则 $M'_k[E'_k]=M_k[E_k]$;

(2) 如果 $k=i$,则 $M'_k[E'_k]=M_k[E_k]-M[E_{k1}]$,其中,$M[E_{k1}]\subseteq M_k[E_k]$,$M[E_{k1}]$是 $M_k[E_k]$的一个子集,满足 $E_{k1}\subseteq E_k$ 并且$|M[E_{k1}]|=m$;

(3) 如果 $k=j$,则 $M'_k[E'_k]=M_k[E_k]+M[E_{k2}]$,其中,$M[E_{k2}]=\{a|a[2]=a'[2]+f(t)[2],a'\in M[E_{k1}]\}$;

(4) 如果 $k=n$,则当$\alpha'[k]=\alpha[k]-m\times f(t)[2]$;$k\neq n$ 时,$\alpha'[k]=\alpha[k]$.

3) 若 $f(t)=(0,0)$,并且 $In_t,Out_t\subseteq(P\times N)^*$,则有:

$$\frac{(In_t,t,n,Out_t)\in H}{((p_1,M_1[E_1])...(p_k,M_k[E_k]),\alpha)\to((p_1,M'_1[E'_1])...(p_k,M'_k[E'_k]),\alpha')},$$

其中,

(1) 如果$(p,_)\notin In_t,Out_t$,则 $M'_k[E'_k]=M_k[E_k]$;

(2) 如果$(p_k,m)\notin In_t$,则 $M'_k[E'_k]=M_k[E_k]-M[E_{k1}]$,其中,$M[E_{k1}]\subseteq M_k[E_k]$,$M[E_{k1}]$是 $M_k[E_k]$的一个子集,满足 $E_{k1}\subseteq E_k$,并且$|M[E_{k1}]|=m$;

(3) 如果$(p_k,m)\notin Out_t$,则 $M'_k[E'_k]=M_k[E_k]+M[E_{k2}]+M[\{a,0\}]$,其中,$M[E_{k2}]\subseteq M[E_{k1}]$,$M[E_{k2}]$是 $M[E_{k2}]$的一个子集,满足 $E_{k2}\subseteq E_{k1}$,$\{(a,0)\}=\{(a,0)|(a,\_)\in M[E_{k1}]\}$,并且两个多重集合满足$|M[E_{k2}]|+|M[(a,0)]|=m$,且$M[E_{k2}]$互不相交;

(4) $\alpha'=\alpha$.

4) $f(t)=(a,b)$,其中,$a>b>0$ 并且 $In_t,Out_t\subseteq P\times N$,则有:

$$\frac{(In_t=(p_i,m),t,n,Out_t=(p_j,m))\in H}{((p_1,M_1[E_1])...(p_k,M_k[E_k]),\alpha)\to((p_1,M'_1[E'_1])...(p_k,M'_k[E'_k]),\alpha')},$$

其中,

(1) 如果 $k\neq i,j$,则 $M'_k[E'_k]=M_k[E_k]$;

(2) 如果 $k=i$,则 $M'_k[E'_k]=M_k[E_k]-M[E_{k1}]$,其中,$M[E_{k1}]\subseteq M_k[E_k]$,$M[E_{k1}]$是 $M_k[E_k]$的一个子集,满足 $E_{k1}\subseteq E_k$ 并且$|M[E_{k1}]|=m$;

(3) 如果 $k=j$,则 $M'_k[E'_k]=M_k[E_k]+M[E_{k2}]$,其中,$M[E_{k2}]=\{a|a[1]=a'[1]+f(t)[1],a'\in M[E_{k1}]\}$;

(4) $\alpha'=\alpha$.

5) $f(t)=(a,b)$,其中,$b>a>0$ 并且 $In_t,Out_t\subseteq P\times N$,则有:

$$\frac{(In_t=(p_i,m),t,n,Out_t=(p_j,m))\in H}{((p_1,M_1[E_1])...(p_k,M_k[E_k]),\alpha)\to((p_1,M'_1[E'_1])...(p_k,M'_k[E'_k]),\alpha')},$$

其中,

(1) 如果 $k\neq i,j$,则 $M'_k[E'_k]=M_k[E_k]$;

(2) 如果 $k=i$,则 $M'_k[E'_k]=M_k[E_k]-M[E_{k1}]$,其中,$M[E_{k1}]\subseteq M_k[E_k]$,$M[E_{k1}]$是 $M_k[E_k]$的一个子集,满足 $E_{k1}\subseteq E_k$ 并且$|M[E_{k1}]|=m$;

(3) 如果 $k=j$,则 $M'_k[E'_k]=M_k[E_k]+M[E_{k2}]$,其中,$M[E_{k2}]=\{a|a[1]=a'[1]+f(t)[1],a'\in M[E_{k1}]\}$;

(4) $\alpha'=\alpha$.

直观而言,第 3 种形式的转移与普通 Petri 网的转移是相同的,都是把一些库所的令牌转移到另一些库所,并且会在某些库所生成一些新的令牌,某些令牌也会消失掉.只不过,在通信 Petri 网这个模型里,因为每个令牌是一个二维向量,因此可能会有无限多种令牌,因而需要用多重集合的方式来进行描述.

前两种形式的转移会对全局向量α:第 1 种形式下,在通信 Petri 网上的转移会增加整个模型全局向量α的值;而在第 2 种形式下,转移的发生同时需要向量α保持一定的大小,使得减小后的向量α'依旧是一个在 N^{N_0} 的向量,同时使得参与转移的令牌的第 2 维数值加 1.而后两种转移则会对令牌的第 1 维数值产生影响:第 4 种转移会减少令牌的第 1 维向量,而第 5 种转移则会增加令牌的第 1 维向量.

在关于转移的形式化定义里,有意省略了一些除了第 3 种形式下的转移,包括限制住了这些转移的 In_t 与 Out_t 以及对于普通令牌迁移的情况.在上述定义中,除了第 3 种转移以外,其余 4 种转移每次相当于只能从一个库所里消耗令牌,再在一个库所里产生新的令牌.第 3 种转移则并没有作这方面的限制,并且我们可以假设令牌的转移和生成在第 3 种规则下是固定的,即:对于第 3 类规则,比如$((p,1),t,1,(q,1)(m,2))$,我们在库所 p 中消耗掉一个(1,2)的令牌,可以假定这个令牌转移到了 m 库所里,而在 q 和 m 库所里则各自生成了一个(1,0)的令牌.由于有第 3 种转移的能力,因此这种省略并不会影响模型本身的能力,并且有助于理解这个模型.事实上,考虑一个夹杂了普通令牌迁移和多个 In_t 与 Out_t 的转移 t 的情况,只需要注意:由于一个通信 Petri 网里的转移(In_t,t,n,Out_t)只有有限多个,因此可以定义出一个新的通信 Petri 网,添加一些新的库所与转移,可以将原来复杂的转移 t 分开来变成若干步线性的转移,分别处理普遍令牌的迁移和 In_t 与 Out_t 的转移,使得这个新模型里所有的转移都符合上面所定义的转移.

特别地,继续来限定一下模型,即使令 f 为 $T\to\{(0,0),(1,0),(0,1),(2,1),(1,2)\}$,所定义出来的模型也有着相同的能力.原因也很简单,就是因为转移仅仅只有有限多个.对于初始的通信 Petri 网,假设存在 1 个转移 t 使得 $f(t)=(m,0)$,便可以在新的通信 Petri 网上针对这个转移 t 构建 m 个辅助转移的转移 $t_1,t_2,\ldots,t_m$ 与辅助的库所 $f(t_i)=(1,0)$,使得原来的通信 Petri 网上转移 t 被触发当且仅当新构建的通信 Petri 网 $t_1,t_2,\ldots,t_m$ 这一系列的转移.因此在接下来的讨论中,要使得通信 Petri 网里映射转移的函数 f 为 $T\to\{(0,0),(1,0),(0,1),(2,1),(1,2)\}$.

3 通信 Petri 网与异步通信程序

现在来介绍通信 Petri 网与异步通信程序之间的联系.对于我们所研究的程序,它有如下特点:(1) 函数存在没有记忆能力的函数调用;(2) 存在 spawn 操作,也即开一个新的线程;(3) 程序之间存在异步通信,并且用来通信的信道和信息类型都只有有限个.其中,没有记忆能力的函数调用对于调用次数不敏感,比如一个函数 A 自我调用,A 调用 k 次和 A 调用 m 次在程序栈的表示上是没有区别的.这里,我们以文前中给出的程序为例来加以说明.我们考虑 *student*()函数,注意到,其两个 *student*()进程里不会产生内部的联系,因此最后一句 *student*()对自己的调用次数不会在程序栈上有不同的表示.在 *student*()函数里,其对自己调用的次数并不敏感.注意到:因为每一个线程都有一个栈,而多栈的系统图灵等价,因此没有记忆能力的栈是对原来栈的一种抽象,这样的抽象没有实现限定栈深度,因而要比 k-型更加一般化.

在通信 Petri 网中,将每个库所视作程序中的某个状态,而一个令牌代表了程序的一个线程,令牌的转移则表示通过程序的执行,每个线程会达到程序中的不同状态.这样,新令牌的产生便能表示新开线程操作的产生,而程序本身的运行便能体现在整个模型里面.比如在上例中,*setup_network*()中第 1 行的 spawn 函数便可认为是在通信 Petri 网中新增加了一个令牌.还需要说明的是如何编码整个通信,这一点我们通过通信 Petri 网的全局向量 α来完成.在通信 Petri 网的转移中,使得 $f(t)=(0,1)$的转移则可以理解为一个程序接受一个消息,收取信息的时候便会在第 2 维向量加 1,也就是一个令牌所对应的第 2 维向量存储了收取信息的数量.同样,以文前的例子来进行说明.在本文开始部分,我们已经假设共有 k 种信息,图 1、图 2 中给出的程序一共有 3 个信道,因此我们要用一个 $3k$ 维的向量来进行模拟,并且假设 3 个信道的顺序是 toResource,toStudents 和 toTeacher,则第 $i\times k+j$ 维上的数值就表示为第$(i+1)$个信道上第 j 种信息的数量.当 *student*()函数企图从 toStudents 上接受第 1 种信息的时候,通信 Petri 网就可以检测第 $k+1$ 维上的值是否为 0 以决定是否能够收到相关的信息.如果大于 0,则通过将这一维向量减 1 的方式表示该信息已被接受;发送信息可以作类似的考虑.由于只需要在对应维度上做加法,因此不会使得程序无法进行.此外,注意到,信道上的内容是全局的,因此在通信 Petri 网中,我们使用全局向量α来模拟.而对于每个令牌来说,每当进行一次程序调用时,自身的第 1 维向量便会加 1 进行存储,结束调用时,第 1 维向量减 1.这里需要考虑如何模拟程序的运行,我们将令牌视作一个线程,而程序的模拟则通过令牌在库所间的游走来加以表示.这里,为了考虑函数调用以及通信的能力,我们在每个令牌里记录了调用层数与收取信息的数量.之前已经提到,第 2 维记录了收取信息的数量,而令牌的第 1 维则记录了调用的层数.在这样的编码下,一个满足上述条件的程序便可以编码到通信 Petri 网上,并且不妨限定每次只有一个消息的传送和接收.

形式化地来说,对于一段异步通信程序,假设一共有 $A_1,A_2,\ldots,A_n$ 个函数,其中,每个函数 A_i 共有 a_i 行语句.接下来构造如下 $\sum na_i$ 个库所,记为 A_{ijk},其中,$i,j\in[n],k\in[a_j]$.库所 A_{ijk} 的意思是指,函数 A_i 调用函数 A_j 进行至第 k 行语句的状态.其次,假设程序共有 k 个信道和 m 种信息,则添加一个 $k\times m$ 维的向量α,用以表示信息在进程间的传递,初始状态下,α为零向量.每当向第 i 个信道发送第 j 种信息时,则$\alpha[(i-1)\times m+j]$加 1;而需要从第 i 个信道接受第 j 种信息时,$\alpha[(i-1)\times m+j]$减 1.在经过这样的编码后,程序已将程序交互上信息的存储与控制流所能达到的状态编码到了通信 Petri 网上的状态和全局向量上.接下来将考虑程序的动作如何编码至转移上去.

令通信 Petri 网的转移根据之前的介绍分为规则 1~规则 5.对于原程序的一条语句,如果产生了信息的发送,则添加一条规则 1,令牌在转移的同时,使全局向量的对应分量加 1;如果需要接受信息,则添加一条规则 2,令牌在能转移的同时,使全局向量的对应分量减 1,并且使得转移的令牌的第 2 维值加 1;当程序产生函数调用的时候,添加一条规则 4,除了令牌的转移以外,令转移的令牌的第 1 维值加 1,与状态一起模拟该线程的程序栈;而程

序结束函数调用的时候,则添加一条规则 5,令转移的令牌的第 1 维值减 1;每当需要产生一个新的线程的时候,产生一条规则 3,使得产生一个新的第 1 维值相同,第 2 维值为 0 的令牌;其余情况只需要根据程序的正常执行产生对应令牌的相应转移即可.这样,便把一个异步通信程序编码到了通信 Petri 网上,并且模型上的一个令牌便代表了一个线程,令牌在模型里的转移则代表了程序的控制流,其中,令牌的第 1 维和令牌所处的状态则刻画出一种受限制的程序栈,也就是没有记忆能力的程序栈.

而对于异步通信程序的研究,验证方法关注的兴趣在于是否存在一些不期望见到的程序运行状态.比如,假设以行号来表示标记程序运行到的状态,则是否会存在两个学生在调用分析作业的时候同时运行到行号 20 的状态产生,也就是说,是否会发生死锁的行为,亦即一个不期望达到的状态,验证方法便是去检验这样的状态是否会发生.对于这一点,接下来会加以说明,这个问题可以通过研究通信 Petri 网的可覆盖性问题来得到解决.

问题 3.1. 给定一个通信 Petri 网 k-CPN=(P,T,H,In,Out,f,α)和它的一个初始标记(c_{ini},α_{ini})=((p_1,$M_1[N^2]$)(p_2,$M_2[N^2]$)…(p_k,$M_k[N^2]$),α_{ini})以及一个目标标记 $(c,\alpha)=((p'_1,M'_1[N^2])(p'_2,M'_2[N^2])...(p'_k,M'_k[N^2]),\alpha)$. 如果存在一个转移序列 $t_1t_2…t_n$ 使得(c_{ini},α_{ini})$\rightarrow*$(c',α'),其中满足 $c'\geqslant c$,则称(CPN,c_{ini})能覆盖住 c.因此,通信 Petri 网的可覆盖性问题便是是否存在这样一个满足要求的 c',使得(c_{ini},α_{ini})$\rightarrow*$(c',α'),其中满足 $c'\geqslant c$.

事实上,基于之前给出的异步通信程序到通信 Petri 网的编码不难发现:程序会达到某个不期望的“坏”状态,当且仅当编码后的通信 Petri 网会达到一个状态可以覆盖掉这个不期望的“坏”状态的标记.

需要注意的是:在通信不受限制的情况下,通信 Petri 网的可覆盖性问题是不可判定的.

接下来将在通信 Petri 网上增加一种比较自然的限制,对于每个令牌来说,它所代表的二维向量的第 2 维会被一个预先给定的固定常数 k 所绑定.这种限制对于这种模型来说是合理的,因为考虑到在异步通信程序中,很多程序尽管会有不受次数限制的消息发送与接收,然而对于每个线程来说,它接受信息的次数是被绑定的.比如,在前文给出的程序中,对于每个函数来说,它只会接受有限多次信息,因而这样的程序便满足了这个限制.

定义 3.2. 一个 k-型限制下的通信 Petri 网是一个通信 Petri 网元组 k-CPN=(P,T,H,In,Out,f,α),设定由初始标记 c_{ini} 出发,定义集合 $Reach$={c|(c_{ini},α_{ini})$\rightarrow*$(c',α')},使得 $Reach\geqslant\{(p_i,(0,k))\}$不成立.

通信 Petri 网可以通过修改转移的语义来完成这一限制,事实上,只需要修改满足 $f(t)$=(0,1)的转移 t.

$$\frac{(In_t=(p_i,m),t,n,Out_t=(p_j,m))\in H\wedge\alpha[n]-m\ \&\ f(t)[2]\geqslant 0}{((p_1,M_1[E_1])...(p_k,M_k[E_k]),\alpha)\rightarrow((p_1,M'_1[E'_1])...(p_k,M'_k[E'_k]),\alpha')},$$

其中,(1) 若 $k\neq i,j$,则 $M'_k[E'_k]=M_k[E_k]$; (2) 若 $k=i$,则 $M'_k[E'_k]=M_k[E_k]-M[E_{k1}]$, 其中,$M[E_{k1}]\subseteq M_k[E_k]$,$|M[E_{k1}]|=m$;并且对于 $M[E_{k1}]$中的任意一个元素 a,$a[2]\leqslant k-1$;(3) 若 $k=j$,则 $M'_k[E'_k]=M_k[E_k]+M[E_{k2}]$, 其中,$M[E_{k2}]$={$a$|$a[2]$=$a'[2]+f(t)[2]$,$a'\in M[E_{k1}]$},并且 $a'[n]=a[n]-m\times f(t)[2]$.

在本节的最后,我们将用一个例子来简单阐述程序到模型的转换,如图 3 所示.

```
1 main()->A()
2     return;
3
4 A()->
5   spawn B()
6   receive C:
7     ready->C!ready
8   end;
9
10 B()->
11   fun()->C!ready
12   receive C:
13     ready->()
14   end
```

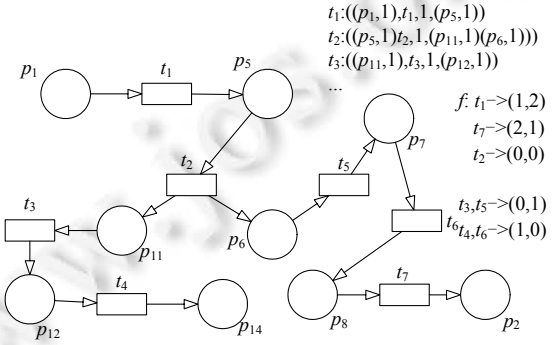

Fig.3 An example transition

图 3 一个转换的例子

图 3 中左边是一个简单的程序,A 和 B 两者的区别是,一个是先发信息再接收信息,另一个是先接收信息再

发信息.图 3 中右图则是通信 Petri 网的一个编码,注意到,只有 C 一个信道和 *ready* 一个信息,该全局变量α是一维的.整个编码与程序的控制流是非常相似的.这里,库所的名字与左边程序的行号直接挂钩.另外,如果一个令牌在库所 p_i 里,则表示这个令牌所代表的进程运行到了第 i 行,则这个程序能否 return 便取决于库所 p_2 里是否能有令牌.

4 *k*-型限制下的通信 Petri 网到数据 Petri 网的编码

本节将给出 *k*-型限制下的通信 Petri 网的可覆盖性可判定性的证明,证明的核心思路是:通过说明在 *k*-型限制下的通信 Petri 网中每个令牌所对应的一个二维向量可以认为是在全序集上的一个映射,从而证明受限的通信 Petri 网的可覆盖性问题可以规约到数据 Petri 网的可覆盖性问题上,而数据 Petri 网的可覆盖性问题可以通过规约至良序转移系统(WSTS)的可覆盖性问题上,从而证明其是可判定的[6].

直观上来理解,数据 Petri 网的拓展是将令牌直接绑定到了一个数据集上的一个数据,而在每次转移的过程中,令牌绑定的数据可能会发生变化,但这个变化是由规则绑定的,所以尽管有无穷多种令牌,但是由于有限条规则的限制,使得数据的变化是有限种.对于通信 Petri 网上性质的证明正是利用了这一点,使得在 *k*-型限制下的通信 Petri 网能够规约到数据 Petri 网的问题上.

给定一个 *k*-型限制下的通信 Petri 网 $k\text{-}CPN=(P,T,H,In,Out,f,\alpha)$和它的一个初始标记$(c_{ini},\alpha_{ini})=((p_1,M_1[N^2])(p_2,M_2[N^2])\dots(p_k,M_k[N^2]),\alpha_{ini})$以及一个目标标记 $(c,\alpha)=((p'_1,M'_1[N^2])(p'_2,M'_2[N^2])...(p'_k,M'_k[N^2]),\alpha)$, 其转换为一个数据 Petri 网的可覆盖性问题说明如下.

第 1 步,将 *k-CPN* 的全局向量α编码进库所里.额外地,令 d 个库所 $p_{m1},p_{m2},\dots,p_{md}$,其中,$d$ 为向量α的维度,每当存在一条需要对α做加法的转移 t,即(In_t,t,n,Out_t),则将在转移 t 出发后,在 p_{mn} 里产生一个向量为(0,0)的令牌;每当存在一条需要对α做减法的转移 t,即(In_t,t,n,Out_t),则将在转移 t 出发后,在 p_{mn} 里消耗一个向量为(0,0)的令牌.在这些额外的 d 个库所中,由于是被用来相当于当成信道存储程序发送的信息,因此特别令在这些额外的 d 个库所中的令牌所代表的向量都是一样的.所以通过额外的 d 个库所的编码,便将全局向量α编码进了库所.

第 2 步,将每个令牌所代表的第 2 维向量编码进库所里.这一点由于有 *k*-型限制,可如下实现.

对于原先通信 Petri 网的每一个库所 p_i,定义对应的 k 个库所 $p_{i1},p_{i2},\dots,p_{ik}$,其中,$p_{ij}$ 存储在原本通信 Petri 网的库所 p_i 中存储的第 2 维向量 j 的令牌.在这样的情形下,新的模型里的库所数量是原来通信 Petri 网的 k 倍.因为库所被分离了,所以需要对现有的转移进行一些修改.下面根据原来的 5 类转移一一进行说明:第 1、第 3、第 4 和第 5 种转移是比较容易说明的,事实上,由于其不会修改里面库所中令牌第 2 维向量的值,因此对于一个需要消耗在 $p_1,p_2,\dots,p_s$ 库所里的令牌并且在 $p'_1,p'_2,...,p'_t$ 库所里生成对应令牌的转移,只需要添加类似的转移,使得转移是需要消耗在 $p_{1i_1},p_{2i_2},...,p_{si_s}$ 库所里的令牌并且在 $p'_{1i_1},p'_{2i_2},...,p'_{ti_t}$ 库所里生成对应令牌,其中,$i_m,j_n\in[k]$.我们注意到:在这种情形下,对于原先的一条转移,由于第 2 维向量并没有发生变化,因此为了在将第 2 维向量编码进状态的数据 Petri 网中模拟该转移,我们需要考虑到所有的情况.简单来说,比如:如果要消耗 p_1 库所里的 m_1 个令牌,注意到,这 m_1 个令牌的第 2 维向量可能是 0~k 中的任一数字,也就是说一共有 $(k+1)^{m_1}$ 种情况,对每个库所消耗的令牌进行考虑,假设在 p_i 库所里需要消耗 m_i 个令牌,则一共会有 $(k+1)^{\sum_{i=1}^{s}m_i}$ 种不同的可能,也就是说需要在数据 Petri 网中添加这么多种的转移.

现在来说明第 2 种转移的修改.注意到:这种类型的转移每次被触发时,一个令牌的第 2 维向量只需要加 1,因此这种转移的添加也比较方便.对于属于第 2 种转移的 t,即$((p_i,m),t,(p_j,m))$这样一个转移,只需要同样构造$((p_{ik},m),t_k,(p_{j(k+1)},m))$的 k 条转移即可.图 4 给出了一个在 2-型下的通信 Petri 网的转移的修改方式.直观上来理解,就是由于第 2 维向量是有限制的,因此可以在状态里控制第 2 维向量.

在对上述的转移进行修改以后,编码已经将 *k*-型限制下的通信 Petri 网的每一个令牌的第 2 维向量编码进库所里,所以从现在开始,将限定这里的每个令牌对应的只是一个自然数值.注意到,N 是一个全序集,因此可以采用数据 Petri 网使用的标记方法来表示现在一个通信 Petri 网下的标记.

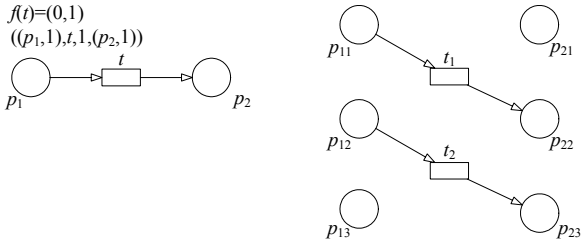

Fig.4 An example of 2-shaped C-Petri net transition

图 4 一个 2-型限制下的通信 Petri 网的转移修改

最后将证明,经过了如此变换的 k-型限制下的通信 Petri 网是一个数据 Petri 网.事实上,只需要说明每一个转移都可以表示成固定常数的一个 N^k 上的列即可.由于每次通信 Petri 网上的操作至多只需要对令牌所表示的向量完成+1,–1 的操作,也就是说,每个通信 Petri 网下涉及到 n 个令牌的转移至多只需写成 n 个 N^k 上的列即可,因此,原本 k-型限制下的通信 Petri 网已经被编码成了一个数据 Petri 网,并且保证了两个模型所能产生的路径是相同的.现在来形式化整个过程.

定义 4.1. 给定一个 k-型限制下的通信 Petri 网 $k\text{-}CPN=(P,T,H,In,Out,f,\alpha)$,其中,

1) P 是一个有限的元素集合 $P=\{p_1,p_2,\ldots,p_m\}$;

2) T 是一个有限的元素集合 $T=\{t_1,t_2,\ldots,t_n\}$;

3) $\alpha\in N^d$,

则 $D(k\text{-}CPN)$所对应的一个 $k\times m+d$ 维的数据 Petri 网是一个元组 $PDN=(P',T',F,H)$,其中,

1) $P'=\{P_{11},P_{12},\ldots,P_{1k},P_{21},\ldots,P_{m1},P_{m2},P_{mk},P_{msg1},P_{msg2},P_{msgd}\}$;

2) 对于 $k\text{-}CPN$ 中的每个转移 t:

i) 如果 $f(t)=(1,0),(In_t=(p_i,1),t,n,Out_t=(p_j,1))\in H$,则 $\{t_1,t_2,\ldots,t_k\}\subseteq T'$,并且 $F_{tl},H_{tl}\in N^{k\times m+d}$,其中,$F_{tl}((i-1)\times k+l)=1$,其余为 0;$H_{tl}((j-1)\times k+l)=1,H_{tl}(m\times k+n)=1$,其余为 0.

ii) 如果 $f(t)=(0,1),(In_t=(p_i,1),t,n,Out_t=(p_j,1))\in H$,则 $\{t_1,t_2,\ldots,t_k\}\subseteq T'$,并且 $F_{tl},H_{tl}\in(N^{k\times m+d})^2$,其中,$F_{tl}[1]((i-1)\times k+l)=1,H_{tl}[1](m\times k+n)=1$,其余为 0;$H_{tl}[2]((j-1)\times k+l)=1$,其余为 0.

iii) 如果 $f(t)=(0,0)$,并且 $(In_t=(p_{i1},1)(p_{i2},1)\ldots(p_{ip},1),t,n,Out_t=(p_{j1},a_1)(p_{j2},a_2)\ldots(p_{jq},a_q))\in H$,则 $\{t_{11},\ldots,t_{12}^{p-1},t_{21},\ldots,t_{k2}^{p-1}\}\in T'$, 并且对于 $x\in[k],y\in\left[\sum_{i=0}^{n-1}(p,i),\sum_{i=0}^{n}(p,i)\right),F_{txy},H_{txy}\in(N^{k\times m+d})^n$, 其中,$p_{i1},p_{i2},\ldots,p_{ip}$ 重新记标号为 $P_{11},P_{12},\ldots,P_{1b1},P_{2b2},\ldots,P_{nbn}$.其中,$\sum_{i=1}^{n}b_i=p$, 则 $F_{txy}[c]((i_d-1)\times k+x)=1,p_{id}$ 被重新记为 p_{ce},其余为 0;$H_{txy}[c]((j_d-1)\times k+x)=1,H_{txy}[c]((j_d-1)\times k+1)=a_d-1$,其余为 0.

iv) 如果 $f(t)=(2,1),(In_t=(p_i,1),t,n,Out_t=(p_j,1))\in H$,则 $\{t_1,t_2,\ldots,t_k\}\subseteq T'$,并且 $F_{tl},H_{tl}\in(N^{k\times m+d})^2$,其中,$F_{tl}[2]((i-1)\times k+l)=1$,其余为 0;$H_{tl}[1]((j-1)\times k+l)=1$,其余为 0.

v) 如果 $f(t)=(1,2),(In_t=(p_i,1),t,n,Out_t=(p_j,1))\in H$,则 $\{t_1,t_2,\ldots,t_k\}\subseteq T'$,并且 $F_{tl},H_{tl}\in(N^{k\times m+d})^2$,其中,$F_{tl}[1]((i-1)\times k+l)=1$,其余为 0;$H_{tl}[2]((j-1)\times k+l)=1$,其余为 0.

vi) $F_{d1},H_{d1},F_{d2},H_{d2}\in(N^{k\times m+d})^2$,其中,$F_{d1}[2](m\times k+d)=1$,其余为 0;$H_{d1}[1](m\times k+d)=1$,其余为 0;$F_{d2}[1](m\times k+d)=1$,其余为 0;$H_{d2}[2](m\times k+d)=1$,其余为 0.

定义 4.2. 定义函数 $m:(P,M_1[N^2])^*\times N^d\to(N^{k\times m+d})^*$和函数 $h:(P\times M_1[N^2])^*\to N$,其中,

1) $h(c)=\max\ d,\prod_{M_i}(d,\_)>0$;

2) $m(c,\alpha)\in(N^{m\times k+d})^{k(c)}$,其中,$m(c,\alpha)[i]((k\times(a-1)+j)=d$,如果在 c 中 $\prod_{M_a}((i,j))=d$.特别地,

$$\sum m(c,\alpha)[i](k\times m+j)=\alpha[j].$$

我们对这两个函数定义进行一定的说明.h 函数实际上是一个作用在一个通信 Petri 网的标记上的函数,返回在该通信 Petri 网当前标记下令牌中第 1 维向量的最大值.这个函数的作用在于服务接下来的 m 函数.实际上,m 函数是一个通信 Petri 网标记转换成将令牌第 2 维编码进状态的数据 Petri 网标记的转换函数.也就是说,如果(c,α)是 k-型限制下的通信 Petri 网的一个标记,则 $m(c,\alpha)$是 $D(k\text{-}CPN)$上对应的一个标记.并且由 m 函数的定义,我们不难定义出 m 函数的逆函数 $m^{-1}:(N^{k\times m+d})^*\to(P,M_1[N^2])^*\times N^d$.关于这两个函数,我们有如下的性质.

命题 4.1. 对于函数 h,m 以及其逆函数 m^{-1},我们有如下性质.

1. 对任意 $c,c'\in(P,M_1[N^2])^*$,如果有 $c\geqslant c'$,我们有 $h(c)\geqslant h(c')$;
2. (c,α)是 k-型限制下的通信 Petri 网的一个标记,则 $m(c,\alpha)$是 $D(k\text{-}CPN)$上对应的一个标记;
3. 对任意 $d,d'\in(N^{k\times m+d})^*$,并且 $m^{-1}(d)=(c,\alpha),m^{-1}(d')=(c',\alpha')$,若$(c,\alpha)=(c',\alpha')$,则对$\forall i\in N,\forall j\in N$,有:
$$d[i](j)=d'[i](j);$$
4. 对任意$(c,\alpha),(c',\alpha')\in(P,M_1[N^2])^*\times N^d,c\geqslant c'$,当且仅当 $m(c,\alpha)[i](j)\geqslant m(c',\alpha')[i](j)$,其中,$i\in[h(c)],j\in[k\times m]$.

证明:从函数 h,m 和函数 m^{-1} 的定义不难得到.

1. 由函数 h 的定义立即得出;
2. 由函数 m 的定义及 $D(k\text{-}CPN)$的定义立即得出;
3. 因为有 $m^{-1}(d)=m^{-1}(d')$,因此有 $c=c'$,所以对于任意的 $i+k\times j\in[k\times m],\prod_{M_i}((a,j))=\prod_{M_i'}((a,j))$ 对任意 a 均成立.因而有 $d[a][i\times k+j]=d'[a][i\times k+j]$;
4. 由 $c\geqslant c'$可知:对于任意的$(i,j)\in N^2$,有$\prod_{M_a}((i,j))\geqslant\prod_{M_a'}((i,j))$,其中,$a\in[m],j\in[k]$,因此对于任意的 $i_0\in[h(c)],j_0\in[k\times m]$,有 $m(c,\alpha)[i_0](j_0)\geqslant m(c',\alpha')[i_0](j_0)$;反之,同理.因而结论成立. □

接下来将说明 k-型限制下的通信 Petri 网 $k\text{-}CPN$ 和一个对应的数据 Petri 网 $D(k\text{-}CPN)$之间的关系.

引理 4.1. 对于一个 k-型限制下的通信 Petri 网 $k\text{-}CPN$ 和一个对应的数据 Petri 网 $D(k\text{-}CPN)$以及 $k\text{-}CPN$ 的两个标记$(c,\alpha),(c',\alpha')$,$(c,\alpha)\to(c',\alpha')$当且仅当在 $D(k\text{-}CPN)$中 $m(c,\alpha)\to m(c',\alpha')$.

证明:只需证明对于在 $k\text{-}CPN$ 上$(c,\alpha)\to(c',\alpha')$,当且仅当在 $D(k\text{-}CPN)$上 $d\to d',m(c,\alpha)=d,m(c',\alpha')=d'$.

($\Rightarrow$)的证明:

假设$(c,\alpha)\to_t(c',\alpha')$,并且

$$(c,\alpha)=((p_1,M_1[N^2])(p_2,M_2[N^2])...(p_k,M_k[N^2]),\alpha),(c',\alpha')=((p_1,M_1'[N^2])(p_2,M_2'[N^2])...(p_k,M_k'[N^2]),\alpha').$$

接下来将说明存在 $D(k\text{-}CPN)$的一条转移 t',使得 $m(c,\alpha)\to_{t'}m(c,\alpha)'$,并且 $m(c,\alpha)'=m(c',\alpha')$.其中,$m(c,\alpha)'$表示为 $m(c,\alpha)$作转移 t'后新的数据 Petri 网的标记.这里,以第 1 类、第 3 类转移作为说明,其余 3 类转移的说明是类似的,这里不再重复.

1) 如果 $f(t)=(1,0),(In_t=(p_i,1),t,n,Out_t=(p_j,1))\in H$,则存在一个大小为 1 的多重集合 $M[N^2]$,使得:
$$M_{j'}[N^2]=M_j[N^2]+M[N^2],M_{i'}[N^2]=M_i[N^2]+M[N^2],\alpha'[n]=\alpha[n]+1.$$

令 $M[N^2]=\{(b,e)\}$.由 $D(k\text{-}CPN)$的构造可知,存在 $F_{t'},H_{t'}\in N^{k\times m+d},F_{t'}((i-1)\times k+e)=1,H_{t'}((j-1)\times k+e)=1,H_{t'}(m\times k+n)=1$.在 $m(c,\alpha)$中考察第 b 个向量 $m(c,\alpha)[b]$,注意到$(b,e)\in M_i[N^2]$,因此,由 m 的规则,$m(c,\alpha)[b](k\times(i-1)+e)>0$.因此,$m(c,\alpha)$可作该条转移 t',使得 $m(c,\alpha)'$满足:

$$\begin{aligned}&m(c,\alpha)'[b](k\times(i-1)+e)=m(c,\alpha)[b](k\times(i-1)+e)-1,\\&m(c,\alpha)'[b](k\times(j-1)+e)=m(c,\alpha)[b](k\times(j-1)+e)+1,\\&m(c,\alpha)'[b](k\times m+n)=m(c,\alpha)[b](k\times m+n)+1.\end{aligned}$$

其余位置与 $m(c,\alpha)$相等.

另一边,由于 $\alpha'[n]=\alpha[n]+1,\prod_{M_i'}(b,e)=\prod_{M_i}(b,e)-1,\prod_{M_j'}(b,e)=\prod_{M_j}(b,e)+1$, 因此,由定义 $m(c',\alpha')$,满足:

$$\begin{aligned}&m(c',\alpha')[b](k\times(i-1)+e)=m(c,\alpha)[b](k\times(i-1)+e)-1,\\&m(c',\alpha')[b](k\times(j-1)+e)=m(c,\alpha)[b](k\times(j-1)+e)+1,\\&m(c',\alpha')[b](k\times m+n)=m(c,\alpha)[b](k\times m)+n)+1.\end{aligned}$$

其余位置与 $m(c,\alpha)$相等.因此有 $m(c,\alpha)'=m(c',\alpha')$.

2) 如果 $f(t)=(0,0)$,并且$(In_t=(p_{i1},1)\ldots(p_{ip},1),t,n,Out_t=(p_{j1},a_1)\ldots(p_{jq},a_q))\in H$,则存在 p 个大小为 1 的多重集合 $M_u[N^2]$. 令 $M_u[N^2]=\{(b_u,e_u)\}$, 使得 $M_{iu'}[N^2]=M_{iu}[N^2]-M_u[N^2]M_{jk'}[N^2]=M_{jk}[N^2]+M_u[N^2]+M[\{(b_u,0)\}]$, 并且 $|M[\{(b_u,0)\}]|=a_k-1$.不妨令 b_i 有 k 个不同的值.由 $D(k\text{-}CPN)$的构造可知,存在 $F_t,H_t\in(N^{k\times m+d})^k$,使得 $F_t[g]((i-1)\times k+e)=1$, $H_t[g]((z-1)\times k+e)=1,H_t[g]((i-1)\times k+1)=a_i-1$.其中,$i,g$ 满足 b_i 在这 p 个值里排在第 g 小的位置,$z=j_s$.由于 $h(c)\geqslant\max b_i$,因此在 $m(c,\alpha)$中存在 k 个位置上的向量,使得 $m(c,\alpha)[x_i]\geqslant F_t[i]$,于是,$m(c,\alpha)$可作该条转移 t',使 $m(c,\alpha)'$满足:

$$m(c,\alpha)'[x_i]((i-1)\times k+e_i)=m(c,\alpha)[x_i]((i-1)\times k+e_i)-1,$$
$$m(c,\alpha)[x_i]((k_i-1)\times k+e_i)=m(c,\alpha)[x_i]((k_i-1)\times k+e_i)+1,$$
$$m(c,\alpha)'[x_i]((k_i-1)\times k+1)=m(c,\alpha)'[x_i]((k_i-1)\times k+e_i)+a_i-1.$$

其余与 $m(c,\alpha)$相等.

另一方面,由 $M_{jk'}[N^2]$的变化及 m 函数的定义可知:

$$m(c',\alpha')[x_i]((i-1)\times k+e_i)=m(c,\alpha)[x_i]((i-1)\times k+e_i)-1,$$
$$m(c',\alpha')[x_i]((k_i-1)\times k+e_i)=m(c,\alpha)[x_i]((k_i-1)\times k+e_i)+1,$$
$$m(c',\alpha')[x_i]((k_i-1)\times k+1=m(c',\alpha')[x_i]((k_i-1)\times k+e_i)+a_i-1.$$

其余位置与 $m(c,\alpha)$相等.

因此有 $m(c,\alpha)'=m(c',\alpha')$.因此由归纳假设,结论成立.

(⇐)的证明:

反过来的证明是相对比较容易的,只要注意到:对于 $D(k\text{-}CPN)$的每个转移 t,都是由原先通信 Petri 网上的一个转移 t 生成而来.因此,如果在 $D(k\text{-}CPN)$上有 $d\to d'$,在 $k\text{-}CPN$ 上即有 $m^{-1}(d)\to m^{-1}(d')$. □

因此,k-型下的通信 Petri 网可覆盖性问题规约到了数据 Petri 网的可覆盖性问题上去.由引理 4.1、定理 1.1,便得到了下面的定理.

定理 4.1. k-型限制下的通信 Petri 网的可覆盖性问题是可判定的.

5 相关工作

异步通信程序分析理论与方法是最近几年计算机理论和程序理论研究者关注的热点,研究者们针对上述 5 个无限状态因素进行限制和抽象,目前主要的研究分支和研究结果如下.

- 限制递归.

即:每个进程不允许函数调用,同时假设各个进程通过无界容量的无序缓存进行通信.这样的系统是异步通信系统,其可覆盖性可通过规约至 Petri 网的可覆盖性,因此是可判定的[7].基于该模型,研究者们设计了一个 Erlang 语言的分析器 Soter[8].文献[9]中,研究者对递归进行了 k-型的限制,在这样的情况下,同时对输入通信的次数进行有界限制,系统的可覆盖性依然可判定,且表达能力严格超过 Petri 网,达到 Tower 完备[9].

- 限制并发.

假设只有一个处理器执行程序,且线程执行时不允许被挂起,也即假设异步并发程序可以线性化.在这种限制下,只有一个程序栈就足够了.即使这样的模型,其表达能力依然很强,是否是图灵完备目前仍然是一个开放问题.文献[2]在上述限制的同时,对通信也进行了限制——只有在栈空的时候才能发生通信,这样的程序模型的状态可达性(EXPSPACE 难)[2]和可覆盖性[10]是可判定的.

- 限制通信.

最极端的限制方法是不允许不同线程间进行通信,这样的程序也即异步程序.在这样的限制下,模型的可覆盖性可以通过规约至 Petri 网的可覆盖性得到可判定的结论[11],其活性(liveness)可以通过规约至 Petri 网的可达性,因此也是可判定的[12].文献[13]详尽地列出了在不允许通信的情况下,各种程序模型子集的判定性结论,对应于不同的程序类型.对于这样的程序模型系统,研究者开发了一个程序分析器原型[14].最新的研究扩展了不允许线程之间的通信,得到了一个限制性比较强的可判定性结论[6].

还有一类型研究是为了对异步的 Web 程序进行分析,假设系统有一个主线程以及无界数量个同类型的辅线程,通信只发生在主辅线程之间,辅线程之间没有通信.因此,辅线程可以参数化.这样的模型叫做参数化异步通信系统.文献[15]首次提出了这样的系统,并给出这一系统的可覆盖性的复杂性上界是 EXPSPACE 的.最后,可覆盖性被确定为 PSPACE 完备[16].在最新的研究成果中,该系统的活性上界被证明是 NEXPTIME 的[17].同时,一些更为扩展的模型的可判定性结论也被提了出来[18].

此外,计算机理论界的一些研究也对本研究具有很强的借鉴作用,比如最新的关于下推向量加法系统(pushdown vector addition system)[19],其实际表达是单进程异步通信程序模型的一个超集,它的可判定结论可以直接影响后者的判定性结论.目前的研究成果是下推向量加法系统在一维情况下可覆盖性可判定[20],这一结论尚不能指导异步通信程序的理论模型(因为程序模型不可能只有一维向量).良结构传递系统[21]及其扩展[10,22]也可以对程序模型的判定性结论起到指导以及证明作用,其优点是可得到通用的证明过程,但不足之处是很难得到复杂性结论以及可实现的算法,因此不太被程序语言理论界所认可.

由这些研究可知,目前大多数的可判定性结论都是规约至 Petri 网的,因此使得这一古老的模型重新焕发新机.最近,很多 Petri 网的高效验证器[23,24]被研究开发出来,作为程序分析器的引擎.文献[25]在不完备的情况下,基于通过 SMT-solver 实现了基于模式检测的 Petri 网可覆盖性验证.文献[11]将异步程序的进程间数据流分析形式化成了一个 AIFDS 问题,并提出了相应的算法.其结论表示:尽管解决该问题是 EXPSPACE 难的,但在实际中效率还是相当快的.

在 20 世纪末,大量 Petri 网的扩展模型也开始应用于并发程序.文献[26]提出了颜色 Petri 网(coloured Petri net),对同步的通信进行了研究.文献[27,28]则提出了对象 Petri 网(object Petri net),对面向对象的程序进行了建模研究.近些年,一些更为具体的应用方面的研究也已经产生.文献[8]针对 Erlang 语言提出了一个叫做 Soter 的自动研究工具,文献[14]也提出了一个针对异步程序开发的程序分析器原型.

此外,除了基于 Petri 网的研究,在应用中也有不少其他方式的探索.文献[29]对 MHP 问题,也就是问两个活动能否在程序里并发地进行这一基础的问题进行了探索,并且在基于类 x10 语言上提出了一种静态 MHP 分析的算法,有着较快的效率.文献[30]则对异步程序进行了依靠/保证(rely/guarantee)的抽象,即,每个异步进程都需要满足这两个断言.这样,整个程序的正确性就依赖于这些断言需要被满足,文献[30]基于此给出一些 C 语言中的例子.文献[31]为了能够编程有效的异步程序,定义了一种高可靠的程序语言 P#,通过静态数据竞争检测等手段来满足有效的要求.

但是,基于这样的模型存在着一些问题,比如:目前,所有的程序模型通信都不考虑传值(value-passing);在文献[3]中提出的异步通信下推系统(asynchronous communicating pushdown systems)中,便将消息限制在了有限种;文献[6]在对基于事件同步的异步程序分析的过程中,也将消息种类作了限定,因此无法表达和分析像 future 和 promise 这样的语言特性.再比如:目前,所有模型也无法表达某线程是否可以知道所接受的信息是某个特定线程所发,等等.因此,研究者们希望研究出表达能力更强的理论模型来描述和分析异步通信程序.与此同时,研究者们也希望能基于此模型开发出高效的验证器来自动实现这些程序的程序分析.

6 总结以及未来的工作

本文针对异步通信程序给出了一个抽象的模型通信 Petri 网来编码,证明了在 k-型限制下的通信 Petri 网的可覆盖性问题是可判定的,并且简单地给出了一个复杂性的下界.通信 Petri 网是将交互与程序调用合理地加以分离,更方便直观理解和实现.

未来的工作包括:

- 在理论上,我们希望得到更为一般化的可判定模型来对异步通信程序进行建模和验证;同时验证其他性质,比如活性.此外,我们希望能对其与其他类似的模型做出能力上的比较,希望能严格证明其是当前最大的可判定模型;
- 在实现上,我们希望能够通过该模型提出有效使用的算法,以此实现强大的通信 Petri 网的验证工具和

从 Erlang 语言到该工具的自动转化程序,从而得到自动化 Erlang 语言的验证工具.

References:

[1] Ramalingam G. Context-Sensitive synchronization-sensitive analysis is undecidable. ACM Trans. on Programming Languages and Systems (TOPLAS), 2000,22(2):416−430. [doi: 10.1145/349214.349241]

[2] Sen K, Viswanathan M. Model checking multithreaded programs with asynchronous atomic methods. In: Proc. of the Int'l Conf. on Computer Aided Verification. Berlin, Heidelberg: Springer-Verlag, 2006. 300−314. [doi: 10.1007/11817963_29]

[3] Kochems J, Ong CHL. Safety verification of asynchronous pushdown systems with shaped stacks. In: Proc. of the Int'l Conf. on Concurrency Theory. Berlin, Heidelberg: Springer-Verlag, 2013. 288−302. [doi: 10.1007/978-3-642-40184-8_21]

[4] Petri CA. Kommunikation mit automaten [Ph.D. Thesis]. University of Bonn, 1962.

[5] Lazić R, Newcomb T, Ouaknine J, Roscoe AW, Worrell J. Nets with tokens which carry data. Fundamenta Informaticae, 2008, 88(3):251−274.

[6] Emmi M, Ganty P, Majumdar R, Rosa-Velardo F. Analysis of asynchronous programs with event-based synchronization. In: Proc. of the European Symp. on Programming Languages and Systems. Berlin, Heidelberg: Springer-Verlag, 2015. 535−559. [doi: 10.1007/978-3-662-46669-8_22]

[7] D'Osualdo E, Kochems J, Ong CHL. Automatic verification of Erlang-style concurrency. In: Proc. of the Int'l Static Analysis Symp. Berlin, Heidelberg: Springer-Verlag, 2013. 454−476. [doi: 10.1007/978-3-642-38856-9_24]

[8] D'Osualdo E, Kochems J, Ong L. Soter: An automatic safety verifier for Erlang. In: Proc. of the 2nd Edition on Programming Systems, Languages and Applications Based on Actors, Agents, and Decentralized Control Abstractions. ACM Press, 2012. 137−140. [doi: 10.1145/2414639.2414658]

[9] Kochems J. Verification of asynchronous concurrency and the shaped stack constraint [Ph.D. Thesis]. University of Oxford, 2015.

[10] Cai X, Ogawa M. Well-Structured pushdown systems. In: Proc. of the Int'l Conf. on Concurrency Theory. Berlin, Heidelberg: Springer-Verlag, 2013. 121−136. [doi: 10.1007/978-3-642-40184-8_10]

[11] Jhala R, Majumdar R. Interprocedural analysis of asynchronous programs. ACM SIGPLAN Notices, 2007,42(1):339−350. [doi: 10.1145/1190215.1190266]

[12] Ganty P, Majumdar R, Rybalchenko A. Verifying liveness for asynchronous programs. ACM SIGPLAN Notices, 2009,44(1): 102−113. [doi: 10.1145/1594834.1480895]

[13] Ganty P, Majumdar R. Algorithmic verification of asynchronous programs. ACM Trans. on Programming Languages and Systems (TOPLAS), 2012,34(1):6. [doi: 10.1145/2160910.2160915]

[14] Majumdar R, Wang Z. BBS: A phase-bounded model checker for asynchronous programs. In: Proc. of the Int'l Conf. on Computer Aided Verification. Springer Int'l Publishing, 2015. 496−503. [doi: 10.1007/978-3-319-21690-4_33]

[15] Hague M. Parameterised pushdown systems with non-atomic writes. In: Proc. of the 31st Int'l Conf. on Foundations of Software Technology and Theoretical Computer Science. 2011. 457. http://arxiv.org/abs/1109.6264

[16] Esparza J, Ganty P, Majumdar R. Parameterized verification of asynchronous shared-memory systems. In: Proc. of the 25th Int'l Conf. on Computer Aided Verification (CAV 2013), Vol.8044. Saint Petersburg: Springer-Verlag, 2013. 124. [doi: 10.1007/978-3-642-39799-8_8]

[17] Durand-Gasselin A, Esparza J, Ganty P, Majumdar R. Model checking parameterized asynchronous shared-memory systems. In: Proc. of the Int'l Conf. on Computer Aided Verification. Springer Int'l Publishing, 2015. 67−84. [doi: 10.1007/978-3-319-21690-4_5]

[18] La Torre S, Muscholl A, Walukiewicz I. Safety of parametrized asynchronous shared-memory systems is almost always decidable. In: Proc. of the 26th Int'l Conf. on Concurrency Theory. 2015. 72. http://drops.dagstuhl.de/opus/volltexte/2015/5381

[19] Leroux J, Praveen M, Sutre G. Hyper-Ackermannian bounds for pushdown vector addition systems. In: Proc. of the Joint Meeting of the 23rd EACSL Annual Conf. on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symp. on Logic in Computer Science (LICS). ACM Press, 2014. 63. [doi: 10.1145/2603088.2603146]

[20] Leroux J, Sutre G, Totzke P. On the coverability problem for pushdown vector addition systems in one dimension. In: Proc. of the Int'l Colloquium on Automata, Languages, and Programming. Berlin, Heidelberg: Springer-Verlag, 2015. 324−336. [doi: 10.1007/978-3-662-47666-6_26]

[21] Finkel A, Schnoebelen P. Well-Structured transition systems everywhere. Theoretical Computer Science, 2001,256(1):63−92. [doi: 10.1016/S0304-3975(00)00102-X]

[22] Chadha R, Viswanathan M. Decidability results for well-structured transition systems with auxiliary storage. In: Proc. of the Int'l Conf. on Concurrency Theory. Berlin, Heidelberg: Springer-Verlag, 2007. 136−150. [doi: 10.1007/978-3-540-74407-8_10]

[23] Rodríguez C. Verification Based on Unfoldings of Petri Nets with Read Arcs. École Normale Supérieure de Cachan, 2013.

[24] Siebert M, Flochová J. Pnets—The verification tool based on Petri nets. In: Proc. of the World Congress on Engineering. 2013. 1. http://www.iaeng.org/publication/WCE2013/WCE2013_pp369-373.pdf

[25] Esparza J, Ledesma-Garza R, Majumdar R, Meyer P, Niksic F. An SMT-based approach to coverability analysis. In: Proc. of the Int'l Conf. on Computer Aided Verification. Springer Int'l Publishing, 2014. 603−619. [doi: 10.1007/978-3-319-08867-9_40]

[26] Christensen S, Hansen ND. Coloured Petri nets extended with channels for synchronous communication. In: Proc. of the 15th Int'l Conf. on the Application and Theory of Petri Nets. LNCS 815, Zaragoza: Springer-Verlag, 1994. 159−178. [doi: 10.1007/3-540-58152-9_10]

[27] Lakos C. The consistent use of names and polymorphism in the definition of object Petri nets. In: Proc. of the 17th Int'l Conf. on the Application and Theory of Petri Nets. LNCS, Springer-Verlag, 1996. 380−399. [doi: 10.1007/3-540-61363-3_21]

[28] Valk R. Petri nets as token objects? An introduction to elementary object nets. In: Proc. of the 19th Int'l Conf. on the Application and Theory of Petri Nets. LNCS 1420, Springer-Verlag, 1998. [doi: 10.1007/3-540-69108-1_1]

[29] Lee JK, Palsberg J, Majumdar R, Hong H. Efficient may happen in parallel analysis for async-finish parallelism. In: Proc. of the Int'l Static Analysis Symp. Berlin, Heidelberg: Springer-Verlag, 2012. 5−23. [doi: 10.1007/978-3-642-33125-1_4]

[30] Gavran I, Niksic F, Kanade A, Rupak M, Viktor V. Rely/Guarantee reasoning for asynchronous programs. In: Proc. of the LIPIcs-Leibniz Int'l Proceedings in Informatics. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015. 42. http://drops.dagstuhl.de/opus/volltexte/2015/5390

[31] Deligiannis P, Donaldson AF, Ketema J, Lal A, Thomson P. Asynchronous programming, analysis and testing with state machines. ACM SIGPLAN Notices, 2015,50(6):154−164. [doi: 10.1145/2813885.2737996]

杨启哲(1994－),男,上海人,博士生,主要研究领域为形式化方法,可计算学习理论.

李国强(1979－),男,博士,副教授,CCF 专业会员,主要研究领域为形式化方法,程序语言理论,可计算学习理论.