

# 基于 Petri 网的 Java 多线程程序分析

郭娟<sup>1</sup>, 许志才<sup>2</sup>, 方贤文<sup>1</sup>

(1. 安徽理工大学 理学院, 安徽 淮南 232001;

2. 滁州学院, 安徽 滁州 239012)

**摘要:**多线程程序可以更好地利用计算机资源, 提高程序运行效率。但是由于多线程执行的不确定性, 程序设计中的潜在错误不易被发现。作为系统建模和分析的工具, Petri 网适合对具有同步、并发、冲突的离散事件系统进行建模和分析。文中探讨利用 Petri 网对 Java 多线程程序建模, 并通过分析模型的结构性质和动态性质定位程序设计中的错误。通过对一个多线程程序实例的建模以及对模型的分析, 定位了引发错误的原因, 并给出了解决方案。从而说明 Petri 网在分析多线程程序和提高程序稳定性方面具有优越性。

**关键词:** Petri 网; 抑制弧; 多线程; 模型

**中图分类号:** TP391.9

**文献标识码:** A

**文章编号:** 1673-629X(2009)04-0051-03

## Multi-Thread Analysis about Java Program Based on Petri Net

GUO Juan<sup>1</sup>, XU Zhi-cai<sup>2</sup>, FANG Xian-wen<sup>1</sup>

(1. School of Science, Anhui University of Science and Technology, Huainan 232001, China;

2. Chuzhou University, Chuzhou 239012, China)

**Abstract:** Multi-thread programs can operate more efficiently by making good use of computer sources. However, due to the uncertainty of multi-threads, the mistakes in programming cannot be found easily. The Petri Net is suitable to model and analyze discrete event systems with properties of being synchronizing, concurrent and conflicting. Probes into how to model programs with the Petri Net and locate the mistakes by analyzing the structural property and dynamic property of the model. Specifically, it finds the reasons for mistakes and then offers solutions by modeling and analyzing a concrete multi-thread program. It comes to the conclusion that the Petri Net has its advantage in analyzing multi-thread programs and improving the programs' stabilization.

**Key words:** petri net; inhibitor arcs; multi-threads; model

## 0 引言

多线程是指同时存在几个执行体, 按几条不同的执行线索共同工作的情况。多线程的程序能更好地表述和解决现实世界的具体问题, 是计算机应用开发和程序设计的趋势。Java 是高级语言中第一个在语言的核心中支持多线程的编程语言, 利用多线程技术, 在 Java 中可以很方便地实现多道程序的并发执行<sup>[1]</sup>。Java 语言内置了对多线程的支持, 使得编程人员方便地开发具有多线程功能, 同时处理多任务的功能强大的应用程序<sup>[2]</sup>。然而, 多线程程序一旦出现错误, 对于

错误的定位往往比较困难。在很多情况下, 一个线程必须与其他线程合作才能共同完成任务, 若配合不当会造成系统的严重错误。比如, 当几个线程共享某个或某些数据时, 如果线程配合不好, 可能会造成数据处理错误甚至更严重的错误。要预防这样的错误发生, 必须保证线程在一个完整操作的所有步骤的执行过程中能独占相关资源而不被打断, 这就是线程的同步。对线程进行了同步不意味着不会发生错误, 如果同步不当可能会造成死锁。当若干线程各自分别占用某些资源, 而同时需要对方的资源, 结果造成相互无限制地等待对方放弃资源, 谁也不能执行下去, 形成死锁。因此在编写多线程程序时, 要有全局的观念, 协调多线程间的配合, 在必要的时候进行线程的同步。

由于多线程并行运行的不确定性, 程序运行一旦出错, 往往难以定位错误。而 Petri 网对于发现系统中的并发与冲突具有其独特的优势。可以通过建立程序的 Petri 网模型, 借助 Petri 网丰富的分析工具发现和

收稿日期: 2008-07-25

基金项目: 国家自然科学基金(30570431); 安徽省青年教师基金(2006jq1077)

作者简介: 郭娟(1980-), 女, 河南濮阳人, 讲师, 硕士, 研究方向为 Petri 网及应用; 许志才, 教授, 博士, 研究方向为优化算法; 方贤文, 副教授, 博士, 研究方向为 Petri 网及应用。

定位错误。

Petri网是分布式系统的建模和分析工具。它特别便于描述系统中进程或部件的顺序、并发、冲突以及同步等关系。Petri网具有严格的数学基础,理论结果非常丰富。文献[3]详细介绍了Petri网的理论知识和Petri网的应用。由于原型Petri网的模拟能力有限,为了能模拟更多的系统,对原型Petri网进行了很多扩展。文中研究的是对多线程程序的建模,程序中有控制语句流程的条件语句,而带抑止弧的Petri网能容易模拟与或等逻辑关系,因此利用了带抑止弧的Petri网进行建模和分析。文献[4]利用原型Petri网对由于同步造成的死锁进行了建模和分析。针对无同步的多线程程序进行建模,在此基础上利用Petri网丰富的分析工具定位程序错误。

## 1 带抑止弧 Petri 的概念及建模规则

### 1.1 基本概念

在这里仅给出几个与文中密切相关的概念,其它Petri网术语可参见文献[1,2]。

定义 1<sup>[3]</sup> 带抑止弧的 Petri 网系统是一个五元组  $\Sigma = (S, T, F, I, M)$ , 其中  $(S, T, F)$  是一个网,  $M$  是网的一个标识,  $I \subset S \times T$ , 称为抑止弧集,  $I \cap F = \emptyset$ , 即  $\forall s \in S \wedge \forall t \in T; (s, t) \in F \rightarrow (s, t) \notin I$ , 对  $t \in T$ , 如果 (a)  $\forall s \in S; (s, t) \in F \rightarrow M(s) \geq 1$ , (b)  $\forall s \in S; (s, t) \in I \rightarrow M(s) = 0$ , 则  $t$  在标识  $M$  有发生权, 记为  $M[t >]$ 。

带抑止弧的 Petri 网是在原型 Petri 网的基础上增加一种连接库所和变迁的弧形成的。这种弧只对具备发生条件的变迁是否允许发生起控制作用。变迁一旦发生, 抑止弧对由此引起的标记变化不产生影响。用图形表示带抑止弧的 Petri 网时, 一条抑止弧相当于把一条有向边的箭头换成一个小圆圈<sup>[3]</sup>。由定义 1 知, 当抑止弧一端的库所中含有标记时, 对应的变迁不能被引发。反之, 若库所中没有标记, 变迁是否被引发与原型 Petri 网的规则相同。同时文献[3]中指出对于带抑止弧的 Petri 网, 也可以采用可达标识图以及关联矩阵和状态方程的分析方法, 只是由于增加了抑止弧, 使得分析的工作量增大。

### 1.2 建模规则

用 Petri 网中的变迁模拟可执行语句, 库所模拟状态。结构化程序包含顺序、分支和循环三种基本结构<sup>[4]</sup>。库所模拟分支和循环结构中的条件表达式时, 对库所进行附加条件限制, 分别表示表达式取真值和假值的情况。若条件表达式包含子表达式, 根据其或与关系利用下面的转换规则进行转换。

规则 1 程序 Petri 网模型有惟一的开始库所和结束库所, 程序结束时只有结束库所中有标记。

规则 2 程序中顺序结构对应 Petri 网结构(见图 1)。

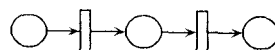


图 1 顺序结构

规则 3 条件语句的转换规则(与结构和或结构见图 2、3)。

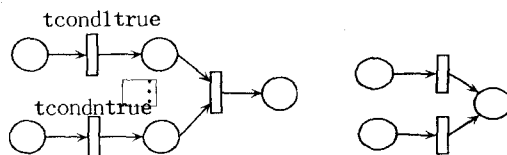


图 2 与结构

图 3 或结构

规则 4 循环语句转换规则(见图 4)。

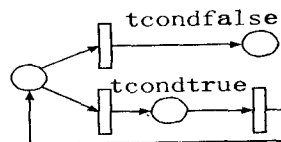


图 4 循环结构

规则 5 含有附加条件的变迁必须同时满足附加条件和 Petri 网中变迁引发规则才能引发。

按上述规则建立 Petri 网模型后, 生成其可达标识图, 如果包含使得 end 库所不能获得标记的死标识, 则说明多线程的运行有问题存在, 根据库所含有标记的情况, 对程序进行相应调试。对于同步不当引发的线程死锁, 利用 Petri 网的结构性质, 可以从模型中找到死锁和陷阱结构。文中主要针对没有同步的多线程程序引发的错误给予解决。

## 2 应用实例

“生产者/消费者”问题是说明线程同步控制的典型例子, 文中选择一个类似的多线程程序—存票售票程序<sup>[5]</sup>说明问题。程序中 Producer 线程负责存票, Consumer 线程负责在有票的时候售票。Tickets 类的对象是存票线程和售票线程共同访问的数据。

程序如下:

```
public class proandcon
{
    public static void main(String args[])
    {
        Tickets t = new Tickets(5);
        new Producer(t).start();
        new Consumer(t).start();
    }
}

class Tickets
{
    int number = 0; int size; boolean available = false;
```

```

public Tickets(int size)
{ this.size = size; }
}

class Procedure extends Thread
{ Tickets t = null;
  public Procedure(Tickets t)
  { this.t = t; }
  public void run()
  { while(t.number < t.size)
    { System.out.print("puts ticket" + (++t.number));
      t.available = true; }
  }
}

class Consumer extends Thread
{ Tickets t = null; int i = 0;
  public Consumer(Tickets t)
  { this.t = t; }
  public void run()
  { while(i < t.size)
    { if(t.available == true && i < t.number)
      System.out.println("buys ticket" + (++i));
      if(i == t.number) t.available = false; }
  }
}

```

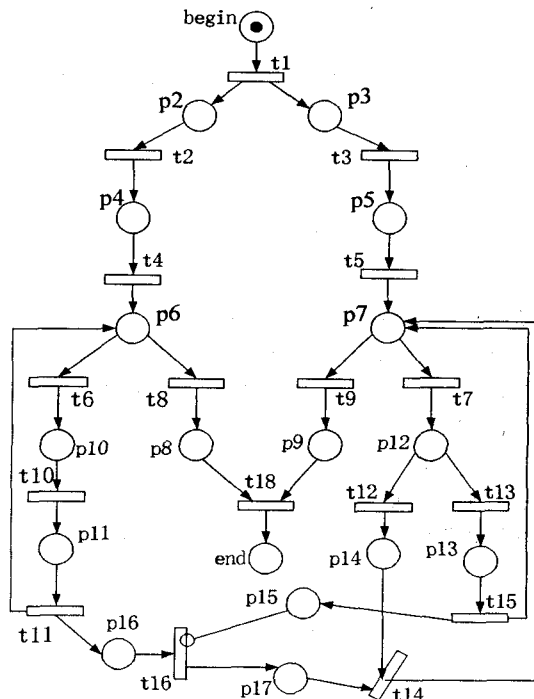


图5 程序的Petri网模型

至此导致程序出错的原因找到,只要对售票线程进行同步即可,即使得售票线程在执行时可以独占资源。在更复杂的多线程程序中,对线程同步之后也要进行建模分析,以确定不会产生死锁或者陷阱。

### 3 结束语

文中通过对一个没有进行同步的多线程程序建模分析找到了程序出错状态,同样对于线程死锁问题也可以使用Petri网模型清楚地分析出来。借助Petri网模型定位错误,然后进行相应处理,提高了多线程程序的安全性和运行的稳定性。不足之处是在对程序按规则建模后,由于模型中的库所和变迁的数量很多,生成的可达标识图规模很大,造成了分析工作量的增大。因此还可以在模型化简方面做出改进,以提高分析的效率。

#### 参考文献:

- [1] 龚成清. Java多线程的创建和线程同步的实现[J]. 宁波职业技术学院学报, 2007, 11(2): 61-63.
- [2] 印旻. Java与面向对象程序设计教程[M]. 北京: 高等教育出版社, 1999.
- [3] 吴哲辉. Petri网导论[M]. 北京: 机械工业出版社, 2006.
- [4] 鲍敢峰, 尤晋元. 多线程Java程序的Petri网模型[J]. 上海交通大学学报, 1998, 32(10): 124-126.
- [5] 郑莉, 王行言, 马素霞. Java语言程序设计[M]. 北京: 清华大学出版社, 2006.

该程序没有对线程进行同步,开始运行程序时没有任何问题,但多次运行程序,陷入死机,程序出错。这个潜在的错误发生概率很小,错误的原因也不易查找。根据以上的建模规则建立该程序的Petri网模型,借助Petri网的理论工具帮助分析和定为错误。程序的Petri网模型如图5,图中省略了某些变迁的附加运行条件。其中,库所begin表示程序的开始,end表示程序结束,t2存票线程启动,t4执行存票run方法,t3售票线程启动,t5执行售票run方法,t6符合while循环条件,t8不符合while条件,t9不符合循环条件,t7符合循环条件,t12  $i < t.number$  成立,t13  $i = t.number$  成立,t16判断  $t.available$  是否为true。

当库所p8,p14,p15,p16中有标记时,Petri网中没有一个变迁可以引发,即当前的状态M是一个死标识,end库所中不可能获得标记,说明程序不能正常结束。对应的多线程的执行过程是执行存票线程存入票后,转去执行售票线程,当售出最后一张可售票后要把  $t.available$  置为false,但在执行  $t.available = false$  语句之前,售票线程被中断,系统执行了存票线程,存入若干票后,存票线程结束,再调用售票线程并执行  $t.available = false$  语句,最终导致售票线程判断有票可售但是  $t.available = false$ ,售票线程陷入死循环,程序不能正确结束。