基于SPIN/Promela的并发系统验证*)

肖美华[1,3] 薛锦云[2,3]

(南昌大学计算中心 南昌330029)[1] (江西师范大学计算机信息工程学院 南昌330027)[2]

(中国科学院软件研究所计算机科学重点实验室 北京100080)[3]

摘 要 并发系统安全性分析是当前计算机科学中一个重要的研究领域。模型检测是最成功的自动验证技术之一,其成功应用归功于有效验证工具的支持。SPIN是一种著名的分析验证并发系统逻辑一致性的工具。本文在阐述SPIN工作机理的基础上,详细分析了基于SPIN的系统建模语言Promela中通道操作、基本数据结构及其功能,并设计了SPIN形式化验证软件系统的基本算法,最后运用SPIN对一个并发系统实例进行验证,得出了相应验证输出图。

关键词 模型检测,并发系统,软件可靠性,SPIN/Promela

Verification of Concurrent Systems Using SPIN/Promela

XIAO Mei-Hua[1,3] XUE Jin-Yun[2,3]

(Computing Center, Nanchang University, Nanchang 330029)[1]

(College of Computer Information, JiangXi Normal University, Nanchang 330027)[2]

(Key Laboratory for Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080)[3]

Abstract Safety analysis for concurrent systems is a vital research field of computer science. Model Checking is one of the most successful technologies of automatic verfication, its successful application is ascribed to the support of valid verification tools. SPIN is a famous tool for analyzing and verfying the logical consistency of concurrent system. Based on the introduction of the theory ofSPIN, this paper analyzes the channel operation, basic data structures and their functions of SPIN/Promela in detail. The fundamental verification algorithm is designed for software systems. We use the SPIN to verify an example of concurrent system, the corresponding verification output charts are obtained.

Keywords Model checking, Concurrent system, Software reliability, SPIN/Promela

1 引言

形式化方法对开发高可信软件具有重要的意义。模型检测(Model Checking)[1]是近十几年来最成功的自动验证技术之一,此技术的成功应用归功于有效验证工具的开发和支持。模型检测的研究大致涵盖以下内容:模态/时序逻辑、模型检测算法及其模型检测工具的研制[2]。SPIN[3~5]是一种著名的分析验证并发系统逻辑一致性的工具,以其简洁明了和自动化程度高而备受注目。SPIN已成功应用在安全协议验证、控制系统验证、软件验证及最优化规划等领域。应用模型检测工具验证具体的系统的过程大致可以分为三步:(1)为系统建立形式化的模型,在建模过程中,需对系统进行抽象或采用渐进式建模策略,以解决状态爆炸(State Explosion)问题;(2)应用逻辑公式表示所要验证的性质,逻辑公式能否正确完整地刻画所关心的性质是得到正确验证结果的前提;(3)验证。狭义的验证指的是状态空间的穷举搜索过程,而从广义来讲,是一个验证→报错→错误信息分析和模型修改→再检测的循环过程。

本文第2节介绍SPIN的工作机理;第3节论述基于SPIN的系统建模语言Promela[5]中通道操作及基本数据结构;第4节设计一个SPIN形式化验证软件系统的基本算法,结合并发系统实例进行验证;最后作结论。

2 SPIN工作机理

作为一种形式化自动验证工具,SPIN的目的是提供:1)系统建模语言Promela,用于直观、明确地描述系统Promela模型规约,而不考虑具体实现细节;2)功能强大而简明的描述系统应满足性质(属性要求)的逻辑表示法(LTL);3)提供一套验证系统建模逻辑一致性及系统是否满足所要验证性质的方法。

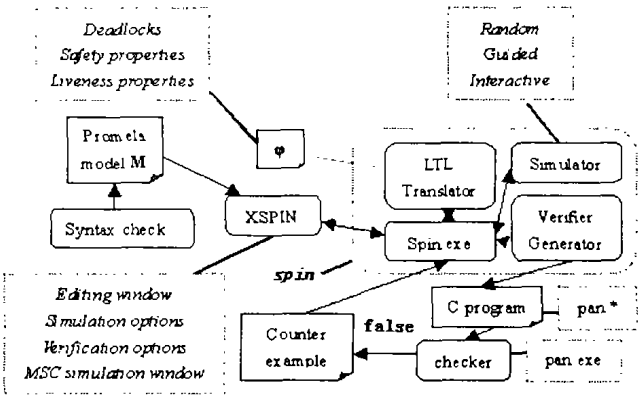

图1 SPIN模拟与验证结构流程图

XSPIN是一个用Tcl/Tk书写的驱动执行SPIN的图形

*)本课题得到国家自然科学基金(60273092)、江西省自然科学基金(0411041)、南昌大学03年度科研基金项目共同资助。肖美华 副教授,博士生,研究方向:软件形式化与可靠性、模型检测;薛锦云 教授,博导,研究方向:软件形式化与自动化、智能教学软件。

前端界面。验证器能快速安装，占用内存小。由 SPIN 执行的穷尽验证结果能明确地确定系统行为是否是无差错的(error-free)。SPIN 模拟与验证流程见图1。

3 SPIN 通道操作及基本数据结构

SPIN 采用的系统规约语言 Promela 中，不考虑与进程交互无关的细节，只对相关进程行为建模并用 SPIN 验证。Promela 程序由进程、信息通道和变量组成。进程刻画系统的行为，通道和全局变量用来定义进程执行的环境。它的建模方式是以进程为单位，进程异步组合，同步则需要进行显式的声明。进程描述的基本要素包括赋值语句，条件语句，通讯语句，非确定性选择和循环语句。

3.1 通道操作

Promela 通过信息通道实现并发系统中不同进程之间的通讯，可以定义为二种方式：同步(rendez-vous)、异步(buffered)。下面具体讨论通道操作(Channel operations)，操作格式为：

(1)*q*!*var1*,*const*,*var2*,… *or*,*equivalently q*!*var1*(*const*,*var2*,…)

(2)*q*?*var1*,*const*,*var2*,… *or*,*equivalently q*?*var1*(*const*,*var2*,…)

·为标准通道操作；

·第一条为信息发送(*send*)语句，第二条为接收(*receive*)语句；

·要使 *send* 和 *receive* 语句使能(*enabled*)，通道 *q* 必须初始化。*var1*,*const*,*var2*,…中数据类型要与通道 *q* 定义的数据类型相容；

·若通道为 *buffered*，当缓冲区未溢出时，*send* 语句是使能的；当缓冲区非空时，*receive* 语句是使能的；

·若通道为 *unbuffered*，一个 *send*(*receive*)语句是使能的，只有当对应的 *receive*(*send*)语句能同时执行；*receive* 读到的信息就是 *send* 发送的信息；

·一个 *receive* 语句执行时读取通道中的最老信息(通道即为队列，读取队列头部信息)；*send* 语句发送的信息为最新信息(位于队列的尾部)；

·在 *receive* 语句中出现的常量 *const*，用来约束它的使能性，通过强制比较它所接收到的信息是否与 *const* 相等，若不等，此 *receive* 语句为阻塞状态；具体地，可通过函数 *eval*( )使用一个局部或全局变量作参数判断，例如：*q* ? *var1*,*eval*(*var2*),*var3*。当所匹配的第二个发送的值与变量 *var2*的值不等时，此 *receive* 语句便阻塞。若通道为 *buffered* 时，有下列附加操作：(1)*q*?[*var1*,*const*,*var2*]：返回真值，当对应的接收操作使能时；(2)*q*?：标准接收操作，*buffer* 中的信息没有移除；(3)*q*!!*var1*,*const*,*var sorted send*：有序发送，往有序信息中发送信息；(4)*q*??*var1*,*const*,*var random receive*：若有缓冲区中的一些信息能使接收操作执行，查找其中的最老信息；(5)*q*??[*var1*,*const*,*var2*]：返回真值，当对应的接收操作是使能的；(6)*q*??：随机接收操作，*buffer* 中的信息没有移除。

上述缓冲区的操作可以通过 SPIN 行命令开关-*m* 来改变：若选用此项，当缓冲区满时，对通道的 *send* 操作不会处于阻塞状态，只不过此时往满缓冲区发送信息将失掉。

3.2 基本数据结构

在 SPIN 中基本数据结构有：状态矢量(*State vector*)、*Depth-first stack* 和 *Seen state set*，它们的含义及功能阐述如下：

(1)状态矢量(*State vector*) 在 Promela 刻画系统中，用状态矢量唯一标识状态，一个状态矢量包含：全局和局部变量，对应系统中每个进程实例的 *process counter*(当前执行位置)。Promela 程序实例见图2，执行违反断言 $N=2$时状态执行序列及对应的状态矢量表示(图3)。

(2)*Depth-first stack* 在 SPIN 中，*Depth-first stack* 作用有二：当到达一条路径末尾需要回溯搜索其它路径时，指出回溯方向；当找出一个反例时，此栈给出引起违反的状态迁移序列(计算路径)，具体通过文件 *.*trail* 存放 *Depth-first stack* 中内容。若对图1所给出的 Promela 程序执行状态迁移：0.1→0.2→0.3→2.1→2.2→…，其对应 *Depth-first stack* 存放内容见图4。

```
byte x,t1,t2;/*var declaration*/
proctype Thread() /*system process*/
  { do :: t1=x;            1.1
          t2=x;            1.2
          x=t1+t2          1.3
    od }
proctype Thread2() /*system process*/
  { do :: t1=x;            2.1
          t2=x;            2.2
          x=t1+t2          2.3
    od }
init /*initial process */
{ x=1;                     0.1
  run Thread1();           0.2
  run Thread2();           0.3
  assert(x!=N) }           0.4
```

图2

```
state vectors for Violating schedule for N=2
(initial vector )   [ 0.1,----,----, 0,0,0 ]
                        ①      ②
①values of program counter for each process
②values of program variables x,t1,t2
←0.1→[0.2,----,----,1,0,0]
←0.2→[0.3,1.1,----,1,0,0] Unstarted processes
←0.3→[0.3,1.1,2.1,1,0,0]
←2.1→[0.4,1.1,2.2,1,1,0]  Process started
←2.2→[0.4,1.1,2.3,1,1,1]
←2.3→[0.4,1.1,----,2.1,2.1,1]
←0.4→[*,1.1,2.1,2,1,1]

Finished process
                  Violation
```

图3

(3)*Seen state set* 在 SPIN 验证过程中，经常会出现此

种现象:沿着状态 S 的不同路径搜索(若状态 S 已搜索过),此时状态 S 以及在计算树中的所有子女都没必要继续遍历(因为已搜索过),SPIN 中通过 *Seen state set* 来达到减少状态压缩、提高搜索效率之目的。对于图1给出实例,通过计算树方式由图5说明(若把状态看成为图的顶点,则可通过可达图说明,即此时同享一个已访问的顶点)。通过 SPIN 中 *Seen state set* 的使用,对非终止性系统的执行则可理解为可终止的,若其只有有限个状态,需要强调的是,在 SPIN 所有验证的系统都是有限的,因为其基本数据结构是受限的(bound)。

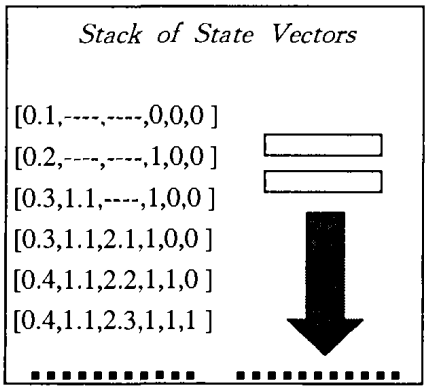

图4

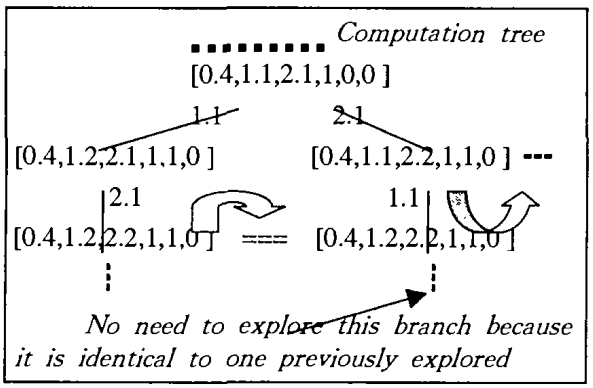

图5

4 验证算法及实例

熟悉模型检测工具 SPIN 最可取的方法是使用 Xspin,一个独立运行于 SPIN 的图形界面软件,在使用 Xspin 主菜单的同时产生相应的 SPIN 行命令,Xspin 后台运行 SPIN 以获得期望的输出结果,并能把输出结果以图形、图表方式直观地表示,Xspin 不仅知道何时以及怎样对由 SPIN 生成的模型检测器 PAN.C 进行编译,而且知道何时以及怎样执行它,在使用 Xspin 过程中,具有良好的图形界面及可操作性,要求用户记忆的很少,文[7]给出了进一步精通 SPIN 的方法。值得注意的是,为保证 Xspin 正常运行,相关软件必须预先安装:wish、Spin 、Xspin、ANSI-C Compiler。SPIN 分析验证算法构建如下:

```
verify(state,trace,depth)
  { if (depth>depth_limit)/ * check depth limit (from m option) * /
        {System. out. println(error. max search depth too small);
         trace_limit_reached =true;}
else {if (state is error state)/ * check an assertion is violated * /
        {dump. trail file,print error message,throw exception;}
      if(state !in seen)/ * if have not explored this state before * /
       {seen ={state }union seen; #add it to seen set
        for each active process p at state do
           {for each enabled transition t in p at state do
              {state =eval_tran(p,t,state); / * get the succ state of
               this state * /
               trace =trace append t; / * update trace and depth info
               * /
               depth =depth +1;
              verify(state,trace,depth)/ * explore successor state * /
}}}}}
```

在 Cygwin 系统虚拟 Linux 环境下,运行 Xspin 的主界面如图6所示。

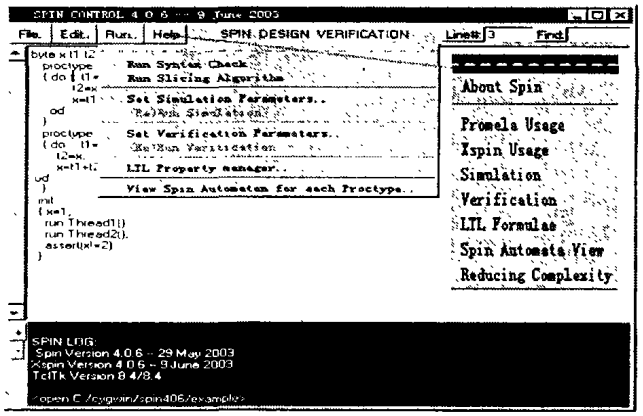

图6 Xspin 运行主界面

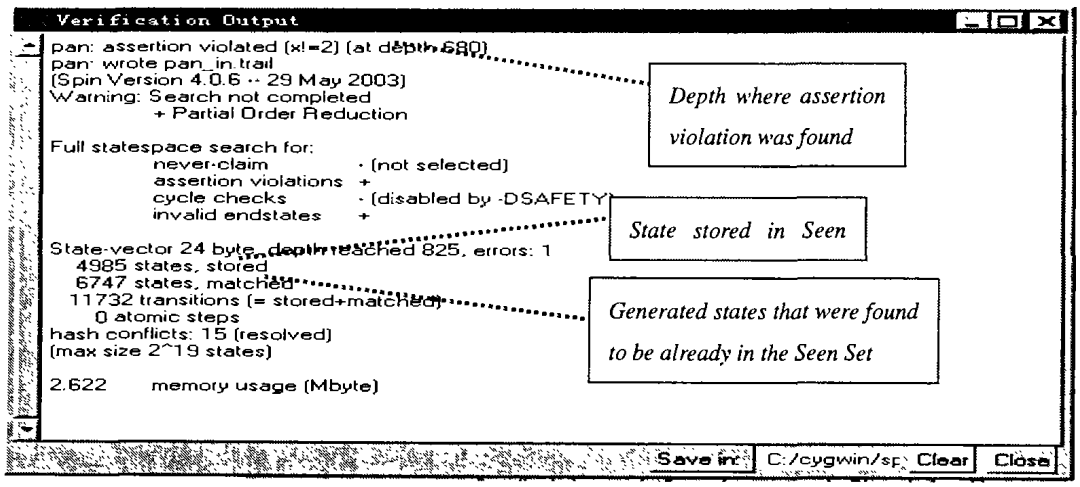

图7 验证输出图

运行图1提供的并发系统验证实例,我们设置断言 *assert* $(x!=2)$,在 Xspin 环境下执行操作:(1)*Set Simulation Parameters…/(Re)Run Simulation*;(2)*Set Verification Parameters…/(Re)Run Verification*;

得到 SPIN 模拟与验证输出图:*Verification Output*(图

(下转第208页)

根据表3，新算法能够大大减少运行时间，待排序表越长，减少的比列越大。在最坏情况下，$\frac{TO/TN}{\frac{m+n}{(m+n)\sqrt{m+n}}}$应趋于某个常数，表3中该项的值反映了一般情况相对于最坏情况而言算法运行时间的减少程度。表3中该项的值随着表的增长而增加，这表明待归并表越长新算法的运行时间减少越显著。

表3中新算法运行时间的减少比例小于表2中新算法元素移动次数的减少比列，这是由于新算法所增加元素的比较次数和辅助操作，部分抵消了元素移动次数的改进效果。

总结 通过采用数据分块、对数据块进行排序等方法归并长度分别为 m 和 n 的两个有序子表($m\geq n$)，新算法最多用 $\frac{m}{2}+4\sqrt{m+n}+(m+n+\sqrt{m+n})\log_2\sqrt{m+n}$ 次比较和 $4m+3n+\sqrt{m+n}+\frac{(m+n)\sqrt{m+n}}{2}$ 次移动。新算法在最坏情况下将经典原地归并算法的比较次数从$\bigcirc(m+n)$增加到$\bigcirc((m+n)\log_2\sqrt{m+n})$，但实验表明在一般情况下比较次数增加并不明显，而且随着待排序表的增长还会比经典算法的比较次数少。新算法将经典原地归并算法的移动次数从$\bigcirc(m\times n)$降低到$\bigcirc((m+n)^{3/2})$，从而大大缩短了算法的运行时间，待排序序列越长，算法改进效果越显著。因此该快速原地归并算法具有较高的理论和实用价值。

参 考 文 献

1 严蔚敏，吴伟民. 数据结构(C 语言版). 北京：清华大学出版社，2002

2 Kruse R L，Ryba A J. Data structure and program design in C++ (Copyright 1999). 北京：高等教育出版社，2001

3 龙昭华. 面向对象程序设计教程. 西安：西安电子科技大学出版社，2003

(上接第203页)

7)、*Simulation Output*(图8)、*Message Sequence Chart*(图9)。Data Values 和 Finite States Machine(FSM)二图从略。

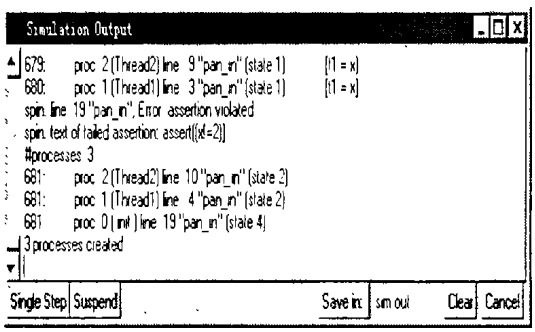

图8 模拟输出图

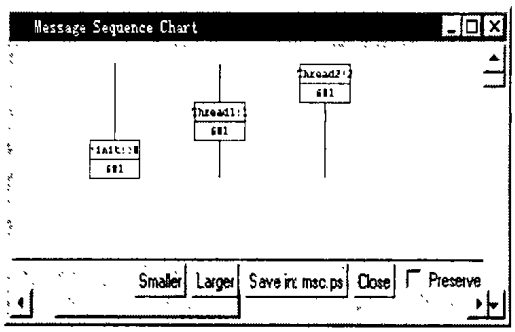

图9 MSC 图

结论 随着对模型检测技术的深入研究，出现了许多与所验证系统以及系统性质表示有密切相关的验证工具，比如：SPIN、SMV、CWB、DESIGN/CPN，UPPAAL，KRONOS，HYTECH，JPF 等。相比而言，SPIN 以其简洁明了和自动化程度高而备受注目，2001、2002年著名的 ACM"Software System Awards"分别授予 SPIN、JAVA 就是一个明证。SPIN 使用 Promela 语言对所需验证系统建模，系统所要验证的性质用 LTL 公式表示，它不仅具有很好的 GUI 以及"press on the button"验证，而且内嵌了用于状态空间压缩的优化算法(partial order redution、bitstate hashing、minimised automaton、dataflow analysis、slicing algorithm 等)，SPIN 已成功应用在安全协议验证、控制系统验证、软件验证及最优化规划等领域。

参 考 文 献

1 Clarke E M，Grumberg O，Peled D A. Model checking，Cambridge，MA：MIT Press，1999

2 林惠民，张文辉. 模型检测：理论、方法与应用. 电子学报，2002，12(A)

3 Holzmann G J. The SPIN Model Checker，Primer and Reference Manual. Addison-Wesley，2003

4 Bérard B，Bidoit M，Finkel A. System and Software Verification：Model Checking Techniques and Tools. Springer-Verlag，2001

5 http://netlib.bell-labs.com/netlib/spin

6 SPIN Online Documentation，Concise Promela，Reference，Rob Gerth，Eindhoven University，Accessible from[5]

7 Ruys T C. SPIN Tutorial：How to Become a SPIN Doctor，LNCS2318，2002

8 http://www.acm.org/awards/ssaward.html