通信下推系统的一种有界可达算法

缪 力,张大方
MIAO Li,ZHANG Da-fang

湖南大学 软件学院,长沙 410082
Software School of Hunan University,Changsha 410082,China
E-mail:miaoli2000@163.com

MIAO Li,ZHANG Da-fang.Bounded reaching algorithm for communicated pushdown systems.Computer Engineering and Applications,2008,44(24):19-21.

Abstract: Qadeer presented a bounded reaching algorithm for concurrent pushdown systems,which based on limiting the number of context switch,can analyze interprocedural concurrent programs.But concurrent pushdown systems simulate concurrency by global variables,which are not able to model event-drive concurrent programs.This paper presents a bounded reaching algorithm for communicated pushdown systems,which based on limiting the number of synchronization schedule and double schedule technology,can analyze event-drive concurrent programs.

Key words: bounded reaching algorithm;communicated pushdown system;interprocedural concurrent program analysis;model checking

摘 要: Qadeer 首次针对并发下推系统提出一种有界可达算法,通过限定上下文切换的次数使得算法可终止,可有效地分析过程间并发程序。但是并发下推系统以全局变量模拟同步,不适应于当前广泛使用的基于事件驱动的并发程序。针对通信下推系统,提出一种基于双重调度的有界可达算法,通过限定同步调度的次数,结合线程间的同步调度和线程内的路径调度解决通信下推系统的可达性问题,从而为事件驱动的过程间并发程序分析提供了算法基础。

关键词: 有界可达算法;通信下推系统;并发过程间程序分析;模型检查
DOI: 10.3778/j.issn.1002-8331.2008.24.006 **文章编号:** 1002-8331(2008)24-0019-03 **文献标识码:** A **中图分类号:** TP311.5

1 引言

并发软件系统往往需要保证某些特殊性质没有逻辑错误,而软件测试技术作为主要的系统可靠性检验手段,只能说明软件存在某种错误而不能说明软件不存在某种错误。通过模型检查等验证技术可以对有限状态变迁系统验证其时态逻辑性质,补充测试技术之不足。模型检查技术的限制在于需验证的系统必须是有限状态的,并且现有模型检查工具的算法主要是针对 Kripke 结构和 ω 自动机设计,软件程序由于其复杂的数据结构和控制结构,使得程序模型检查成为一个极具挑战性的问题。

基于程序控制流提取的模型检查方法是直接分析程序的源代码,构造扩展程序控制流图(或控制流自动机),利用程序分析技术和抽象技术,构造程序的抽象有限状态模型,通过模型检查算法对该有限状态模型进行性质验证。一个程序的实际可执行路径必然是其控制流图所有路径的子集,控制流图上不存在的路径,就不可能是程序的实际执行路径。

并发软件程序的面向过程特性以及并发特性使得过程间并发程序分析技术显得更加重要,并且越来越得到广泛的关注,然而,即使在只有两个并发任务的情况下,过程间并发程序分析仍然可能是一个不可判定问题[1]。

下推系统[2]是以下推自动机为基础建立的一种程序模型,可方便地对面向过程的程序建模,并通过高效的分析算法验证模型的性质,是当前模型检查和程序分析技术研究的热点。将下推系统扩展为并发下推系统或通信下推系统可模拟过程间并发程序特性。过程间并发程序的分析虽然是一个不可判定问题,但可以通过限定程序的性质[3],使用半算法[4](不一定终止的算法),以及基于抽象解释[5]的近似算法和基于有界可达的近似算法[6]进行计算和分析。

Bouajjani[7]针对通信并发下推系统提出一种可达性分析的理论框架,以抽象解释为理论基础,通过有限链抽象和 Kleene 代数方法建立了不同层次的近似算法框架。与抽象解释方式不同的另一类近似算法是有界可达算法。通过限定算法可能执行的步数上界,保证算法可终止,从而得到实用的分析算法。Qadeer[8]首次针对并发下推系统提出一种有界可达算法,通过限定上下文切换的次数使得算法可终止,可有效地分析过程间并发程序。但是并发下推系统以全局变量模拟同步,不适应于当前广泛使用的事件驱动的并发程序。本文针对通信下推系统,提出一种基于双重调度的有界可达算法,通过限定同步调度的次数,结合线程间的同步调度和线程内的路径调度解决通

基金项目: 国家自然科学基金(the National Natural Science Foundation of China under Grant No.60673155,No.90718008)。
作者简介: 缪力(1972-),男,博士,主要研究领域:程序分析,模型检测;张大方(1959-),男,博士,教授,主要研究领域:可信系统与网络、容错计算等。
收稿日期: 2008-04-11 **修回日期:** 2008-05-12

信下推系统的可达性问题,从而为事件驱动的过程间并发程序分析提供了算法基础。

2 下推系统与并发下推系统

定义 1(下推系统) 一个下推系统是一个三元组 $P=(S,\Gamma,\Delta)$,其中 S 是控制位置,Γ 是栈符号表;下推系统 P 的一个格局 c 是一个二元组(s,w),其中 $s\in S,w\in\Gamma^*$;Δ 是规则的一个有限集合,规则的表达形式为$(s,\gamma)\rightarrow(s',w)$,其中 $s,s'\in S$,$w\in\Gamma^*,\gamma\in\Gamma$;规则定义了格局之间的变迁关系,对于规则 $r\in\Delta$,如果$(s,\gamma)\rightarrow(s',w)$,则有$(s,\gamma w')\rightarrow(s',w\ w')$成立,其中 $w'\in\Gamma^*$。

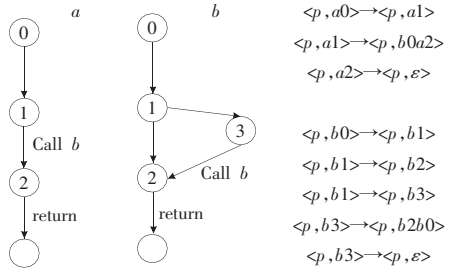

图 1 下推系统

定义 2(P 自动机) 对于一个下推系统 $P=(S,\Gamma,\Delta)$,其对应的 P 自动机是一个五元组 $A=(Q,\Gamma,->,S,F)$,其中 $Q\supseteq S$ 表示状态集合,$->\subseteq Q\times\Gamma\times Q$ 是变迁集合,$F\subseteq Q$ 表示最终状态集合,P 自动机的初始状态集合是 S;一个格局(s,w)可被 P 自动机接受,当且仅当$(s,w)\rightarrow*(s,q)$,其中 $q\in F$。

P 自动机用来表示一个下推系统的合法格局。文献[1]给出了基于 P 自动机计算下推系统中一个格局 c 的所有后继格局 $POST^*(c)$和前驱格局 $PRE^*(c)$的算法,基于该算法可解决下推系统的可达性问题并进而对下推系统进行模型检查。

定义 3(并发下推系统) 一个并发下推系统是一个多元组 $CP=(P_1,P_2,\cdots,P_n)$,表示多个下推系统的集合。

对于一个并发下推系统 CP,其一个全局格局是一个多元组 $g=(c_1,c_2,\cdots,c_n)$,对应于组成该并发下推系统的各个下推系统的格局。

设通信下推系统 CP 的两个全局格局为 $g=(c_1,c_2,\cdots,c_n)$,$g'=(c'_1,c'_2,\cdots,c'_n)$,有:

$g\rightarrow_i g'$当且仅当 $c_i\rightarrow_i c'_i$,且 $c_k=c'_k$若 $k\neq i$;对于并发下推系统,各下推系统之间的通信通过全局变量模拟,因此并发下推系统便于抽象出共享内存的并发程序模型。

对于某个并发下推系统 CP 中的两个全局格局 g 和 g',是否有 $g\rightarrow g'$,即并发下推系统的可达性问题。该问题可转化为两个下推自动机之交是否为空的问题,而这是一个不可判定问题,因此,并发下推系统的可达性问题没有精确解,只能通过近似算法计算近似结果。

不可判定问题之所以没有精确的算法求解,是因为精确求解的算法不一定终止,比如典型的停机问题,没有算法可以判定任意给定的程序是否终止。但是通过限定算法的最大执行步,可以得到一个可终止的判定算法。对于可达性问题,这种算法称为有界可达算法,有界可达算法的关键在于如何定义执行步。

Qadeer 等提出了一种针对并发下推系统的有界可达算法。上下文有界(context-bounded)可达算法对线程之间的交互次数进行限定,是 k 界可达问题(k-bounded reachability problem)。一个上下文(context)定义为在单一线程(单一堆栈)中的计算或变迁序列,计算从一个线程切换至另一个线程,定义为上下文切换。对于一个并发下推系统 P,给定其中两个格局为 g_1、g_2,g_1是否可以通过 k 次上下文切换(context switch)到达格局 g_2,即是否有 $g_1\overset{k}{\rightarrow}g_2$成立?该问题是一个可判定问题。设并发下推系统 CP 中的一个格局为 g,k 界可达算法是一棵计算树,其根节点是初始格局,第 i 层节点表示所有经过 i-1 次切换的可达格局。

3 通信下推系统

Qadeer 的算法简明实用,但是并发下推系统以全局变量模拟并发并不适合当前软件系统通过事件进行交互和同步的特性,直接计算某个格局的后继格局集合 $POST^*(c)$,不考虑同步事件的约束可能导致分析精度不够。

定义 4(通信下推系统) 一个通信下推系统是一个多元组 $CP=(P_1,P_2,\cdots,P_n)$,表示多个下推系统的集合,有相同的动作符号集合 Lab,Lab 用来表示各个下推系统之间的通信同步方式。由于本文重点讨论可达性问题,与通信同步无关的各下推系统内部动作可统一抽象为动作 τ。

对于一个通信下推系统 CP,其一个全局格局是一个多元组 $g=(c_1,c_2,\cdots,c_n)$,对应于组成该通信下推系统的各个下推系统的格局。

设通信下推系统 CP 的两个全局格局为 $g=(c_1,c_2,\cdots,c_n)$,$g'=(c'_1,c'_2,\cdots,c'_n)$,有:

$g\rightarrow_\tau g'$若对于 $1\leqslant i\leqslant n$,$c_i\rightarrow_\tau c'_i$且 $c_j=c'_j$若 $j\neq i$;

$g\rightarrow_a g'$若对于 $1\leqslant i,j\leqslant n$ 且 $j\neq i$,$c_i\rightarrow_a c'_i$且 $c_j\rightarrow_a c'_j$,且 $c_k=c'_k$若 $j\neq k\neq i$。

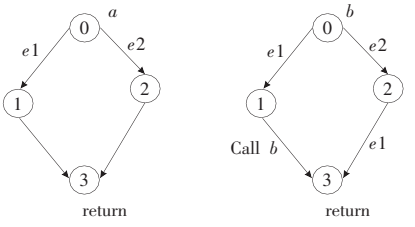

图 2 通信下推系统

通信下推系统的可达性问题同样是一个不可判定问题,但是,由于通信下推系统中各个下推系统以事件而不是全局变量进行通信并保持同步,Qadeer 的有界可达算法不能直接用于通信下推系统。例如,Qadeer 通过 $POST^*(c)$计算通信下推系统中某个下推系统 P 的格局 c 的可达格局集合,但是,对于通信下推系统,该可达集合未必是可达的,到达该集合的路径中包括了多个同步事件序列,必须要考虑同步事件的约束。

与 Qadeer 的算法解决切换问题不同,本文的算法要解决的问题是:对于一个通信下推系统 P,给定其中两个格局为 g_1、g_2,g_1是否可以通过 k 次同步调度到达格局 g_2,即是否有 $g_1\overset{k}{\rightarrow}g_2$成立?这里关键的问题是如何进行调度。最简单的调度是直接计算每个格局的后继格局,这种方式可以包括所有可能的线程

交互,但是会遇到状态空间爆炸问题,并且,很多调度仅仅是不同线程的调度,没有对在有同步事件的约束条件下某个格局是否可达的求解做出贡献,有意义的调度是事件同步调度必须进行的,因此同步调度应该在同步事件出现时进行。

注意到不同的路径可能出现不同的同步事件,从通信下推系统的某个下推系统的一个控制位置出发,紧接出现的所有同步事件是和路径有关的一个同步事件集合,从发生了同步的控制位置出发也是如此,因此有界可达算法还需要考虑如何得到调度点。

4　有界可达算法

以上一章的分析为基础,本文提出一种基于双重调度思想的有界可达算法分析通信下推系统的可达性问题。

双重调度是指线程间的同步调度和单线程的路径调度。通过同步调度使得线程可以并发执行,并保存同步调度的次数,当同步调度次数超过一个给定的上界 K 时终止算法;分支路径调度得到单个线程的某个控制位置紧接的所有同步事件,以及产生该事件时该线程的格局,结合这两种调度方式可得到通信下推系统的有界可达算法。

算法描述:

Input:Communicated pushdown system $(P_1,P_2,\cdots,P_n)$ and bound k

0:$Gin=(C1,C2,\cdots,Cn)$ //给定初始的格局;

1:forall (i=0…n) //初始路径调度

2:　$new\_Ci=GetNextHandoffPoint(Ci)$;对某个线程进行路径调度的结果是该线程的一个格局集合

3:forall ($C'i\in new\_C1,\cdots,C'i\in new\_Ci,\cdots,C'i\in new\_Cn$)

4:　$G=(C'1,\cdots,C'i,\cdots,C'n)$;

5:　$WL:=(G,0)$;　//WL 是工作表,初始的调度次数为 0;

6:　$Reach:=Reach\cup G$;

7:while (WL not empty)

8:　$(G,i):=remove(WL)$;

9:　if ($i<k$)　//切换次数不能超过上界 k

10:　forall ($m,p<n$)

11:　if ($label(Cm)=label(Cp)$) //相同事件标记的进行同步调度

12:　$new\_Cm:=GetNextHandoffPoint(Cm)$;

13:　$new\_Cp:=GetNextHandoffPoint(Cp)$;

14:　forall ($C'm\in new\_Cm,C'p\in new\_Cp$) //对线程 m 和 p 进行路径调度

15:　$G=(C1,\cdots,C'm,\cdots,C'p,\cdots,Cn)$;

16:　add(WL;(G,i+1));

17:　$Reach:=Reach\cup A$;

Output:Reach

算法的输入是一个通信下推系统和一个给定的上界 K,输出是从初始格局 Gin 出发的 K 界可达集合 Reach。

算法第 0 行到 6 行进行初始的路径调度,假定初始格局 Gin 中,各个线程的格局没有同步事件,通过 $GetNextHandoffPoint$ 计算每个线程的下一个同步事件发生时的格局,$GetNextHandoffPoint$ 计算结果是某个线程的一个格局集合,计算结果是所有的线程都"阻塞"在一个需要进行同步调度的格局。

算法第 7 行到 17 行是算法的主体。WL 工作表是格局集合,从该集合中移出一个元素,因为格局中所有的线程都"阻塞"需要进行同步调度,所以寻找是否存在两个线程的当前格局有相同的同步事件标记,即 11 行判断 $label(Cm)=label(Cp)$,如果有则这两个线程可以进行同步,即并发继续执行,由此 12、13 行对这两个线程的当前格局进行路径调度,14、15 行得到新的格局,因为进行了同步调度,所以 16 行新格局的同步调度次数加 1。如此反复计算,直至工作表为空。

5　结论

随着并发和分布式系统的广泛应用,对并发程序的可靠性提出更高的要求。由于并发程序有着不确定的执行路径,难以通过测试技术提供可靠性保障,需要进行性质验证以提升并发程序的可靠性。

下推系统可以有效地对过程间程序建模并以可达算法进行性质验证,为了分析过程间并发程序,将下推系统扩展为并发下推系统或通信下推系统。过程间并发程序分析是一个不可判定问题,Qadeer 首次针对并发下推系统提出一种有界可达算法,通过限定上下文切换的次数使得算法可终止,可有效地分析过程间并发程序。但是并发下推系统以全局变量模拟同步,不适应于当前广泛使用的基于事件驱动的并发程序。本文针对通信下推系统,提出一种基于双重调度的有界可达算法,通过限定同步调度的次数,结合线程间的同步调度和线程内的路径调度解决通信下推系统的可达性问题,从而为事件驱动的过程间并发程序分析提供了算法基础。

参考文献:

[1] Ramalingam G.Context sensitive synchronization sensitive analysis is undecidable[J].ACM Trans on Programming Languages and Systems,2000,22:416-430.

[2] Esparza J,Hansel D,Rossmanith P,et al.Efficient algorithms for model checking pushdown systems[C]//Volume 1855 of Lec Notes in Comp Sci:Computer Aided Verif,2000:232-247.

[3] Esparza J,Podelski A.Efficient algorithms for pre* and post* on interprocedural parallel flow graphs[C]//POPL 00:Principles of Programming Languages,ACM,2000:1-11.

[4] Qadeer S,Rajamani S K,Rehof J.Summarizing procedures in concurrent programs[C]//POPL 04:ACM Principles of Programming Languages,2004:245-255.

[5] Cousot P,Cousot R.Abstract interpretation:a unified lattice model for static analysis of programs by construction of approximation of fixed points[C]//Princ of Prog Lang,1977:238-252.

[6] Qadeer S,Wu D.KISS:keep it simple and sequential[C]//PLDI 04:Programming Language Design and Implementation,ACM,2004:14-24.

[7] Bouajjani A,Esparza J,Touili T.A generic approach to the static analysis of concurrent programs with procedures[C]//POPL 03:Principles of Programming Languages,ACM,2003:62-73.

[8] Qadeer S,Rehof J.Context-bounded model checking of concurrent software[C]//TACAS,2005:93-107.