一种宽容的多线程程序内部时间信息流类型系统

李 沁 袁志祥
(安徽工业大学计算机学院 马鞍山 243032)

摘 要 提出了一种针对多线程程序的内部时间信息流的宽容的类型系统。在隐藏竞争变量集合的基础上定义了非干扰属性的形式化规范;在类型系统中区分了隐藏线程,细化了对内部时间信息流发生场景的分析。相对于已有的基于类型理论的方法,本类型系统可以允许更多实质上安全的代码通过类型检查。另外,类型系统的可靠性是在独立于调度模型的情况下证明的。

关键词 非干扰,内部时间信道,类型理论
中图法分类号 TP301 **文献标识码** A

Permissive Type System for Internal Timing Information Flow in Multi-thread Programs

LI Qin YUAN Zhi-xiang
(School of Computer, Anhui University of Technology, Maanshan 243032, China)

Abstract This paper proposed a permissive type theory for checking internal timing channels in multi-thread programs. A formal specification of non-interference was defined based on the set of hidden-racing variables. In the type system, threads were discriminated according to the ways by which they deal with low level variables, so that we could refine the analysis about the scenario in which internal timing channel occurs. In contrast to existing methods, type checking in this work is more permissive because of less false positive. In addition, the soundness of the type system is proved in independent of the scheduler model.

Keywords Non-interference, Internal timing channel, Type theory

1 引言

安全信息流问题研究在程序运行时如何保证信息的机密性或完整性。安全信息流策略可以在操作系统内核中通过访问控制机制实现,但是有其自身的局限性;它也可以在程序源代码中通过约束实现,由于其灵活、广泛的应用范围受到越来越多的重视[1]。基于程序设计语言的信息流分析(Language-Based Information Flow, LBIF)研究程序是否满足非干扰属性(non-interference),即:高安全级别变量的变化不会影响到低级别程序变量的取值。本质上说,LBIF分析的目的是在静态分析阶段发现程序中可能引起违反安全策略的隐蔽信道,其中时间信道在并发程序中较为常见。

时间隐蔽信道通过特定动作发生的时间传递机密信息,时间信道是通过控制流以及特定代码段的执行时间控制程序的时间性特征实现的。观察者(潜在的攻击者)通过观察公开可见的程序状态(比如特定低级别变量的值)来推断机密信息的内容。

本文研究并发程序内部时间信道的信息流分析。多线程程序内部时间隐蔽信道源于线程代码的执行时间依赖于高安全级别变量(高级变量)的取值,进而影响到低安全级别变量(低级变量)的取值。例如在以下代码中:

例1 c1:i:=0;if (high=0) then while(i<10000) i:=i+1;else skip;low:=0 ; ‖ c2:skip;skip;low:=1;

‖是线程并行符号,低级变量low的值是否为1依赖于语句high=0的真值。这是因为高级变量high的取值会影响线程c1执行至语句low:=0所需的时间,从而影响low最终的取值。比如,若high=0为假,则分支内部的while循环不会被执行,所以low首先被赋值为0,然后线程c2被调度,low获得最终的值为1;反之若high=0为真,在while循环进行时发生线程调度,c2中的low首先被赋值为1,然后线程c1再次被调度,最终low被赋值为0。攻击者在了解程序源码的情况下可通过观察low的值来获悉high的值,从而导致机密性的破坏。

为分析类似上例的内部时间信道,已经提出了一系列基于类型理论的方法[2-6]。这些方法试图通过禁止隐藏线程拒绝那些可能存在内部时间信道的程序。隐藏线程是指这样的线程:线程中存在直接非法数据流;或者线程中存在非法控制流,即while或if语句的判断依赖于高级变量(上例中的变量high),而且在这两种语句(下文中称之为WH:while-high, IH:if-high)之内或之后出现了对低级变量(上例中的变量low)的赋值。最后一种情况是构成内部时间信道的主要原因,如果隐藏线程(c1)同其它与之有竞争的低级线程(c2)并

到稿日期:2013-04-25 返修日期:2013-08-05 本文受国家自然科学基金(61170070)资助。
李 沁(1976—),男,博士,副教授,主要研究方向为形式化方法,E-mail:linuxos2@163.com;袁志祥(1973—),男,硕士,副教授,CCF会员,主要研究方向为信息安全。

发运行，攻击者就有可能通过观察竞争的低级变量来获取高级变量的信息。在已有类型理论中识别隐藏线程的方法是在WH和IH语句之内或之后检查是否出现对低级变量的赋值[2]；或者，在IH的两个分支的执行时间相同的情况下允许在IH后出现对低级变量的赋值，或如果执行时间不同的话，则利用padding技术平衡分支的执行时间[7]。

对于内部时间信道的信息流问题，已有的类型论方法的局限性在于：

1)对于轮转调度模型下可能产生的时间信道的检查结果缺乏可靠性[2]；

2)这些方法在表达非干扰属性时均以观察必然为定义手段，导致在程序中完全排除对低级变量的竞争，即免于竞争(Race freedom)[4]。这个要求过于严格(restrictive)，以致一些明显安全的程序被类型检查拒绝。

上述问题的核心在于没有把隐藏线程从一般线程中区分出来：只有隐藏线程和低级线程(仅改变低级变量的线程，高级线程的定义是类似的)在低级变量上的竞争才会引起信息泄露。

基于以上考虑，本文提出了一种宽容的检查内部时间信息流的基于类型理论的方法，其贡献在于：

(1)利用隐藏竞争变量集合定义非干扰属性。通过排除隐藏低级变量之间的竞争关系达到杜绝内部时间信道的目的。此定义的出发点在于产生内部时间信道的机制，而非信息流导致的结果，使得属性规范更为简明，同时类型检查独立于调度模型；

(2)在类型系统中区别对待隐藏线程和低级线程，结合对每个线程所涉及到的低级变量的分析，判断对低级变量的竞争访问是否允许，使得类型系统更为宽容。

本文仅考虑内部时间信道的隐式信息流，不考虑直接或间接信息流问题，忽略后者并不会引起可靠性的问题，但是本文的类型系统仍然包含了相应的规则，以检查相应的程序行为。

本文第3节给出一个简单的多线程命令式语言的语法和操作语义；第4节给出非干扰属性的形式化定义；类型系统及其可靠性证明分别在第5和6节给出；第7节给出了类型检查的实例；最后是结论。

2 相关工作

本节着重于信息流分析的已有工作中与时间信道分析相关的研究。

Sabelfeld在文献[7]中针对包含同步原语的并发程序信息流提出了类型检查的方法，并给出了一种关闭时间信道的代码填充技术，作者又在文献[3]中给出了优于代码填充的转换技术。代码填充降低了程序运行的效率，而转换技术虽然避免了代码填充引起的效率问题，却增加了线程调度的开销。

Volpano和Smith在文献[8]中针对静态线程提出了概率性非干扰，在语言中加入了protect原语以防止产生时间信道，然而protect原语本身在实现上的可能性还值得商榷。文献[6]针对类似于CML语言的并发演算提出了基于观察必然的非干扰属性，并给出了相应的类型系统。文献[2,9]提出了基于马尔科夫链的观察必然的非干扰属性，分别从类型理论和模型检验的角度对时间信道进行了分析。这些工作是基于统一均匀分布调度模型的，不能应用于轮转调度模型。

文献[10]为提高类型检查方法的可扩展性，在类型系统中引入对变量使用的约束，以注释的形式说明线程对变量的访问模式，两个线程只有在对变量使用上达成一致才能并行组合在一起，该文没有考虑线程的时间性特征对信息流安全的影响。

3 简单的多线程命令式语言SMIPL：语法和操作语义

本节给出一个简单的多线程命令式语言SMIPL(Simple Multi-thread Imperative Programming Language)作为本文讨论之方法的载体，以下两小节分别给出其语法和操作语义。SIMPL语法和语义的定义参考了文献[2]的部分内容。

3.1 语法

SIMPL的语法包括表达式和命令，它们的BNF范式如下：

$$
\begin{aligned}
e ::= \quad & \text{(表达式)} \\
& x:H \mid x:L \quad \text{(变量)} \\
& \mid n \quad \text{(整型)} \\
& \mid b \quad \text{(布尔型)} \\
& \mid op\, e_1 e_2 \quad \text{(操作)} \\
c ::= \quad & \text{(命令)} \\
& x := e \mid c_1 ; c_2 \mid \text{while } e \text{ do } c \mid \text{if } b \text{ then } c_1 \text{ else } c_2 \\
& \mid c_1 \parallel c_2 \mid \text{skip}
\end{aligned}
$$

表达式和命令的语法构件与常见的命令式语言相同，以下仅给出必要的解释和假设：变量的安全类型H或L需根据安全策略指定；表达式(操作)中的操作符op并不需要在这里指定其含义；$c_1 \parallel c_2$表达两个命令的并置操作，即这两个命令并发执行，并置操作包含了多线程的并发操作，并且从第3.2节可知并发线程是允许交替执行的。我们假设表达式是无副作用且是完全的(total)。

3.2 操作语义

为定义SMIPL的操作语义，首先需要一个表达存储状态的函数$\mu: \mathcal{V} \mapsto \mathbb{Z}$，它是从变量名集合$\mathcal{V}$到整数集合$\mathbb{Z}$的函数；其次是表达式的语义函数$[\![\,]\!]$，其归纳定义如下：

$$
[\![e]\!] = \begin{cases} [\![n]\!], & e = n \\ \mu(x), & e = x \\ op[\![e_1]\!][\![e_2]\!], & e = op\, e_1\, e_2 \end{cases}
$$

，$[\![n]\!]$是n所代表的数

SIMPL命令(线程)的操作语义是通过程序格局上的迁移关系$\rightarrow$来定义的，格局C是一个序对(c,μ)或仅仅是一个存储状态μ。如果格局是序对，则c是当前正要执行的命令；如果格局是一个存储状态，则当前所有命令执行完毕，命令的执行效果保留在最后的存储状态中。符号$\rightarrow^*$表示迁移关系的自反传递闭包，用$\rightarrow^i, i \geqslant 0$表示迁移关系的$i$重自我复合。SIMPL命令的结构化操作语义定义如下：

$$
(skip,\mu) \rightarrow \mu \qquad \frac{x \in dom(\mu) \quad [\![e]\!] = n}{(x := e,\mu) \rightarrow \mu[x \mapsto n]}
$$

$$
\frac{[\![e]\!] \neq 0}{(\text{if } e \text{ then } c_1 \text{ else } c_2,\mu) \rightarrow (c_1,\mu)}
$$

$$
\frac{[\![e]\!] = 0}{(\text{if } e \text{ then } c_1 \text{ else } c_2,\mu) \rightarrow (c_2,\mu)}
$$

$$
\frac{[\![e]\!] \neq 0}{(\text{while } e \text{ do } c,\mu) \rightarrow (c; \text{while } e \text{ do } c,\mu)}
$$

$$\frac{[\![e]\!]=0}{(\text{while } e \text{ do } c,\mu)\to\mu}$$

$$\frac{(c_1,\mu)\to\mu'}{(c_1;c_2,\mu)\to(c_2,\mu')}\quad\frac{(c_1,\mu)\to(c_1',\mu')}{(c_1;c_2,\mu)\to(c_1';c_2,\mu')}$$

$$\frac{(c_1,\mu)\to(c_1',\mu')}{(c_1\parallel c_2,\mu)\to(c_1'\parallel c_2,\mu')}\quad\frac{(c_2,\mu)\to(c_2',\mu')}{(c_1\parallel c_2,\mu)\to(c_1\parallel c_2',\mu')}$$

多线程程序的执行模型由程序格局表示，由线程池和线程共享存储状态构成：(P,μ)。线程池是由自然数到线程集合的映射，$P:\mathbf{N}\to\mathscr{C}$，$\mathscr{C}$ 是命令集合。若 $P(k)=\{\}$，则当前线程池中位置 k 上无线程执行，否则 $P(k)$ 为当前线程池中位置 k 上尚未执行的命令。一个多线程程序在调度策略下交替执行。调度模型由 μ 中存储 sz（线程池中线程数目）以及 $next$ 函数（给出下一个被调度的线程号）定义，此调度模型在全局程序格局上定义关系 $\xrightarrow{k}$，$(P,\mu)\xrightarrow{k}(P',\mu')$ 表示格局 (P,μ) 在调度下执行命令 $P(k)$ 后转变为格局 (P',μ')。根据此模型定义多线程程序的全局操作语义如下：

$$(\text{Global})(\{\},\mu)\xrightarrow{0}(\{\},\mu)$$

$$\frac{next()==k,(P(k),\mu)\to(P(k)',\mu')}{(P,\mu)\xrightarrow{k}(P[k\mapsto P(k)'],\mu')}$$

$$\frac{next()==k,P(k)=c_1\parallel c_2}{(P,\mu)\xrightarrow{k}(P[k\mapsto c_1,\mu(sz)+1\mapsto c_2],\mu[sz\mapsto sz+1])}$$

说明：本文采用的调度模型是抽象的，$next$ 函数的返回值依赖于调度模型的具体实现。

定义 1（程序轨迹） 给定程序格局 (P,μ)，一个由中间格局构成的列表 $T=C_0,C_1,C_2,\cdots,C_n$ 是 (P,μ) 的程序轨迹，记为 $(P,\mu)\Downarrow T$，当且仅当：

- $C_0=(P,\mu)$；且
- 对所有 $i<n$，存在 k，$C_i\xrightarrow{k}C_{i+1}$。

下文中 $T(\mu)$ 表示程序轨迹在格局存储上的投影；$T(v)$ 表示 $T(\mu)$ 在标识符 v 上的投影；T_i 表示程序轨迹中第 i 个程序格局。

4 非干扰属性的形式定义

SIMPL 程序中所有标识符都具有安全标签（security label），安全标签构成格 $\mathscr{L}=\langle\{H,L\},\subseteq\rangle$ 且具有 H 的标识符比具有 L 的标识符的安全级别高，即标识符的集合

$$\mathscr{V}=\mathscr{V}_H\cup\mathscr{V}_L,\ \mathscr{V}_L=\mathscr{V}_{LL}\cup\mathscr{V}_{HL},\ \forall x\in\mathscr{V}_L,y\in\mathscr{V}_H,x\subseteq y$$

其中，集合 $\mathscr{V}_{HL}$ 包括那些出现在隐藏线程中的低级变量，$\mathscr{V}_H$ 和 $\mathscr{V}_{LL}$ 分别是高级变量集合与仅出现在低级线程中的低级变量集合。

对于引起直接或间接信息流的低级变量，要求它们仍然满足观察必然的要求；对于在低级线程中与隐藏线程无竞争访问关系的低级变量，不施加任何约束；最后，对于隐藏线程中的低级变量，要求其不在其它低级线程中出现，消除出现竞争访问关系的可能。基于以上分类，给出多线程程序非干扰属性的形式定义。

首先定义集合 $race(\mathscr{C}_{HL})$，此集合是有竞争访问的隐藏低级变量集合。定义过程 $FL(c)=\{x:x$ 为线程 c 中低级别自由变量$\}$（可由常规的自由变量计算方法归纳求解），可知被两个线程 c_1 和 c_2 竞争访问的低级变量集合为 $FL(c_1)\cap FL(c_2)$。给定线程集合 $\mathscr{C}_L$（包含低级线程集合 $\mathscr{C}_{LL}$ 和隐藏线程集合 $\mathscr{C}_{HL}$）和隐藏线程 c，c 中出现的低级竞争变量的集合为：

$$race(c)=\bigcup_{c'\in\mathscr{C}_L}\{FL(c)\cap FL(c')\}$$

而 $\mathscr{C}_L$ 中出现的隐藏级别竞争变量集合为：$race(\mathscr{C}_{HL})=\bigcup_{c\in\mathscr{C}_{HL}}race(c)$。$race(\mathscr{C}_{HL})$ 非空意味着存在隐藏线程与其它低级线程在某些低级变量上有竞争访问关系，构成产生内部时间信道的必要条件。

定义 2（免于隐藏竞争并发） 对多线程程序 P 及其类型环境 Γ，若在 Γ 下 P 的任意程序轨迹的任意程序格局中的线程集合 $\mathscr{C}_{HL}$ 满足 $race(\mathscr{C}_{HL})=\emptyset$，则称 P 是在 Γ 下免于隐藏竞争并发的。

说明：隐藏级别竞争变量集合的计算是基于语法树递归进行的。容易证明如果程序中所有低级线程涉及的低级变量的总数为 n，那么计算此集合的时间复杂度为 $O(n^2)$。

5 类型系统

本文的类型系统包含以下几种基本类型：

$$\tau::=L\text{ of }n\mid H\text{ of }n\mid L\mid H$$

$$\rho::=\tau\mid\rho\text{ var}\mid\langle\tau_1,\tau_2\rangle\mid\langle\tau,n\rangle$$

若不关心运行时间，则最后一则类型表达式可写为 $\langle\tau,\_\rangle$。安全级别仅包含高、低两个级别，将其一般化为更多级别的安全格（lattice）并非难事。类型 $\langle\tau_1,\tau_2\rangle$ 指语句的执行结果会影响 τ_1（或更高）类型的变量，且执行时间依赖于 τ_2（或更低）类型的变量；类型 $\langle\tau_1,n\rangle$ 指语句会影响 τ_1（或更高）类型的变量，并且其执行时间为 n 步。

在进行类型检查的同时还需要引入一个标记过程，当识别出 c 为隐藏线程的时候将其标记为 $c^{\#}$。这个标记的作用域是可以在顺序组合的情况下扩张的，即：由隐藏线程顺序组合的进程亦为隐藏线程。

类型规则的形式为 $\Gamma\vdash e:\tau$ 或者 $\Gamma\vdash c:\rho$，指在类型环境 Γ 下表达式 e 具有类型 τ 或命令 c 具有类型 ρ。首先给出不包含控制结构的简单语句的类型规则：

$$\text{VAR}\ \frac{\Gamma(x)=\tau\text{ var}}{\Gamma\vdash x:\tau\text{of }0}\quad\text{INT}\ \Gamma\vdash n:L\text{ of }0$$

$$\text{SKP}\ \Gamma\vdash\text{skip}:\langle H,1\rangle\ \text{OP}$$

$$\frac{\Gamma\vdash e_1:\tau_1\text{ of }n_1,\Gamma\vdash e_2:\tau_2\text{ of }n_2}{\Gamma\vdash op\ e_1\ e_2:\tau_1\vee\tau_2\text{ of }n_1+n_2+1}(i\in I)$$

$$\text{ASS}\ \frac{\Gamma(x)=\tau_1\text{ var},\Gamma\vdash e:\tau_2\text{ of }n,\tau_2\subseteq\tau_1}{\Gamma\vdash x:=e:\langle\tau_1,n+1\rangle}$$

规则（OP）指复合表达式的类型由参与计算的子表达式类型的最小上界确定，同时其执行时间为子表达式的执行时间之和；规则（ASS）中的前提 $\tau_2\subseteq\tau_1$ 保证了类型检查将拒绝直接信息流，并且赋值占用一个时间单位的执行时间。

控制语句的类型规则为：

$$\text{IF1}\ \frac{\Gamma\vdash e:\tau_1\text{ of }n,\Gamma\vdash c_1:\langle\tau_1,n\rangle,\Gamma\vdash c_2:\langle\tau_3,m\rangle,\tau_1\subseteq\tau_2\wedge\tau_3}{\Gamma\text{ if }e\text{ then }c_1\text{ else }c_2:\langle\tau_2,n+m\rangle}$$

$$\text{IF2}\ \frac{\Gamma\vdash e:\tau_1\text{ of }n,\Gamma\vdash c_1\langle\tau_1,n\rangle,\Gamma\vdash c_2:\langle\tau_3,m'\rangle}{\Gamma\text{ if }e\text{ then }c_1\text{ else }c_2:\langle\tau_2\wedge\tau_3,\tau_1\rangle}\ m\neq m',\tau_1\subseteq\tau_2\wedge\tau_3$$

$$\text{IF3}\ \frac{\Gamma\vdash e:\tau_1\text{ of }\_,\tau_1\subseteq\tau_2,\Gamma\vdash c_1\langle\tau_2,\tau_3\rangle,\Gamma\vdash c_2:\langle\tau_2,\tau_3\rangle}{\Gamma\text{ if }e\text{ then }c_1\text{ else }c_2:\langle\tau_2,\tau_1\vee\tau_3\rangle}$$

$$\text{WHL}\ \frac{\Gamma\vdash e:\tau_1,\tau_1\subseteq\tau_2,\tau_2\subseteq\tau_3,\Gamma\vdash c:\langle\tau_2,\tau_3\rangle}{\Gamma\vdash\text{while }e\text{ do }c:\langle\tau_2,\tau_1\vee\tau_3\rangle}$$

规则（IF1）处理的情况是两个分支的执行时间相同，即使在其后有低于类型 τ_1 的赋值语句，也不具有构成隐藏线程的

可能，因此可赋类型；规则(IF2)表明当分支执行时间不一致时，并不拒绝此语句，而是将其标记为在类型 τ_1 上的控制语句，待其与后续语句顺序组合时根据后续语句类型再做判定，分支的执行时间在后续判断中不再起作用；规则(IF3)和(WHL)与(IF2)相似，只能确定其有可能构成隐藏线程，所以将其标记为在类型 $\tau_1 \vee \tau_3$ 上的控制语句。规则(IF3)要求在判断类型低于分支类型时，条件语句影响的类型由其分支语句影响的类型来决定，而执行时间所依赖的类型由判断和分支语句影响时间的类型的最小上界决定。规则(WHL)通过前提条件 $\tau_1 \subseteq \tau_2 \wedge \tau_3 \subseteq \tau_2$ 保证了不会出现包含高级变量的判断影响低级循环体执行的情况，也保证了命令 c 本身的执行不会影响更高级变量的状态。注意这 4 条规则中的前提 $\tau_1 \subseteq \tau_2$ 杜绝了间接信息流。

顺序组合与并行组合语句的类型规则为：

$$\text{COMP1}\ \frac{\Gamma \vdash c_1 : \langle \tau_1, m\rangle, \Gamma \vdash c_2 : \langle \tau_2, n\rangle}{\Gamma \vdash c_1 ; c_2 : \langle \tau_1 \wedge \tau_2, m+n\rangle}$$

$$\text{COMP2}\ \frac{\Gamma \vdash c_1 : \langle \tau_1, \tau_2\rangle, \tau_2 \subseteq \tau_3, \Gamma \vdash c_1 : \langle \tau_3, \tau_4\rangle}{\Gamma \vdash c_1 ; c_2 : \langle \tau_1 \wedge \tau_3, \tau_2 \vee \tau_4\rangle}$$

$$\text{COMP3}\ \frac{\Gamma \vdash c_1 : \langle \tau_1, \tau_2\rangle, \tau_3 \subset \tau_2, \Gamma \vdash c_2 : \langle \tau_3, \tau_4\rangle}{\Gamma \vdash \{c_1 ; c_2\}^{\#} : \langle \tau_1 \wedge \tau_3, \tau_2 \vee \tau_4\rangle}$$

$$\text{PAR1}\ \frac{\Gamma \vdash c_1 : \langle \tau_1, n\rangle, \Gamma \vdash c_c : \langle \tau_2, m\rangle}{\Gamma \vdash c_1 \parallel c_2 : \langle \tau_1 \wedge \tau_2, m+n\rangle}$$

$$\text{PAR2}\ \frac{\Gamma \vdash c_1 : \langle \tau_1, \tau_2\rangle, \Gamma \vdash c_2 : \langle \tau_3, \tau_4\rangle}{\Gamma \vdash c_1 \parallel c_2 : \langle \tau_1 \wedge \tau_3, \tau_2 \vee \tau_4\rangle}$$

$$\text{PAR3}\ \frac{\Gamma \vdash c_1^{\#} : \langle \tau_1, \tau_2\rangle, \Gamma \vdash c_2 : \langle \tau_3, \tau_4\rangle, \tau_1 \subset \tau_3}{\Gamma \vdash c_1^{\#} \parallel c_2 \langle \tau_1 \wedge \tau_3, \tau_2 \vee \tau_4\rangle}$$

$$\text{PAR4}\ \frac{\Gamma \vdash c_1 : \langle \tau_1, \tau_2\rangle, \Gamma \vdash c_2^{\#} : \langle \tau_3, \tau_4\rangle, \tau_3 \subseteq \tau_1}{\Gamma \vdash c_1 \parallel c_2^{\#} : \langle \tau_1 \wedge \tau_3, \tau_2 \vee \tau_4\rangle}$$

$(FL(c_1) \cap FL(c_2) = \emptyset)$

子类型规则为：

BASE $L \subseteq H$ REF $\rho \subseteq \rho$

$$\text{STEP}\ \frac{n \leqslant m}{\langle \tau, n\rangle \subseteq \langle \tau, m\rangle} \quad \text{CMD1}\ \frac{\tau_1' \subseteq \tau_1, \tau_2 \subseteq \tau_2'}{\langle \tau_1, \tau_2\rangle \subseteq \langle \tau_1', \tau_2'\rangle}$$

$$\text{CMD2}\ \frac{\tau' \subseteq \tau}{\langle \tau, n\rangle \subseteq \langle \tau', n\rangle} \quad \text{CMD3}\langle \tau, n\rangle \subseteq \langle \tau, L\rangle$$

$$\text{TRANS}\ \frac{\rho_1 \subseteq \rho_2, \rho_2 \subseteq \rho_3}{\rho_1 \subseteq \rho_3}$$

$$\text{SUBSUMP}\ \frac{\Gamma \vdash c : \rho_1, \rho_1 \subseteq \rho_2}{\Gamma \vdash c : \rho_2}$$

在子类型规则中需特别注意规则(CMD1)与(CMD2)，由于 $\langle \tau_1, \tau_2\rangle$ 中的 τ_1 在格中是向上封闭的，因此在(CMD2)中类型为 $\langle \tau', n\rangle$ 的命令可以替换为类型为 $\langle \tau, n\rangle$ 的命令而不改变程序的可赋类型性，对(CMD1)亦然。

6 类型系统的性质

本节证明类型系统的一些重要性质，并证明如果程序在此类型系统中是可赋类型的，则程序在免于隐藏竞争并发的意义上是安全的(可靠性 Soundness)。

分别对表达式和命令进行归纳证明可得两个易见的引理。

引理 1 若 $\Gamma \vdash e : L$，则 e 仅含类型为 L 的变量。

引理 2 若 $\Gamma \vdash c : \langle H, -\rangle$，则 c 不对类型为 L 的变量赋值。

定理 1(类型保持 subject reduction) 假设 $(c, \mu) \rightarrow (c', \mu')$。

- 若 $\Gamma \vdash c : \langle \tau_1, \tau_2\rangle$，则 $\Gamma \vdash c' : \langle \tau_1, \tau_2\rangle$。
- 若 $\Gamma \vdash c^{\#} : \langle \tau_1, \tau_2\rangle$，则 $\Gamma \vdash c'^{\#} : \langle \tau_1, \tau_2\rangle$。
- 若 $\Gamma \vdash c : \langle \tau, n\rangle$，$n > 1$，且此规约的计算步数为 m，则 $\Gamma \vdash c' : \langle \tau, n-m\rangle$。

证明：我们对命令 c 的结构进行归纳证明。

首先，根据假设，c 不会形如 $x := e$ 或 $skip$。

若 c 形如 if e then c_1 else c_2，则 c' 为 c_1 或 c_2。若 c 的类型是 $\langle \tau, n\rangle$ 且 e 的类型为 $\langle \tau, n'\rangle$，$n' < n$，则 c 的类型是由类型规则 IF1 决定的，因此 c_1 和 c_2 的类型为 $\langle \tau, n-n'\rangle$。或者若 c 的类型为 $\langle \tau_1, \tau_2\rangle$，则其类型由类型规则 IF1 或 IF2 决定：利用规则 IF1 的原因在于 $\langle \tau_1, n\rangle \subseteq \langle \tau_1, \tau_2\rangle$；当类型由 IF2 决定时，存在类型 τ'、n 使 $\Gamma \vdash e : \langle \tau', n\rangle$，且存在类型 $\tau_3 \subseteq \tau_2$ 使命令 c_1 和 c_2 具有类型 $\langle \tau_1, \tau_3\rangle$，进而由规则 SUBSUMP 可知，$c$ 具有类型 $\langle \tau_1, \tau_2\rangle$。

若 c 形如 $whileedoc_1$，则 c' 形如 $c_1 ; c$。根据规则 WHL 命令 c 必具有形如 $\langle \tau_1, \tau_2\rangle$ 的类型。因此存在类型 $\tau_3 \subseteq \tau_2 \wedge \tau_1$ 使 c_1 的类型为 $\langle \tau_1, \tau_3\rangle$，又由规则 COMP2 可知 $c_1 ; c$ 具有类型 $\langle \tau_1, \tau_2\rangle$。

若 c 形如 $c_1 ; c_2$，在 $(c_1, \mu) \rightarrow \mu'$ 的情形下 c' 为 c_2，或者在 $(c_1, \mu) \rightarrow (c_1', \mu')$ 的情形下 c' 为 $c_1' ; c_2$。若 c 的类型为 $\langle \tau, n\rangle$，则根据规则 COMP1 存在 k 和 l，$k+l=n$，使 c_1 和 c_2 的类型分别是 $\langle \tau, k\rangle$ 和 $\langle \tau, 1\rangle$。如果 c' 为 c_2，则 c 形如 $skip$ 或 $x := e$，若为前者则 $k=1$，因此 c' 的类型由归纳假设可知为 $\langle \tau, n-1\rangle$；后者的计算步数由 e 的类型 $\langle \tau, k\rangle$ 中的 k 决定，即 $k+1$，因此 c' 的类型由归纳假设可知为 $\langle \tau, n-k-1\rangle$。若 c 的类型为 $\langle \tau_1, \tau_2\rangle$，则其类型由规则 COMP2 决定，即存在 τ_3、τ_4、τ_5 和 τ_6，且 $\tau_4 \subseteq \tau_5 \wedge \tau_2$，$\tau_6 \subseteq \tau_2$，$\tau_1 \subseteq \tau_3 \wedge \tau_5$，使得 c_1 和 c_2 的类型分别是 $\langle \tau_3, \tau_4\rangle$ 和 $\langle \tau_5, \tau_6\rangle$。如果 c' 为 c_2，根据规则 SUBSUMP，由 $\langle \tau_5, \tau_6\rangle \subseteq \langle \tau_1, \tau_2\rangle$ 可知其类型为 $\langle \tau_1, \tau_2\rangle$；又若 c' 为 $c_1' ; c_2$，根据归纳假设有 c_1' 的类型为 $\langle \tau_3, \tau_4\rangle$，因此 c' 的类型为 $\langle \tau_1, \tau_2\rangle$。

若 c 形如 $c_1 \parallel c_2$，由归纳假设知若 $(c_1, \mu) \rightarrow (c_1', \mu')$，则 c_1' 的类型与 c_1 相同，所以 $(c_1 \parallel c_2, \mu) \rightarrow (c_1' \parallel c_2, \mu')$ 后 $c_1' \parallel c_2$ 的类型保持不变；$(c_2, \mu) \rightarrow (c_2', \mu')$ 的情形是类似的。

下面考虑类型系统的可靠性，对并发程序而言，仅仅考虑程序的初始格局是不够的，还需要考虑程序执行中产生的中间格局，需保证一个良类型的程序在任意存储上的程序轨迹不会产生非空的 $race(\mathscr{C}_{HL})$。

引理 4 对程序 P，若其在类型环境 Γ 下可赋类型，则 P 的 $race(\mathscr{C}_{HL})$ 为空集。

证明：对线程池 $|P|$ 的体积进行归纳证明：

当 $|P|=1$(P 中不含并发操作)时，在格局中线程数为 1，所以不存在多线程在某个低级变量上形成竞争访问关系，集合 $race(\mathscr{C}_{HL})$ 必为空集，结论自然成立。

下面考虑多线程的情形。需证明当 $|P|=s$ 时，初始格局的 $race(\mathscr{C}_{HL})$ 为空集。

若 $s=2$，则 P 形如 $c_1 \parallel c_2$。按 $c_1 \parallel c_2$ 的类型分情形讨论：

若类型为 $\langle \tau, m\rangle$，则存在 $\tau_1, \tau_2, \tau = \tau_1 \wedge \tau_2, m_1, m_2, m = m_1 + m_2$，$\Gamma \vdash c_1 : \langle \tau_1, m_1\rangle$，$\Gamma \vdash c_2 : \langle \tau_2, m_2\rangle$。注意到 $c_1 \parallel c_2$ 并非隐藏线程，由类型规则 PAR1 可知，它们之间不存在对隐藏低级变量的竞争关系，所以当前格局的 $race(\mathscr{C}_{HL})$ 是空集，满足免于隐藏竞争并发的要求。若类型为 $\langle \tau, \tau'\rangle$，情况与上则

分析类似，不再赘述。若 $c_1 \parallel c_2$ 中存在隐藏线程且类型为$\langle \tau, \tau' \rangle$，则(1)根据 PAR3，存在 $\tau_1, \tau_2, \tau_3, \tau \subset \tau_2, \tau' = \tau_1 \vee \tau_3, \Gamma \vdash c_1^{\#} : \langle \tau, \tau_1 \rangle, \Gamma \vdash c_2 : \langle \tau_2, \tau_3 \rangle$；或者(2)根据 PAR4，存在 $\tau_1, \tau_2, \tau_3, \tau_4, \tau = \tau_1 \wedge \tau_3, \tau_1 \supseteq \tau_3, \tau' = \tau_2 \vee \tau_4, \Gamma \vdash c_1 : \langle \tau_1, \tau_2 \rangle, \Gamma \vdash c_2^{\#} : \langle \tau_3, \tau_4 \rangle$，并且两个线程的自由低级变量的交集为空($FL(c_1) \cap FL(c_2) = \emptyset$)。在情况(1)中，$\Gamma \vdash c_1^{\#} : \langle \tau, \tau_1 \rangle$，$c_1$ 为隐藏线程，而 $\Gamma \vdash c_2 : \langle \tau_2, \tau_3 \rangle$，它所影响的变量级别 $\tau_2 \supset \tau$，所以两个线程在 τ 的级别上没有竞争关系，所以 $race(\mathscr{C}_{HL})$ 为空集；在情况(2)中，$\Gamma \vdash c_2 : \langle \tau_3, \tau_4 \rangle$，虽然 $\tau_3 \subseteq \tau_1$(意味着隐藏线程 c_2 所影响的变量类型低于线程 c_1 所影响的变量类型)，但是附加条件($FL(c_1) \cap FL(c_2) = \emptyset$)保证了两个线程对 c_2 中的隐藏低级变量没有竞争访问关系，所以当前格局下的 $race(\mathscr{C}_{HL})$ 为空集。

归纳假设为当 $|P| = s > 2$ 时，当前格局的 $race(\mathscr{C}_{HL})$ 为空集。不妨假设存在 $i = \mu(lst) + 1 \leqslant s$，使 $P(i) = c_1 \parallel c_2$，则经轮转调度后 P 分别为 $P' = \{P(1), \cdots, P(i-1), c_1, \cdots, P(s), c_2\}$，当前被调度线程分别为 c_1。需证明对 $P' = \{P(1), \cdots, P(i-1), c_1, \cdots, P(s), c_2\}$，其 $race(\mathscr{C}_{HL})$ 为空集。首先证明 $P' - \{c_2\}$ 满足定理结论，而后证明 P' 亦满足。下面仍然就 $c_1 \parallel c_2$ 的类型分情形讨论：

若类型为$\langle \tau, m \rangle$，则存在 $m_1, m_2, m = m_1 + m_2$，$\tau_1, \tau_2, \tau = \tau_1 \wedge \tau_2, \Gamma \vdash c_1 : \langle \tau_1, m_1 \rangle$，$\Gamma \vdash c_1 : \langle \tau_2, m_2 \rangle$。注意到 $c_1 \parallel c_2$ 并非隐藏线程，所以它们各自也不是隐藏线程，并且 $FL(c_1 \parallel c_2) = FL(c_1) \cup FL(c_2)$，由归纳假设可知此集合不包含格局中出现在其它隐藏线程中的低级变量，所以 $race(\mathscr{C}_{HL})$ 仍然是空集，因此满足免于隐藏竞争并发的要求。若类型为$\langle \tau, \tau' \rangle$，情况与上则分析类似，不再赘述。

若 $c_1 \parallel c_2$ 中存在隐藏线程且类型为$\langle \tau, \tau' \rangle$，则(1)存在 $\tau_1, \tau_2, \tau_3, \tau \subset \tau_2, \tau' = \tau_1 \vee \tau_3, \Gamma \vdash c_1^{\#} : \langle \tau, \tau_1 \rangle, \Gamma \vdash c_2 : \langle \tau_2, \tau_3 \rangle$，$c_1$ 为隐藏线程，其影响类型 τ 与 $c_1 \parallel c_2$ 相同，所以将 $c_1 \parallel c_2$ 替换为 c_1 后，$P' - \{c_2\}$ 保持了 P 的性质。同时由于归纳假设保证了对任意属于 $c_1 \parallel c_2$ 的低级变量不会出现在其它隐藏线程中，所以线程 c_2 与 $P' - \{c_2\}$ 中各隐藏线程之间不存在低级变量上的竞争访问关系。因此 P' 的 $race(\mathscr{C}_{HL})$ 是空集。(2)存在 $\tau_1, \tau_2, \tau_3, \tau_4, \tau = \tau_1 \wedge \tau_3, \tau_1 \supseteq \tau_3, \tau' = \tau_2 \vee \tau_4, \Gamma \vdash c_1^{\#} : \langle \tau_1, \tau_2 \rangle, \Gamma \vdash c_2 : \langle \tau_3, \tau_4 \rangle$，并且两个线程的自由低级变量的交集为空($FL(c_1) \cap FL(c_2) = \emptyset$)。由 $\tau = \tau_1 \wedge \tau_3$ 可知 $\tau \subseteq \tau_1$，而 $\Gamma \vdash c_1^{\#} : \langle \tau_1, \tau_2 \rangle$，由归纳假设可知 $c_1 \parallel c_2$ 虽然是隐藏线程，但其与 P 中其它线程并无隐藏竞争关系，所以影响类型更高的线程 c_1 在格局 $P' - \{c_2\}$ 中亦不会导致隐藏竞争的出现。同理，无论 c_2 是否为隐藏线程，它的影响类型 τ_3 亦不低于 τ，因而格局 P' 的 $race(\mathscr{C}_{HL})$ 为空集。

定理 2(类型系统的可靠性)　对程序 P，若其在类型环境 Γ 下可赋类型，则 P 在 Γ 下免于隐藏竞争并发。

证明：由类型保持定理可知 P 的程序轨迹中任意格局的程序分量都是可赋类型的，并且由引理 4 知在所有可能出现的格局中隐藏线程的竞争变量集合 $race(\mathscr{C}_{HL})$ 均为空集。因此 P 是在 Γ 下免于隐藏竞争并发的。

7　实例

7.1　实例分析

本节对利用本文类型系统进行类型检查的过程给出一个例子。以下为两个线程并行组合的例子，例 1：

c_1 : l_1 : L :=1; ‖

c_2 : {if h : H

then {skip; skip; skip;}

else skip;

l_1 :=0;}

为讨论方便将两个线程分别标记为 c_1 和 c_2。

首先 c_1 的类型由规则 ASS 确定，$\Gamma \vdash c_1 : \langle L, 1 \rangle$。

其次，对线程 c_2，由规则 SKIP 和 COMP 可知条件控制的两个分支的类型分别为$\langle H, 3 \rangle$和$\langle H, 1 \rangle$，继而由规则 IF2 可知条件控制的类型为$\langle H, H \rangle$。同时有 $\Gamma \vdash l_1 := 0 : \langle L, 1 \rangle$。对于条件控制和赋值的顺序组合，根据规则 COMP3 有 $\Gamma \vdash c_2^{\#} : \langle H, L \rangle$，即 c_2 为隐藏线程。

最后，因为 $l_1 \in race(c_1 \parallel c_2) = FL(c_1) \cap FL(c_2)$，$c_1$ 和 c_2 的并行组合不能应用规则 PAR4，所以程序不能通过类型检查。反之如果将 c_1 中的变量 l_1 换为 $l_2 \notin FL(c_2)$，则由规则 PAR4 可知，即使 c_2 中高级变量对低级变量的赋值存在时间上的影响，程序仍然是可赋类型的。

7.2　与相关工作的比较

本节利用近年来提出的并发程序信息流的类型系统对例 2 进行类型检查，并和本文的方法进行比较。作为比较的对象，所提方法的对象语言与本文基本一致。与例 1 不同，例 2 中 c_2 的变量 l_1 被替换为 $l_2 \notin FL(c_1)$，替换后的代码在本文的类型系统中是可赋类型的。

c_1 : l_1 :=1; ‖

c_2 : {if h

then {skip; skip; skip;}

else skip;

l_2 :=0;}

例 2　首先考察文献[11,12]中的类型系统。这两篇文献的类型系统中对条件分支语句的类型规则为：

$$\frac{\Gamma \vdash e : \theta, \Gamma \vdash P_i : (\tau, \sigma) cmd, \theta \subseteq \tau}{\Gamma \vdash \text{if } e \text{ then } P_0 \text{ else } P_1 : (\tau, \theta \vee \sigma) cmd}$$

不难发现此规则要求条件分支的两个分支所修改的变量的安全级别必须相同。例 2 的 c_2 中的条件分支将无法通过类型检查。

其次考察文献[2]，其要求如果在条件分支的判断语句中包含高级变量，那么两个分支在执行时间上必须平衡：

$$\frac{\Gamma \vdash e : \tau, \Gamma \vdash P_i : \tau \; cmd \; n}{\Gamma \vdash \text{if } e \text{ then } P_0 \text{ else } P_1 : \tau \; cmd \; n+1}$$

而 c_2 中显然两个分支不平衡，所以例 2 不能通过类型检查。

综上，现有针对内部时间信道的类型系统在处理条件控制结构时对分支的平衡性过于严格，并没有考虑时间信道的产生和非平衡控制流有关。其次，在这些方法中没有考虑线程对变量的竞争访问方式，禁止在不平衡的条件下控制语句后存在对低级变量的赋值行为。实际上，例 2 说明在并发程序中内部时间信道的产生是非平衡控制流与低级变量赋值的不恰当组合以及对低级变量的竞争访问共同导致的。相比之下，本文的类型系统对这些内部时间信道的产生原因做了更为细致的划分，使其具有宽容性。

结束语　本文提出了一种验证多线程程序内部时间信道的类型系统，在类型系统中引入了对隐藏线程的判断，消除了产生内部时间信道的必要条件，允许低级线程之间的竞争访

问，构造了一个宽容的类型系统；在独立于调度模型的情形下证明了类型系统的可靠性。本质上说，本文的方法借助于静态分析提高了类型检查的能力。

在将来的工作中我们将扩展本文方法的应用范围，研究更为接近实际的并发程序语言，包括在语言中引入方法调用、指针等语法构件；开发相应的类型检查算法。进一步研究在基于类型理论的方法中更多地融入静态分析的思路，提高类型检查的精确度。

参 考 文 献

[1] Sabelfeld A, Myers A C. Language-based information-flow security[J]. IEEE Journal on Selected Areas in Communications, 2003, 21(1): 5-19

[2] Smith G. Improved typings for probabilistic noninterference in a multi-threaded language[J]. J. Comput. Secur., 2006, 14(6): 591-623

[3] Russo A, et al. Closing internal timing channels by transformation[C]// Proceedings of the 11th Asian computing science conference on Advances in computer science: secure software and related issues 2007. Tokyo, Japan: Springer-Verlag, 2007

[4] Terauchi T. A Type System for Observational Determinism [C]// Computer Security Foundations Symposium, 2008. CSF '08. IEEE 21st. 2008

[5] Russo A, Sabelfeld A. Securing interaction between threads and the scheduler in the presence of synchronization[J]. Journal of Logic and Algebraic Programming, 2009, 78(7): 593-618

[6] Zdancewic S, Myers A C. Observational determinism for concurrent program security [C] // Computer Security Foundations Workshop 2003. Proceedings 16th IEEE. 2003

[7] Sabelfeld A. The Impact of Synchronisation on Secure Information Flow in Concurrent Programs[C] // Revised Papers from the 4th International Andrei Ershov Memorial Conference on Perspectives of System Informatics. Akademgorodok, Novosibirsk, Russia: Springer-Verlag, 2001: 225-239

[8] Volpano D, Smith G. Probabilistic noninterference in a concurrent language[C] // Computer Security Foundations Workshop 1998. Proceedings 11th IEEE. 1998

[9] Huisman M, Worah P, Sunesen K. A temporal logic characterisation of observational determinism [C] // Proceedings of the 19th IEEE Workshop on Computer Security Foundations. 2006

[10] Mantel H, Sands D, Sudbrock H. Assumptions and Guarantees for Compositional Noninterference [C] // Computer Security Foundations Symposium 2011 IEEE 24th. 2011

[11] Boudol G, Castellani I. Noninterference for concurrent programs and thread systems[J]. Theoretical Computer Science, 2002, 281 (1/2): 109-130

[12] Barthe G, Nieto L P. Formally verifying information flow type systems for concurrent and thread systems[C]// Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering 2004. ACM: Washington DC, USA, 2004: 13-22

(上接第 148 页)

[6] Haluk T, Salim H, Wu M Y. Performance-effective and Low-complexity Task Scheduling for Heterogeneous Computing[J]. Parallel and Distributed Systems, 2002, 13(3): 260-274

[7] Gilbert C S, Edward A L. A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures[J]. Parallel and Distributed Systems, 1993, 4(2): 75-87

[8] Cao J W, Stephen A J, Sunhash S, et al. GridFlow: Workflow Management for Grid Computing[C] // 3rd IEEE International Symposium on Cluster Computing and the Grid. Tokyo, IEEE Computer Society, May 2003: 198-205

[9] Berman F, et al. New Grid Scheduling and Rescheduling Methods in the GrADS Project[J]. Parallel Programming, 2005, 33 (2): 209-229

[10] Hönig U, Schiffmann W. A Meta-algorithm for Scheduling Multiple DAGs in Homogeneous System Environments[C]// Parallel and Distributed Computing and Systems. Dallas, IEEE Computer Society, November 2006: 147-152

[11] Zhao He-nan, Sakellariou R. Scheduling Multiple DAGs onto Heterogeneous Systems[C] // 20th International Parallel and Distributed Processing Symp. Piscataway: IEEE, 2006

[12] Yu Zhi-feng, Shi Wersong. A Planner-Guided Scheduling Strategy for Multiple Workflow Applications [C]// International Conference on Parallel Processing, Portland, 2008. IEEE Computer Society, September 2008: 1-8

[13] 孔维梁，刘清堂，杨宗凯，等. 基于动态 QoS 的 Web 服务组合[J]. 计算机科学，2012，39(2)：268-272

[14] Amudha T, Dhibyaprabha T T. QoS Priority Based Scheduling Algorithm and Proposed Framework for Task Scheduling in a Grid Environment [C] // International Conference on Recent Trends in Information Technology, Chennai, 2011. IEEE, June 2011: 650-655

[15] Ankur K, Soo-young L. A Stochastic Approach to Estimating Earliest Start Times of Nodes for Scheduling DAGs on Heterogeneous Distributed Computing Systems [C]//19th International Parallel and Distributed Processing Symposium, Denver, 2005. IEEE Computer Society, April 2005: 1530-2075

[16] Kwok Y, Ahmad I. On Multiple Processor Task Scheduling Using Efficient State Space Search Approaches[J]. Parallel and Distributed Computing, 2005, 65(12): 1515-1532

[17] 王万森. 人工智能原理及其应用(第 2 版)[M]. 北京：电子工业出版社，2005：115-120

[18] 付云虹. 基于 backfill 的并行计算作业调度算法研究[D]. 长沙：湖南大学，2007

[19] 田国忠，肖创柏，徐竹胜，等. 异构分布式环境下多 DAG 工作流的混合调度策略[J]. 软件学报，2012，23(10)：2720-2734