计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2019.20180513
56(9): 2001-2011, 2019

二进制翻译正确性及优化方法的形式化模型

傅立国 庞建民 王 军 张家豪 岳 峰
(数学工程与先进计算国家重点实验室(战略支援部队信息工程大学) 郑州 450002)
(flg_njlg@163.com)

Formal Model of Correctness and Optimization on Binary Translation

Fu Liguo, Pang Jianmin, Wang Jun, Zhang Jiahao, and Yue Feng
(*State Key Laboratory of Mathematical Engineering and Advanced Computing* (*Strategic Support Force Information Engineering University*), *Zhengzhou* 450002)

Abstract Binary translation has attracted more and more attention in the fields of architecture optimization, program performance optimization, security analysis, software transplantation and so on. Although there are totally different requirements on the application of binary translation techniques in these various fields, the researchers are always focusing their attention on two aspects: the correctness of translation and the efficiency of translation. The correctness of translation refers to the equivalent logical functions between the program before and after the translation, so an appropriate formal model for the translation needs to be built at first. According to the current research needs on correctness and optimization of binary translation technique, twice mapping model based on subsequent relations could have been constructed firstly; then the properties and construction methods of a correct process on translation should have been described formally; based on the formal description, the optimization methods of translation would be classified and analyzed on their diverse characteristics and various properties finally. This paper provides a more powerful theoretical support for the further research on the strategy pattern combination of translation methods and optimization methods in the binary translation technology by constructing a formal model of binary translation based on correctness and optimization.

Key words binary translation; translation correctness; efficiency; translation optimization; formal model

摘 要 二进制翻译在体系结构设计、程序性能优化、安全性分析以及软件移植等领域的研究中备受关注.不同应用场景对二进制翻译的需求各不相同,却总聚焦于翻译的正确性和翻译的效率2个方面.翻译的正确性用于评判翻译前后程序在逻辑功能上是否具有等价性,而等价的证明依赖于适当的形式化模型.为了满足研究二进制翻译正确性以及翻译优化方法对理论模型的需求,对已有理论模型进行了深入的剖析,并进一步构建了新的基于后继关系的映射模型.该模型既能够形式化地描述正确翻译的二进制翻译过程所具备的性质和构造方法,也可以在翻译过程形式化描述的基础上对翻译过程优化方法的特征和性质进行描述.通过构建翻译正确性及翻译过程优化方法的形式化模型,为二进制翻译技术中关于翻译过程的实现以及优化方法的策略组合等进一步研究提供了更强的理论支撑.

收稿日期:2018-07-17;**修回日期**:2019-04-17
基金项目:国家自然科学基金项目(61472447)
This work was supported by the National Natural Science Foundation of China (61472447).
通信作者:庞建民(jianmin_pang@126.com)

关键词 二进制翻译;翻译正确性;效率;翻译优化;形式化模型

中图法分类号 TP314

传统意义上的二进制翻译是指将源平台指令集的指令序列翻译成其他平台指令集的指令序列的过程.随着虚拟化技术应用的兴起,二进制翻译技术作为跨指令集架构虚拟化实现的核心技术被广泛关注[1].二进制翻译中的动态翻译方式既能够在程序运行期间动态地解释代码片段,也能够结合程序运行的时空局部性特征实现性能调优,亦可以再现程序在源平台复杂的运行时行为,从而成为体系结构设计、程序性能优化、安全性分析以及软件移植的研究热点.

为实现二进制程序到其他平台的有效移植,需要确保二进制翻译过程的正确性和有效性.正确性要求源平台二进制程序的逻辑功能在宿主平台上被予以等价的实现;有效性关注的是源平台代码在宿主平台上运行时的效率.因此,二进制翻译技术在应用研究中的问题总是可以归结为翻译的正确性问题和翻译的效率问题.

现有相关问题的研究大多是对限定翻译系统工程实现的探讨,局限于具体翻译系统缺陷的修复.以选定的翻译器或者翻译框架为条件,不利于深入研究二进制翻译的正确性和运行效率.如用于精确处理中断、精确处理异常还原的二进制翻译器,则更关注其翻译的正确性;而用于软件无源移植的翻译器则更关注其运行效率[2-3].

在设计二进制翻译器时,将翻译过程的正确性和效率作为整体进行研究,能够预见和避免类似限定框架下存在的种种问题.例如,在新型体系架构研发的研究中,为了解决传统的通过功能模拟的方法收集典型行为特征速度很慢的问题,使用了动态二进制翻译技术来加快程序行为分析.然而该方法却也引入了分析误差,可见追求翻译的有效性时翻译的正确性会受到影响[4].因此,翻译正确性和翻译效率以及两者之间关联性的研究对于二进制翻译系统框架的设计具有重要意义,特别是在对一些优化方法的正确性进行理论验证的过程中.

在现有的二进制翻译正确性验证相关的研究中,采用自动化形式化的验证方法已经比较普遍.如文献[5]于2011年提出了对动态翻译过程中SIMD指令的翻译进行运行时验证的方法;文献[6]于2015年提出同时支持动态翻译和静态翻译的验证方法,却局限于仅对部分制定翻译结果的验证;文献[7]于同年提出了使用形式化的方法讨论操作数和操作数约束的优化过程,然而该模型对于二进制优化方法的描述不具备通用性.因此为了能够对翻译的正确性和优化方法的有效性进行一致性的研究,本文对翻译过程及其优化方法构建适当的形式化模型.

1 基于指令解释的映射模型

文献[8]借鉴文献[9]中可计算代数系统中同构的性质,构建了机器模拟过程和动态二进制翻译过程的形式化模型.该模型将机器模拟的过程定义为由源机器状态和指令解释函数组成的有序对到由宿主机器状态和指令解释函数组成的有序对的映射;继而在此基础上,将动态二进制翻译过程归约为机器模拟过程的动态优化方法,从而描述了动态二进制翻译过程相较于机器模拟过程所优化的对象和方法.

1.1 机器模拟的形式化描述

定义 1. 机器描述 1.机器为 $M=(S,I,\gamma)$.其中,S 表示机器的状态集;I 表示机器的指令集;γ:$I\times S\to S$ 表示机器指令在指定机器状态下的解释函数.

当指令 $i\in I$ 作用于状态 $s\in S$,其后继状态 s' 可表示为 $s'=\gamma(i,s)$.

定义 2. 解释函数.指令路径 $i^*=\langle i_0,i_1,\cdots,i_{m-1}\rangle\in I^*$ 的解释函数为 $\gamma^*:I^*\times S\to S$,表示进行 m 次 γ 的迭代操作,即 $\gamma(i_{m-1},\cdots,\gamma(i_2,\gamma(i_1,\gamma(i_0,s_0)))\cdots)=\gamma^*(i^*,s_0)$.

1.2 机器模拟的性质

在宿主机器模拟源机器,即对源机器的指令执行行为进行模拟.因此,源机器的组成要素需要在宿主机器上一一实现.用 $M_R=(S_R,I_R,\gamma_R)$ 表示源机器,用 $M_H=(S_H,I_H,\gamma_H)$ 表示宿主机器,设 φ_S:$S_R\to S_H$ 为源机器状态集合到宿主机器状态集合的映射.设 $\varphi_I:I_R^*\to I_H^*$ 为源机器指令路径集合到宿主机器指令路径集合的映射.机器模拟中源机器和宿主机器关于状态和指令路径的映射关系如图1所示.

源机器 M_R 执行指令序列 i_R^* 由状态 s_R 到达状态 $\gamma_R^*(i_R^*,s_R)$,γ_R^* 为指令序列 i_R^* 的解释函数.宿主机器 M_H 对 M_R 的模拟行为表现在2个方面:

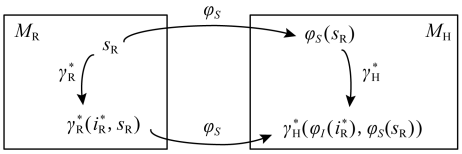


Fig. 1 The mapping relation in machine simulation

图 1 机器模拟中的映射关系

1)对状态 s_R 和 $\gamma_R^*(i_R^*, s_R)$有相对应的状态 $\varphi_S(s_R)$ 和 $\varphi_S(\gamma_R^*(i_R^*, s_R))$;2)对指令序列 i_R^* 有映射得到的本地指令序列 $\varphi_I(i_R^*)$,该序列的解释函数为 γ_H^*.有效的模拟过程中,机器状态、指令序列及其解释函数应满足

$$\varphi_S(\gamma_R^*(i_R^*, s_R)) = \gamma_H^*(\varphi_I(i_R^*), \varphi_S(s_R)). \quad (1)$$

源机器 M_R 从状态 s_R 经过指令序列 i_R^* 得到状态 $\gamma_R^*(i_R^*, s_R)$ 的过程,被宿主机器 M_H 从状态 $\varphi_S(s_R)$经过指令序列 $\varphi_I(i_R^*)$到状态 $\gamma_H^*(\varphi_I(i_R^*), \varphi_S(s_R))$的过程所模拟.

文献[8]将动态二进制翻译归为机器模拟过程的动态优化技术,指出其实质是源机器平台上执行路径在宿主机器上的再次编译,并参照式(1)给出了动态二进制翻译的一般性质.

1.3 现有模型存在的不足

基于指令解释的映射模型通过式(1)描述了机器模拟过程和动态二进制翻译过程的一般性质.文献[10]在协同设计虚拟机的模拟研究中使用文献[8]中的模型对模拟过程中的一些性质给予了形式化的论证,文献[5-7]在使用这种描述方法的同时也暴露了该模型存在的一些不足.

1.3.1 确定的指令路径

在该模型中,宿主机器对源机器的模拟和翻译默认源机器执行前的初始状态以及之后执行的指令序列是确定的.而在实际翻译过程中,仅加载待翻译可执行文件后的初始状态以及机器指令执行行为的相关解释是已知的.在无法确定具体指令执行路径时,便无法通过映射得到宿主机的指令执行路径.

1.3.2 指令路径的映射

假设源机器执行的指令路径是确定的.通过映射得到宿主机器的指令路径,执行该指令路径使得宿主机器从初始状态得到满足条件的结束状态.然而对于模拟和翻译过程,具体执行的指令路径只是一个实现.以指令和固定的指令序列作为映射元素的翻译方式,破坏了指令序列更高层次等价翻译的可能性,这对于构建更高形式优化方法的描述是不利的.

1.3.3 指令路径的执行效率

该模型主要用于描述模拟和翻译过程的正确性.仅使用状态映射函数及其优化函数描述本地指令路径的执行效率,不足以深入讨论翻译方法的优化问题.将翻译过程表示为指令路径间的映射过于抽象,无法刻画动态翻译过程的开销.该模型的翻译过程与源机器具体的指令路径相关,优化也是基于具体指令路径的.不同的指令或者指令路径其指令解释函数可能是相同的,然而这些等价的指令(序列)若按照指令为单位的翻译方式,其在宿主机器翻译得到的指令(序列)之间执行开销的差距可能极其大.翻译过程的优化也包括求解更为有效的本地实现.然而在指令映射的方法不足以描述生成的本地指令路径的效率.

综上所述,在分析翻译效率时,该模型中的状态映射函数及其优化函数并不能涵盖翻译过程中所有重要的因素.能够对翻译过程正确性和优化方法有效性进行分析的形式化模型,需要对影响翻译正确性和翻译效率的所有因素的概念加以定义,然后结合这些因素对二进制翻译的正确性和翻译中优化方法的定义和性质进行讨论.

2 二进制翻译过程的抽象及其形式化

二进制翻译过程需要源机器架构特征描述和二进制可执行文件一同作为输入.由于宿主机器只能执行本地指令序列,待翻译的程序在宿主机器可以有 2 种存在形式:1)完整的宿主平台可执行文件;2)需要其他程序配合的零散的指令序列片段.然而不论是以哪种形式,本地存储的指令序列在执行时都是宿主机器对二进制程序在源机器上执行过程的模仿.

系统性地讨论翻译过程的正确性和翻译过程的优化,需要对二进制翻译过程进行抽象.对二进制翻译过程的抽象即对源机器程序的执行过程以及对该过程模仿过程的抽象,且两者在本质上都是机器的指令执行过程.所以抽象机器上指令执行的过程是刻画二进制翻译过程的基础.

2.1 机器的抽象描述

基于指令解释的映射模型本质上是利用指令的操作语义描述指令执行过程中机器状态的变化过程,以算数表达式的形式用指令解释函数定义了指令执行的过程.机器 $M=(S, I, \gamma)$从状态 $s \in S$ 执行

指令 i 到达状态 $s'\in S$ 的过程,指令解释函数的操作语义可描述为

$$\langle i,s\rangle\rightarrow s'. \tag{2}$$

然而,基于这种描述的指令解释函数在描述指令序列的解释函数时,在对每条指令的解释函数进行复合操作的时候需要明确每条指令的解释函数,这需要满足以下 2 点:一是指令序列是已知的;二是每条指令的解释函数是可表示的.为了弱化指令路径对指令迭代执行过程描述的限制,需要对指令执行过程进一步抽象.

文献[11]将动态二进制翻译过程概括为查找、翻译和执行 3 个过程的迭代.该描述与处理器从存储器取指、译指(取值)和执行(返值)的执行过程相符.因此,指令执行的路径是和机器状态密切相关的,每个机器状态的后继状态是通过执行从该状态中获得的指令而得到的.因此机器指令的执行过程可以描述为 $\gamma(i,s)=\gamma(f(s),s)=s'$.其中 f 表示从执行前的状态 s 获取指令 i 的过程.在状态解释函数 γ 中,2 个参数都是状态 s 的函数,可将其缩略为

$$\gamma(s)=s'. \tag{3}$$

如此,函数 γ 定义了一个机器状态间的映射关系,描述了机器状态在指令执行中的状态迁移过程.这种描述是对机器执行指令的行为加以抽象,而并非对具体指令的执行效果进行描述.此时 γ 不再是机器指令解释函数,而是机器指令执行的指称函数.

使用状态转移函数可以表示任意指令和指令序列的执行.根据指称语义及其操作语义的一致性[12],结合式(2)可将指令 $i\in I$ 的状态转移函数表述成

$$\gamma(i)=\{(s,s')\mid\langle i,s\rangle\rightarrow s'\}. \tag{4}$$

式(4)中使用函数 γ 描述了指令 i 的指称语义,称为指令 i 的指称.当讨论范围扩大为整个指令集 I 时,可以将指令集 I 的指称语义以函数的形式表述为

$$\gamma=\gamma(I)=\bigcup_{i\in I}\{(s,s')\mid\langle i,s\rangle\rightarrow s'\}. \tag{5}$$

此时,函数 γ 作为指令集 I 的指称定义了一个关系集合 $\bigcup_{i\in I}\{(s,s')\mid\langle i,s\rangle\rightarrow s'\}$,这与式(3)所构建的状态元素关系是一致的.通过指称语义的抽象,可以归纳具有相同功能的指令和指令序列,从而刻画机器指令和指令序列执行过程的本质而不必考虑指令和指令序列的具体形式.

综合以上分析,对机器定义如下:

定义 3. 机器描述 2.机器为 $M=(S,\gamma)$.其中,S 表示机器的状态集;$\gamma:S\rightarrow S$ 表示机器指令集的指称函数.

指称函数 γ 定义了状态集 S 上的一个二元关系,表示指令执行前后状态的变换关系,故也可以将其称作状态迁移函数.

2.2 程序执行的抽象

重新定义机器描述后,程序执行的过程也可以得到进一步的抽象.设加载完可执行文件后的机器状态为 $s_0\in S$,程序执行的过程可以用机器指令的迭代执行过程表示,而程序执行过程状态和程序执行结果的求解问题则被归约为状态 s_0 后继状态的求解过程.

状态 s_0 及其后续状态可以表述为有序集合 $\{\gamma^0(s_0),\gamma^1(s_0),\gamma^2(s_0),\cdots,\gamma^n(s_0),\cdots\}$.$s_0\in S$,$n\in\mathbb{N}$,其中 $\gamma^0(s_0)=s_0$.为了深入刻画程序执行的性质,需要对该集合中元素间的关系进一步讨论.

定义 4. 状态间的二元关系.在机器状态集合上存在二元关系$\leqslant$,对于状态 a 和状态 b,如果 $b\in\gamma^*(a)$,记作 $a\leqslant b$.

性质 1. 当集合 $\gamma^*(s_0)$中不存在 2 个相等的元素时,集合 $\gamma^*(s_0)$上的二元关系$\leqslant$为全序关系.

证明.

1) 对于任意状态 $s=\gamma^0(s)\in\gamma^*(s)$,因此有 $s\leqslant s$,自反性可证.

2) 设有状态 a 和状态 b 使得$(a\leqslant b)\wedge(b\leqslant a)$.根据二元关系$\leqslant$的定义可知 $b\in\gamma^*(a)$和 $a\in\gamma^*(b)$同时成立.当 $b\in\gamma^*(a)$成立,由函数 γ^* 的定义,不妨设 $b=\gamma^i(a)$,$i\geqslant 0$,带入前提条件 $a\in\gamma^*(b)$有 $a\in\{\gamma^{i+0}(a),\gamma^{i+1}(a),\gamma^{i+2}(a),\cdots,\gamma^{i+n}(a),\cdots\}$,$i\geqslant 0$,因为 $a=\gamma^0(a)$,所以仅当 $i=0$ 时$(a\leqslant b)\wedge(b\leqslant a)$满足.此时 $b=\gamma^0(a)=a$,反对称性可证.

3) 设集合 $\gamma^*(s_0)$中有状态 a,b,c 满足 $a\leqslant b$ 且 $b\leqslant c$,设 $a=\gamma^i(s_0)$,$b=\gamma^j(s_0)$,$c=\gamma^k(s_0)$.$i,j,k\in\mathbb{N}$,根据二元关系$\leqslant$的定义可知,$i\leqslant j$,$j\leqslant k$ 即 $i\leqslant k$,可得 $a\leqslant c$,传递性可证.综上,二元关系$\leqslant$为偏序关系可证.

4) 集合 $\gamma^*(s_0)$中的任意元素均存在 $\gamma^i(s_0)$,$i\geqslant 0$ 的表达形式,因此任意 2 个元素总是可比较的.

证毕.

状态迁移函数定义了指令执行前后机器状态的一个二元关系.因此程序在机器上的执行过程可以用全序集$\{\gamma^*(s_0),\leqslant\}$来表示,其中 s_0 为程序加载完后的机器初始状态.

2.3 模拟行为的抽象

模拟是二级制翻译的核心本质,抽象描述程序执行过程后,就可以对模拟的概念加以阐述.

设源机器 $M_R=(S_R,\gamma_R)$ 在初始状态 $s_0\in S_R$ 执行程序 p 得到状态有序集合

$$S_R^p=\{\gamma_R^0(s_0),\gamma_R^1(s_0),\gamma_R^2(s_0),\cdots,\gamma_R^m(s_0),\cdots\},m\in\mathbb{N}.$$

设宿主机器 $M_H=(S_H,\gamma_H)$ 在初始状态 $s_0'\in S_H$ 执行程序 q 得到状态有序集合

$$S_H^q=\{\gamma_H^0(s_0'),\gamma_H^1(s_0'),\gamma_H^2(s_0'),\cdots,\gamma_H^n(s_0'),\cdots\},n\in\mathbb{N}.$$

定义 5. 指令序列的模拟.若满足 2 个条件:1)存在映射 φ_S 使得集合 S_R^p 中的任意元素在集合 S_H^q 中有相对应的像;2)若集合 S_R^p 中有状态 a,b 满足 $(a\leqslant b)\wedge(a\neq b)$,且有 $a'\in\varphi_S(a),b'\in\varphi_S(b)$,那么集合 S_H^q 中的状态 a',b' 满足 $(a'\leqslant b')\wedge(a'\neq b')$.此时,则称在宿主机器 M_H 上的程序 q 是对源机器 M_R 上程序 p 的一个模拟.

2.4 二进制翻译过程的抽象

二进制翻译最终实现了指令序列到指令序列的映射.根据宿主机器对源机器上程序模拟执行的定义,可以定义二进制翻译:

定义 6. 指令序列的翻译.设 $(a\leqslant b)\wedge(a\neq b)$,$a,b\in S_R^p$,映射 φ_S 使得状态 a 在宿主机器 M_H 上的像中存在 $a'\in\varphi_S(a)$ 满足 $(\gamma^*(a')\cap\varphi_S(b))\neq\varnothing$.宿主机器 M_H 上存在状态序列 $\{a',\gamma^1(a'),\gamma^2(a'),\cdots,\gamma^{n-1}(a'),b'\}$,$\gamma^i(a')\notin\varphi_S(b)$,称状态 a' 到状态 b' 所执行的指令序列为源机器 M_R 上状态 a 到状态 b 所执行的指令序列的一个翻译.而生成宿主机器上这段指令序列的过程称为对源机器上对应指令序列的翻译过程.

3 二进制翻译的正确性

翻译源机器平台上的程序,是为了在宿主机器平台上得到具有相同逻辑功能的程序.因此翻译得到程序的逻辑功能相同与否决定了翻译的正确性.然而当讨论翻译正确性的时候,需要先统一描述程序的功能,继而给出关于逻辑功能等价的定义,最终探讨正确的二进制翻译过程具有的一般性质.

3.1 等价的一般性描述

程序逻辑功能的等价可以通过相同逻辑功能的正确性证明[13].即通过证明不同程序满足相同逻辑功能的正确性,从而证明程序间在某种程度上具有等价性.

证明程序 a,b 等价,需要证明对于任意前置条件 P 和后置条件 Q,$\{P\}a\{Q\}$ 可证当且仅当 $\{P\}b\{Q\}$ 可证.该描述方法借用了程序功能验证描述的一般形式——霍尔三元组,将程序的功能等价定义为功能相同.然而此处定义的前置条件和后置条件是不加限制的.因此,当讨论对象限制在源机器和宿主机器之间的指令序列执行其指称语义的等价,有效的二进制翻译具有如下性质.

性质 2. 设机器 M_R 到机器 M_H 的正确翻译是指对于 M_R 上任意满足关系 $(p\leqslant q)\wedge(p\neq q)$ 的状态 p,q,在 S_H 上总存在状态满足关系 $(p'\leqslant q')\wedge(p'\neq q')$ 的状态 p',q' 与之相对应,可表示为

$$trans(M_R,M_H)\Leftrightarrow(\forall p,q\in S_R.(p\leqslant q)\wedge(p\neq q)\Rightarrow\exists p',q'\in S_H.(p'\leqslant q')\wedge(p'\neq q')).\quad(6)$$

式(6)中,限制状态 p,q 不相等,因为对于不改变机器状态的指令进行翻译是没有意义的.性质 2 和式(6)是对具体程序在不同机器上翻译、模拟性质的推广.因而在此不予证明.通过将源机器和宿主机器抽象为 2 个偏序集合 $\{S_R,\leqslant\}$ 和 $\{S_H,\leqslant\}$,翻译过程和程序模拟执行过程被有效区分,有利于二进制翻译正确性的深入分析.

3.2 状态映射函数的性质

在给出二进制翻译正确性的一般性质之后,我们需要对翻译过程所具有的性质进行分析.即为了实现源机器上程序的模拟执行,如何在宿主机器上构建与之相对应的状态序列.

根据性质 2 和式(6)的描述,当成功地翻译并模拟执行了一个程序时,即映射 φ_S 将偏序集 $\{S_R,\leqslant\}$ 上的任意全序集 $\{\gamma_R^*(s_0),\leqslant\},s_0\in S_R$ 映射到偏序集 $\{S_H,\leqslant\}$ 上的全序集 $\{\gamma_H^*(s_0'),\leqslant\},s_0'\in S_H$.对于状态 $a,b\in\gamma_R^*(s_0)$ 以及状态映射函数 φ_S,总有状态 $a'\in(\varphi_S(a)\cap\gamma_H^*(s_0'))$ 和状态 $b'\in(\varphi_S(b)\cap\gamma_H^*(s_0'))$ 且 $a\leqslant b\Rightarrow a'\leqslant b'$ 总为真.因此,映射 φ_S 总是可以将偏序集 $\{S_R,\leqslant\}$ 上的任意全序映射为偏序集 $\{S_H,\leqslant\}$ 上的一个全序.根据保序映射的定义,此时偏序集 $\{S_R,\leqslant\}$ 到偏序集 $\{S_H,\leqslant\}$ 的映射 φ_S 是保序映射.

然而在具体到任意代表程序执行的全序集到全序集的映射时,根据函数单调性的定义,可知机器状态的映射函数 φ_S 是单调的.根据单调的性质可知,映射函数 φ_S 的反函数 $\overline{\varphi_S}$ 必然存在.

性质 3. 态映射函数 φ_S 是保序映射,单调且存在反函数.

尽管状态映射函数 φ_S 作用的对象是源机器的

机器状态,然而其本质是对源机器状态集合中的后继关系到宿主机器状态集合中的后记关系的再现.

因此,翻译的正确性体现在状态映射函数 φ_S 能否构建一个满足模拟源机器程序执行的状态集在本地的一个模拟实现.

3.3 状态转移函数的表达形式

综合 3.2 节所述,映射 φ_S 描述了状态转移函数 γ_R 和 γ_H 执行前后状态之间的关系.

根据式(6)可知,满足正确性的翻译过程中所实现的二元关系的映射,是从 γ_R 到 γ_H 的映射,即从已知源平台的指令序列到不确定的宿主平台指令序列的映射.此时可以用描述机器状态序列所蕴含的指令序列指称语义的函数来描述二进制翻译中映射关系的保持.

状态转移函数 γ_H 所描述的前后继关系显然是受 γ_R 的描述约束的.根据紧致性定理,若对状态转移函数 γ_R 作用域中每个子函数均有对应的状态转移函数满足有效翻译间的映射关系,那么每个子函数映射得到函数的集合所构建得到状态转移函数 γ_H 是满足翻译的有效性的.

因此,复杂机器指令的指称语义函数可以使用其子函数指称语义函数集合的形式描述[14].描述翻译过程需要明确状态转移函数的表示形式.根据工程实践中待翻译指令的可确定性,本文将状态转移函数的描述划分为基于子函数的描述和基于连通图的描述 2 种.

3.3.1 基于子函数的描述

$\gamma=\{f_i(s)\}, s\in A_i$ 给出了符合机器指令功能描述的状态转移函数表示形式.根据状态转移过程中所执行的指令进行分类,将状态转移函数划分为若干个描述不同指令功能的子函数.其中,$\{A_i \mid i\in I\}$表示对状态转移函数 γ 在状态集合 S 上的 1 个划分.当划分所得的子集数目有限时,根据紧致性定理,如果对于机器状态集 S 划分的每个子集,状态转移函数 γ 都有 1 个有效的子函数,那么状态转移函数 γ 对于机器状态集 S 的状态转移关系定义是有效的.

然而这种划分需要对作用的状态先进行关于定义域划分的归类,因此在使用状态函数计算时需要先进行归类计算.

3.3.2 基于连通子图的描述

使用基于子函数的描述方式通常能够较为完备地描述状态转移函数.然而当研究范围局限于部分机器状态时,基于子函数的描述方式会构建大量冗余的关系定义.并且当需要对状态转移函数进行修改且状态后继关系不确定的时候,使用子函数进行描述所定义的状态转移函数更加复杂.

$\gamma=\{\{[\gamma^n(s_i),\gamma^{n+1}(s_i)]\}\}, s_i\in A_i$ 给出了符合特定指令执行序列描述的状态转移函数表示形式,其中$\{[\gamma^n(s),\gamma^n(s)]\}, n\in\mathbb{N}$表示状态 s 及其所有后继状态作为状态转移函数定义集所定义的后继形成的集合.其现实意义即每个初始状态对应机器执行过程中一个特定的执行序列.

此时可以使用有向图来描述二元后继关系.对满足$(a\leqslant b)\wedge(a\neq b)$的机器状态 a,b,以 a 为起点、b 为终点作有向图.因此,当给定任意状态 s 时,求解 s 所定义状态转移图中所有的连通关系,通过求解以 s 为起点,依次求解每个后继节点的后继,得到 1 个后继关系图,该图是状态转移状态图的1 个连通子图.这种描述则是将机器状态根据后继关系划分成了若干个连通子图.

状态转移函数适合描述待翻译指令具有不确定性的动态翻译过程,因为待翻译指令不确定,机器状态序列间的前后继关系较为复杂,使用子函数能够较好地描述这种复杂的前后继关系;而在可以静态翻译的程序中,分析得到确定的指令序列片段,因此每个指令序列执行前后机器状态的变化较为简单,使用状态连同图描述较为便捷,可以简化新构建的状态转移函数.

3.4 指称语义等价状态序列的构建

在定义了状态转移函数的描述形式之后,在宿主机器构建满足翻译正确性的状态序列可以通过构建具有语义等价的指称函数实现[15-16].设 $\gamma_{RH}\subset\gamma_H^*$ 为满足 γ_R 在机器 M_H 翻译正确性的指称函数,若 γ_R 是基于子函数的表达形式,γ_{RH} 的构建过程如图 2 所示:

$$\begin{array}{ll}\gamma_R & \gamma_{RH}\\ \left\{\begin{array}{l} f_{R1}(s), s\in A_1\\ f_{R2}(s), s\in A_2\\ \vdots\\ f_{Rn}(s), s\in A_n\end{array}\right. & \left\{\begin{array}{l} f_{RH1}(s), s\in\varphi_S(A_1)\\ f_{RH2}(s), s\in\varphi_S(A_2)\\ \vdots\\ f_{RHn}(s), s\in\varphi_S(A_n)\end{array}\right.\end{array}$$

Fig. 2 Construction of state transition function based on subfunction description

图 2 基于子函数描述的状态转移函数构建

通过这种描述,将状态转移函数 γ_{RH} 的构建问题细化为对状态转移函数 γ_R 中子函数 $f_{Ri}(s), s\in A_i$ 对应的状态转移函数 $f_{RHi}(s), s\in\varphi_S(A_i)$ 的构建.因为状态转移函数 γ_R 的作用域 S_R 为有限域,

其上的划分$\{A_i \mid i \in I\}$也是有限的,选择的划分方式决定了 i 的值,从而决定了构建状态转移函数 γ_{RH}的复杂程度.

这种描述的现实意义即按照机器状态转移的方式进行分类,从而可以对不同指令的处理过程进行分类讨论.如果 γ_R 是基于连通子图描述的,γ_{RH}的构建如图 3 所示:

$$\begin{array}{cc} \gamma_R & \gamma_{RH} \\ \left\{\begin{array}{l} \{[\gamma_R^i(s_1), \gamma_R^{i+1}(s_1)]\} \\ \{[\gamma_R^i(s_2), \gamma_R^{i+1}(s_2)]\} \\ \vdots \\ \{[\gamma_R^i(s_n), \gamma_R^{i+1}(s_n)]\} \end{array}\right. & \longrightarrow \left\{\begin{array}{l} \{[\varphi_S(\gamma_R^i(s_1)), \varphi_S(\gamma_R^{i+1}(s_1))]\} \\ \{[\varphi_S(\gamma_R^i(s_2)), \varphi_S(\gamma_R^{i+1}(s_2))]\} \\ \vdots \\ \{[\varphi_S(\gamma_R^i(s_n)), \varphi_S(\gamma_R^{i+1}(s_n))]\} \end{array}\right. \end{array}$$

Fig. 3 Construction of state transition function based on connected subgraph description

图 3 基于连通子图描述的状态转移函数构建

基于连通子图描述的状态转移函数构建过程如图 3 所示.其中,$s_i \in A_i$,$0<i<n$,$\{A_i \mid i \in I\}$是 γ_R 定义域的一个划分.此时状态转移关系 γ_{RH}的构建被归约为 2 个连通子图之间关系的构建.

当在宿主机上构建具有语义等价的指称函数之后,便可以使用代表宿主机指令指称函数的状态转移函数 γ_H 去实现构建的 γ_{RH},通过 γ_H 实现 γ_{RH}所构建的状态序列若满足和源机器序列语义等价,则满足翻译正确性[13].至此,二进制翻译的形式化模型可以描述如图 4 所示:

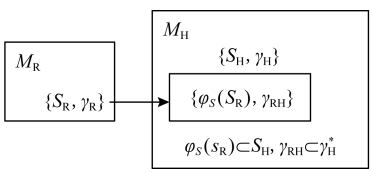


Fig. 4 Formal model of binary translation

图 4 二进制翻译的形式化模型

图 4 中明确表示出翻译过程即将源机器 M_R 定义的偏序集$\{S_R, \gamma_R\}$映射到宿主机器偏序集 M_H 定义的偏序集$\{S_H, \gamma_H\}$的过程,其映射所得的偏序集合$\{\varphi_S(S_R), \gamma_{RH}\}$满足 $\varphi_S(S_R) \subset S_H, \gamma_{RH} \subset \gamma_H^*$.

3.5 状态转移函数构建的正确性验证

图 4 构建了二进制翻译统一的形式化模型,然而在工程实现中,宿主机器状态转移函数构建的正确性验证与翻译器使用的翻译方式是相关的.

动态二进制翻译中代码发现过程与翻译过程是交叉执行的,因此可以处理执行路径较为复杂和不确定的指令序列.处理这种指令序列,使用基于子函数的形式描述源机器指令执行前后机器状态的变化是较为合适的.如翻译器 QEMU 自带的正确性验证程序 test-i386,即通过 x86 处理器指令的翻译测试验证翻译系统的正确性.

静态二进制翻译中代码发现过程相对于翻译过程是独立的,可以构建待翻译程序的执行路径流图.此时,使用基于联通子图的状态转移函数描述源机器上一段确定指令序列的状态转移函数是较为合适的.静态二进制翻译通常以指令段为基本单位,在翻译的过程中结合语义分析进行优化,最终翻译得到的结果很难使用基于指令翻译正确性的验证方法去验证翻译的正确性.因此,静态二进制翻译通常以翻译后指令最终执行结果作为正确性测试的结果.常用的测试集包括 nbench,spec2006 等.

4 优化过程的形式化描述

设机器 M_R,有 $\forall p \in S_R, q \in \gamma_R^*(p)$,为了获取指令序列 q 在初始状态 p 下的执行结果,可以通过正确的翻译,将该求解过程归约为宿主机器 M_H 上指令序列 $\varphi(q)=\gamma_H^*(\varphi(p))$的执行.然而在使用基于 γ_H 定义的状态序列描述 γ_{RH}所表述的指称函数时,其解并非唯一,即满足同一指称语义的指令序列可以有多个.此时为了获取状态序列中后续状态到达最快,也就是求解 $\gamma_H^*(\varphi_S(p))$速度最快的状态序列,需要对使用指令序列的指称函数计算后继状态的过程进行分析,从而确定翻译过程中影响翻译结果效率的各种因素.

本节从状态转移函数不同表述形式的使用、状态转移函数在宿主机器状态序列的实现以及状态转移函数的修改 3 个方面,阐述各种实现方法对翻译得到的状态序列效率的影响.

4.1 状态转移函数表述形式对效率的影响

翻译后得到的宿主机状态序列,其目标后继状态的计算过程越快,计算执行的速度越好.即生成的指称函数 γ_{RH}在使用基于 γ_H 定义的子函数实现时,其实现的复杂程度越低越好.状态转移函数 γ 同时也是指令序列的指称函数,具有 2 种描述形式.因此后续状态的计算形式也有 2 种.

首先给出使用基于子函数描述的状态转移函数计算后继状态的计算过程.

设 $\gamma=\{f_i(s), s \in A_i\}$,$0<i<n$,计算状态 $\gamma^{i-1}(s)$的后继状态 $\gamma^i(s)=\gamma(\gamma^{i-1}(s))$时,需要首先根据状态过转移函数 γ 的定义域对状态 $\gamma^{i-1}(s)$进行归类,

然后再选择对应的状态转移子函数进行计算.该过程与式(3)中表示从当前状态获取执行指令的过程相一致.此时后继状态的计算过程可以表述为

$$\gamma^i(s)=f_m(\gamma^{i-1}(s)),\gamma^{i-1}(s)\in A_m.$$

若将对当前状态归类的过程看作一个函数,该函数以当前状态为参数,查询当前状态在状态转移函数定义域中所属的子域,以该子域对应的子函数为返回值.设 δ 为状态归类函数,$\delta(s)=f_i,s\in A_i$,则基于子函数表示的状态转移函数 γ 计算后继状态 $\gamma(s)$ 的计算过程可表示为

$$\begin{cases}\delta(s)=f_i,s\in A_i\\ \gamma^i(s)=f_m(\gamma^{i-1}(s)),\gamma^{i-1}(s)\in A_m\end{cases}\Rightarrow$$
$$\gamma^i(s)=\delta(\gamma^{i-1}(s))(\gamma^{i-1}(s)). \tag{7}$$

式(7)中状态归类函数 δ 是一个关于 $\gamma^{i-1}(s)$ 的二阶函数.该函数抽象描述了基于子函数表示的状态转移函数做后继状态计算的过程,忽略了状态转移函数的表示形式,可以将任意起始状态 s_0 的后继状态的计算递归地表示为

$$\begin{cases}\gamma^0(s)=s_0,\\ \gamma^i(s)=\delta(\gamma^{i-1}(s))(\gamma^{i-1}(s)).\end{cases}$$

用 $\delta_i=\delta(\gamma^{i-1}(s))$ 表示第 i 次后继状态计算的状态转移函数,则后继递归计算递归地表示为

$$\begin{cases}\gamma^0(s)=s_0,\\ \gamma^i(s)=\delta_i(\delta_{i-1}\cdots(\delta(s_0)\cdots)).\end{cases}$$

此时状态转移函数定义了由若干状态子集划分的后继关系集合 $\gamma=\{\delta^{S_i},\delta^{S_j},\cdots\}$,其中状态集合 S_i,S_j 是状态全集 S 的1个有效划分中的不同元素.

由此可见,在针对部分指定状态的后继计算过程中,使用基于子函数表示的状态转移函数与使用基于连通子图表示的状态转移函数具有一致性.

性质4. 设有状态集合 S_i,如果其在源机器状态转移函数中有基于连通子图的定义,那么,在宿主机上构建状态转移函数时,针对状态集 S_i 的映射状态 $\varphi_S(S_i)$ 构建的状态转移函数,基于连通子图构建的状态转移函数效率会比基于子函数构建的状态映射函数效率高.

性质4的证明需要通过分析图5中使用2种不同状态转移函数计算后继状态的复杂程度.

假设源机器 M_S 上的状态集 S_i 有状态转移函数 $\gamma_R=\delta^{S_i}$,在计算后继状态 $\gamma_R^t(S_i)$ 时,若使用基于子函数表示的状态转移函数会先对状态集 S_i 进行归类计算求取状态转移子函数 f_m,然后再求取后继状态,如图5中虚线箭头所示;而基于同构图描述的状态转移函数,可以直接通过转移函数求解后继状态,如图5中实线箭头所示.因此,每次求得的 f_m 和对应的 $\delta_n^{S_i}$ 是等价的,基于同构图的状态转移函数蕴含有对应的状态转移函数信息,因此省略了选择或者生成对应转移函数的过程,性质4可证.

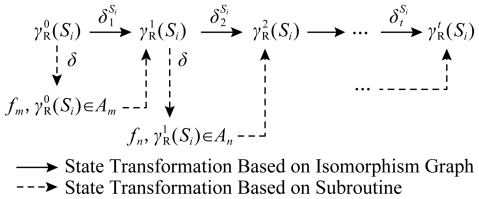


Fig. 5 Subsequent state calculation progress (1)

图5 后继状态计算过程(1)

在实际应用中,动态翻译在运行时翻译和维护代码的过程实现了模拟状态转移指令序列的生成,但从功能上并没有实现模拟源平台状态的转移,所以执行效率低于静态翻译.与此同时,也显示出了静态翻译的局限性,因为需要预测源机器执行的状态序列.

将指令及其解释函数抽象为状态转移函数,能够对二进制翻译过程进行更系统的刻画,主要是可以对机器模拟、动态翻译和静态翻译进行统一描述.而这种形式化的统一描述对于构建动静结合的二进制翻译模式是至关重要的.

4.2 状态转移函数在宿主机器状态序列的实现

在二进制翻译模型中,基于偏序集 $\{S_H,\gamma_H\}$ 翻译得到了偏序集合 $\{\varphi_S(S_R),\gamma_{RH}\}$.因此在实现 γ_{RH} 时,所使用的状态序列是需要满足 γ_H 所定义的后继关系限制的.这种限制体现在宿主机器体系结构特征对二进制翻译结果的限制.

这种限制形式化描述为:若 $\gamma_H=\{f_1,f_2,\cdots,f_n\}$,那么 $\gamma_{RH}=f_{\cdots}(\cdots f_j(f_i(s))\cdots),s\in S_H$.

现实中,任何机器平台最终只能执行该平台所能识别的指令和操作.基于子函数描述的状态转移函数 γ_H 抽象了宿主机器的体系架构,定义了一系列的功能(指令).由此可见,不同指令序列所对应的状态序列会影响 γ_{RH} 中后继状态计算的效率.这种影响和具体的机器架构相关,不做过多讨论.这种优化方式在应用中主要体现为寄存器映射一类的优化方法中,将源平台上使用频繁的存储单元(通常为寄存器)映射到宿主平台的寄存器上.因为使用频繁,节省了映射在内存中反复读写而占用的时间[17-18].

4.3 状态转移函数的修改

在二进制翻译中,有时为了获取更高的效率,会忽略宿主机器翻译执行过程中的部分状态是否与源

机器的执行状态相互对应、而只是关注在部分关心的状态节点是否能够相互对应.这种忽略源机器执行序列过程,而只在意部分节点对应的方法,是在具体问题中对源机器所定义的偏序集$\{S_R,\gamma_R\}$进行了修改、在删减部分不关心的状态之后,并使用其前驱元素和后继元素构建成新的状态转移函数,如此提高后继状态的计算速度.经过这种修改后,在宿主机器得到的执行状态序列与源平台的状态序列其对应关系不再严格满足,此时式(6)中关于正确性的描述是不成立的.为了使得翻译中正确性和优化带来的影响能够有区别的讨论,因此对二进制翻译的模型修改如图 6 所示:

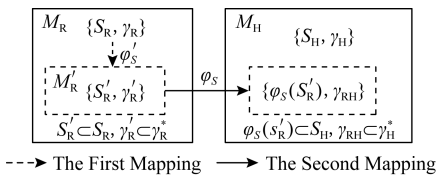


Fig. 6 The quadratic mapping process of binary translation

图 6 二进制翻译双映射过程

图 6 描述翻译过程中的双映射.在翻译过程中,首先根据具体问题缩小论域范围,即根据应用需求删减源机器的状态元素,修改其状态转移函数,该过程被抽象为映射 φ'_S.通过 φ'_S 将源平台所定义的偏序集$\{S_R,\gamma_R\}$映射为一个虚拟的机器 M'_R 所定义的偏序集$\{S'_R,\gamma'_R\}$,即取偏序集$\{S_R,\gamma^*_R\}$的 1 个子集.该过程的正确性由具体问题的性质决定,只删除和修改问题不关心的状态或者序列.此时当映射 φ_S 满足式(6)时,则问题关心的翻译结果也是正确的.

4.4 复合优化翻译模式

删减状态元素有利于降低问题的复杂程度,忽略一些不关注的机器状态及其处理过程.在现实中,根据用户的需求和关注的对象,如进程级的翻译系统在操作系统层及其以下做了本地虚拟化的处理,忽略了一些底层实现的翻译,而在软件无源移植的过程中相较于系统级的翻译系统具有巨大优势.

然而此类优化是基于对具体问题的求解需求进行分析的.处理方法是将不关心的过程的起始状态及其终止状态进行界定,为了保证问题讨论的完备性,对起始状态进行划分,划分的依据即对于问题的具体讨论,然后根据专业知识构建起始状态每个子类到终止状态的状态转移函数.

本质上,修改状态转移函数的优化方法也是需要预先知道优化状态及其状态转移函数的信息才能进行的.因此通过映射 φ'_S 得到的虚拟机器 M'_R 的状态转移函数 γ'_R 可以是复合了 γ_R 的 2 种表达形式而定义的.

为了保持描述的一致性,此处仍使用 M'_R 定义的符号讨论的后继状态的计算.尽管 M'_R 的后继状态求解也是 M_R 的求解,并且 M_H 模拟的是 M'_R.

设机器 M'_R 上的状态集 S_i 有基于同构图的状态转移函数 $\gamma=\{\delta_1^{S_i},\delta_2^{S_i},\delta_3^{S_i},\cdots\}$,$S_i\in A_i$,$i\in I$.当对其进行有限迭代计算总能得到后继状态集 S_j,并且对于状态集 S_i 的子集 S'_i 存在函数 $f_{S'_i}^{S_j}$,使得 $f_{S'_i}^{S_j}(S'_i)=\gamma'_R(S'_i)\subset S_j$ 成立.

图 7 中使用优化后的基于状态集与状态转移函数描述了后继状态的计算过程,使用该方法形成的状态转移函数同时使用了基于子函数以及基于同构图 2 中状态转移函数的表示方法.

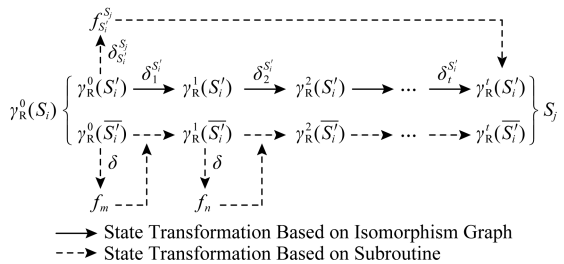


Fig. 7 Subsequent state calculation progress (2)

图 7 后继状态计算过程(2)

不同于图 5 中逐单条源指令后继状态的变换以及后继转移函数的推导,图 7 中计算状态 S_i 的子集 S'_i 的后继状态 $\gamma'_R(S'_i)$时所使用的状态转移函数是基于对状态子集 S'_i 在状态集 S_i 中进行归类计算生成的状态转移函数 $f_{S'_i}^{S_j}$,而 $f_{S'_i}^{S_j}$ 生成则需要对状态子集 S'_i 在基于同构图的转移函数进行分析,结合从状态集 S'_i 到状态集 S_j 的每个后继转移函数 $\delta^{S'_i}$ 迭代计算的操作语义,归纳为该过程对应的较为高级的表示形式,即状态转移函数 $f_{S'_i}^{S_j}$.与图 5 的计算过程相较,此类优化方法结合 2 种表示方式,对动态翻译过程中的局部问题进行了静态翻译,被称为动静结合的二进制翻译模式.这种综合的优化方法因为优化的灵活性好、针对性强,在二进制翻译的应用中被广泛应用.

然而,该优化的效果因为优化复杂度的提升而难以获得保证,针对状态子集 S'_i 构建的状态归类函数 $\delta_{S'_i}^{S_j}$ 所增加的开销,以及通过状态转移函数

$f_{S_i'}^{S_j}$ 替换 $\delta^{S_i'}$ 迭代计算所获取的收益,共同决定了能否获得整体上的收益.并且对于其他的状态子集 $\overline{S_i'}$ 而言,加了对于 S_i' 的划分过程,也是对整个翻译过程开销的增加.

综合来看,若是构建的状态归类函数 $\delta_{S_i'}^{S_j}$ 在映射到宿主机上能够获得本地的良好支持,通常是明显优于 $\delta^{S_i'}$ 迭代计算的效率.而状态子集 S_i' 构建的状态归类函数所增加的开销与 S_i 子集划分的数量相关.其他状态子集 $\overline{S_i'}$ 增加的关于 S_i 的划分过程的开销与具体实例中状态在 S_i' 和 S_i 中的比例相关.

在软件移植的应用中,对应用程序的主体采用动态的二进制翻译技术,对程序执行所依赖的过程函数,根据函数的特性,可以不做任何处理,模仿源平台调用过程的动态翻译;也可以根据函数说明,分析参数传递,做本地函数封装调用的替换处理;也可以对依赖的自定义库函数进行静态翻译成本地库函数,再封装调用替换.在上述相关工作中,特别是库函数的替换处理以及库函数的静态翻译,对该模型的性质进行了验证.

5 总 结

二进制翻译技术在体系结构优化、程序性能优化、安全性分析以及软件移植的研究中具有重要作用.

本文首先分析了基于指令解释的映射模型在二进制翻译关于正确性以及翻译效率研究中的不足,给出了基于后继关系的映射模型;继而定义了正确翻译并形式化地描述了正确翻译的过程;最后就翻译过程汇中翻译效率的优化方法分别进行了讨论.

在分析翻译正确性和翻译优化方法的过程中,结合实际也对翻译正确性的性质以及翻译优化方法的效果进行了论证.并在最后指出了二进制翻译研究中的热点以及难点——动静结合的二进制翻译模式.

本文所论述的二进制翻译中的正确性以及优化方法的形式化也是该翻译模式研究的基础.针对如何通过动静结合的二进制翻译模式有效提高二进制翻译技术的实用性将作为下一步的研究目标.

参 考 文 献

[1] Tan Yusong, Wu Qingbo. A study of virtualization and operating system technologies [J]. Computer Engineering & Science, 2011, 33(4): 62-68 (in Chinese)
(谭郁松, 吴庆波. 虚拟化与操作系统辨析[J]. 计算机工程与科学, 2011, 33(4): 62-68)

[2] Tang Feng, Wu Chenggang, Zhang Zhaoqing, et al. Exception handling in application level binary translation [J]. Journal of Computer Research and Development, 2006, 43(12): 2166-2173 (in Chinese)
(唐锋, 武成岗, 张兆庆, 等. 二进制翻译应用级异常处理[J]. 计算机研究与发展, 2006, 43(12): 2166-2173)

[3] Yang Hao, Tang Feng, Xie Haibin, et al. Library function disposing approach in binary translation [J]. Journal of Computer Research and Development, 2006, 43(12): 2174-2179 (in Chinese)
(杨浩, 唐锋, 谢海斌, 等. 二进制翻译中的库函数处理[J]. 计算机研究与发展, 2006, 43(12): 2174-2179)

[4] Zhao Tianlei, Tang Yuxing, Fu Guitao, et al. Accelerating program behavior analysis with dynamic binary translation [J]. Journal of Computer Research and Development, 2012, 49(1): 35-43 (in Chinese)
(赵天磊, 唐遇星, 付桂涛, 等. 利用动态二进制翻译加速应用程序行为特征分析[J]. 计算机研究与发展, 2012, 49(1): 35-43)

[5] Nakamura T, Miki S, Oikawa S. Automatic vectorization by runtime binary translation [C] //Proc of the 2nd Int Conf on Networking and Computing. Washington: IEEE Computer Society, 2011: 87-94

[6] Chen Jiuye, Yang Wu, Shen Boye, et al. Automatic validation for binary translation [J]. Computer Languages Systems & Structures, 2015, 43(C): 96-115

[7] William Y J, Bauman M A, Kao Fengjung, et al. Operand and limits optimization for binary translation system: America, US9021454 [P]. 2015-04-28

[8] Tröger J. Specification-driven dynamic binary translation [D]. Brisbane, Australia: Queensland University of Technology, 2005

[9] Rabin M O. Computable algebra, general theory and theory of computable fields [J]. Transactions of the American Mathematical Society, 1960, 95(2): 341-360

[10] Chen Wei, Xu Weixia, Wang Zhiying, et al. A formalization of an emulation based co-designed virtual machine [C] //Proc of the 5th Int Conf on Innovative Mobile and Internet Services in Ubiquitous Computing. Washington: IEEE Computer Society, 2011: 164-168

[11] Liao Yin. Dynamic binary translation modeling and parallelization research [D]. Hefei: University of Science and Technology of China, 2013 (in Chinese)
(廖银. 动态二进制翻译建模及其并行化研究[D]. 合肥: 中国科学技术大学, 2013)

[12] Winskel G. The Formal Semantics of Programming Languages [M]. Cambridge, MA: MIT Press, 2004

[13] Song Fangmin. Proof of equivalence of programs [J]. Journal of Computer Research and Development, 1989,26(4): 34-38 (in Chinese)
(宋方敏. 程序的等价性证明[J]. 计算机研究与发展, 1989, 26(4): 34-38)

[14] David M. Model Theory: An Introduction [M]. Berlin: Springer, 2007

[15] Shen Yuming, Ma Yue, Cao Cungen, et al. Faithful and full translations between logics [J]. Journal of Software, 2013, 24(7): 1626-1637 (in Chinese)
(申宇铭, 马越, 曹存根, 等. 逻辑之间的语义忠实语义满翻译[J]. 软件学报, 2013, 24(7): 1626-1637)

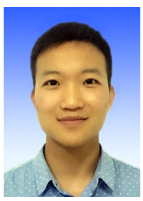
[16] Tao Qiuming, Zhao Chen, Guo Liang. Proving soundness of program transformations in optimizing compilation based on temporal logic [J]. Journal of Software, 2009, 20(8): 2074-2086 (in Chinese)
(陶秋铭, 赵琛, 郭亮, 等. 基于时序逻辑证明编译优化程序变换的保义性[J]. 软件学报, 2009, 20(8): 2074-2086)

[17] Dai Tao, Shan Zheng, Lu Shuaibing, et al. Register allocation algorithm of dynamic binary translation based on priority [J]. Journal of Zhejiang University: Engineering Science, 2016, 50(7): 1338-1346 (in Chinese)
(戴涛, 单征, 卢帅兵, 等. 基于优先级动态二进制翻译寄存器分配算法[J]. 浙江大学学报: 工学版, 2016, 50(7): 1338-1346)

[18] Wen Yanhua, Tang Daguo, Qi Fengbin. Register mapping and register function cutting out implementation in binary translation [J]. Journal of Software, 2009, 20(Supplement): 1-7 (in Chinese)
(文延华, 唐大国, 漆锋滨. 二进制翻译中的寄存器映射与剪裁的实现[J]. 软件学报, 2009, 20(增刊): 1-7)

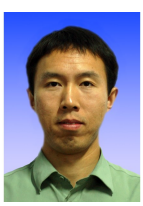
Fu Liguo, born in 1989. PhD. His main research interests include advanced compilation and high performance computing.

Pang Jianmin, born in 1964. PhD, professor, PhD supervisor. Senior member of CCF. His main research interests include computer architecture, information security and high performance computing.



Wang Jun, born in 1992. PhD candidate. His main research interests include computer architecture and high performance computing.

Zhang Jiahao, born in 1991. Master. His main research interests include computer architecture and high performance computing.



Yue Feng, born in 1985. PhD. Member of CCF. His main research interests include advanced compilation and high performance computing.