****

**硕士研究生学位论文中期检查报告**

**Mid-term Examination Report for SJTU Master Student**

|  |  |
| --- | --- |
| **学号 Student ID** | 121033910117 |
| **姓名 Name** | 刘思雨 |
| **学生类别 Degree Program** | 专业型硕士生 Professional Master |
| **学习形式 Study Mode** | 全日制 Full-time |
| **导师 Supervisor(s)** | 汪宇霆 |
| **专业 Major** | 计算机技术(085404) |
| **学院 School** | 电子信息与电气工程学院 |
| **考核日期 Date** |  |

填报说明 **Instruction**

1. 硕士研究生学位论文中期检查应通过[数字交大](http://my.sjtu.edu.cn/)在线提交申请，填写本表并上传系统。特殊情况下经研究生院事先同意，可不上传系统，并使用《上海交通大学硕士论文中期检查评审表》完成评审。

The application for thesis mid-term examination should be submitted online through [My SJTU](http://my.sjtu.edu.cn/). The student shall filled this form and upload it in the system. Under special circumstance, this form does not need to be uploaded and the review can be proceeded with the review form with prior consent from the graduate school.

1. 本报告请用A4纸双面打印，于左侧订在一起。各栏空格不够时，请自行加页。考核前提前一周送交导师、评审专家审阅。

This report should be printed with A4 papers and bound together on the left. If the space left is not enough, please feel free to add extra pages. The print version shall be sent to the supervisor, and the review committee members for review at least one week before the oral presentation.

1. 中期检查报告通过后，定稿版报告由研究生、导师各存档一份，无需上传系统。

Upon passing the review, the final version of this report shall be archived by the graduate student and his/her supervisors for future reference.

**中期检查报告 Mid-term Examination Report**

|  |  |
| --- | --- |
| 论文题目  Thesis Title | **基于静态单赋值中间语言的函数式编译器验证方法** |
| 研究课题来源  Source of Research Project | 请在合适选项前画√ Please select the proper options by “√”.  国家自然科学基金课题 NSFC Research Grants  国家社会科学基金 National Social Science Fund of China  国家重大科研专项 National Key Research Projects  其它纵向科研课题 Other Governmental Research Grants  企业横向课题 R&D Projects from Industry  自拟课题 Self-proposed Project |
| 论文开题日期  Thesis Proposal Date | 2022-12-14 |

1. **报告正文 Report。**请阐述开题报告以来学位论文研究工作的进展情况及所取得的阶段性成果，并简述下一阶段研究计划，不少于4000汉字。Please summarize your research progress and achievements since your thesis proposal as well as your plan for next step. No less than 3200 words if written in English.
   1. 研究工作进展情况

从论文开题至今，本课题研究工作按计划有序开展。首先，经过了对相关工作的调研，完成了直接风格PCF语言与CPS形式PCF语言的定义，设计并实现对源程序进行CPS转换的算法。接下来，根据已经确定的源程序及目标程序，设计并实现了从函数式程序到类似于LLVM IR的SSA程序编译算法的实现部分。最后，使用模拟技术对函数式源程序的CPS转换及CPS到SSA的转换进行了正确性验证。

1. 源程序语言的定义及CPS转换。

该编译过程的源语言是直接风格的PCF，目标语言是CPS语言。它们的语法归纳定义如下图1。PCF语言的语法表示形式参考了Gilles Dowek和Jean-Jacques Lévy对PCF语言的描述，包括了基本的λ演算、四则运算、分支跳转、不动点。

确定了直接风格的PCF代码项，下一步就是定义CPS代码项的语法。本文参考Andrew Kennedy中的CPS表示，定义了本文中使用的CPS语言，如图10。为了得到清晰确定的控制流，CPS形式的PCF程序引入了延续变量。

(a) 直接风格PCF代码项语法

(b) CPS代码项语法

图1 直接风格PCF与CPS语法定义

本文中对PCF、CPS及SSA语言都定义了其小步操作语义，即程序状态进行一步转换的规则，是程序状态之间最小的二元关系。使用表示从程序状态可以通过一步转换到达状态。对于一种编程语言，它可以有多种正确的操作语义。在本文中，为了使直接风格的PCF程序更明确地与 CPS 程序的语义对应起来，降低PCF到CPS转换过程的证明复杂度，在PCF程序状态 中包含了上下文信息。

现在我们来设计直接风格PCF语言到CPS的转换算法。它由函数描述，转换规则定义为，表示由输入的PCF代码项和参数经过编译得到输出。其中，是已经被使用的变量名，新生成的变量名应当与中已有的不同。使用的形式，表示当此时处理的代码项归约到一个值时，接下来要把它应用到哪个代码项上进行后续计算。具体转换规则如图2所示。

**Input** **Output**

**Input** : **Output**:   
(1). letval in   
(2).   
(3). .(letcont in )))

**where**   
(4). (ifz .(ifz

**where**   
(5). (op .(letop op y in )))

**where**   
(6). (let in   
(7). (fix letfun in

**where**

图2 PCF程序的CPS转换算法

1. 目标SSA语言的定义及CPS到SSA的转换。

本文中所使用的SSA目标语言是对LLVM IR进行简化的版本，保留了LLVM IR的基本结构，贴近工业界实际运用的SSA语言的结构，如图3所示。

该SSA语言的顶层翻译单元对应LLVM中的模块（Module）。它作为一个SSA命令式程序，由一系列组成，是函数的定义。函数定义在LLVM中非常复杂，有很多可选的参数，例如链接类型、垃圾收集器名称等。本文中只保留了函数定义中必须指定的内容，即函数名称、参数和代码块序列。如果函数的定义中没有指定入口代码块，就从第一个代码块开始。第一个代码块的标签默认为。代码块由代码块标签、若干函数，若干赋值命令和一条代码块结束命令构成。本文中使用的命令包括了四则运算表达式、对数值进行比较的、函数调用。指令的结构也和LLVM中一样，由代码块标签和变量值成对的序列组成。LLVM中有许多终止指令，本文中选取了基本的和指令，实现跳转和返回功能即可。

图3 SSA目标语言语法

对于这样的程序结构，程序计数器无法只通过数值定位到具体指令。将的结构设置为一个三元组，即，包括函数的标签、代码块的标签以及指令在代码块中的位置。这样，通过函数就能迅速定位到所指向的指令。

SSA语言的小步操作语义表示为程序状态的转换。中包括当前值、上一次块间跳转前的值、当前变量名与值的映射 、调用函数的指令栈。

CPS到SSA的转换算法由函数描述，读入CPS形式的函数式程序，得到SSA命令式程序。CPS与SSA程序结构上存在对应关系，CPS中的延续指定了当前代码片段求值完成后程序下一步该如何执行，即指明了控制流的方向。所以，将CPS程序转换为SSA程序时，只需根据这种结构上的对应关系设置跳转语句，使控制流抵达正确的代码块。

CPS到SSA编译算法的转换规则定义如图4所示。定义从CPS代码项及相关参数转换为SSA程序的转换规则。转换算法的输入包括CPS代码项、已经生成的部分SSA程序、当前程序计数器。在SSA程序基本代码块的生成过程中，部分代码块的标签可以使用延续变量的名字。但是，由条件语句生成的新代码块标签需要新的、不重复的名字。所以在转换过程中还需要维护条件语句基本代码块的计数器，以生成新的代码块标签。为了判断延续变量的具体类型是局部延续变量还是函数的返回延续，需要把引入延续变量信息存入中。基本上，局部延续变量对应着新的SSA代码块，当前计算结束后通过追加跳转指令使控制流跳转到标签为的代码块。

**Input** : **Output** :

where nil main, main,   
 **Input** : **Output** :   
(1). letval in

(2). letop op in

(3). letfun in

**where**   
(4). letcont in

**where**   
(5). ifz

**where** and

(6).

(7).

图4 CPS到SSA的转换算法

1. 基于模拟方法的转换算法正确性验证。

为了证明转换算法的正确性，我们需要将上述两步转换分别进行模拟，证明它们都实现了目标程序对源程序的语义保存。将整个转换过程定义为转换函数。该转换过程的语义保存可由定理1和定理2描述。其中，表示的值为，表示会发散。

定理1：

定理2：

我们使用前向模拟（Forward Simulation）技术证明对于源程序语义中的每一步，目标程序都有与之对应的步骤达到相匹配的程序状态。根据目标程序和源程序的性质（即目标程序为Determinate而源程序为Receptive）可以将正向模拟转换为后向模拟（Backward Simulation），从而推导出上述正确性定理。

首先，我们对两步转换分别进行正向模拟，然后组合在一起。

PCF到CPS的正向模拟不存在PCF走一步、CPS程序驻留的情况，因此使用多步模拟即可。对于某个PCF程序状态，按照小步操作语义走一步，相应的CPS程序状态走一步或多步与之等价。在对CPS到SSA的转换进行正向模拟时，CPS程序状态经过一步转换，相应的SSA程序状态可能是不变的，其他情况与多步模拟相同。它们的证明结构类似，所以，我们将会具体讨论CPS到SSA转换过程的正确性验证。

在该转换过程中，首先需要定义CPS和SSA程序状态的匹配关系。首先，在初始状态下，这种状态匹配应当是成立的。之后，在源程序和目标程序一步步执行的过程中，程序状态的匹配应当一直成立。

对于CPS到SSA的Star模拟，存在无限驻留问题，需要定义正确的测量函数。本文在测量函数 的设计上，选择了代码项中结构的数量，如图21。这是因为该结构的一步转化对应了SSA程序的零步转化。当SSA程序驻留时，转换为，SSA程序状态不变，且，测量函数在遇到零步对应转换时严格递减。

定理3：

对定理3进行证明时，使用归纳证明方法，对CPS程序状态走一步时所遵照的小步操作语义进行分类讨论。对于每一个子目标，构造出与相匹配的CPS程序状态，并找出由到需要经过哪些SSA小步操作语义步骤。

将两步转换的正向模拟进行组合，并根据语言性质推出反向模拟，即可证明整体编译过程的语义保存，即定理1和定理2。

* 1. 研究阶段性成果

该阶段完成了一个部分经过形式化验证的可信函数式编译器。该编译器前端读入PCF程序文本，经过多步编译后生成LLVM IR程序。将CPS到SSA的转换算法应用到直接风格的函数式语言PCF时，首先需要对源程序进行CPS转换，然后将这一步转换与CPS到SSA的编译算法组合起来，最后由SSA程序生成LLVM IR代码。主要由以下四个部分组成。

1. PCF语法分析器（Parser）。处理PCF程序需要得到其明确的语法结构。对于文本形式的PCF程序，首先用词法解析器将它解析为标记（token）流，再对标记流进行语法分析，得到直接风格PCF程序项。
2. 直接风格的PCF程序转换为CPS程序。将直接风格的PCF程序转换为CPS形式程序的算法主要是根据原PCF程序的代码项和上下文来确定的。
3. CPS程序转换为SSA程序。经过此步编译之后，输入的CPS程序项转换为一个包含主函数的SSA程序。
4. 由SSA程序生成LLVM IR代码。将前几步编译得到的类似于LLVM IR的SSA程序编译为Vellvm抽象语法树（AST），即可以利用Vellvm得到最终的LLVM IR程序。

其中，(2) (3) 部分的转换算法经过形式化验证，使用基于模拟的方法证明了目标SSA程序保存了原来PCF程序的语义。

编译框架及相关定理的形式化证明使用交互式定理证明器Coq实现。Coq主要是用OCaml实现的，支持数学断言的表示，并找到形式化的证明。其中，PCF语法分析器部分在OCaml中实现，相关代码位于Parser目录下，在顶端Coq模块中添加需要进行编译的PCF程序在Coq中的定义。Ocamllex和Ocamlyacc[25]是OCaml中词法和语法解析器的生成器，使用它们可以便捷地得到对PCF程序文本进行解析的OCaml代码。将Coq中的PCF程序添加到顶层模块后，就可以运行顶层编译模块top.v，将源程序编译到SSA程序。

在编译算法正确性证明过程中使用的小步操作语义是关系型的，即两个程序状态之间的关系。这样的设计对于证明来说很方便，但是无法直接运行程序得到结果。为了对操作语义和转换算法进行初步测试，使用Coq为三种语言分别构建解释器（Interpreter），从而能够执行相应的程序，得到程序返回的结果。其中，发散的定义要取决于对最大步数的限定，即解释器的fuel。解释器每走一步都会消耗一个fuel，如果fuel消耗完程序还没有终止或出错，就可以认为程序发散了，返回超时状态（Timeout）。进行解释器测试主要是为了找出操作语义和转换算法中的问题，为接下来的证明减少阻碍。形式化验证过程本身与解释器没有关系。

* 1. 下阶段研究计划

在下一阶段的研究中，主要计划是使用局部无命名的方法表示函数式程序中的变量。

当前函数式语言的设计中，程序项中使用有名字的变量表示方法。在这种表示方法中，每个变量都有一个唯一的名称，通常是字符串或符号。这种表示方法使代码项容易被人类理解，但在处理绑定变量时可能会引入名称冲突，需要进行重命名。如果使用局部无命名表示，变量不具有全局唯一的名称，而是使用一种特殊的标记，将绑定变量表示为de Bruijn形式的索引，从而避免了α等价类的引入。而自由变量保持为有名字的表示方式。如果能够完成该计划，就可以避免对源程序使用α重命名。

1. **成果清单 List of Achievements。**请列出开题报告以来或上次年度进展报告以来新发表的学术论文、授权专利、国际会议论文、专著等成果清单。作者、标题、杂志、卷、期、页码等信息请填写完整。Please provide a list of academic publications (papers, patents, international academic conference talks/presentations, monographs, etc.) since your thesis proposal. Information on author list, title, journal name, volume, number, and pages shall be complete.

国际会议论文：

Liu, S., & Wang, Y. (2023, June). *Verified Transformation of Continuation-Passing Style into Static Single Assignment Form*. In International Symposium on Theoretical Aspects of Software Engineering (pp. 20-37). Cham: Springer Nature Switzerland.

**本人承诺：报告中的内容真实无误，若有不实，愿承担相应的责任和后果。** **I hereby declare and confirm that the details provided in this Form are valid and accurate.** **If anything untruthful found, I will bear the corresponding liabilities and consequences.**

**学生签字/Signature of Student： 日期/Date：** 2023-10-16