本科生科研汇报

各位老师大家上午好！我是2011级本科生吴逸鸣，今天我来做我和吴阳怿同学的本科生科研基金答辩。我们今天的题目是《双向变换的理论与若干应用研究》。我个人的项目题目是《双向变换的应用－一个双向预编译器》。这是我今天汇报的纲要。

项目背景

（双向变换是什么）

（自动修复是什么，为什么需要双向预编译器）

代码自动修复工具的设计是最近学界一直讨论的问题。大型工程的开发者必须要确认，回溯并定位bug之后才能去修复这些bug。人工debug不仅耗时，也消耗大型公司的财力，因此人们最近一直在做代码自动修复工具的研究。比如GenProg, SemFix, Minthint，都是近来优秀的代码自动修复工具。

但是，所有的自动修复工具在实用化之前都不可避免地要面临一个问题：预处理，Preprocessing。对于C, C++等其他语言，一定要在预处理之后才能对代码进行分析。现在主流的代码修复工具，比如说现在学界公认最好的Genprog，选择直接跳过预处理，要求输入的代码，项目都是预处理后的代码。

这样的做法有两个坏处。其一，它大大增加了程序员使用自动修复工具的难度，程序员需要把项目都预编译了再放到工具里去。其二，这样自动修复工具生成的修复都是针对预编译后的代码。这些修改所指定的代码和预编译前的代码有时相去甚远。随着自动修复工具的不断革新，预编译带来的这两个问题的严重性将愈发明显。

然而，在做了大量的宏的使用调研后，我们发现如果让每个自动修复工具的设计者去解决预编译的问题并不是一件容易的事情。有的时候把宏返回到源代码有许多解；还有的时候不考虑宏地插入一些代码反而会错。

第一个例子FREE

第二个例子多阶宏

因此，我们认为我们需要一个双向预编译器。他不仅能遵守普通预编译器的规则进行正向预编译，还能在预编译后的代码被修改了以后进行反向预编译，把修改传播更新到预编译前的代码上。

我们还定义我们的这个双向预编译器的输入输出要满足这四个性质

（四个性质）
正确性
最优性
coverage
（报错一定时错的）完整性

于是，我们使用双向变换的设计原理，在这里设计提出一个双向预编译器。这样的一个双向预编译器不仅可以让代码修复的开发者们省去预编译的烦恼，我们相信他在未来软件工程的领域里也会有实用价值。

另外，我们还做了一次Linux内核上的宏的调研，把大型项目开发中会用到的宏分类并做了数字上的调查。并会在未来证明我们的双向预编译器可以处理所有类型的宏。

展开图技术

如果要设计一个双向预编译器，最重要的就是宏的处理。我们设计了一个宏展开依赖关系图，简称MEDG。让我们从一个Quick Example开始。

这个quick example中的宏需要展开两层。每次展开，我们会记录宏展开的依赖关系。这种关系表示为一根线，我们称之为expandFrom。依赖关系中有三个域，分别记录了这个token是被什么宏展开的，参数列表和展开他的宏在原来的代码里的位置pos在哪里。

对于一段代码我们会反复迭代构造这样的结构，可以证明这样的结构在预处理的问题里一定是树状的。如图所示，这里的a+a中的每个token都是被g的宏展开，参数分别是a,b，原来在代码中的位置是pos。

当出现修改时

当出现修改时，比方说这里插入了b+。我们先定位这次修改落在哪个宏展开的范围。如果在宏展开之外则直接返回。如果判定修改在一个宏展开的地方，则会依据展开的依赖关系，回溯找到最开始的宏，然后把其他宏尽可能保留地输出，做到最大程度地保存宏，减少修改后代码和原有代码之间的差距。

关于证明

下一步设想

1. 用Boost库实现完整的支持所有c规则的我们的双向预编译器。做出一个优秀的工具。现在已经可以实现正向的预编译。
2. Soundness性质证明，和真正符合编程人员习惯的宏变换的定义。

我在项目里主要做什么

1. 参与《基于上下文无关文法的可逆变换模型》的模型理论设计
2. 参与双向预编译的算法设计
3. 调研linux内核的宏使用情况
4. 正在使用Boost实现我们的双向预编译器

总结

感谢熊老师