Software Engineering and Applications 软件工程与应用, 2015, 4(5), 89-95
Published Online October 2015 in Hans. http://www.hanspub.org/journal/sea
http://dx.doi.org/10.12677/sea.2015.45012

Hans汉斯

Design and Application of a Process-Visible Compiler

Hanfei Lin, Xiwen Chen, Yufei Liang, Xiaoming Ju

Sophia Team of East China Normal University, Shanghai
Email: xmju@sei.ecnu.edu.cn

Received: Sept. 29th, 2015; accepted: Oct. 13th, 2015; published: Oct. 16th, 2015

Abstract

The course of Compiling Principles has always been difficult for students to understand, for the related algorithms are quite abundant and complex, and not all students possess the ability to comprehend those algorithms easily. In order to facilitate the learning of compiling techniques, we have designed a special compiler which displays the whole process of compiling for observing, and it's suitable to assist teaching work. Based on the subset of C grammar, this compiler can dynamically show the executing process of compiling algorithms and operation mechanism with abundant graphs and messages. As an assistant tool for teaching, not only does it contribute to the quick, efficient understanding of compiling principles, but it also adds some fun to the course.

Keywords

Compiler, Process-Visible, Operation Mechanism, Teaching-Assistant

过程可视化编译器的设计与应用

林涵菲，陈希文，梁雨霏，琚小明

华东师范大学Sophia实验室，上海
Email: xmju@sei.ecnu.edu.cn

收稿日期：2015年9月29日；录用日期：2015年10月13日；发布日期：2015年10月16日

文章引用: 林涵菲, 陈希文, 梁雨霏, 琚小明. 过程可视化编译器的设计与应用[J]. 软件工程与应用, 2015, 4(5): 89-95.
http://dx.doi.org/10.12677/sea.2015.45012

摘　要

针对编译原理课程中算法种类繁多、算法难度大、学生基础层次不统一、理解能力不一致的现状，如何让学生学习和掌握编译课程的算法与编译运行机制是教学过程中急待解决的问题。本文设计开发了一套基于教学辅助的可视化编译器，该编译器实现了C语言子集的编译功能，并将编译过程中的关键算法原理和运行机理通过图、表等方式实时、动态地展示在界面上。该编译器作为教学辅助软件，有助于学生快速、高效、清晰地理解编译原理算法，提高学生对编译原理课程的理解和兴趣。

关键词

编译器，过程可视化，运行机理，教学辅助

1. 引言

随着软件行业的日益革新，社会急需培养具有扎实编程基础的软件开发人员。编译原理课程一直作为计算机科学技术与软件工程专业的核心课程，它不仅能够帮助学生理解编译过程的实质，而且让学生在理解编译算法的基础上，思考复杂的算法原理，并设计优秀的算法，因此这门课程对于培养具有扎实基础的软件人才起着不可忽视的作用。

由于编译算法的比较复杂和抽象，在实际教学过程中，采用常规的多媒体课件教学很难让学生理解算法原理和运行机理，致使达不到教学效果。目前，有些可视化编译软件可用于辅助教学，尽管可视化的内容和实现的方法各有不同，但都是对编译过程中各功能模块结果的显示，不能体现编译算法运行机理的内在过程，致使辅助教学的效果不理想。

为了解决编译教学过程中存在的这一问题，本文设计了一套过程可视化教学辅助编译器。该编译器与现有编译器最大的不同之处在于能够动态地呈现编译过程中每个步骤的结果，并对编译算法在运行时的机理(如：语法分析中的分析栈)也能动态显示，这种方式让学生能够达到对编译算法和算法运行机理的深度理解。

2. 相关研究工作

高级语言编译器可视化在辅助教学方面的研究工作主要聚焦在中间结果的显示方面，文献[1]对编译原理经典算法进行了可视化实现，使用 LEX 完成词法分析，将获得结果保存到文件里，并在界面上展示语法分析过程中 First 与 Follow 集合的计算以及得到的 LL(1)分析表；文献[2]采取了一种不同的方法，它使用 YACC 在源程序的基础上插入可视化代码，之后按照进行常规编译。插入的代码被执行后即呈现出可视化效果，例如能够显示当前编译的语句位置；而文献[3]则重点在于编译过程的可视化，使得词法分析时状态机状态之间的相互转移、语法分析时的堆栈状态都能实时反映在界面上，同时还提供错误信息；文献[4]基于算符优先算法，设计了可进行优先级比较的可视化编译基本功能；文献[5]与文献[4]类似，设计了基于 LL(1)语法分析器的可视化；文献[6]和[7]均设计实现了 C 语言的可视化编译器，能够显示各过程中的运行输出结果。

上述编译算法可视化的研究各有侧重，其中文献[1]详细展示了语法分析时两个集合的计算过程，但只完成了简单的词法与语法的可视化工作；文献[2]利用 YACC 工具软件，可供插入的可视化代码有限，只能显示表面上的编译过程；文献[3]的不足的是语义以及中间代码等方面的可视化尚欠缺；文献[4]和[5]均是针对编译过程中的一个局部功能的可视化；文献[6]和[7]针对编译全过程的可视化设计，但仅对中间

过程中的结果进行了输出显示。

在上述可视化编译研究的基础上，本文结合辅助教学的实际需求，依据教科书[8]的编译算法，构建了一套过程与运行机理可视化的的编译系统：不仅完整实现了词法、语法、语义、中间代码生成的逻辑及其过程可视化功能，还支持编译算法软件运行时工作机理(如堆栈，动态语法树构建等)的动态显示，更加有力地辅助编译原理的教学。

3. 可视化编译器架构设计

3.1. 整体结构

本系统分为算法功能模块和可视化模块，其中算法功能模块与普通编译器相似，分为“词法分析器”、“语法分析器”、“语义分析器”和“中间代码生成器”四个部分。可视化模块与算法功能模块一一映射，将传统编译器中的算法实现的输入与输出结果动态地展现在可视化界面上，整个系统的架构如图 1所示。

教学辅助的可视化编译器除了实现传统编译器的词法分析、语法分析、语义分析、中间代码生成外，还包含对应功能模块的可视化功能：

1) 词法分析过程中动态显示构词缓冲区中指针的改变，高亮显示正在分析的词素，被分析过的词素按照不同的类型呈现不同的颜色；

2) 语法分析过程中动态展示语法分析栈的变化，包括堆栈内容、堆栈指针以及堆栈生长方向；

3) 语法分析过程中动态构造语法树，语义分析过程中动态改变语法树上被调用节点的颜色和属性值；

4) 中间代码生成过程中动态展现正在分析的语句和正在生成的三地址代码；

5) 符号表中的值随着编译的进行动态改变；

6) 实时产生和定位的错误处理系统。

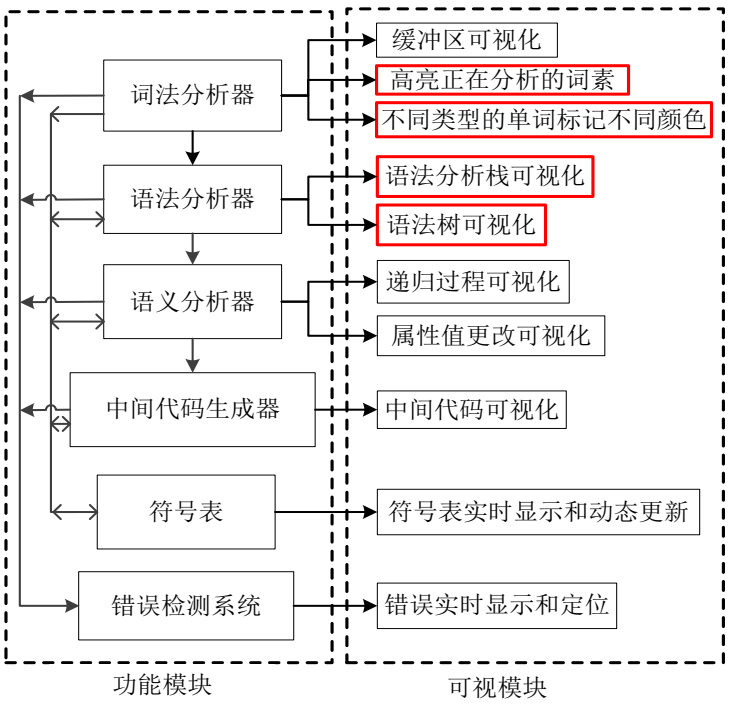

Figure 1. The architecture of compiler and visualization
图 1. 编译器和可视化的整体结构图

3.2. 编译算法及其可视化

3.2.1. 词法分析算法与可视化

词法分析器对源程序进行扫描，按照 token 类型如标识符、数字等对应的正则表达式，将整个程序代码分解成为一个个的 token 以供语法分析器进行后续分析。词法分析算法的 token 构造是在缓冲区内进行的，构词缓冲区即当前分析进度的可视化是关键。

缓冲区设置有两个指针：beginning 指针和 forward 指针，其中 beginning 指针指向当前 token 的第一个字符，在读取字符过程中，如果读入字符符合正则表达式，则 forward 指针依次向前移动，直到 beginning 与 forward 之间的内容构成一个 token。此时将获得的 token 传出，并重置 beginning 和 forward 以开始下一个 token 的分析。为了展现这一过程，在具体分析时添加了高亮显示当前分析到的词素(即一个 token 在源程序文本中对应的内容)的功能。被分析过的词素按照不同的 token 类型还可以呈现不同的颜色，如标识符是 X 色，数字是 Y 色。这样能更加直观地实时展示词法分析情况，方便学生跟踪分析过程。

3.2.2. 语法分析算法与可视化

语法分析器是编译器前端的核心结构。语法分析器中将调用词法分析器，对词法分析出来的一个个的 token 依据上下文无关文法进行分析。对不符合语法规则的语句，则需要进行报错处理。

本系统为手工编写的可视化编译器，采用了 LL(1)文法。依据文法计算所需要的 first 和 follow 集合，构造出 LL(1)语法分析表，并建立语法分析栈对输入的源程序进行分析。语法分析器调用词法分析器获得 token，并构造语句，在每个语句块完成时都会构建一棵语法树，并供后续语义分析和三地址生成所用。

语法分析栈的可视化。在语法分析过程中最重要的数据结构就是语法分析栈，这也是学生最难以理解的地方。而基于教学辅助的可视化编译器将语法分析过程中语法分析栈的每一个状态都动态呈现在界面上，这样学生可以清晰地观察到栈内输入输出的变化，进而快速理解 LL(1)语法分析过程。

语法树的可视化。在语法分析栈中每使用一个产生式，都会在语法树上相应的节点生成一棵新的子树。学生可以选择单步执行，这样当执行到产生式输出的步骤，语法树就会自动生长。学生可以直观地看到语法树的生长状态，使语法分析过程不再抽象难懂。

3.2.3. 语义分析与可视化

语义分析是编译过程中的重要阶段，主要功能包括类型检查以及符号表的更新。除此之外，语义分析将能够检查一些基本的语义错误，例如除数是否为零等，语义分析器还是中间代码生成的基础。

本系统的分析过程为语法制导翻译过程和递归调用语法树过程的结合，通过为每一个产生式定义属性文法，递归遍历语法树执行相应节点上的属性文法规则来实现。语法树由语法分析器传入，其中每个节点包含相应产生式、终结符或非终结符以及它们对应的属性。

递归过程可视化。基于教学辅助的可视化编译器通过改变正在被调用的节点的颜色来达到递归调用可视化的目的。学生可以选择单步调试也可以直接编译，二者效果相似。当编译器执行语义分析函数时，每递归调用一次，相应节点的颜色就发生变化，这样学生能直观地感受到语义分析的递归调用过程。

属性值更改可视化。在语义分析过程中，通过动态地改变节点上属性的值来引导学生理解综合属性的含义，每当一个节点进行了一次语法制导翻译时，节点上的属性值都有可能被更新，学生可以通过观察属性值的变化来理解语义分析的过程。

3.2.4. 三地址代码生成与可视化

三地址生成器是在语义分析的基础上进行的，同样采用递归调用的思路，在每个语句块分析结束时将其转化成等效的三地址代码。

三地址代码可视化。本系统的三地址生成器在每生成一个汇编语句时都会动态地将源代码和目标(三地址)代码同时展现在界面上，这样学生可以直观地理解汇编语句和源语句的翻译关系。

3.2.5. 符号表动态显示

符号表用来存放被编译的源程序中出现的标识符属性信息，符号表中所登记的信息在编译的不同阶段都要用到。比如在语义分析中，符号表所登记的内容将用于语义检查(如检查一个名字的使用和原先的说明是否一致)和产生中间代码。

动态更新显示。本系统中符号表中的信息和数据会随着编译器的分析过程动态地更新，学生可以实时观察到符号表的变化，便于学生理解符号表贯穿在整个编译过程中的实质，使学生不会产生诸如“符号表是词法分析器的数据结构”这样的误区。

3.2.6. 出错处理

出错处理系统是编译分析与调试过程中的重要组成部分。本系统有着完善的错误反馈机制，当次分析遇到的所有错误都会显示在一张表格中，报错信息尽可能准确、丰富，并附有行出错位置的行列号。双击某一条报错信息时，引发其出错的地方会在源程序文本中高亮显示，同时展示语法分析结果的文本框也会转到相应位置，方便老师以及同学们分析出错原因。

4. 教学辅助实例

下面通过一个简单的例子，参见图 2 中的源文件，按照可视化编译器的功能顺序，详细介绍可视化编译器的辅助教学功能。

4.1. 主界面

本编辑软件可通过菜单栏“文件”->“打开源文件”选择已有的 txt 文本文件，也可以在主界面中手动输入。选择是否查看语法树，然后点击“确定”按钮，编译器的其余功能按钮即会亮起。例如图 3 为导入了图 2 中的源文件之后的界面。我们的演示重点就是语法树，因此在此我们点击“看语法树”。

4.2. 词法分析标亮

可以选择点击“分析下一个”或者“分析到结尾”按钮，开始编译分析。图 3 中符号“n”底色被标黄，表示编译分析的当前位置，已经分析完成的 token，根据每个 token 属性的不同，标注了不同的颜色。

4.3. 语法分析语法栈可视化

每当分析出来一个 token 时，堆栈可视化窗口将动态显示语法分析中各符号压栈和出栈情况。图 4 显示的是编译过程中通过查 LL(1)表执行“arithexprprime->ε”语法规则，将栈顶元素“arithexprprime”弹出，然后压入“ε”，动态展现了一条语法规则在堆栈中的分析过程。

4.4. 语义分析语法树可视化

通过标亮语法树节点来表示递归语法树的过程，图 5 为正在递归计算赋值语句 arithexpr 的值。首先递归计算 assgstmt 的第三个孩子节点 arithexpr 的值，然后更改 assgstmt 第一个孩子节点对应的符号表中 ID 的值。通过层层递归现在计算到了 simpleexpr 的值。语义分析是编译原理中难度最大的模块，通过可视化的动态语法树，可以直观的看到语法树生成的过程和递归调用的过程。

4.5. 三地址代码生成可视化

在每个语句的语义分析结束之后，该语句将生成三地址代码，并在主界面上动态显示。图 3 中三地

址代码栏下的第 16 行、17 行为“sum = sum + j”这句话的三地址代码。

4.6. 符号表可视化

符号表不仅显示了所有符号的值，还记录了符号第一次赋值的位置。符号表的可视化能够动态看到符号表中符号及其各属性值的动态变化情况。

4.7. 错误处理可视化

在本例中，源文件中存在 4 个语法错误，错误列表清晰的显示了这 4 个错误的相关信息。图 3 中错误表格中的第三个错误为第 6 行第 44 列发生错误“不能输入单独的 ID”。双击该错误后，编译器定位并标亮了错误位置，便于对源代码的修改。

```
{
  i=10;
  j=100.00;
  while (i>0) {n=n+1;i=i-1}
  if (j>=50) then sum=sum+j; else {sum=sum+I;n;}
}
```

Figure 2. The source codes
图 2. 源文件代码

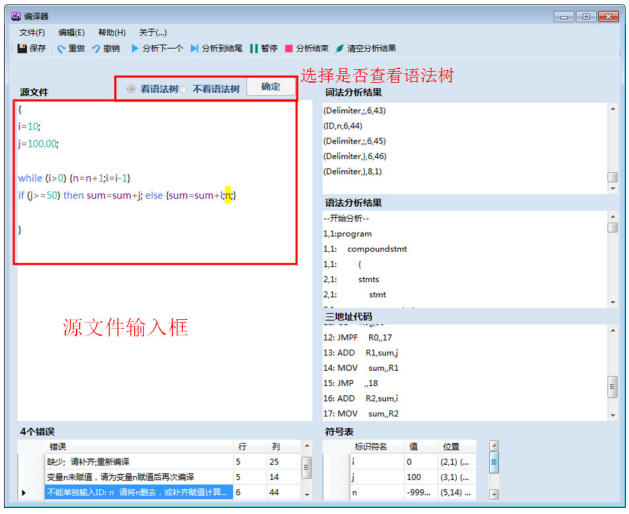

Figure 3. The main interface
图 3. 主界面

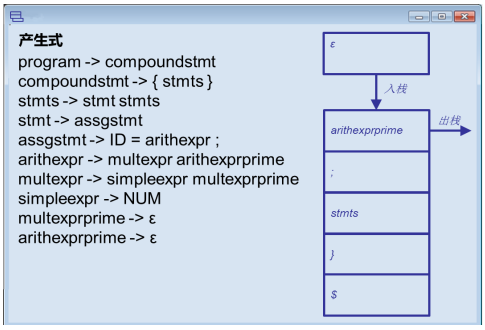

Figure 4. The visualization of parsing stack
图 4. 语法分析语法栈可视化

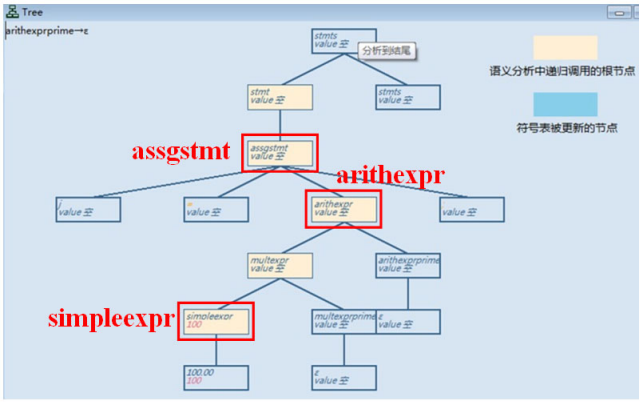

Figure 5. The visualization of syntax tree in semantic analysis
图 5. 语义分析语法树可视化

5. 结论

本文设计的基于教学辅助的可视化编译器由词法分析器、语法分析器、语义分析器和三地址生成器及其对应的可视化构成，在语法分析和语义分析中，增加了语法分析栈和语法树构建的动态显示，并将整个编译过程通过图、表、颜色等方式直观、动态地展现在界面上。在编译课程的教学中，可以选择“分析下一个”方式，通过各可视化界面和功能，能够将编译算法的过程以及编译运行的机理逐步展示出来，便于深刻理解编译的算法和功能。通过一个学期的教学实践，该可视化编译器在《编译原理》课程的教学中起着良好的辅助作用。

参考文献 (References)

[1] 但静培, 渡边坦 (2002) 基于编译技术的程序可视化. *计算机应用研究*, **10**, 51-52, 70.

[2] 李显 (2011) 编译过程可视化的研究与实现. 河北工业大学, 天津, 1-65.

[3] 佚名 (2012) 编译原理经典算法的可视化实现. 长沙理工大学, 长沙, 1-47.

[4] 赵丽, 齐兴斌, 李雪梅 (2012) 基于编译技术的可视化计算系统设计. *电脑编程技巧与维护*, **18**, 95-96.

[5] 王涛 (2015) 编译原理 LL(1)语法分析的可视化教学方法. *新教育时代底子杂志(教师版)*, **6**, 78-79.

[6] 许智宏, 李显, 高静静 (2012) 高级语言编译过程可视化研究. *教育教学论坛*, **10**, 30-31.

[7] 蒋秀锋, 任志雄 (2010) 可视编译器的设计与实现. *计算机与现代化*, **10**, 63-67.

[8] Aho, A.V., Lam, M.S., Sethi, R. and Ullman, J.D. (1985) Compilers: Principles, techniques, and tools.人民邮电出版社, 北京, 18-112.