课程信息简介

1 课程内容

这门课在内容方面，主要包括语法树的生成（编译器前端）、从语法树到汇编代码的变换（编译器后端）、指称语义理论、霍尔逻辑理论与程序验证。既会讲解一些重要的理论知识，帮助大家准确理解一些概念，也对大家的编程实践有一定的要求。

在讲解程序理论的过程中会教大家使用也要求大家使用 Coq 定理证明器编写严格的定义与证明代码。

2 参考书籍

- Robert Harper. Practical Foundations for Programming Languages.
- Glynn Winskel. The Formal Semantics of Programming Languages: An Introduction.
- Benjamin Pierce, et. al. Software Foundations.
- Andrew W. Appel. Modern Compiler Implementation in C.

3 课程评分

本课程的期末总评分分为四个部分。

- 课堂参与 5 分
- 平时作业 35 分
- 大作业 30 分
- 期末考试 30 分

4 课程资料与重要信息

- QQ 群：1007523862
- 课后答疑：通过 QQ 群进行。其他问题请邮件沟通：caoqinxiang@sjtu.edu.cn.
- 助教：吴熙炜、吴基洋
- 课程网站：`https://jhc.sjtu.edu.cn/public/courses/CS2612/`

5 程序语言的语法树

下面三组 C 程序语句中，每一组内的两句程序语句是相同的程序语句吗？第一组：

```
y=x+1; // 代码中的空格更少
```

```
y = x + 1; // 代码中的空格更多
```

第二组：

```
y = (x) + 1; // 代码中的包含多余的括号
```

```
y = x + 1; // 代码中无多余的括号
```

第三组：

```
y = 1 + x;
```

```
y = x + 1;
```

对于描述程序行为与程序正确性的理论而言，像第一组例子中的多余空格与第二组例子中的多余括号并不重要，因此，我们认为第一组与第二组都包含了相同的程序语句。而第三组中的两句程序语句则是不同的程序语句。

C 表达式 `* (p + 1) ++` 的结构：

```
          ++ (后缀运算符)
           |
           *
           |
          ( )
           |
           +
          /  \
         p    1
```

又例如，C 表达式 `x = * p; free(p);` 可以理解成为下面树结构。

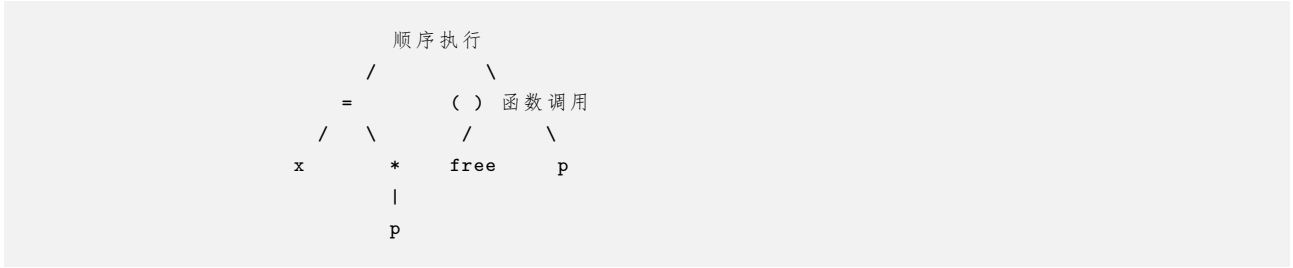
C 表达式 `x = * p; free(p);` 的结构：

```
              顺序执行
            /          \
           ;            ;
           |            |
           =          ( ) 函数调用
         /   \        /    \
        x     *    free     p
              |
              p
```

这些树结构中，有些信息是有些多余的。例如，上一个例子中的括号是多余信息。精简后可以得到以下树结构。

C 表达式 `* (p + 1) ++` 的结构：

类似的，C 表达式 `x = * p; free(p);` 的结构也可以简化。
C 表达式 `x = * p; free(p);` 的结构：



我们就把这样子精简之后的树结构称为这个程序的抽象语法树（Abstract Syntax Tree, AST）。

6 While 语言

- 常数
  - `N ::= 0 | 1 | ...`
  - While 语言的常数是以非 0 数字开头的一串数字或者 0。
- 保留字
  - While 语言的保留字有：var, if, then, else, while, do
- 变量名
  - `V ::= ...`
  - While 语言变量名的第一个字符为字母或下划线，while 语言的变量名可以包含字母、下划线与数字。
  - 保留字不是变量名。
  - 例如：`a0` , `__x` , `leaf_counter` 等等都可以是 while 语言的变量名。
- 表达式
  - `E :: = N | V | -E | E+E | E-E | E*E | E/E | E%E |`
    `E<E | E<=E | E==E | E!=E | E>=E | E>E |`
    `E&&E | E||E | !E`
  - 优先级：`|| < && < ! < Comparisons < +,- < *,/,%`
  - 同优先级运算符之间左结合，可以使用小括号改变优先级。
  - 例如：`a0+1` , `x<=y&&x>0` 等等都是 while 语言的表达式。
- 语句

```
C :: = var V |
       V = E |
       C; C |
       if (E) then { C } else { C } |
       while (E) do { C }
```

7 While 语言 + Dereference + Built-in functions

表达式

```
E :: = N | V | -E | E+E | E-E | E*E | E/E | E%E |
       E<E | E<=E | E==E | E!=E | E>=E | E>E |
       E&&E | E||E | !E | *E |
       malloc(E) | read_int() | read_char()
```

语句

```
C :: = var V |
       write_int(E) |
       write_char(E) |
       E = E |
       C; C |
       if (E) then { C } else { C } |
       while (E) do { C }
```

While 程序示例 1

```
var x;
var n;
var flag;
x = read_int();
n = 2;
flag = 1;
while (n * n <= x && flag) do {
  if (x % n == 0)
  then { flag = 0 }
  else { n = n + 1 }
};
write_int(flag)
```

While 程序示例 2

```
var x;
var l;
var h;
var mid;
x = read_int();
l = 0;
h = x + 1;
while (l + 1 < h) do {
  mid = (l + h) / 2;
  if (mid * mid <= x)
  then { l = mid }
  else { h = mid }
};
write_int(l)
```

While 程序示例 3

```
var n;
var s;
s = 0;
n = 1;
while (n != 0) do {
  n = read_int();
  s = s + n
};
write_int(s)
```