程序语言的语法

一个极简的指令式程序语言

整数类型表达式

```
EI ::= N | V | EI + EI | EI - EI | EI * EI
```

布尔类型表达式

```
EB ::= TRUE | FALSE | EI < EI | EB && EB | ! EB
```

语句

```
C ::= SKIP |
      V = EI |
      C; C |
      if (EB) then { C } else { C } |
      while (EB) do { C }
```

整数类型表达式的语法树

```
Inductive expr_int : Type :=
  | EConst (n: Z): expr_int
  | EVar (x: var_name): expr_int
  | EAdd (e1 e2: expr_int): expr_int
  | ESub (e1 e2: expr_int): expr_int
  | EMul (e1 e2: expr_int): expr_int.
```

```
Check [[1 + "x"]].
Check [["x" * ("a" + "b" + 1)]].
```

While 语言

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
```

语句

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

用 Coq 归纳类型定义二叉树

```
Inductive tree: Type :=
| Leaf: tree
| Node (l: tree) (v: Z) (r: tree): tree.
```

```
Definition tree_example0: tree :=
  Node Leaf 1 Leaf.
```

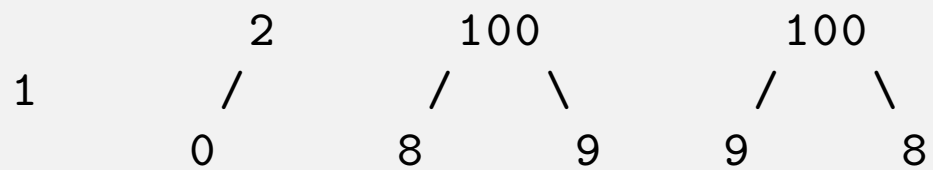
```
Definition tree_example1: tree :=
  Node (Node Leaf 0 Leaf) 2 Leaf.
```

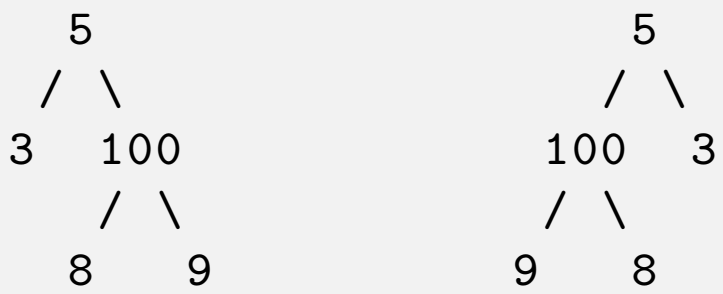
```
Definition tree_example2a: tree :=
  Node (Node Leaf 8 Leaf) 100 (Node Leaf 9 Leaf).
```

```
Definition tree_example2b: tree :=
  Node (Node Leaf 9 Leaf) 100 (Node Leaf 8 Leaf).
```

```
Definition tree_example3a: tree :=
  Node (Node Leaf 3 Leaf) 5 tree_example2a.
```

```
Definition tree_example3b: tree :=
  Node tree_example2b 5 (Node Leaf 3 Leaf).
```


2
100
100
1
0
8
9
9
8


5
3
100
8
9
5
100
3
9
8

```
Fixpoint tree_height (t: tree): Z :=
  match t with
  | Leaf => 0
  | Node l v r => Z.max (tree_height l) (tree_height r) + 1
  end.
```

```
Fixpoint tree_size (t: tree): Z :=
  match t with
  | Leaf => 0
  | Node l v r => tree_size l + tree_size r + 1
  end.
```

```
Example Leaf_height:
  tree_height Leaf = 0.
Proof. reflexivity. Qed.
```

```
Example tree_example2a_height:
  tree_height tree_example2a = 2.
Proof. reflexivity. Qed.
```

```
Example treeexample3b_size:
  tree_size tree_example3b = 5.
Proof. reflexivity. Qed.
```

```
Fixpoint tree_reverse (t: tree): tree :=
  match t with
  | Leaf => Leaf
  | Node l v r => Node (tree_reverse r) v (tree_reverse l)
  end.
```

```
Example Leaf_tree_reverse:
  tree_reverse Leaf = Leaf.
Proof. reflexivity. Qed.
```

```
Example tree_example0_tree_reverse:
  tree_reverse tree_example0 = tree_example0.
Proof. reflexivity. Qed.
```

```
Example tree_example3_tree_reverse:
  tree_reverse tree_example3a = tree_example3b.
Proof. reflexivity. Qed.
```

结构归纳法

数学归纳法

- 奠基步骤：证明`P 0`成立；
- 归纳步骤：证明对于任意自然数`n`，如果`P n`成立，那么`P (n + 1)`也成立。

结构归纳法

- 奠基步骤：证明`P Leaf`成立；
- 归纳步骤：证明对于任意二叉树`l r`以及任意整数标签`n`，如果`P l`与`P r`都成立，那么`P (Node l n r)`也成立。

请看演示