**Script** generated by TTT

Title: Petter: Compiler Construction (25.06.2020) - 44: Symbol Tables

Date: Thu Jun 25 10:31:37 CEST 2020

Duration: 15:45 min

Pages: 14

## Refer Uses to Declarations: Symbol Tables

Check for the correct usage of variables:

- Traverse the syntax tree in a suitable sequence, such that
  - each declaration is visited before its use
  - the currently visible declaration is the last one visited
  - ⇝ perfect for an L-attributed grammar
  - equation system for basic block must add and remove identifiers
- for each identifier, we manage a *stack* of declarations
  1. if we visit a *declaration*, we push it onto the stack of its identifier
  2. upon leaving the *scope*, we remove it from the stack
- if we visit a *usage* of an identifier, we pick the top-most declaration from its stack
- if the stack of the identifier is empty, we have found an undeclared identifier



## Example: Decl-Use Analysis via Table of Stacks

```
1  void f()
2  {
3    int a, b;
4    b = 5;
5    if (b>3) {
6      int a, c;
7      a = 3;
8      c = a + 1;
9      b = c;
10   } else {
11     int c;
12     c = a + 1;
13     b = c;
14   }
15   b = a + b;
16 }
```

⇐

| | |
|---|---|
| 0 | $a$ |
| 1 | $b$ |
| 2 | $c$ |



## Example: Decl-Use Analysis via Table of Stacks

```
1  void f()
2  {
3    int a, b;
4    b = 5;
5    if (b>3) {
6      int a, c;
7      a = 3;
8      c = a + 1;
9      b = c;
10   } else {
11     int c;
12     c = a + 1;
13     b = c;
14   }
15   b = a + b;
16 }
```

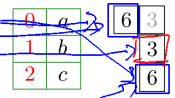
⇐

| | |
|---|---|
| 0 | $a$ |
| 1 | $b$ |
| 2 | $c$ |

| |
|---|
| 3 |
| 3 |

## Example: Decl-Use Analysis via Table of Stacks

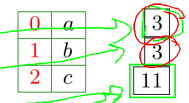
```
1  void f()
2  {
3    int a, b;
4    b = 5;
5    if (b>3) {
6      int a, c;
7      a = 3;
8      c = a + 1;
9      b = c;
10   } else {
11     int c;
12     c = a + 1;
13     b = c;
14   }
15   b = a + b;
16 }
```






## Example: Decl-Use Analysis via Table of Stacks

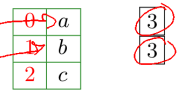
```
1  void f()
2  {
3    int a, b;
4    b = 5;
5    if (b>3) {
6      int a, c;
7      a = 3;
8      c = a + 1;
9      b = c;
10   } else {
11     int c;
12     c = a + 1;
13     b = c;
14   }
15   b = a + b;
16 }
```






## Example: Decl-Use Analysis via Table of Stacks

```
1  void f()
2  {
3    int a, b;
4    b = 5;
5    if (b>3) {
6      int a, c;
7      a = 3;
8      c = a + 1;
9      b = c;
10   } else {
11     int c;
12     c = a + 1;
13     b = c;
14   }
15   b = a + b;
16 }
```





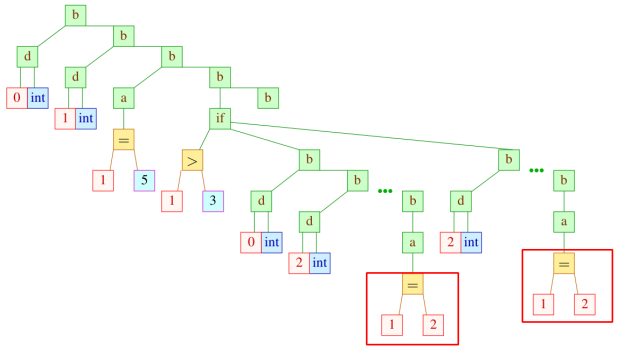

## Example: Decl-Use Analysis via Table of Stacks

```
1  void f()
2  {
3    int a, b;
4    b = 5;
5    if (b>3) {
6      int a, c;
7      a = 3;
8      c = a + 1;
9      b = c;
10   } else {
11     int c;
12     c = a + 1;
13     b = c;
14   }
15   b = a + b;
16 }
```

## Example: Decl-Use Analysis via Table of Stacks

```
1  void f()
2  {
3    int a, b;
4    b = 5;
5    if (b>3) {
6      int a, c;
7      a = 3;
8      c = a + 1;
9      b = c;
10   } else {
11     int c;
12     c = a + 1;
13     b = c;
14   }
15   b = a + b;
16 }
```

d declaration
b basic block
a assignment

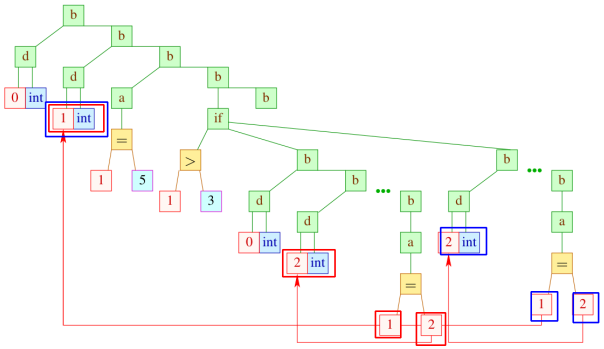




## Example: Decl-Use Analysis via Table of Stacks

```
1  void f()
2  {
3    int a, b;
4    b = 5;
5    if (b>3) {
6      int a, c;
7      a = 3;
8      c = a + 1;
9      b = c;
10   } else {
11     int c;
12     c = a + 1;
13     b = c;
14   }
15   b = a + b;
16 }
```

d declaration
b basic block
a assignment





## Alternative Implementations for Symbol Tables

- when using a list to store the symbol table, storing a marker indicating the old head of the list is sufficient

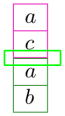

in front of if-statement



## Alternative Implementations for Symbol Tables

- when using a list to store the symbol table, storing a marker indicating the old head of the list is sufficient

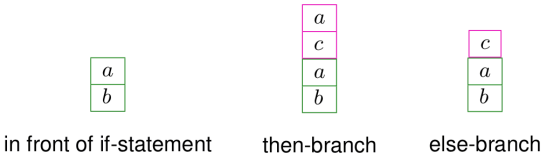

in front of if-statement




then-branch

# Alternative Implementations for Symbol Tables

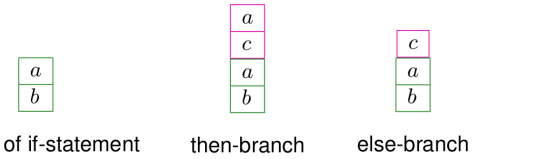
- when using a list to store the symbol table, storing a marker indicating the old head of the list is sufficient






# Alternative Implementations for Symbol Tables

- when using a list to store the symbol table, storing a marker indicating the old head of the list is sufficient

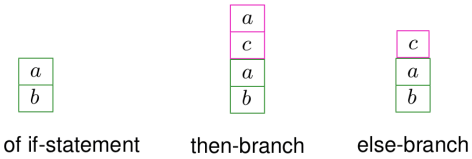



- instead of lists of symbols, it is possible to use a list of hash tables $\rightsquigarrow$ more efficient in large, shallow programs



# Alternative Implementations for Symbol Tables

- when using a list to store the symbol table, storing a marker indicating the old head of the list is sufficient




- instead of lists of symbols, it is possible to use a list of hash tables $\rightsquigarrow$ more efficient in large, shallow programs
- an even more elegant solution: *persistent trees* (updates return fresh trees with references to the old tree where possible)
  - $\rightsquigarrow$ a persistent tree $t$ can be passed down into a basic block where new elements may be added, yielding a $t'$; after examining the basic block, the analysis proceeds with the unchanged old $t$