CISS 375 – COMPILER CONSTRUCTION
Fall 2020
Final Exam Review

- Review
  - Lectures' slides
  - Assignments
  - First test and first test review
  - Second test and second test review

Bottom-up Parsing

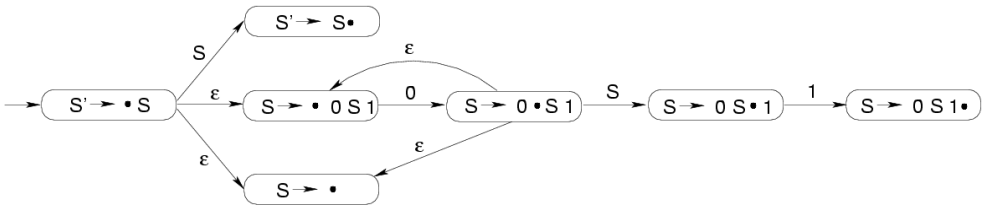
lecture 20

Construct a Viable Prefix NFA for the following grammar:

S → 0S1

S→ ε

Answer



Semantic Analysis

lecture 28 & 29

Consider the following program:

```
1:   int x = 0;
2:   int *y = malloc(sizeof(int));
3:   while(x < 10){
4:     int z = 0;
5:     int y = x;
6:     while(y < 10){
7:       z = z + y;
8:       y = y + 1;
9:     }
10:  }
11:  *y = x;
```

(a) Write a symbol table showing the type of each variable that is in scope at line 4

```
{x : int, y : int*}
```

(b) Write a symbol table showing the type of each variable that is in scope at line 6

```
{x : int, y : int, z: int}
```

(c) Write a symbol table showing the type of each variable that is in scope at line 11

```
{x : int, y : int* }
```

Code optimization

lecture 37

(1) Write the live variable information for the following program:

```
u = v + 1
w = u - v
f = u
x= x + w
y = u - w
z = x + y
// live {n}    <- The question assumes that n is the only live variable at this point
```

The answer

```
// live: {n, x, v}
u = v + 1
// live: {n, x, u, v}
w = u - v
// live: {n, x, u, w}
f = u
// live: {n, x, u, w}
x= x + w
// live: {n, x, u, w}
y = u - w
// live: {n, x, y}
z = x + y
// live {n}
```

(2) Apply Dead Code Elimination to the code above

```
// live: {n, x, v}
u = v + 1
// live: {n, x, u, v}
w = u - v
// live: {n, x, u, w}
f = u          <- delete
// live: {n, x, u, w}
x= x + w
// live: {n, x, u, w}
y = u - w
// live: {n, x, y}
z = x + y      <- delete
// live {n}
```

(3) For the basic block:

```
t6 := 4 * i
x := a[t6]
t7 := 4 * i
t8 :=4 * j
t9 := a[t8]
a[t7] := t9
t10 :=4 * j
a[t10] := x
```

What sources of optimization are performed on the basic block above to produce the following basic block:

```
t6 := 4 * i
x := a[t6]
t8 :=4 * j
t9 := a[t8]
a[t6] := t9
a[t8] := x
```

sources of optimization can be one or more of the following:

- *Common Subexpressions elimination;*
- *Copy Propagation;*
- *Dead-Code elimination;*
- *Constant Folding.*

(4) For the basic block:

```
q = 3
r = 10
s = q + r
t = 2 * r + s
t = q
u = q + r
v = q + t
w = 3 + x
```

State for each of the following basic blocks which optimization was performed on the above:

(a)

```
q = 3
r = 10
s = q + r
t = 2 * r + s
t = q
u = s
v = q + t
w = 3 + x
```

Common Subexpression Elimination

(b)

```
q = 3
r = 10
s = q + r
t = 2 * r + s
t = q
u = q + r
v = q + q
w = 3 + x
```

Copy Propagation

(c)

```
q = 3
r = 10
s = q + r
t = q
u = q + r
v = q + t
w = 3 + x
```

Dead Code Elimination