CS480
Syntax Directed Translation
Extra Slides
March 18, 2009
Semantic Rules

• Problems?

```c
int main()
{
    x = 9;
    return 0;
}

int main()
{
    int y, x;
    y = x;
    return 0;
}

int main()
{
    int y, x;
    if ( input() == 1 )
    {
        x = 0;
        y = x;
    }
    y = x;
    return 0;
}
```
Data Flow Analysis p 608

• Separate from compilers
  • other use of a parser

• think of a parse tree based on statements rather than tokens
  • imagine we replace statement in our grammar like we did EXPRESSION in TD
    – reaching definitions
    – live variables
    – definition-use chain
  • Equation:
    • out[S] = gen[S] U (in[S] - kill[S])
exp -> lvalue assignop exp | * unexp assignop exp | orterm
orterm -> orterm || andterm | andterm
andterm -> andterm && eqterm | eqterm
eqterm -> eqterm equop relterm | relterm
relterm -> relterm relop term | term
term -> term addop factor | factor
factor -> factor mulop unexp | unexp
unexp -> lvalue autoop | & lvalue | * unexp | negop unexp
            | primary
primary -> ( exp ) | lvalue | constant | func
lvalue -> var | ( lvalue )
var -> id | id [ exp ]
func -> id | id ( list )
list -> exp | list , exp
Example

• How does the compiler handle overloaded functions?

  – why can you only do this in C++/Java not C?
  – what does `extern "C"` mean?

```c
void foo(x,y)
    int x;
    int y;
    {}
void foo(x,y)
    int x;
    int *y;
    {}
```
Semantic Rules?

I=int, P=ptr, AX=array (X=I or P)

foo_I\text{I} \quad \text{foo}_I\text{P}