Plan for Today

Recursive descent or predictive parsing
- example predictive parser
- FIRST and FOLLOW sets revisited
- constructing a predictive parser table

Syntax-directed translation

Example Predictive Parser

```
S -> Mesh EOF
Mesh -> num NodeList num ElemList
NodeList -> ε | Node NodeList
Node -> node num real real     // node_id, x, y
ElemList -> ε | Elem ElemList
Elem -> tri num num num num    // elem_id, 3 node ids
Elem -> sqr num num num num num   // elem_id, 4 node ids
```

```
void S() { switch(tok) {
    case NUM:   Mesh(); eat(EOF); break;
    default:    error();
}}

void Mesh() { switch(tok) {
    case NUM:   num_nodes = NUM.val; eat(NUM);
                NodeList();
                num_elem = NUM.val; eat(NUM); break;
    default:    error();
}}
void NodeList() { switch(tok) {
    case NUM:   break;
    case NODE:  Node(); NodeList(); break;
    default:    error();

}}
```


FIRST and FOLLOW sets

nullable(X)
- X is a nonterminal
- nullable(X) is true if X can derive the empty string

FIRST
- FIRST(z) = {z}, where z is a terminal
- FIRST(X) = ∪ FIRST(rhs_i), where X is a nonterminal and X -> rhs_i
  - union all of FIRST(sym) on rhs up to and including first nonnullable

FOLLOW(Y), only relevant when Y is a nonterminal
- look for Y in rhs of rules (lhs -> rhs) and union all FIRST sets for symbols after Y up to and including first nonnullable
- if all symbols after Y are nullable then also union in FOLLOW(lhs)

Constructing the Predictive Parser Table

Algorithm

```
for each X -> gamma
    for each T in FIRST(gamma)
        table[X,T] = X->gamma
    if gamma is nullable
        for each T in FOLLOW(X)
            table[X,T] = X->gamma
```

```
S -> Mesh EOF
Mesh -> num NodeList num ElemList
NodeList -> ε | Node NodeList
Node -> node num real real     // node_id, x, y
ElemList -> ε | Elem ElemList
Elem -> tri num num num num    // elem id, 3 node ids
Elem -> sqr num num num num num   // elemid, 4 node ids
```

Syntax-directed translation

One-pass versus multi-pass compiler
- What do we need for MiniJava?

In a predictive, or top-down, parser
- do actions in functions for nonterminals
- example: storing off the number of nodes and elements in the mesh grammar parser

In a shift-reduce, or bottom-up, parser
- each reduction leads to an action being performed
- example: SableCC builds a parse tree using actions associated with each grammar rule
- syntax-directed translation is equivalent to performing an action on internal nodes in the parse tree during an in post-order traversal
- example: interpreter you wrote for PA1