Plan for Today

FIRST and FOLLOW sets

- formal description of how they are calculated and used in predictive parsers

Error Recovery during

- lexical analysis
- syntax analysis via predictive parsing

Example Predictive Parser

```
(1) start -> mesh EOF
(2) mesh -> NUM nodelist NUM elemlist
(3a & b) nodelist -> ε | node nodelist
(4) node -> NODE NUM REAL REAL     // node_id, x, y
(5a & b) elemlist -> ε | elem elemlist
(6) elem -> TRI NUM NUM NUM NUM    // elem_id, 3 node ids
(7) elem -> SQR NUM NUM NUM NUM NUM //elem_id,4 node ids
```

```
void start() { switch(lookahead) {
    case NUM:   mesh(); match(EOF); break;
    default:    error();
}}
void mesh() { switch(lookahead) {
    case NUM:   num_nodes = lookahead.val; match(NUM);
                nodelist();
                num_elem = lookahead.val; match(NUM);
                elemlist(); break;
    default:    error();
}}
void nodelist() { switch(lookahead) {
    case NUM:   break;                          // nodelist -> epsilon
    case NODE:  node(); nodelist(); break;      // nodelist -> node nodelist
    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)

Error Handling Goals

Provide program with a list of as many errors as possible

Provide USEFUL error messages

- appropriate line and position information
- guidance for fixing the error

Avoid infinite loops or recursion

Add minimal overhead to the processing of correct programs

Predictive parser for Float Assignment Grammar

```
void S() { switch (lookahead) {
    case ID:
    case EOF:// the 2 characters in the FIRST(StmList EOF)
        try { StmList(); match(EOF); } catch { panic(S); } break;
    default: panic(S); break;
}}
void StmList() { switch (lookahead) {
    case ID:  // FIRST( Stm StmList ) = { ID }
        try { Stm(); StmList(); } catch { panic(StmList) } break;
    case EOF: // FOLLOW(StmList) = { EOF }
      break;
    default: panic(StmList); break;
}}
void Stm() { switch (lookahead) {
    case ID:  try { match(ID); match(ASSIGN); match(FLOAT);
              } catch { panic(Stm); } break;
    default: panic(Stm); break;
}}
```