FIRST AND FOLLOW Example

CS 4447 /CS 9545 -- Stephen Watt
University of Western Ontario

FIRST/FOLLOW Main Ideas

- Look at all the grammar rules.
- Examine the possibilities of all substrings of RHS symbols being nullable.
- Nullable RHS prefixes imply adding to FIRST sets.
- Nullable RHS suffixes imply adding to FOLLOW sets.
- Nullable RHS interior substrings imply adding to FOLLOW sets.

Algorithm for FIRST, FOLLOW, nullable

```
for each symbol X
  FIRST[X] := { }, FOLLOW[X] := { }, nullable[X] := false

for each terminal symbol t
    FIRST[t] := {t}

repeat
   for each production X → Y1 Y2 … Yk,
        if all Yi are nullable then
          nullable[X] := true
        if Y1..Yi-1 are nullable then
          FIRST[X] := FIRST[X] U FIRST[Yi]
        if Yi+1..Yk are all nullable then
          FOLLOW[Yi] := FOLLOW[Yi] U FOLLOW[X]
        if Yi+1..Yj-1 are all nullable then
          FOLLOW[Yi] := FOLLOW[Yi] U FIRST[Yj]

until FIRST, FOLLOW, nullable do not change
```

Example Grammar

$G = (\Sigma, V, S, P)$, where

$\Sigma = \{ a, c, d\}$

$V = \{X, Y, Z\}$

$S = Z$

$P = \{$

$Z \rightarrow d, \quad Y \rightarrow \varepsilon, \quad X \rightarrow Y,$

$Z \rightarrow X\ Y\ Z, \quad Y \rightarrow c, \quad X \rightarrow a$

$\}$

Initialization

The initialization loops give

FIRST[X] := {}   FOLLOW[X] := {} nullable[X] := false

FIRST[Y] := {}   FOLLOW[Y] := {} nullable[Y] := false

FIRST[Z] := {}   FOLLOW[Z] := {} nullable[Z] := false

FIRST[a] := {a}

FIRST[c] := {c}

FIRST[d] := {d}

Rules That We Will Iterate

for each production X → Y1 Y2 … Yk,
    if all Yi are nullable **then**
        nullable[X] := true
    if Y1..Yi-1 are nullable **then**
        FIRST[X] := FIRST[X] U FIRST[Yi]
    if Yi+1..Yk are all nullable **then**
        FOLLOW[Yi] := FOLLOW[Yi] U FOLLOW[X]
    if Yi+1..Yj-1 are all nullable **then**
        FOLLOW[Yi] := FOLLOW[Yi] U FIRST[Yj]

How to Interpret Sub-sequences

Yi..Yk	when i < k	has	k-i+1	elements
Yi..Yk	when i = k	has	k-i+1 = 1	elements
Yi..Yk	when i = k+1	has	k-i+1 = 0	elements

Otherwise, not defined.

(Some applications actually do define i > k+1 by making a subsequence taking the original elements backwards, which can be interesting and powerful.)

Specialize for Z → d

Taking Z → d as the production X → Y1..Yk,
we have X is Z and Y1 is d. There are no other Yi.

The *first* rule

if all Yi are nullable **then**
nullable[X] := true

is

if d is nullable (it isn't) **then**
nullable[Z] := true.

Specialize for Z → d

The *second* rule

if Y1..Yi-1 are nullable **then**
FIRST[X] := FIRST[X] U FIRST[Yi]

can be instantiated only for i=1. This gives

if Y1..Y0 = ε is nullable (which it is) **then**
FIRST[Z] := FIRST[Z] U { d }

Specialize for $Z \rightarrow d$

The *third* rule

if Yi+1..Yk are all nullable **then**
FOLLOW[Yi] := FOLLOW[Yi] U FOLLOW[X]

can be instantiated only for i=1. This gives

if Y2..Y1= ε is nullable **then**
FOLLOW[d] := FOLLOW[d] U FOLLOW[Z]

We are not interested in computing FOLLOW of terminals, so we won't bother with this one.

Specialize for $Z \rightarrow d$

The *fourth* rule

if Yi+1..Yj-1 are all nullable **then**
FOLLOW[Yi] := FOLLOW[Yi] U FIRST[Yj]

has no valid values for i and j.

The only Yi is Y1, so we would have to have i=j=1.

But the subsequence Y2..Y0 does not exist.

Rules for X → a, Y → c, Z → d

The same logic applies for all of X → a, Y → c, Z → d.

This gives:

FIRST[X] := FIRST[Z] U {a}
FIRST[Y] := FIRST[Z] U {c}
FIRST[Z] := FIRST[Z] U {d}

There are also some rules to compute FOLLOW of terminals but we don't care about those.

Specialize for X → Y

Taking X → Y as the production X → Y1..Yk,
we have X is X, Y1 is Y, and there are no other Yi.

The *first* rule

if all Yi are nullable **then**
nullable[X] := true

is

if Y is nullable **then**
nullable[X] := true.

Specialize for X → Y

The *second* rule

if Y1..Yi-1 are nullable **then**
FIRST[X] := FIRST[X] U FIRST[Yi]

can be instantiated only for i=1. It becomes

if Y1..Y0 is nullable **then**
FIRST[X] := FIRST[X] U FIRST[Y]

The condition is always satisfied as Y1..Y0 is empty.

FIRST[X] := FIRST[X] U FIRST[Y]

Specialize for X → Y

The *third* rule

if Yi+1..Yk are all nullable **then**
FOLLOW[Yi] := FOLLOW[Yi] U FOLLOW[X]

can be instantiated only for i=1. It becomes

if Y2..Y1 is nullable **then**
FOLLOW[Y] := FOLLOW[Y] U FOLLOW[X].

The test is true (the empty string is nullable).

Specialize for $X \rightarrow Y$

The *fourth* rule

if Yi+1..Yj-1 are all nullable **then**
FOLLOW[Yi] := FOLLOW[Yi] U FIRST[Yj]

has no instantiation.

The only Y is Y1, so we would have to have i=j=1.

But the subsequence Y2..Y0 is not defined.

Rules for X → Y

All the rules together for X → Y are

if Y is nullable **then,** nullable[Z] := true.

FIRST[X] := FIRST[X] U FIRST[Y]

FOLLOW[Y] := FOLLOW[Y] U FOLLOW[X].

Specialize for $Y \rightarrow \varepsilon$

Taking $Y \rightarrow \varepsilon$ as the production $X \rightarrow Y1..Yk$,
we have X is Y, and there are no Yi.

The *first* rule

if all Yi are nullable **then**
nullable[X] := true

is

if all zero of the Y's are nullable (true) **then**
nullable[Y] := true.

Specialize for Y → ε

The *second* rule

if Y1..Yi-1 are nullable **then**
FIRST[X] := FIRST[X] U FIRST[Yi]

cannot be instantiated as there are no Yi.

Specialize for Y → ε

The *third* rule

if Yi+1..Yk are all nullable **then**
FOLLOW[Yi] := FOLLOW[Yi] U FOLLOW[X]

cannot be instantiated as there are no Yi.

Specialize for $Y \rightarrow \varepsilon$

The *fourth* rule

if Yi+1..Yj-1 are all nullable **then**
FOLLOW[Yi] := FOLLOW[Yi] U FIRST[Yj]

has no instantiation as there are no Yi.

Rules for Y → ε

All the rules together for Y → ε are

nullable[Y] := true.

Specialize for $Z \rightarrow X\ Y\ Z$

Taking $Z \rightarrow X\ Y\ Z$ as the production $X \rightarrow Y1..Yk$,
we have X is Z, Y1 is X, Y2 is Y, Y3 is Z.

The *first* rule

if all Yi are nullable **then**
nullable[X] := true

is

if X, Y, Z are all nullable **then**
nullable[Z] := true.

Specialize for Z → X Y Z

The *second* rule

if Y1..Yi-1 are nullable **then**
FIRST[X] := FIRST[X] U FIRST[Yi]

can be instantiated for i=1,2 or 3. These give

if ε is nullable (which it is) **then**
FIRST[Z] := FIRST[Z] union FIRST[X]
if X is nullable **then**
FIRST[Z] := FIRST[Z] union FIRST[Y]
if XY is nullable **then** ou
FIRST[Z] := FIRST[Z] union FIRST[Z]

Specialize for Z → X Y Z

The *third* rule

if Yi+1..Yk are all nullable **then**
FOLLOW[Yi] := FOLLOW[Yi] U FOLLOW[X]

can be instantiated for i=1,2,3. These give

if YZ is nullable **then**
FOLLOW[X] := FOLLOW[X] union FOLLOW[Z]
if Z is nullable **then**
FOLLOW[Y] := FOLLOW[Y] union FOLLOW[Z]
if ε is nullable **then**
FOLLOW[Z] := FOLLOW[Z] union FOLLOW[Z]

Specialize for Z → XYZ

The *fourth* rule

if Yi+1..Yj-1 are all nullable **then**
FOLLOW[Yi] := FOLLOW[Yi] U FIRST[Yj]

has instantiations for (i,j) = (1,2), (1,3), (2,3).

if ε is nullable **then**
FOLLOW[X] := FOLLOW[X] union FIRST[Y]

if Y is nullable **then**
FOLLOW[X] := FOLLOW[X] union FIRST[Z]

if ε is nullable **then**
FOLLOW[Y] := FOLLOW[Y] union FIRST[Z]

Rules for $Z \rightarrow X\ Y\ Z$

if X, Y, Z all nullable **then**	nullable[Z] := true.
if X is nullable **then**	FIRST[Z] := FIRST[Z] union FIRST[Y]
if XY is nullable **then**	FIRST[Z] := FIRST[Z] union FIRST[Z]
if YZ is nullable **then**	FOLLOW[X] := FOLLOW[X] union FOLLOW[Z]
if Z is nullable **then**	FOLLOW[Y] := FOLLOW[Y] union FOLLOW[Z]
if Y is nullable **then**	FOLLOW[X] := FOLLOW[X] union FIRST[Z]

FIRST[Z] := FIRST[Z] union FIRST[X]
FOLLOW[X] := FOLLOW[X] union FIRST[Y]
FOLLOW[Y] := FOLLOW[Y] union FIRST[Z]
FOLLOW[Z] := FOLLOW[Z] union FOLLOW[Z]

All the Rules We Need to Iterate

nullable[X] := true, **if** Y is nullable
nullable[Y] := true.
nullable[Z] := true, **if** X, Y, Z all nullable

FIRST[X] := FIRST[Z] U {a}
FIRST[Y] := FIRST[Z] U {c}
FIRST[Z] := FIRST[Z] U {d}

FIRST[X] := FIRST[X] U FIRST[Y]
FIRST[Z] := FIRST[Z] U FIRST[X]
FIRST[Z] := FIRST[Z] U FIRST[Y], **if** X is nullable
FIRST[Z] := FIRST[Z] U FIRST[Z], **if** XY is nullable // Trivial can drop it

FOLLOW[X] := FOLLOW[X] U FIRST[Y]
FOLLOW[X] := FOLLOW[X] U FIRST[Z], **if** Y is nullable
FOLLOW[Y] := FOLLOW[Y] U FIRST[Z]

FOLLOW[X] := FOLLOW[X] U FOLLOW[Z], **if** YZ is nullable
FOLLOW[Y] := FOLLOW[Y] U FOLLOW[Z], **if** Z is nullable
FOLLOW[Y] := FOLLOW[Y] U FOLLOW[X]
FOLLOW[Z] := FOLLOW[Z] U FOLLOW[Z]

Now Can Iterate to Fill in the Table

$Z \to d$ $\quad Y \to \varepsilon$ $\quad X \to Y$

$Z \to X\ Y\ Z$ $\quad Y \to c$ $\quad X \to a$

	nullable	FIRST	FOLLOW
X	false		
Y	false		
Z	false		

	nullable	FIRST	FOLLOW
X	false	a	c d
Y	true	c	d
Z	false	d	

	nullable	FIRST	FOLLOW
X	true	a c	a c d
Y	true	c	a c d
Z	false	a c d	