Lec 15 CS241

Graham Cooper

June 24th, 2015

```
Nullable[A] <- false and A
repeat
-- for each rule B -> B1 ... Bk
-- -- Nullable[B]<- true
until nothing changes
```

```
First[A] <- empty and A
repeat
-- for each B1 ... Bk
-- -- for i = 1 ... k
-- -- -- if Bi is a terminal a
-- -- -- -- First[B] += {a}; break
-- -- -- else
-- -- -- -- First[B] += First[Bi]
-- -- -- -- If not Nullable[Bi] break
until nothing changes
```

```
Follow[A] <- empty for all not equal to s'
repeat
-- for each B -> B1...Bk
-- -- for i = 1 ... k
-- -- -- if Bi in N
-- -- -- -- Follow[Bi] += First*(Bi +1 ... Bk)
-- -- -- -- if all of Bi+1 ... Bk Nullable (incl. i = k)
-- -- -- -- -- Follow[Bi] += Follow[B]
```

Example:

1. S' $\rightarrow \vdash S \dashv$
2. S $\rightarrow$ b S d
3. S $\rightarrow$ p S q
4. S $\rightarrow$ C
5. C $\rightarrow$ l C

6. C $\rightarrow \epsilon$

Nullable

Iteration	0	1	2	3
S'	false	false	false	false
S	false	false	true	true
C	false	true	true	true

First

Iteration	0	1	2	3
S'	{}	$\{\vdash\}$	$\{\vdash\}$	$\{\vdash\}$
S	{}	{b,p}	{b,p,l}	{b,p,l}
C	{}	{l}	{l}	{l}

Follow

Iteration	0	1	2
S	{}	$\{\dashv$, d, q $\}$	$\{\dashv$, d,q $\}$
C	{}	$\{\dashv$, d, q $\}$	$\{\dashv$, d,q $\}$

Predict(A,a) = $\{A \rightarrow B \mid a \in \text{First*}(B)\} \cup \{A \rightarrow B \mid \text{Nullable}(B), a \in \text{Follow}(A)\}$

Predictor Table

	$\vdash$	$\dashv$	b	d	p	q	l
S'	{1}						
S		4	2	4	3	4	4
C		6		6		6	5

For S, we can use rule 4 to make an l, or to make dissapear!

A grammar is LL(1) it:

- no two distinct productions with the same LHS can generate the same first terminal symbol
- no nullable symbol A has the same terminal symbol a in both its first set and its follow set
- there is only one way to send a nullable symbol to ϵ

Eg:
E $\rightarrow$ E + T | T
T $\rightarrow$ T * F | F

F $\rightarrow$ id

The above is not LL(1)
Why?
- Left recursion.
- Left recursion is never LL(1), here is why:

E $\implies$ E + T $\implies$ T + T $\implies$ F + T $\implies$ id + T
E $\implies$ T $\implies$ F $\implies$ id
These have the same first symbol!!!!! AHHH OH NOOOOO

How to fix it:
Make it right-recursion

E $\rightarrow$ T + E | T
T $\rightarrow$ F * T | F
F $\rightarrow$ id
(still not LL(1), we need to factor)

Factor:
E $\rightarrow$ TE'
E' $\rightarrow$ ϵ| + E
T $\rightarrow$ F T'
T' $\rightarrow$ ϵ| * T
F $\rightarrow$ id

LL(1) conflicts with left-associativity

Bottom Up Parsing

Stack stores partially reduced input read so far.
w $\Longleftarrow$ α_k $\Longleftarrow$ α_{k-1} $\Longleftarrow$ ... $\Longleftarrow$ α_1 $\Longleftarrow$ S

Invariant: Stack and unread input = α_i (or w or s)

EX

S' $\rightarrow \vdash$ S $\dashv$
S $\rightarrow$ A y B
A $\rightarrow$ ab
A $\rightarrow$ cd
B $\rightarrow$ z
B $\rightarrow$ wx

w = $\vdash$abywx$\dashv$

Stack	Read Input	Unread Input	Action
	ϵ	$\vdash$abywx$\dashv$	Shift $\vdash$
$\vdash$	$\vdash$	abyw$\dashv$	shift a
$\vdash$a	$\vdash$a	bywx$\dashv$	shift b
$\vdash$ab	$\vdash$ab	ywx$\dashv$	REduct A $\rightarrow$ ab, pop b, pop a, push A
$\vdash$A	$\vdash$ab	ywx$\dashv$	shift y
$\vdash$Ay	$\vdash$aby	wx$\dashv$	shift w
$\vdash$Ayw	$\vdash$abyw	x$\dashv$	shift x
$\vdash$Aywx	$\vdash$abywx	$\dashv$	reduct B $\rightarrow$ wz; pop x, pop w, push B
$\vdash$AyB	$\vdash$abywx	$\dashv$	reduce S $\rightarrow$ AyB, pop B,y,a, Push S
$\vdash$S $\dashv$	$\vdash$abywx$\dashv$	ϵ	Reduce S' $\rightarrow \vdash$S$\dashv$
S'	$\vdash$abywx$\dashv$	ϵ	Accept

We have a choice at each step:

1. shift a char from input to stack
2. Reduce - TOS is the RHS of a grammar rule - replace it with LHS

Accept if Stack contains S' when the input is ϵ (could equiv. $\vdash$S$\dashv$ on empty input) (But that is the same as being at the end of file, so we could accept at $\dashv$)

How dowe know whether to shift or reduce?
- Use the next character to help us decide
- But the problem is still hard