LL(1) grammar (2016)

Select is the first terminal of alpha if alpha is not null otherwise is first of alpha with the union of follow

Ex 1:

<S> -> <X> y

<X> -> <Y> y + follow(<X>) = {y}

<X> -> x {x}

<X> -> null follow(<X>) = {y}

<Y> -> y {y}

<Y> -> null follow(<X>) = {y}

Find joint -> not LL(1)

Ex 2:

<S> -> <X> <Y> z

<X> -> x {x}

<X> -> null follow<X> = {y, z}

<Y> -> y {y}

<Y> -> null follow<Y> = {z}

LL(1)

Error processing (2016)

Prefix property -> detect error as early as possible

Diff between global and local

Local ->

Global ->

Proc SKIP\_TO(valid symbols, other symbols, symbol list)

{

If not (symbol in valid symbol) then {

//process global error

Error(e);

valid symbols <- valid symbols + other symbols;

while not (symol in valid symbols) do

next symbol;

}

}

l-attributed translation grammar and augmented pushdown machine with stack replacements (2016)

<state> -> <exp>p {print}p

<exp>p -> + <exp>q <exp>r {add}q, r, p {add}p, q, r r = p + q

<exp>p -> \* <exp>q <exp>r {mult}q, r, p {mult}p, q, r r = p \* q

<exp>p -> constq

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | + | \* | const | end |
| <s> | 1 | 1 | 1 |  |
| <e> | 2 | 3 | 4 |  |
|  |  |  |  | Accept |
| {add} | Compute p + q, pop, retain | | | |
| {mult} | Compute p \* q, pop, retain | | | |
| {print} | Print value of exp, pop, retain | | | |

1: replace (<e>), retain

2: replace ({add}<e><e>), advance

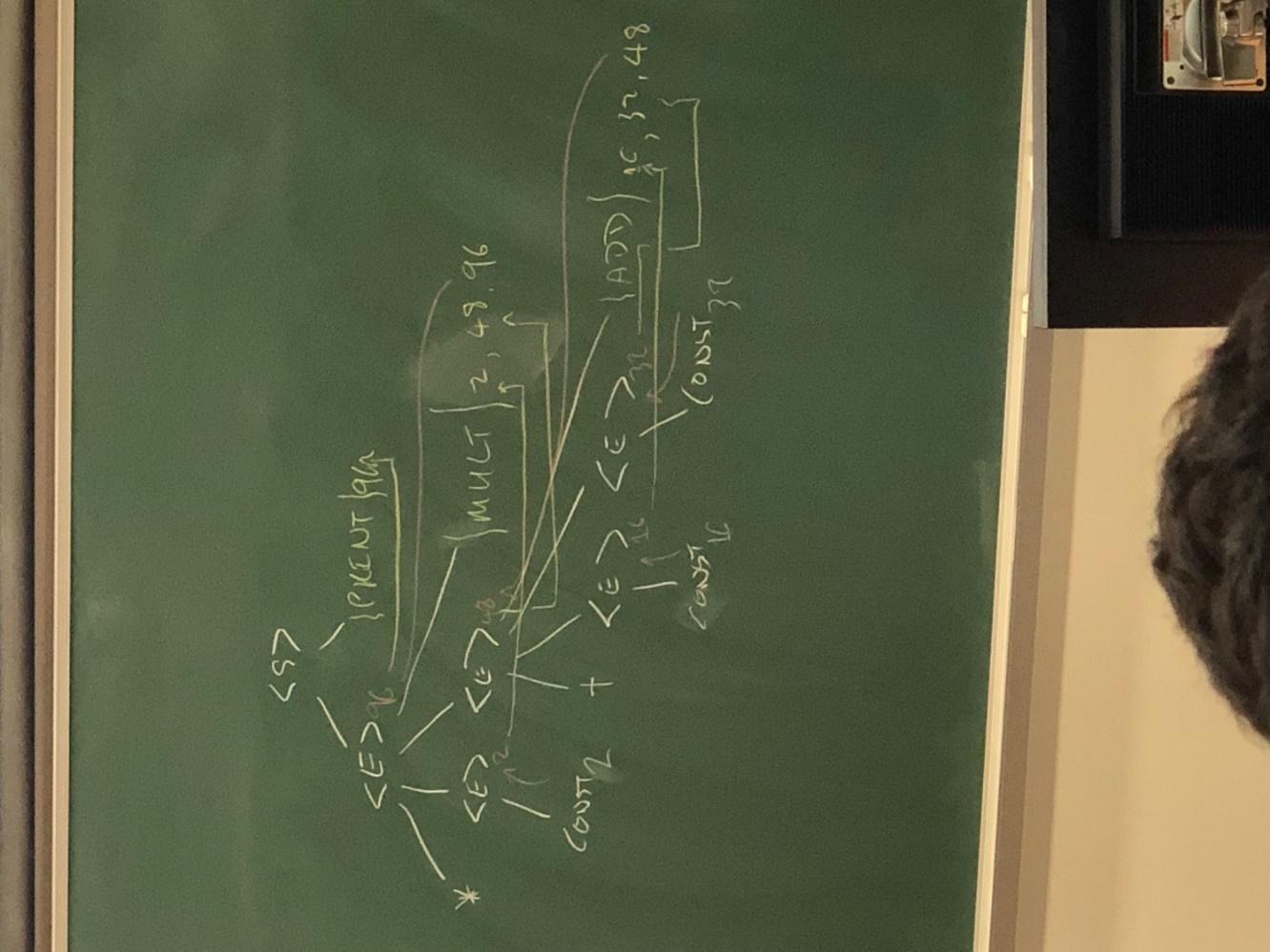
3: replace ({mult}<e><e>), advance

4: pop, advance

All empty table entries represent reject

\* 2 + 16 32





Pushdown machine

Stack symbols

|  |
| --- |
|  |
| <s> |
| <e>  Ptr to place where syn attr is to be stored |
| {mult}  Place where attr is stored |

|  |
| --- |
| <s> |
|  |

-> 7

|  |
| --- |
| <e>  ptr |
| {print}  7 |
|  |

-> +/\*

|  |
| --- |
| <e>  ptr |
| <e>  ptr |
| {add} or {mult}  +/\* |

If/repeat state

<S> -> if <C> then <S>

<S> -> repeat <Ss> until <C>

IF

**Flow**

If -> evaluate <C> -> true -> <S> -> end

-> false -> end

**Translation**

-> <C> -> {jumpF} -> <S> -> {label}

<S> -> if <C>p {jumpFq,r} then <S> {label}

q <- r

(r, s) <- newl

Repeat

**Flow**

Repeat -> execute <Ss> -> evaluate <C> -> true -> end

-> Go back execute <S>

**Translation**

-> {label} -> <Ss> -> <C> -> {jumpF}

<S> -> repeat {labelp} <Ss> until <C>q {jumpFr,s}

r <- q

(p, s) <- newl

Record

Var

struct : record

a, b : float

end;

x : struct;

symbol table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Name | Number of fields | Field name | Field type |
| Record | struct | 2 | a | float |
| var | x | type  ptr |  |  |
|  |  |  |  |  |