# <s> -> while … & <s> -> repeat … & <s> -> if … & <s> -> for …

2017Q1(i, 2016Q3,2015Q3,2014Q3

## Repeat

Flow

1. Execute <s>
2. evaluate <c>
3. If False, back to 1

Translation

1. {label}

2. <s>

3. <c>

4. {jumpF1}

<s> -> repeat {labelp} <s> until <c>q {jumpeFr,s}

* r << q
* (p,s) <<newl

## While do

Flow

1. evaluate <c>
2. True? False?
3. True <s>
4. Go back to 1
5. False, skip 3 and 4

Translation

1. {label}

2. <c>

3. {jumpF6}

4. <s>

5. {jump1}

6.{label}

<s> -> while {labelt} <c>p {jumpFq,r} do <s> {jumpu} {labels}

* (t,u) << newl
* q << p
* (r,s) << newl

## If statement

Flow

1. evaluate <c>
2. True? False?
3. True <s>
4. False, skip 3

Translation

1. <c>
2. {jumpF4}
3. <s>
4. {label}

<s> -> if <c>p {jumpFq,r} then <s> {labels}

* q << p
* (r,s) <<newl

## For loop

Flow

1. <assign>

2. evaluate <c>

3. True <s>

4. Go back to 1

5. False, skip 3, 4

Translation

1. {label}

2. <assign>

3. <c>

4. {jumpF6}

5. {jump1}

6. {label}

<s> -> for {labelt} <assign> <c>p {jumpFq, r} <s> {jumpu} {labels}

* q << p
* (t,u) << newl
* (r,s) << newl

## For all <s> -> …

Where <c>q synthesized p, all action symbol attributes are inherited and newl allocates a new symbol table entry for a table

Symbol Table

|  |  |
| --- | --- |
| Type | Label Value |
| label | A00128… |

* for (r, s), both r and s pointing to the same entry in symbol table
* If your jump come before the label, it’s not in the symbol table yet
* If only 0 in the symbol table, but the address of the following instruction in the symbol table
* When see the label, check if symbol table entry is 0
* If it is, label replaces placeholder value in object code

# If Statement is Not LL

2017Q3(vi, 2014Q1(iii

<s> -> if <c> then <s> else <s>

<s> -> if <c> then <s>

If A = B then

if C = D then P Q

else X Y

Left Factor

1. <s> -> if <c> then <s> <else part>

2. <else part> -> else <s>

3. <else part> -> ε

Follow(<else part>) = Follow(<s>) = {else, ⊣}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | If | Then | Else | ⊣ |
| <s> | 1 |  |  |  |
| <else part> |  |  | 2, 3 | 3 |

Not LL(1)

Push down machine

1. Replace(<else part><s> then <c>), Advance

2. Replace(<s>), Advance

3. Pop, Retain

# Extend grammar and left factor and eliminating left recursion with attributes and pushdown machine

2013Q1, 2015Q1,

<E> -> <E> + <T>

<E> -> <T>

<T> -> <T> \* <P>

<T> -> <P>

<P> -> (<E>)

<P> -> const

## Extension -, \*, /, ^

<E> -> <E> + <T>

<E> -> <E> - <T>

<E> -> <T>

<T> -> <T> \* <F>

<T> -> <T> / <F>

<T> -> <F>

<F> -> <F> ^ <P>

<F> -> <P>

<P> -> (<E>)

<P> -> const

## Left factoring and eliminating left recursion with attributes

1. <E>p -> <T>q<ELIST>r, s r<<q, p<<s

2. <ELIST>p, q -> + <T>r {ADD}s, t, u <ELIST>v, w \*

3. <ELIST> p, q -> - <T> r {SUB}s, t, u <ELIST> v, w \*

4. <ELIST> p, q -> ε q<<p

5. <T> p -> <F> q <TLIST> r, s r<<q, p<<s

6. <TLIST> p, q -> \* <F> r {MUL}s, t, u <TLIST> v, w \*

7. <TLIST> p, q -> / <F> r {DIV}s, t, u <TLIST> v, w \*

8. <TLIST> p, q -> ε q<<p

9. <F> p -> <P> q <EXP> r, s r<<q, p<<s

10. <EXP> p, q -> ^ <F>r {EXP}s, t, u s<<p, t<<r, q<<u

11. <EXP>p, q -> ε q<<p

12. <P> p -> (<E> q) q<<p

13. <P> p -> const q q<<p

\*s<<p, t<<r, v<<u, q<<w

## Pushdown machine

2017Q1(iii, 2016Q1(iii, 2015Q1(vii, 2014Q1(iv, 2013Q1(vi

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | + | - | \* | / | ^ | ( | ) | const | accept |
| <E> |  |  |  |  |  | 1 |  | 1 |  |
| <ELIST> | 2 | 3 |  |  |  |  | 4 |  | 4 |
| <T> |  |  |  |  |  | 5 |  | 5 |  |
| <TLIST> | 8 | 8 | 6 | 7 |  |  | 8 |  | 8 |
| <F> |  |  |  |  |  | 9 |  | 9 |  |
| <EXP> | 11 | 11 | 11 | 11 | 10 |  | 11 |  | 11 |
| <P> |  |  |  |  |  | 12 |  | 13 |  |

1. replace (<ELIST><T>), retain

2. replace (<ELIST>{ADD}<T>), advance

3. replace (<ELIST>{SUB}<T>), advance

4. pop, retain

5. replace (<TLIST><F>), retain

6. replace (<TLIST>{MUL}<F>), advance

7. replace (<TLIST>{DIV}<F>), advance

8. pop, retain

9. replace (<EXP><P>), retain

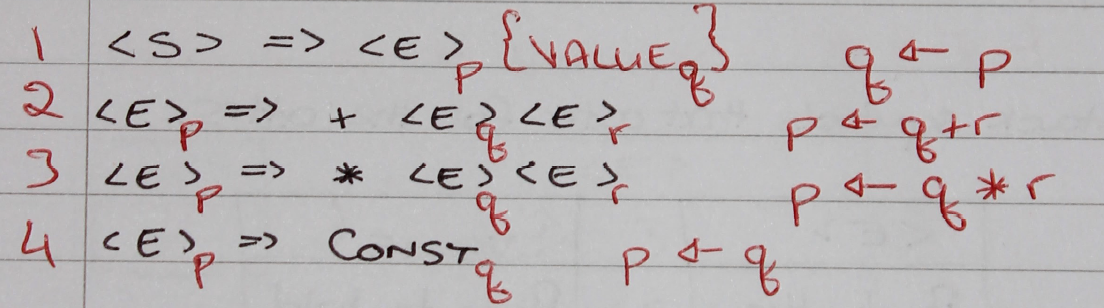
10. replace (<F>{EXP}), advance

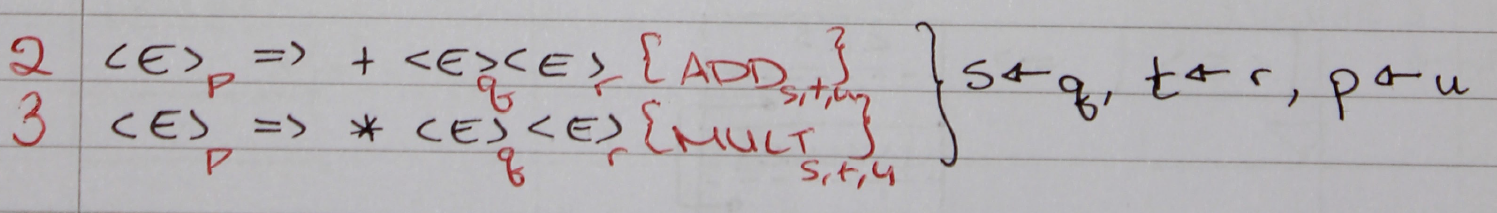
11. pop, retain

12. replace (<E>)), advance

13. pop, advance

Example 2









# LL(1) grammar

2017Q3(i, 2016Q1(i, 2015Q1(iii, 2014Q1(i - iii, 2013Q1(iii

## First

1. If X is a terminal **then** First(X) is just X!
2. If there is a Production X → ε **then** add ε to first(X)
3. If there is a Production X → Y1Y2..Yk**then** add first(Y1Y2..Yk) to first(X)
4. First(Y1Y2..Yk) is **either**
   1. First(Y1) (if First(Y1) doesn't contain ε)
   2. **OR** (if First(Y1) does contain ε) then First (Y1Y2..Yk) is everything in First(Y1) <except for ε > as well as everything in First(Y2..Yk)
   3. If First(Y1) First(Y2)..First(Yk) all contain ε **then** add ε to First(Y1Y2..Yk) as well.

## Follow

1. First put $ (the end of input marker) in Follow(S) (S is the start symbol)
2. If there is a production A → aBb, (where a can be a whole string) **then** everything in FIRST(b) except for ε is placed in FOLLOW(B).
3. If there is a production A → aB, **then** everything in FOLLOW(A) is in FOLLOW(B)
4. If there is a production A → aBb, where FIRST(b) contains ε, **then** everything in FOLLOW(A) is in FOLLOW(B)

## Conflicts

### FIRST/FIRST Conflict

S -> E | E 'a'

E -> 'b' | ε

First(S) = {b, b}

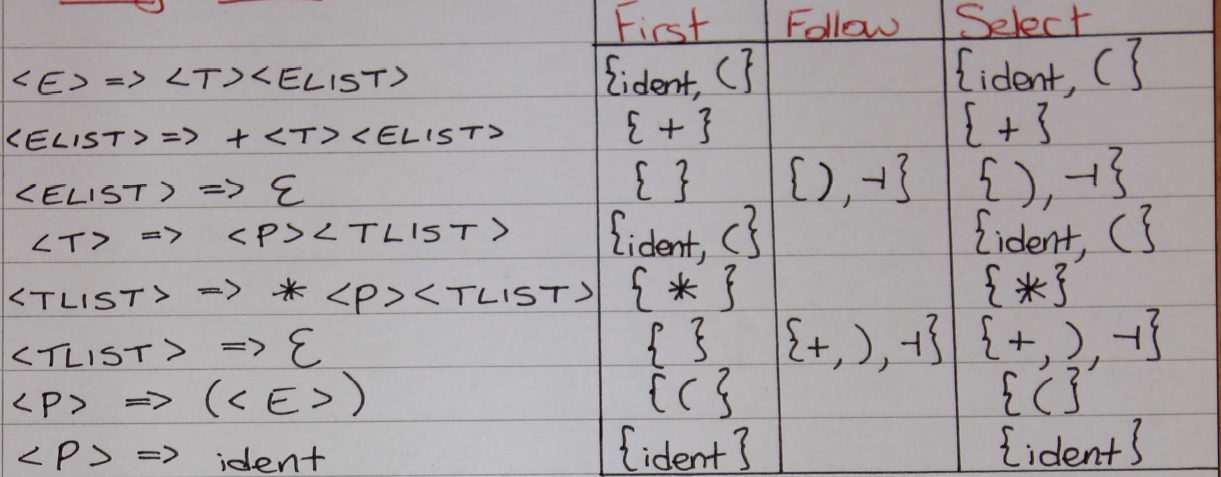
### FIRST/FOLLOW Conflict

S -> A 'a' 'b'

A -> 'a' | ε

First(A) = {a}, Follow(A) = {a}

## Example



|  |  |  |
| --- | --- | --- |
| PRODUCTION | FIRST | FOLLOW |
| <S> -> <X>y | {x, y, ε} | N/A |
| <X> -> <Y> | x | ε | {x, y, ε} | {y} |
| <Y> -> y | ε | {y, ε} | {y} |

Not LL(1) ~ first follow overlay

|  |  |  |
| --- | --- | --- |
| PRODUCTION | FIRST | FOLLOW |
| <S> -> <X><Y>z | {x, y, z, ε} | N/A |
| <X> -> x | ε | {x, ε} | {y} |
| <Y> -> y | ε | {y, ε} | {z} |

Is LL(1)

# Synthesized and inherited

2017Q3(iii

Synthesized Attribute: An attribute that gets its values from the attributes attached to the children of its non-terminal.

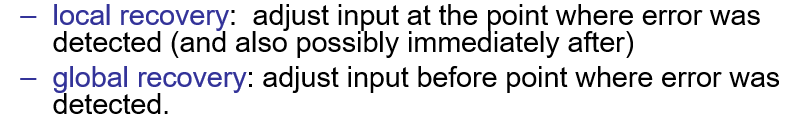
Inherited Attribute: An attribute that gets its values from the attributes attached to the parent (or siblings) of its non-terminal.

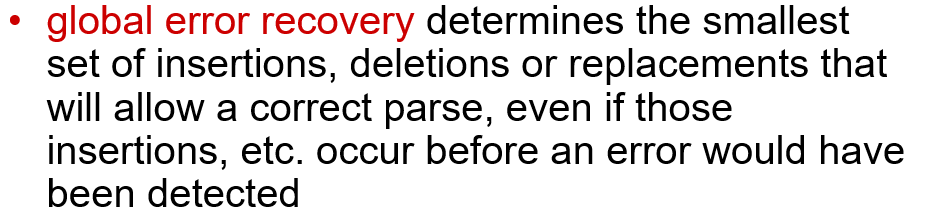
S -> ABC

Synthesized: S

Inherited: A, B, C

# §§§§§§§§§§§Error recovery





## 

## 

## Skip\_to

proc skipTo(validSymbols, otherSymbols: SymbolSet) {

if not(symbol in validSymbols) {

error(0);

validSymbols := validSymbols + otherSymbols;

while not symbol in validSymbols {

do nextSymbol;

}

}

}

# FSM

