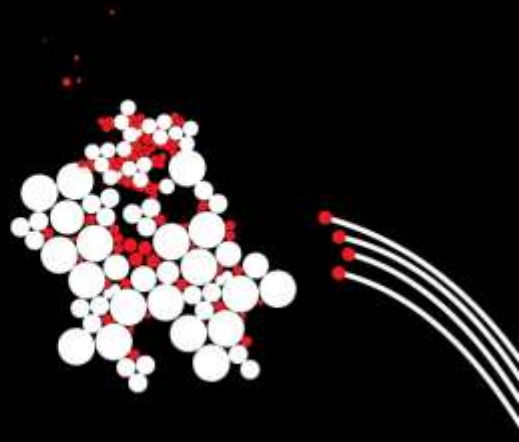
UNIVERSITY OF TWENTE.

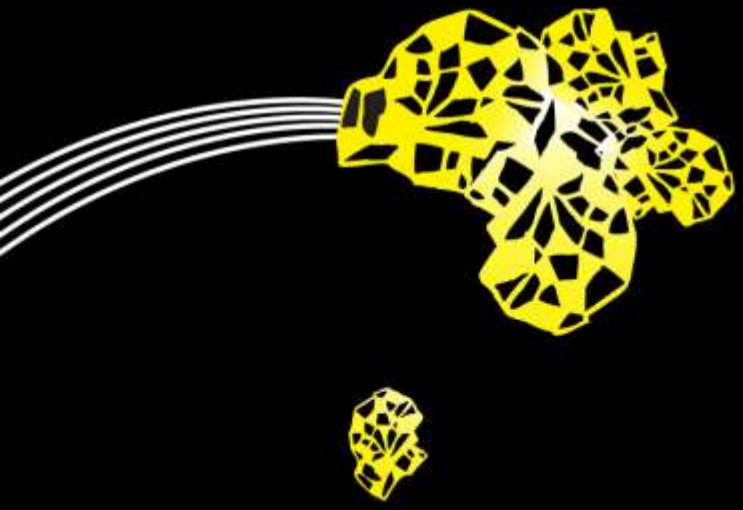


COMPILER CONSTRUCTION

CC 2-3: FOLLOW, FIRST+

MODULE 8: PROGRAMMING PARADIGMS

1 MAY 2019

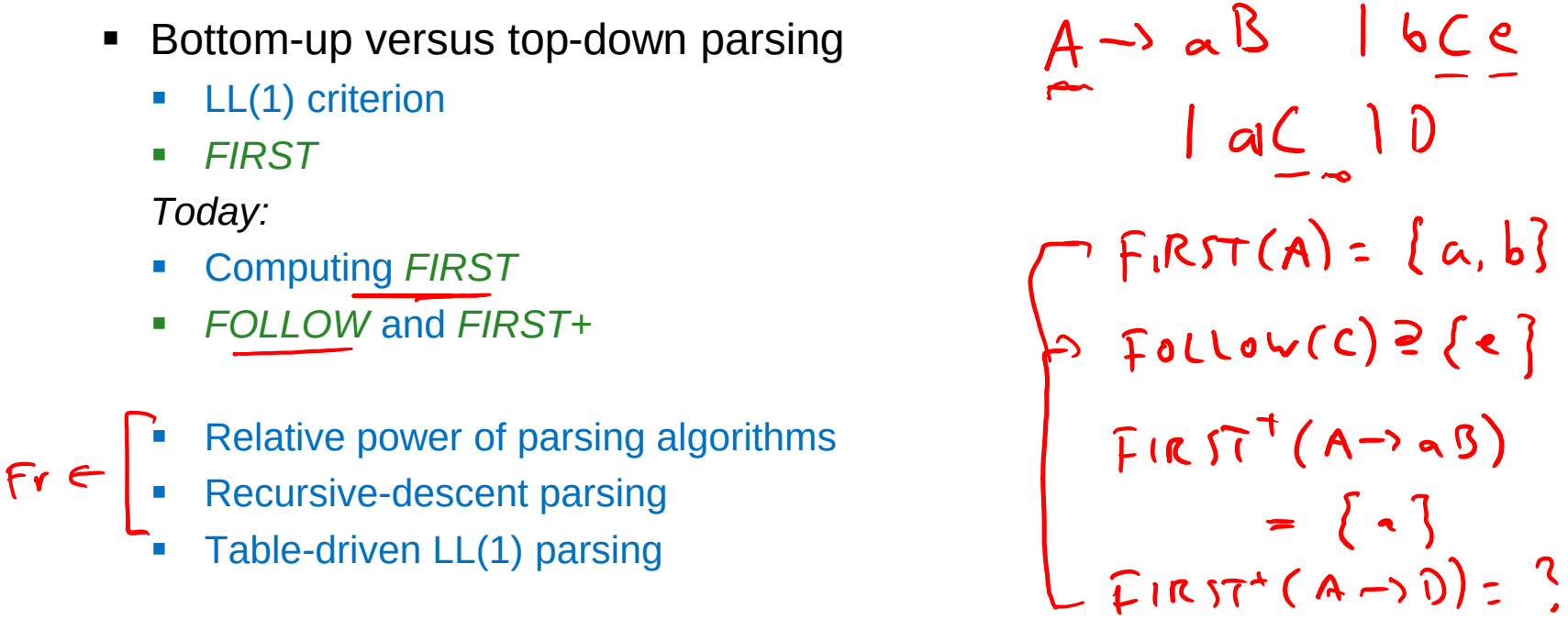


SEEN THIS BLOCK (SO FAR)

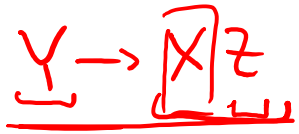
- Bottom-up versus top-down parsing
  - LL(1) criterion
  - *FIRST*

  Today:
  - Computing *FIRST*
  - *FOLLOW* and *FIRST+*

- Relative power of parsing algorithms
- Recursive-descent parsing
- Table-driven LL(1) parsing



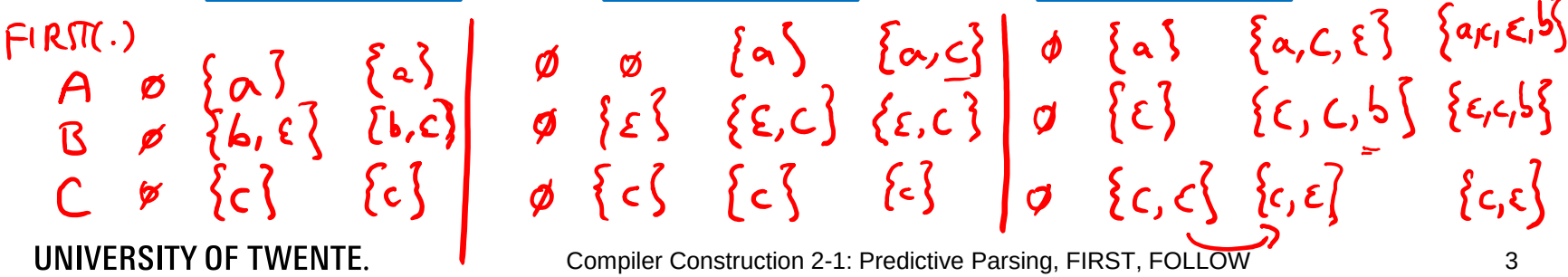
FIRST-COMPUTATION



The idea is straightforward

1. Function from symbols to sets of initial terminals in RHS
   - The *FIRST* of a terminal is that terminal itself
2. Let the RHS start with X (terminal or nonterm.): copy *FIRST*(X) except ϵ
   - If ϵ in *FIRST*(X), also look at *second* symbol of RHS
   - Continue with 3rd etc., as long as ϵ is in *FIRST* of all previous symbols
3. Include ϵ if the RHS consists of only non-terminals with ϵ in their *FIRST*
   - RHS equal to ϵ is a special case of this
4. Iterate the above until all *FIRST*-sets are stable
   - Iteration is necessary to cope with chain of initial non-terminals

$A \rightarrow$ a B $B \rightarrow$ b $C \mid \epsilon$ $C \rightarrow$ c	$A \rightarrow B$ a $B \rightarrow C$ b $\mid \epsilon$ $C \rightarrow$ c	$A \rightarrow B\ C \mid$ a $B \rightarrow C$ b $\mid \epsilon$ $C \rightarrow$ c $\mid \epsilon$



FIRST-COMPUTATION

The idea is straightforward

1. Function from symbols to sets of initial terminals in RHS
   - The *FIRST* of a terminal is that terminal itself
2. Let the RHS start with X (terminal or nonterm.): copy $FIRST(X)$ except ϵ
   - If ϵ in $FIRST(X)$, also look at *second* symbol of RHS
   - Continue with 3rd etc., as long as ϵ is in *FIRST* of all previous symbols
3. Include ϵ if the RHS consists of only non-terminals with ϵ in their *FIRST*
   - RHS equal to ϵ is a special case of this
4. Iterate the above until all *FIRST*-sets are stable
   - Iteration is necessary to cope with chain of initial non-terminals

$A \rightarrow$	a B
$B \rightarrow$	b C | ϵ
$C \rightarrow$	c

	0	1	2
A	$\emptyset$	a	a
B	$\emptyset$	b ϵ	b ϵ
C	$\emptyset$	c	c

$A \rightarrow$	B a
$B \rightarrow$	C b | ϵ
$C \rightarrow$	c

	1	2	3
A	$\emptyset$	a	a c
B	ϵ	ϵ c	ϵ c
C	c	c	c

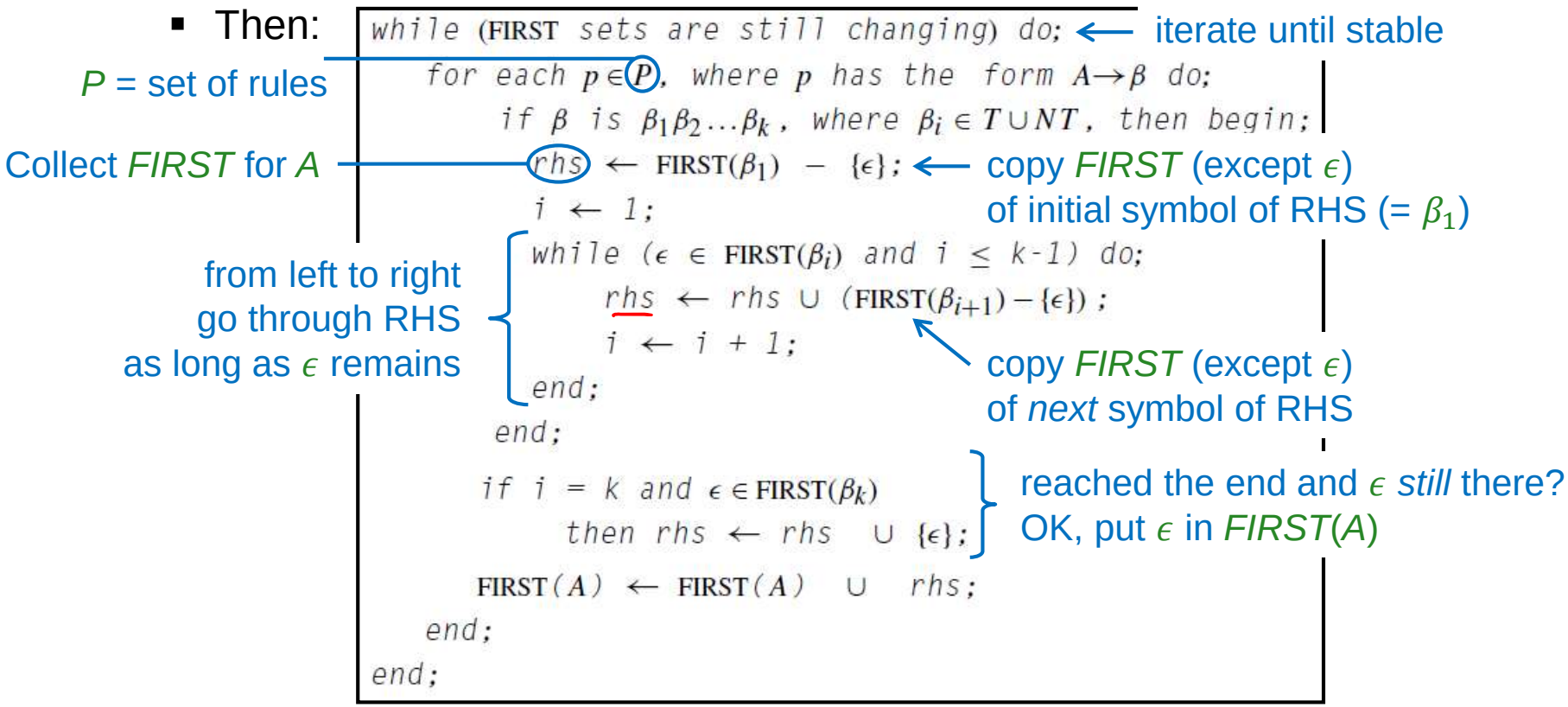
$A \rightarrow$	B C | a
$B \rightarrow$	C b | ϵ
$C \rightarrow$	c | ϵ

	1	2	3
A	a	a c ϵ	a c ϵ b
B	ϵ	ϵ c b	ϵ c b
C	c ϵ	c ϵ	c ϵ

Omitting first & last stage in other examples

FIRST-ALGORITHM FROM THE BOOK

- First: initialisation, set all *FIRST*s to ∅
- Then:



FOLLOW-COMPUTATION

Trickier than *FIRST*

1. Function from non-terminals *A* to sets of terminals that may *follow A*
   - To determine, look at occurrences of *A* in RHSs of *other* rules
2. Initialize *FOLLOW* with { eof } for start symbol, ∅ for all others
3. Work through RHS from right to left (to cater for ϵ)
   - *B* at the end of RHS of *A*: copy *FOLLOW*(*A*) to *FOLLOW*(*B*)
     - Whatever may follow *A* may also follow *B*
   - *B* not at the end: copy *FIRST* of symbol to the right of *B* to *FOLLOW*(*B*)
4. Accumulate as long as ϵ in *FIRST* of RHS symbols
   - If pattern ACb is in RHS and ϵ is in FIRST(C), put b in *FOLLOW*(*A*)

$A \rightarrow a\ B$ $B \rightarrow b\ C \mid \epsilon$ $C \rightarrow c$	$A \rightarrow B\ a$ $B \rightarrow C\ b \mid \epsilon$ $C \rightarrow c$	$A \rightarrow B\ C \mid a$ $B \rightarrow C\ b \mid \epsilon$ $C \rightarrow c \mid \epsilon$

FOLLOW(·)								
A	{eof}	{eof}	{eof}	eof	eof	eof	eof	eof
B	∅	{eof}	{eof}	∅	a	a	∅	c eof
C	∅	∅	{eof}	∅	b	b	∅	eof b

FOLLOW-COMPUTATION

Trickier than *FIRST*

1. Function from non-terminals *A* to sets of terminals that may *follow A*
   - To determine, look at occurrences of *A* in RHSs of *other* rules
2. Initialize *FOLLOW* with { eof } for start symbol, ∅ for all others
3. Work through RHS from right to left (to cater for ϵ)
   - *B* at the end of RHS of *A*: copy *FOLLOW*(*A*) to *FOLLOW*(*B*)
     - Whatever may follow *A* may also follow *B*
   - *B* not at the end: copy *FIRST* of symbol to the right of *B* to *FOLLOW*(*B*)
4. Accumulate as long as ϵ in *FIRST* of RHS symbols
   - If pattern ACb is in RHS and ϵ is in FIRST(C), put b in *FOLLOW*(*A*)

$A \rightarrow$ a B	$A \rightarrow B$ a	$A \rightarrow B\ C \mid$ a
$B \rightarrow$ b $C \mid \epsilon$	$B \rightarrow C$ b $\mid \epsilon$	$B \rightarrow C$ b $\mid \epsilon$
$C \rightarrow$ c	$C \rightarrow$ c	$C \rightarrow$ c $\mid \epsilon$

	0	1	2
A	eof	eof	eof
B	∅	eof	eof
C	∅	∅	eof

	0	1
A	eof	eof
B	∅	a
C	∅	b

	0	1
A	eof	eof
B	∅	eof c
C	∅	eof b

WHAT'S THIS ABOUT EOF?

- Take into account that file *should* end after start symbol
  - Fake terminal eof, conceptually insert at end of file
- Also conceptually insert at end of every RHS of start symbol
  - That can go wrong if start symbol appears in some other RHS

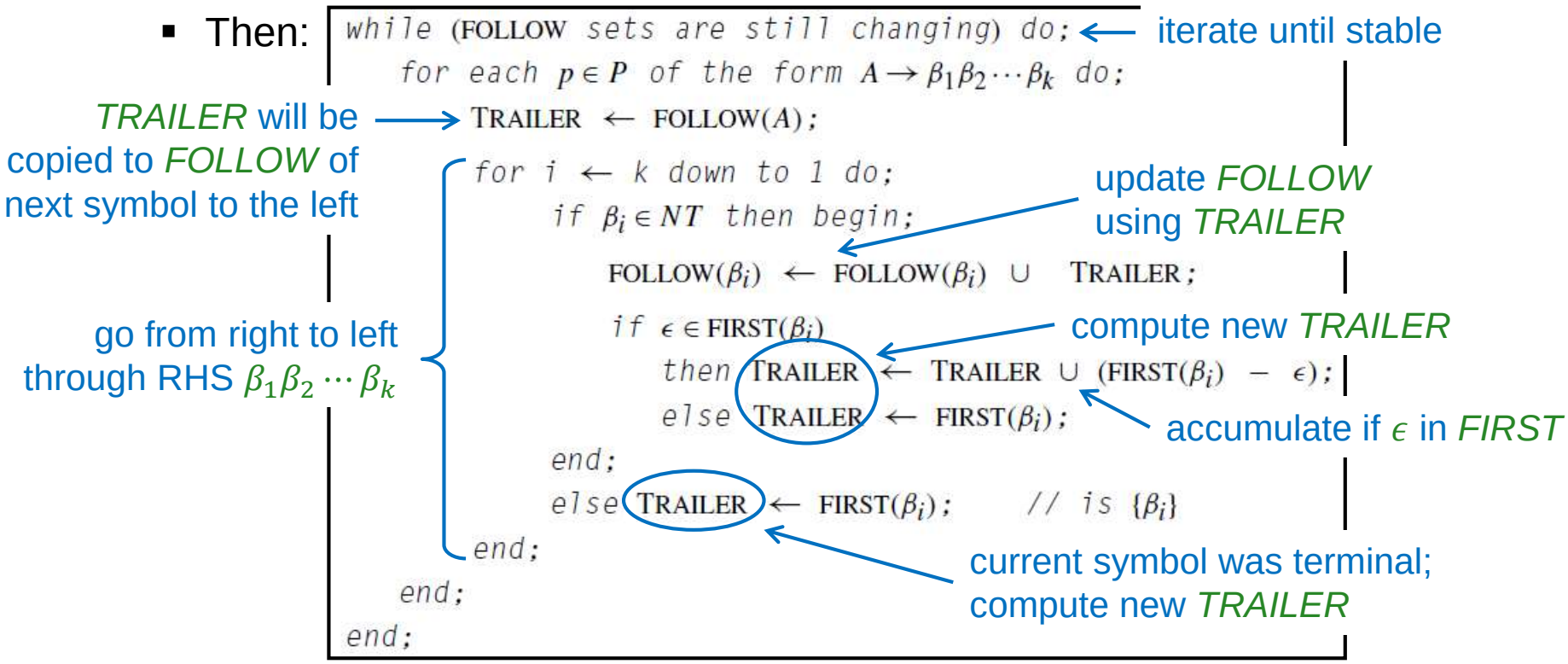
A → a B | ϵ B → b C | ϵ C → A c	not equivalent to	A → a B eof | eof B → b C | ϵ C → A c

  - Don't *really* insert in rule; this is the intuition about the treatment of eof

Try to not make the start symbol appear on any RHS.

FOLLOW-ALGORITHM FROM THE BOOK

- First: initialisation, set *FOLLOW*(*S*) to eof, all others to ∅
- Then:



AND NOW… FIRST+!

- All this work was just to compute the *FIRST+*-set of a rule $A \to \beta$
  - In principle *FIRST+*$(A \to \beta)$ equals *FIRST*(β) (minus ϵ)
    - What's *FIRST*(β)? We only have *FIRST* for single symbols!

$$FIRST(\beta) = \begin{cases} \epsilon & \text{if } \beta = \epsilon \\ FIRST(\beta_1) & \text{if } \beta = \beta_1\ \beta' \text{ and } \epsilon \text{ is not in } FIRST(\beta_1) \\ FIRST(\beta_1)\backslash\{\epsilon\} \cup FIRST(\beta') & \text{otherwise} \end{cases}$$

    - This is essentially the same computation we already did for *FIRST*
  - If ϵ is in *FIRST*(β), we add *FOLLOW*(A) to *FIRST+*$(A \to \beta)$
  - All together now:

$$\text{FIRST}^+(A \to \beta) = \begin{cases} \text{FIRST}(\beta) & \text{if } \epsilon \notin \text{FIRST}(\beta) \\ \text{FIRST}(\beta) \cup \text{FOLLOW}(A) & \textit{otherwise} \end{cases}$$

Note that ϵ may end up in *FIRST+*, but it's quite useless and will be omitted in these slides

- Remember: LL(1)-criterion
  - *FIRST+*$(A \to \beta_1)$ and *FIRST+*$(A \to \beta_2)$ may not overlap *for the same A*!

EXAMPLES

- Grammar

$A \rightarrow a\ B$ $B \rightarrow b\ C \mid \epsilon$ $C \rightarrow c$	$A \rightarrow B\ a$ $B \rightarrow C\ b \mid \epsilon$ $C \rightarrow c$	$A \rightarrow B\ C \mid a$ $B \rightarrow C\ b \mid \epsilon$ $C \rightarrow c \mid \epsilon$

- *FIRST*

A a B b ϵ C c	A a c B ϵ c C c	A a c ϵ b B ϵ c b C c ϵ

- *FOLLOW*

A eof B eof C eof	A eof B a C b	A eof B eof c C eof b

- *FIRST+*

EXAMPLES

- Grammar

$A \rightarrow a\ B$ $B \rightarrow b\ C \mid \epsilon$ $C \rightarrow c$	$A \rightarrow B\ a$ $B \rightarrow C\ b \mid \epsilon$ $C \rightarrow c$	$A \rightarrow B\ C \mid a$ $B \rightarrow C\ b \mid \epsilon$ $C \rightarrow c \mid \epsilon$

- *FIRST*

A a B b ϵ C c	A a c B ϵ c C c	A a c ϵ b B ϵ c b C c ϵ

- *FOLLOW*

A eof B eof C eof	A eof B a C b	A eof B eof c C eof b

- *FIRST+*

$B \rightarrow b\ C$ b $B \rightarrow \epsilon$ eof	$B \rightarrow C\ b$ c $B \rightarrow \epsilon$ a	$A \rightarrow B\ C$ eof c b $A \rightarrow a$ a $B \rightarrow C\ b$ c b $B \rightarrow \epsilon$ eof c $C \rightarrow c$ c $C \rightarrow \epsilon$ eof b

Overlap!

Not LL(1): in input string c b, initial symbol c can be parsed as