CSE443
Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall

Building the finite control for a bottom-up parser

- Build a finite state machine, whose states are sets of items
- Build a table (M) incorporating shift/reduce decisions

Augment grammar

Given a grammar

$G = (N,T,P,S)$

we augment to a grammar

$G' = (N \cup \{S'\}, T, P \cup \{S' \to S\}, S')$, where $S' \notin N$

G' has exactly one rule with S' on left.

CLOSURE(I)

- I is a set of items
- CLOSURE(I) fixed point construction

$CLOSURE_0(I) = I$

repeat {

$CLOSURE_{i+1}(I) =$

$CLOSURE_i(I) \cup \{ B \to \bullet\gamma \mid A \to \alpha \bullet B\beta \in CLOSURE_i(I) \text{ and } B \to \gamma \in P \}$

} until $CLOSURE_{i+1}(I) = CLOSURE_i(I)$

GOTO(I,X)

- GOTO(I,X) is the closure of the set of items A -> αX•β s.t. A -> α•Xβ $\in$ I

- GOTO(I,X) construction for G' (figure 4.32):

```
set-of-items CLOSURE(I) {
    J = I
    repeat {
        for each item A -> α•Bβ ∈ J
            for each production B -> γ ∈ P
                if B->•γ not already in J
                    add B->•γ to J
    } until no more items are added to J
    return J
}
```

Building the LR(0) automaton

```
void items(G') {
   C = { CLOSURE( { S' -> •S } ) }
   repeat {
      for each set of items I ∈ C and
      for each grammar symbol X ∈ (NUT)
      if ( GOTO(I,X) is not empty and not already in C )
         add GOTO(I,X) to C
   } until no new sets of items are added to C
}
```

C is a set of sets of items

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

T

I_2

E -> T •
T -> T • * F

(

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

F

I_3

T -> F •

id

I_5

F -> id •

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

Compute GOTO(I_1,X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

GOTO(I_1,\$) = accept

GOTO(I_1,+) = CLOSURE({ E -> E + • T })

= { E -> E + • T, T -> • T * F, T -> • F, F -> • (E), F -> • id }

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

GOTO(I_1 , '(') = GOTO(I_1 , ')') = GOTO(I_1 , *) = GOTO(I_1 , id) =
GOTO(I_1 , E) = GOTO(I_1 , T) = GOTO(I_1 , F) = ∅

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_2

E -> T •
T -> T • * F

F

id

(

I_3

T -> F •

I_5

F -> id •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

I_2

E -> T •
T -> T • * F

Compute GOTO(I_2,X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

GOTO(I_2,*) = CLOSURE({ T -> T * ⊙ F })

= { T -> T * ⊙ F, F -> ⊙ (E), F -> ⊙ id }

I_7

T -> T * ⊙ F

F -> ⊙ (E)
F -> ⊙ id

GOTO(I_2 , '(') = GOTO(I_2 , ')') = GOTO(I_2 , +) = GOTO(I_2 , id) =
GOTO(I_2 , E) = GOTO(I_2 , T) = GOTO(I_2 , F) = GOTO(I_2 , $) = ∅

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_2

E -> T •
T -> T • * F

*

I_7

T -> T * • F

F -> • (E)
F -> • id

F

id

(

I_3

T -> F •

I_5

F -> id •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

I_3 | T -> F •

Compute GOTO(I_3,X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

GOTO(I_3,E) = GOTO(I_3,T) = GOTO(I_3,F) = GOTO(I_3,+) =
GOTO(I_3,*) = GOTO(I_3,'(') = GOTO(I_3,')') = GOTO(I_3,id) =
GOTO(I_3,$) = $\varnothing$

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_2

E -> T •
T -> T • * F

*

I_7

T -> T * • F

F -> • (E)
F -> • id

F

id

(

I_3

T -> F •

I_5

F -> id •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

Compute GOTO(I_4,X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

I_4

F -> (• E)
E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

GOTO(I4,E) =
CLOSURE({ F -> (E ⊙), E -> E ⊙ + T }) =
{ F -> (E ⊙), E -> E ⊙ + T }

I_8

E -> E ⊙ + T
F -> (E ⊙)

GOTO(I_4,T) = CLOSURE({ E -> T ⊙, T -> T ⊙ * F }) = { E -> T ⊙,
T -> T ⊙ * F } = I_2

GOTO(I_4,F) = CLOSURE({ T -> F ⊙ }) = I_3

GOTO(I_4,'(') = CLOSURE({ F -> (⊙ E) }) = I_7

GOTO(I_4,id) = CLOSURE({ F -> id ⊙ }) = I_5

GOTO(I_4 , ')') = GOTO(I_4 , +) = GOTO(I_4 , *) = GOTO(I_4 , $) = ∅

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_2

E -> T •
T -> T • * F

*

I_7

T -> T * • F

F -> • (E)
F -> • id

F

id

(

T

I_3

T -> F •

I_5

F -> id •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

id

F

(

E

I_8

E -> E • + T
F -> (E •)

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

I_5 F -> id •

Compute GOTO(I_5,X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

GOTO(I_5,E) = GOTO(I_5,T) = GOTO(I_5,F) = GOTO(I_5,+) =
GOTO(I_5,*) = GOTO(I_5,'(') = GOTO(I_5,')') = GOTO(I_5,id) =
GOTO(I_5,$) = ∅

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_2

E -> T •
T -> T • * F

*

I_7

T -> T * • F

F -> • (E)
F -> • id

F

id

(

T

I_3

T -> F •

I_5

F -> id •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

id

F

(

E

I_8

E -> E • + T
F -> (E •)

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

I_6

E -> E + • T

T -> • T * F

T -> • F

F -> • (E)

F -> • id

S' -> E

E -> E + T

E -> T

T -> T * F

T -> F

F -> (E)

F -> id

Compute GOTO(I_6,X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

GOTO(I_6,T) = CLOSURE({ E -> E + T ⊙, T -> T ⊙ * F }) =
{ E -> E + T ⊙, T -> T ⊙ * F }

GOTO(I_6,F) = CLOSURE({ T -> F ⊙ }) = I_3

GOTO(I_6,'(') = CLOSURE({ F -> (⊙ E) }) = I_4

GOTO(I_6,id) = CLOSURE({ F -> id ⊙ }) = I_5

GOTO(I_6 , E) = GOTO(I_6 , ')') = GOTO(I_6 , +) = GOTO(I_6 , *) =
GOTO(I_6 , $) = ∅

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_9

E -> E + T •
T -> T • * F

T

I_2

E -> T •
T -> T • * F

*

I_7

T -> T * • F

F -> • (E)
F -> • id

F

id

(

I_3

T -> F •

I_5

F -> id •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

I_8

E -> E • + T
F -> (E •)

E

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

I_7

T -> T * ◉ F

F -> ◉ (E)
F -> ◉ id

Compute GOTO(I_7,X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

GOTO(I_7,F) = CLOSURE({ T -> T * F ⊙ }) = { T -> T * F ⊙ }

GOTO(I_7,'(') = CLOSURE({ F -> (⊙ E) }) = I_4

GOTO(I_7,id) = CLOSURE({ F -> id ⊙ }) = I_5

I_{10}

T -> T * F ⊙

GOTO(I_7 , E) = GOTO(I_7 , T) = GOTO(I_7 , ')') = GOTO(I_7 , +) = GOTO(I_7 , *) =
GOTO(I_7 , $) = ∅

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

I_1

S' -> E •
E -> E • + T

I_2

E -> T •
T -> T • * F

I_3

T -> F •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

I_5

F -> id •

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

I_7

T -> T * • F

F -> • (E)
F -> • id

I_8

E -> E • + T
F -> (E •)

I_9

E -> E + T •
T -> T • * F

I_{10}

T -> T * F •

E
T
F
+
*
(
)
id
$

accept

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

Compute GOTO(I_8,X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

I_8

E -> E ⊙ + T
F -> (E ⊙)

$S' \to E$
$E \to E + T$
$E \to T$
$T \to T * F$
$T \to F$
$F \to (E)$
$F \to id$

$GOTO(I_8, ')') = CLOSURE(\{ F \to (E) \bullet \}) = \{ F \to (E) \bullet \}$

$GOTO(I_8, +) = CLOSURE(\{ E \to E + \bullet T \}) = I_6$

I_{11}

$F \to (E) \bullet$

$GOTO(I_8 , '(') = GOTO(I_8 , *) = GOTO(I_8 , E) = GOTO(I_8 , T) = GOTO(I_8 , F) =$
$GOTO(I_8 , \$) = GOTO(I_8 , id) = \emptyset$

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

T

I_2

E -> T •
T -> T • * F

F

I_3

T -> F •

id

I_5

F -> id •

(

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_9

E -> E + T •
T -> T • * F

*

I_7

T -> T * • F

F -> • (E)
F -> • id

F

I_{10}

T -> T * F •

(

I_{11}

F -> (E) •

)

I_8

E -> E • + T
F -> (E •)

E

+

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

I_9

E -> E + T ⊙
T -> T ⊙ * F

Compute GOTO(I_9,X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

$GOTO(I_9, *) = CLOSURE(\{ T \to T * \bullet F \}) = I_7$

$GOTO(I_9, '(') = GOTO(I_9, ')') = GOTO(I_9, +) = GOTO(I_9, id) = GOTO(I_9, \$) =$
$GOTO(I_9, E) = GOTO(I_9, T) = GOTO(I_9, F) = \varnothing$

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_9

E -> E + T •
T -> T • * F

T

I_2

E -> T •
T -> T • * F

*

F

I_7

T -> T * • F

F -> • (E)
F -> • id

(

id

id

F

I_3

T -> F •

I_5

F -> id •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

id

(

id

F

(

I_{10}

T -> T * F •

I_{11}

F -> (E) •

)

I_8

E -> E • + T
F -> (E •)

E

+

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

I_{10} | T -> T * F ●

Compute GOTO(I_{10},X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

GOTO(I_{10},E) = GOTO(I_{10},T) = GOTO(I_{10},F) = GOTO(I_{10},+) =
GOTO(I_{10},*) = GOTO(I_{10},'(') = GOTO(I_{10},')') = GOTO(I_{10},id) =
GOTO(I_{10},$) = ∅

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_9

E -> E + T •
T -> T • * F

T

I_2

E -> T •
T -> T • * F

*

F

I_7

T -> T * • F

F -> • (E)
F -> • id

(

id

id

F

I_3

T -> F •

I_5

F -> id •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

id

(

id

F

(

I_{10}

T -> T * F •

I_{11}

F -> (E) •

)

I_8

E -> E • + T
F -> (E •)

E

+

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

I_{11}

F -> (E) •

Compute GOTO(I_{11},X) for each X
in { +, *, '(', ')', id, E, T, F, $ }

$GOTO(I_{11},E) = GOTO(I_{11},T) = GOTO(I_{11},F) = GOTO(I_{11},+) =$
$GOTO(I_{11},*) = GOTO(I_{11},'(') = GOTO(I_{11},')') = GOTO(I_{11},id) =$
$GOTO(I_{11},\$) = \varnothing$

I_0

S' -> • E

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

E

I_1

S' -> E •
E -> E • + T

$

accept

+

I_6

E -> E + • T

T -> • T * F
T -> • F
F -> • (E)
F -> • id

T

I_9

E -> E + T •
T -> T • * F

T

I_2

E -> T •
T -> T • * F

*

F

I_7

T -> T * • F

F -> • (E)
F -> • id

(

id

id

F

I_3

T -> F •

I_5

F -> id •

I_4

F -> (• E)

E -> • E + T
E -> • T
T -> • T * F
T -> • F
F -> • (E)
F -> • id

id

(

id

F

(

I_{10}

T -> T * F •

I_{11}

F -> (E) •

)

I_8

E -> E • + T
F -> (E •)

E

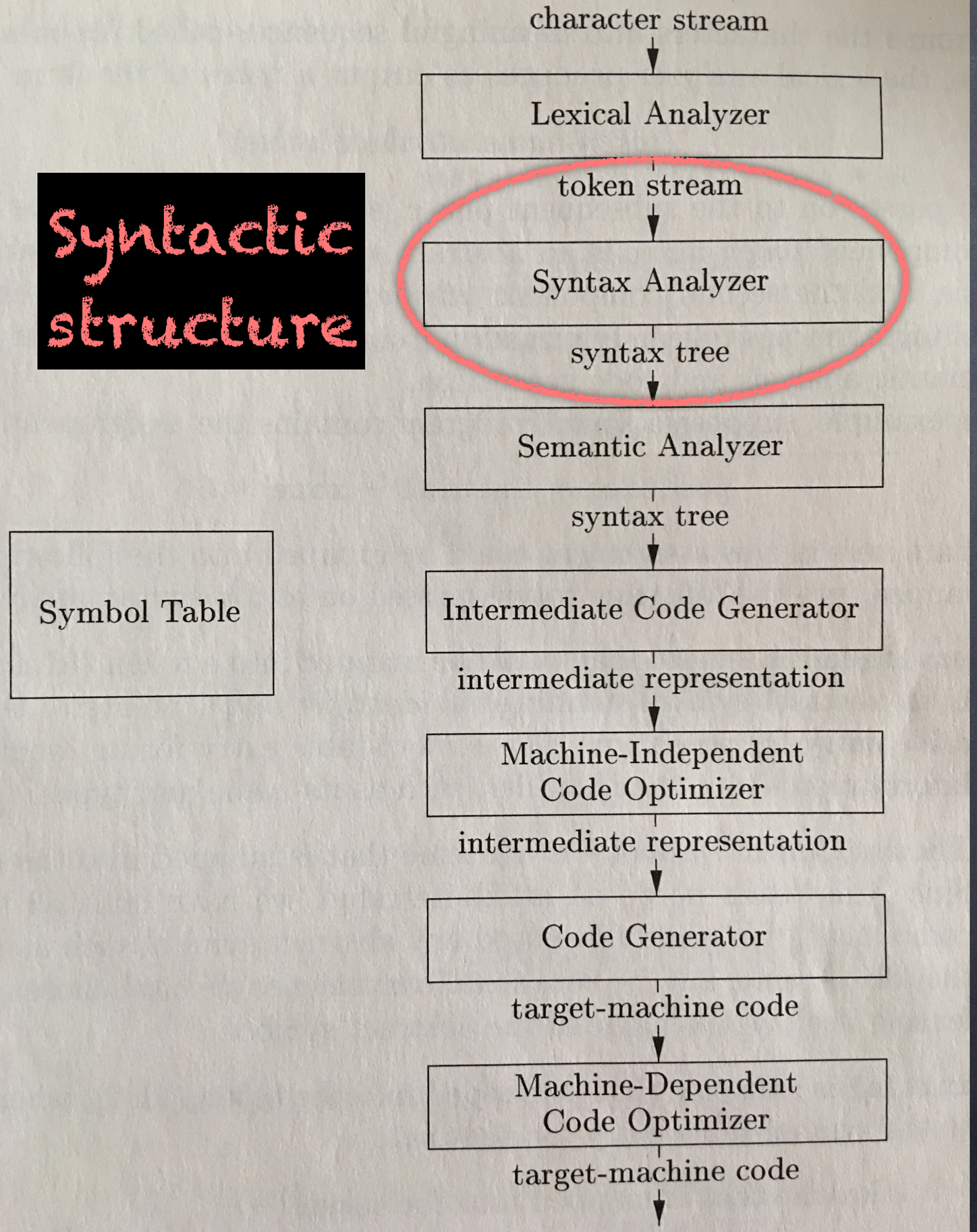
+

S' -> E
E -> E + T
E -> T
T -> T * F
T -> F
F -> (E)
F -> id

Phases of a compiler

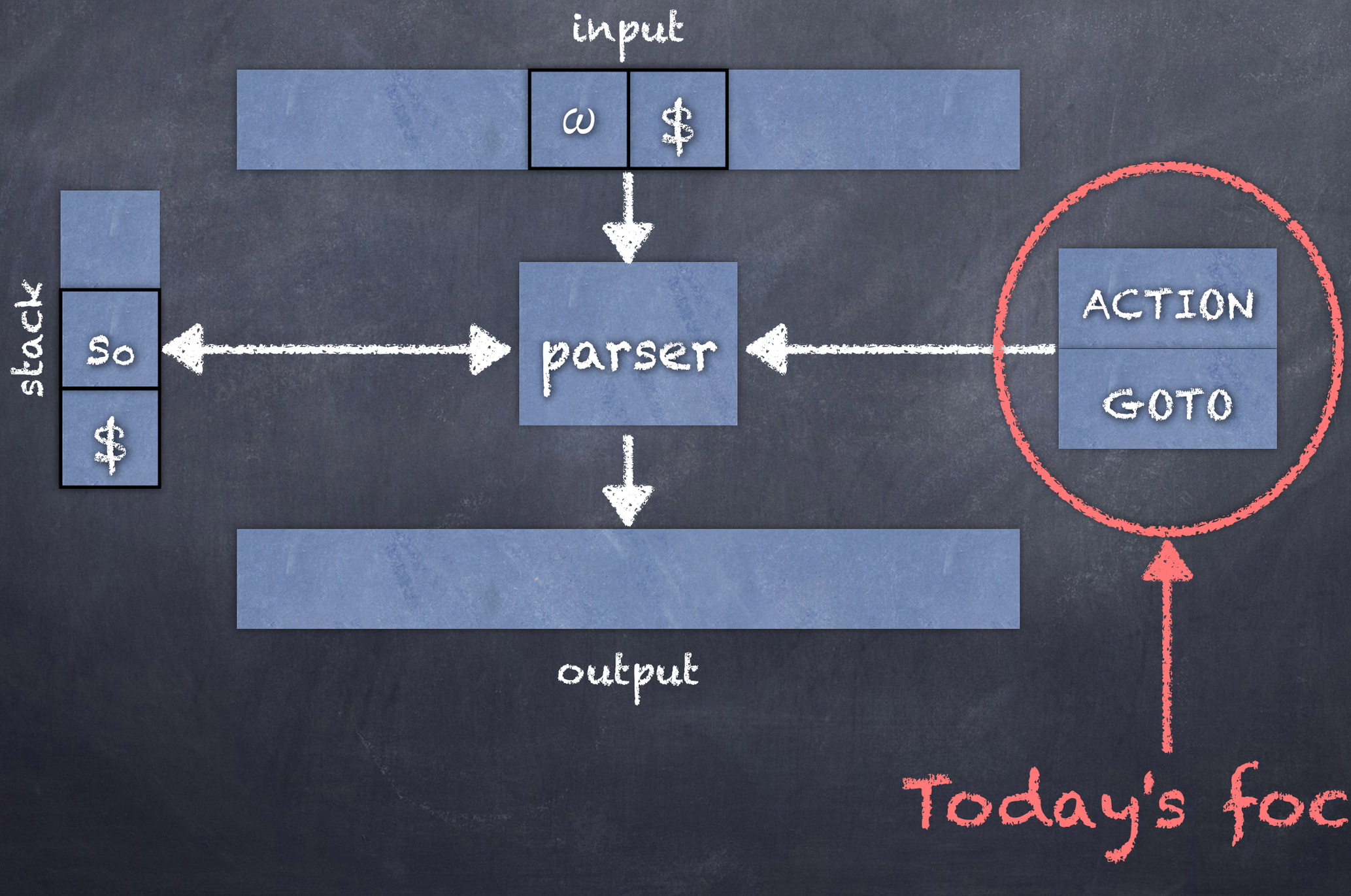
Figure 1.7, page 5 of text

Syntactic structure



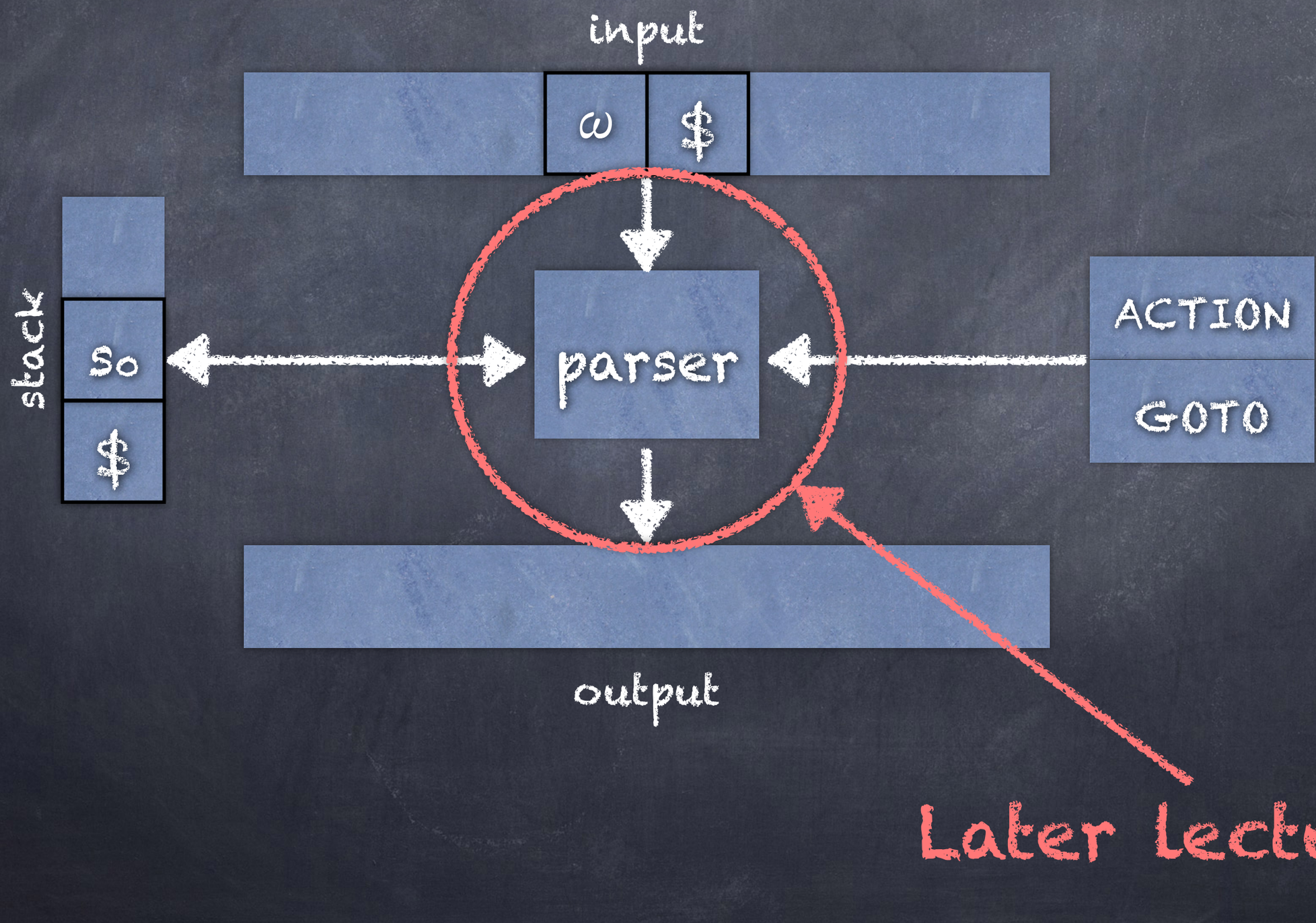
Initial state of the parser
(top of stack is current state in LR(0) automata)

modified from figure 4.35 [p. 248]



Initial state of the parser
(top of stack is current state in LR(0) automata)

modified from figure 4.35 [p. 248]



LR(k)

- LR(k) parser
  - L => left-to-right scanning of input
  - R => rightmost derivation in reverse
  - k => number of lookahead symbols
  - k is typically 0 or 1
  - LR => LR(1)

LR(k)

- LR(k) parser
  - L => left-to-right scanning of input
  - R => rightmost derivation in reverse
  - k => number of lookahead symbols
  - k is typically 0 or 1
  - LR => LR(1)

Lookahead here refers to how many input symbols can be consulted during parsing

[pg. 242]

- "The LR-parsing method is the most general nonbacktracking shift-reduce parsing method known"
- "[The LR-parsing method] can be implemented as efficiently as other [...] shift-reduce methods"
- "An LR parser can detect a syntactic error as soon as it is possible to do so on a left-to-right scan of the input."
- "The class of grammars that can be parsed using LR methods is a proper superset of the class of grammars that can be parsed with predictive or LL methods."