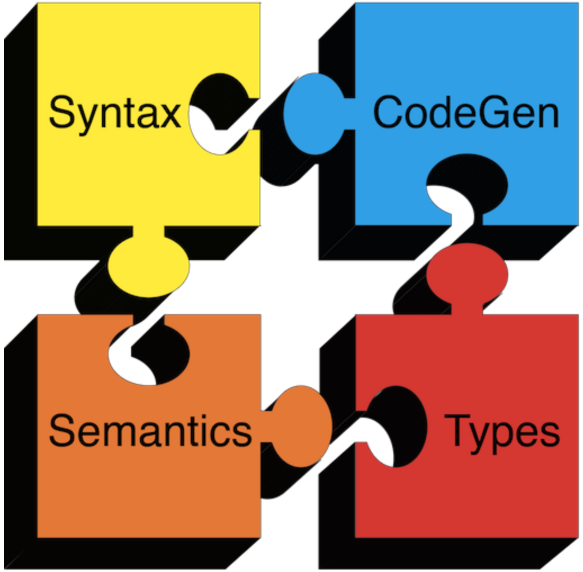


Compilers

Recursive Descent Parsing

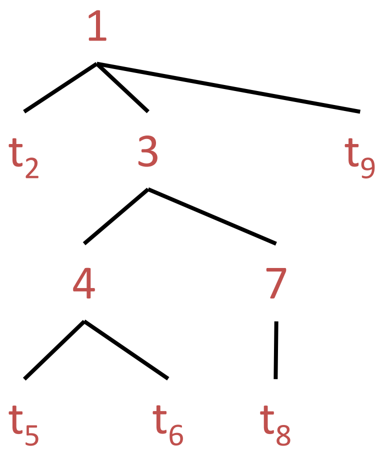
Alex Aiken

Recursive Descent

- The parse tree is constructed
  - From the top
  - From left to right

- Terminals are seen in order of appearance in the token stream:

  $t_2 \ \ t_5 \ \ t_6 \ \ t_8 \ \ t_9$



Recursive Descent

- Consider the grammar

  E → T | T + E

  T → int | int * T | (E)

- Token stream is: (int_5)

- Start with top-level non-terminal E
  - Try the rules for E in order

Recursive Descent

$E \rightarrow T \mid T + E$

$T \rightarrow \text{int} \mid \text{int} * T \mid (E)$

E

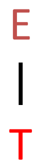
(int_5)

↑

Recursive Descent

E → T | T + E

T → int | int * T | (E)



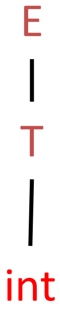
(int_5)

↑

Recursive Descent

$E \rightarrow T \mid T + E$

$T \rightarrow \text{int} \mid \text{int} * T \mid (E)$



E

T

int

Mismatch: int does not match (

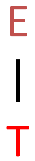
Backtrack …

$(\ \text{int}_5 \ )$

Recursive Descent

E → T | T + E

T → int | int * T | (E)



E

T

(int_5)

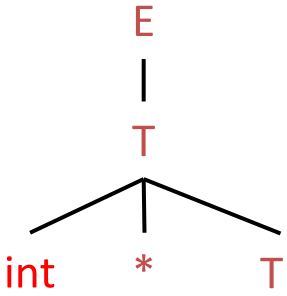
↑

Alex Aiken

Recursive Descent

E → T | T + E

T → int | int * T | (E)



Mismatch: int does not match (

Backtrack …

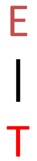
(int_5)

↑

Recursive Descent

$E \rightarrow T \mid T + E$

$T \rightarrow \text{int} \mid \text{int} * T \mid (E)$

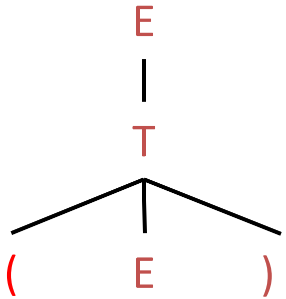


(int$_5$)

Recursive Descent

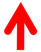
$E \rightarrow T \mid T + E$

$T \rightarrow \text{int} \mid \text{int} * T \mid (E)$



Match! Advance input.

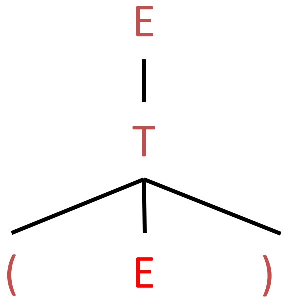
(int_5)



Recursive Descent

E → T | T + E

T → int | int * T | (E)



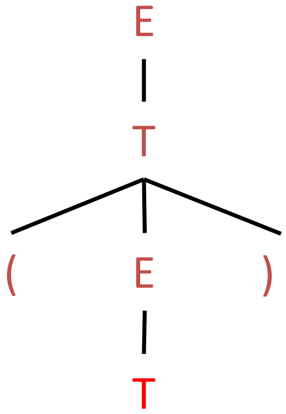
(int_5)

↑

Recursive Descent

E → T | T + E

T → int | int * T | (E)



(int_5)

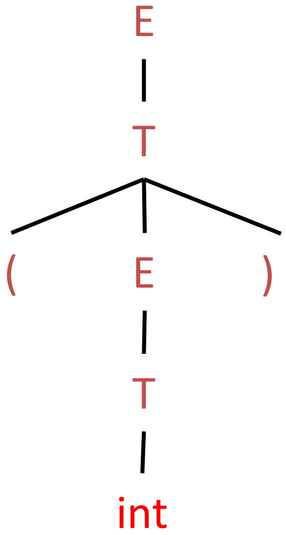
↑

Alex Aiken

Recursive Descent

E → T | T + E

T → int | int * T | (E)



Match! Advance input.

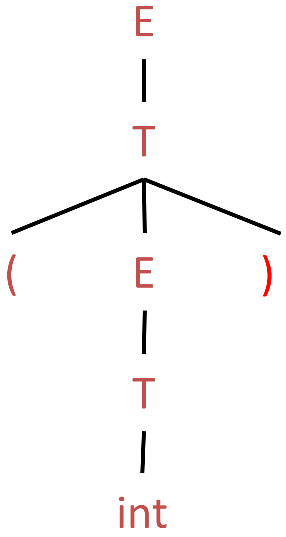
(int_5)

↑

Recursive Descent

$E \rightarrow T \mid T + E$

$T \rightarrow \text{int} \mid \text{int} * T \mid (E)$



Match! Advance input.

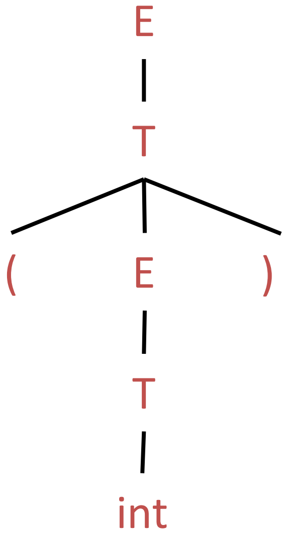
(int_5)

↑

Recursive Descent

$E \rightarrow T \mid T + E$

$T \rightarrow \text{int} \mid \text{int} * T \mid (E)$



End of input, accept.

(int_5)

Recursive Descent

Choose the derivation that is a valid recursive descent parse for the string id + id in the given grammar. Moves that are followed by backtracking are given in red.

E → E’ | E’ + E

E’ → -E’ | id | (E)

○

E
E’
E’ + E
id + E
id + E’
id + id

○

E
E’
-E’
id
(E)
E’ + E
-E’ + E
id + E
id + E’
id + -E’
id + id

○

E
E’ + E
id + E
id + E’
id + id

○

E
E’
id
E’ + E
id + E
id + E’
id + id