

# Bottom-Up Parsing

# What is Bottom-Up Parsing?

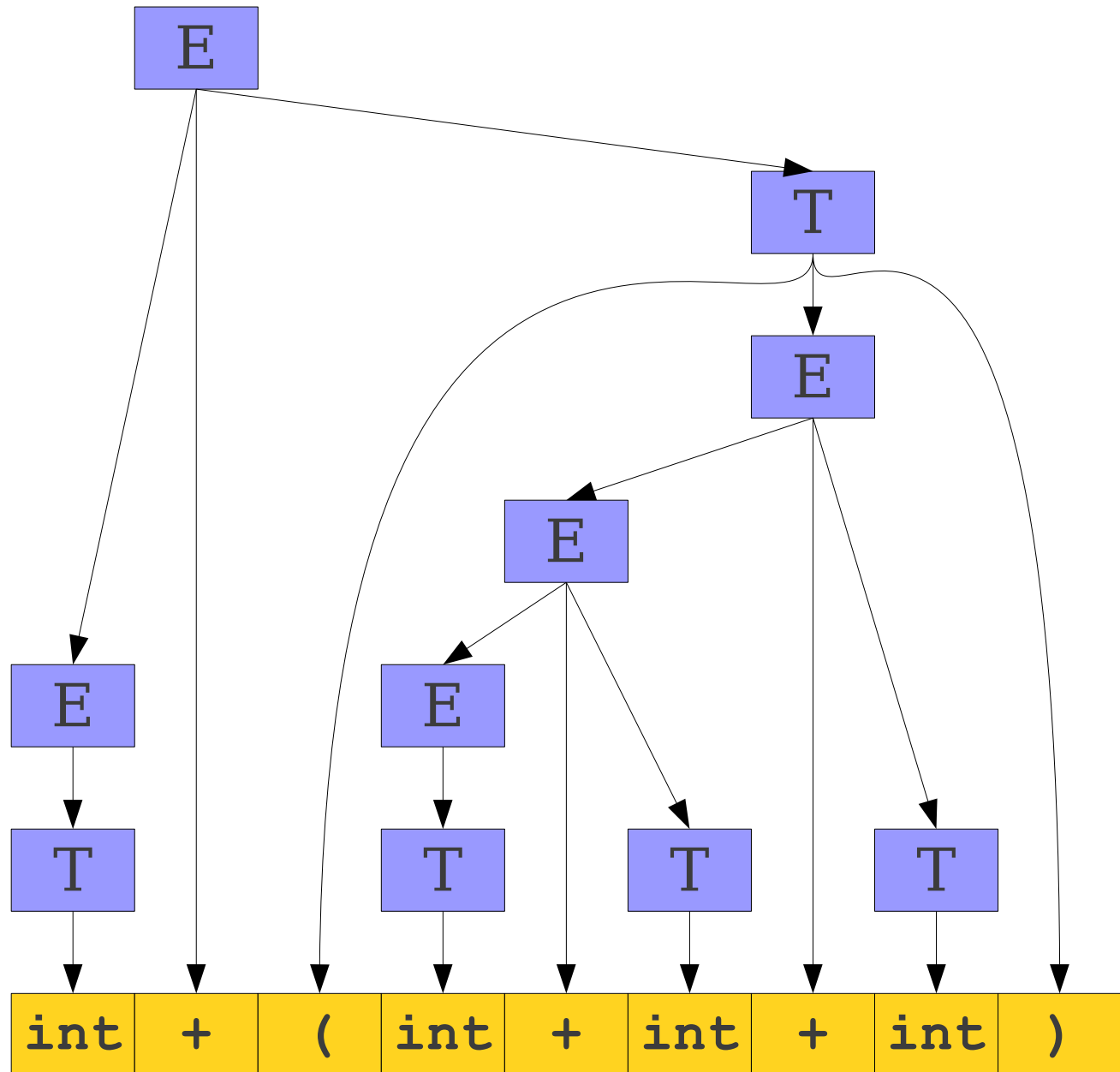
- Idea: Apply productions **in reverse** to convert the user's program to the start symbol.
- As with top-down, could be done with a DFS or BFS, though this is rarely done in practice.
- We'll be exploring four **directional, predictive** bottom-up parsing techniques:
  - **Directional**: Scan the input from left-to-right.
  - **Predictive**: Guess which production should be inverted.

# Bottoms Up!



# One View of a Bottom-Up Parse

$E \rightarrow T$   
 $E \rightarrow E + T$   
 $T \rightarrow \text{int}$   
 $T \rightarrow (E)$



# A Second View of a Bottom-Up Parse

|                            |  |
|----------------------------|--|
| $E \rightarrow T$          | $\text{int} + (\text{int} + \text{int} + \text{int})$    |
| $E \rightarrow E + T$      | $\Rightarrow T + (\text{int} + \text{int} + \text{int})$ |
| $T \rightarrow \text{int}$ | $\Rightarrow E + (\text{int} + \text{int} + \text{int})$ |
| $T \rightarrow (E)$        | $\Rightarrow E + (T + \text{int} + \text{int})$          |
|                            | $\Rightarrow E + (E + \text{int} + \text{int})$          |
|                            | $\Rightarrow E + (E + T + \text{int})$                   |
|                            | $\Rightarrow E + (E + \text{int})$                       |
|                            | $\Rightarrow E + (E + T)$                                |
|                            | $\Rightarrow E + (E)$                                    |
|                            | $\Rightarrow E + T$                                      |
|                            | $\Rightarrow E$  |

# A Second View of a Bottom-Up Parse

|                            |  |
|----------------------------|--|
| $E \rightarrow T$          | $\text{int} + (\text{int} + \text{int} + \text{int})$    |
| $E \rightarrow E + T$      | $\Rightarrow T + (\text{int} + \text{int} + \text{int})$ |
| $T \rightarrow \text{int}$ | $\Rightarrow E + (\text{int} + \text{int} + \text{int})$ |
| $T \rightarrow (E)$        | $\Rightarrow E + (T + \text{int} + \text{int})$          |
|                            | $\Rightarrow E + (E + \text{int} + \text{int})$          |
|                            | $\Rightarrow E + (E + T + \text{int})$                   |
|                            | $\Rightarrow E + (E + \text{int})$                       |
|                            | $\Rightarrow E + (E + T)$                                |
|                            | $\Rightarrow E + (E)$                                    |
|                            | $\Rightarrow E + T$                                      |
|                            | $\Rightarrow E$  |

A left-to-right, bottom-up parse is a rightmost derivation traced in reverse.

# A Third View of a Bottom-Up Parse

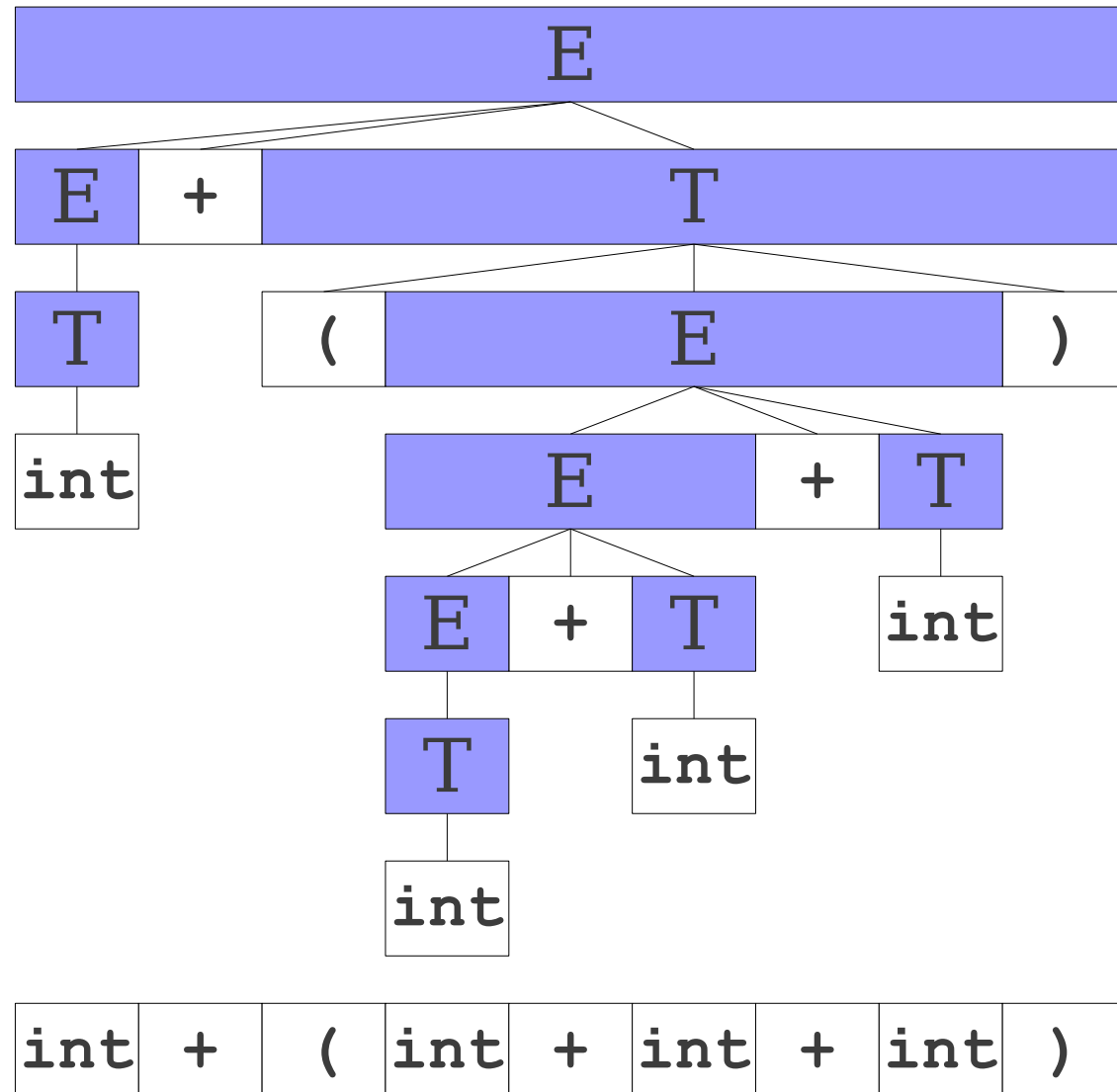
`int + (int + int + int)`  
⇒ `T + (int + int + int)`  
⇒ `E + (int + int + int)`  
⇒ `E + (T + int + int)`  
⇒ `E + (E + int + int)`  
⇒ `E + (E + T + int)`  
⇒ `E + (E + int)`  
⇒ `E + (E + T)`  
⇒ `E + (E)`  
⇒ `E + T`  
⇒ `E`

Each step in this bottom-up parse is called a **reduction**.  
We **reduce** a substring of the sentential form back to a nonterminal.



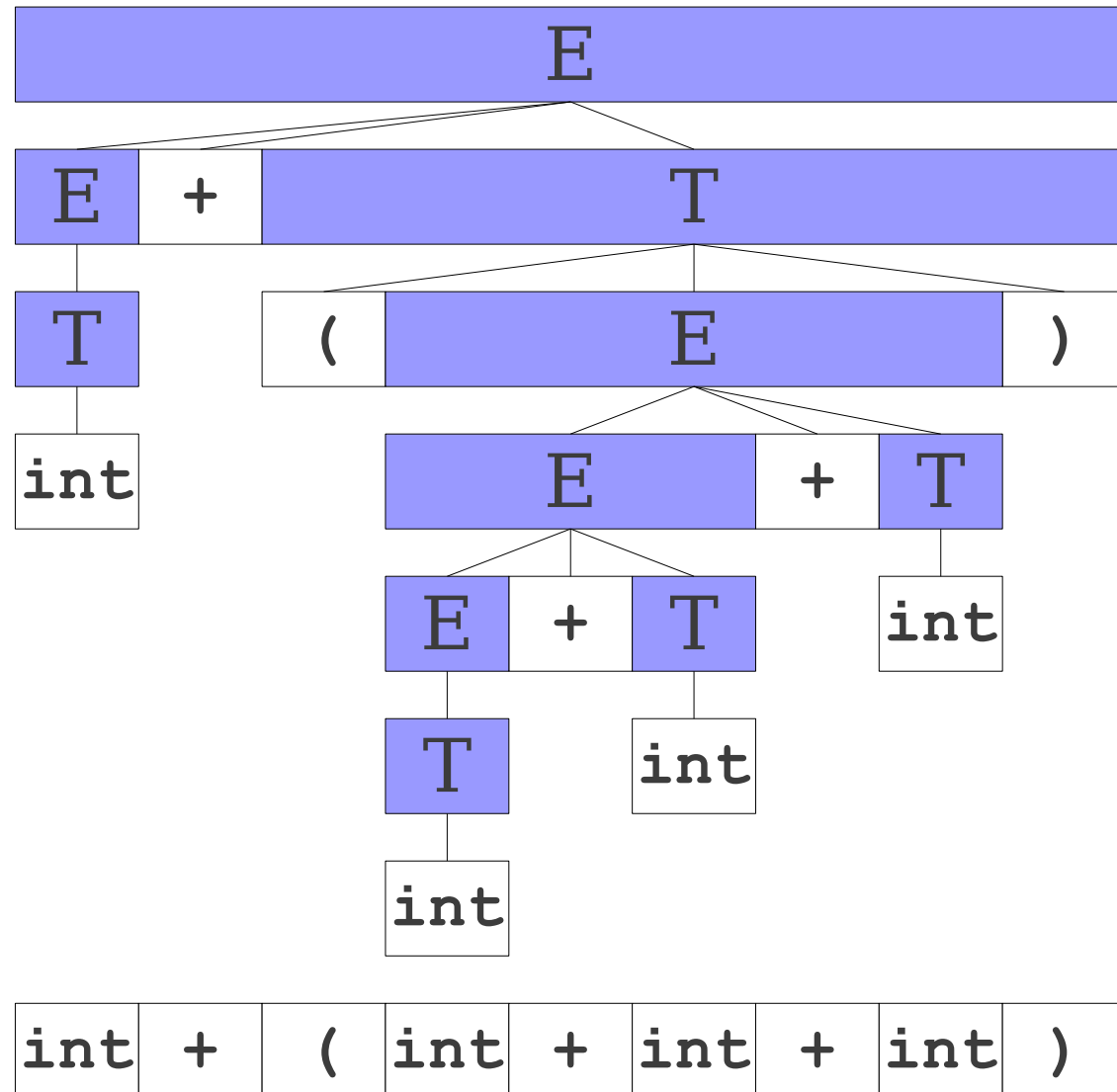
# A Third View of a Bottom-Up Parse

`int + (int + int + int)`  
⇒ `T + (int + int + int)`  
⇒ `E + (int + int + int)`  
⇒ `E + (T + int + int)`  
⇒ `E + (E + int + int)`  
⇒ `E + (E + T + int)`  
⇒ `E + (E + int)`  
⇒ `E + (E + T)`  
⇒ `E + (E)`  
⇒ `E + T`  
⇒ `E`



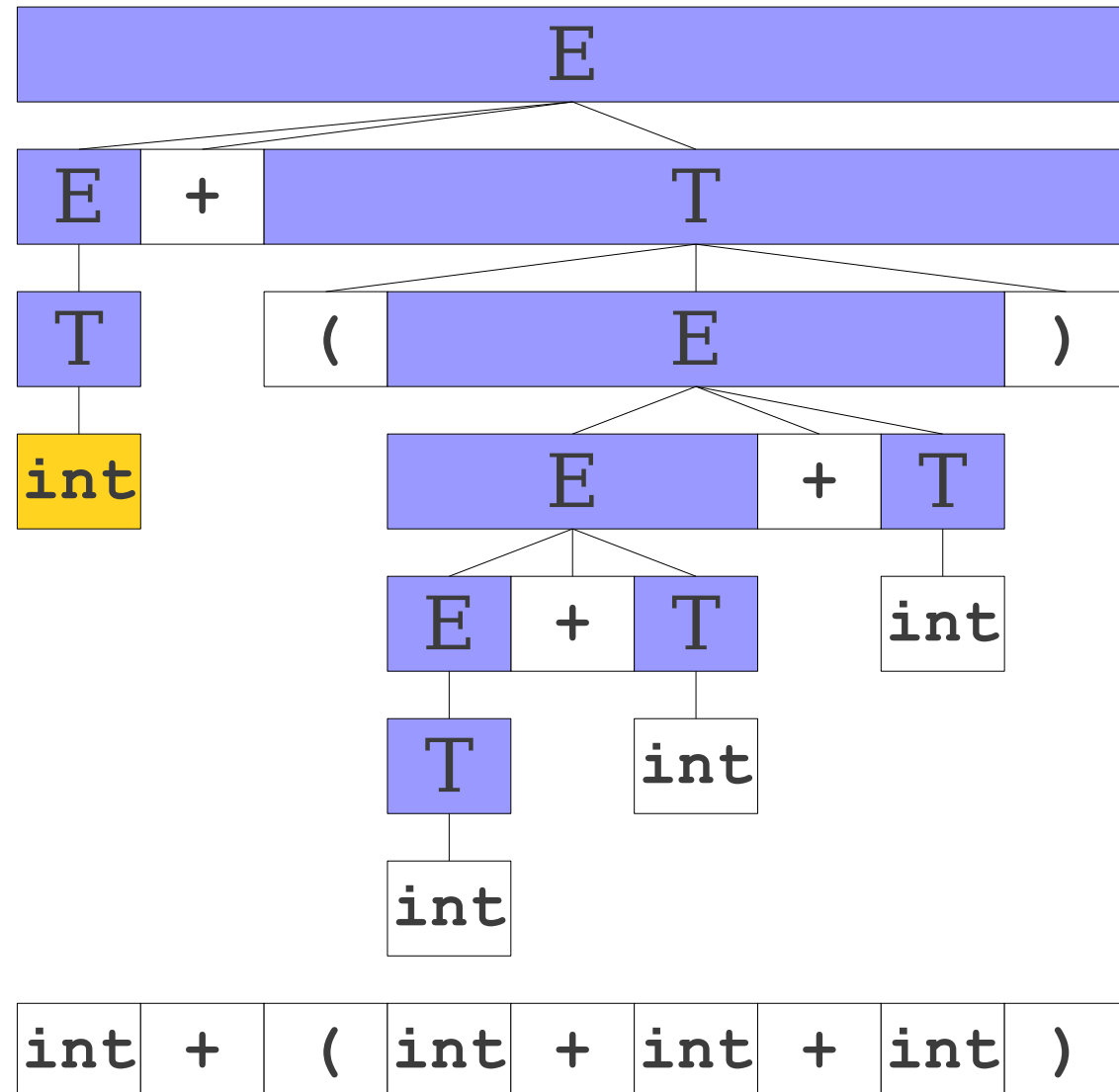
# A Third View of a Bottom-Up Parse

**int** + (int + int + int)  
⇒ **T** + (int + int + int)  
⇒ **E** + (int + int + int)  
⇒ **E** + (**T** + int + int)  
⇒ **E** + (**E** + int + int)  
⇒ **E** + (**E** + **T** + int)  
⇒ **E** + (**E** + int)  
⇒ **E** + (**E** + **T**)  
⇒ **E** + (**E**)  
⇒ **E** + **T**  
⇒ **E**



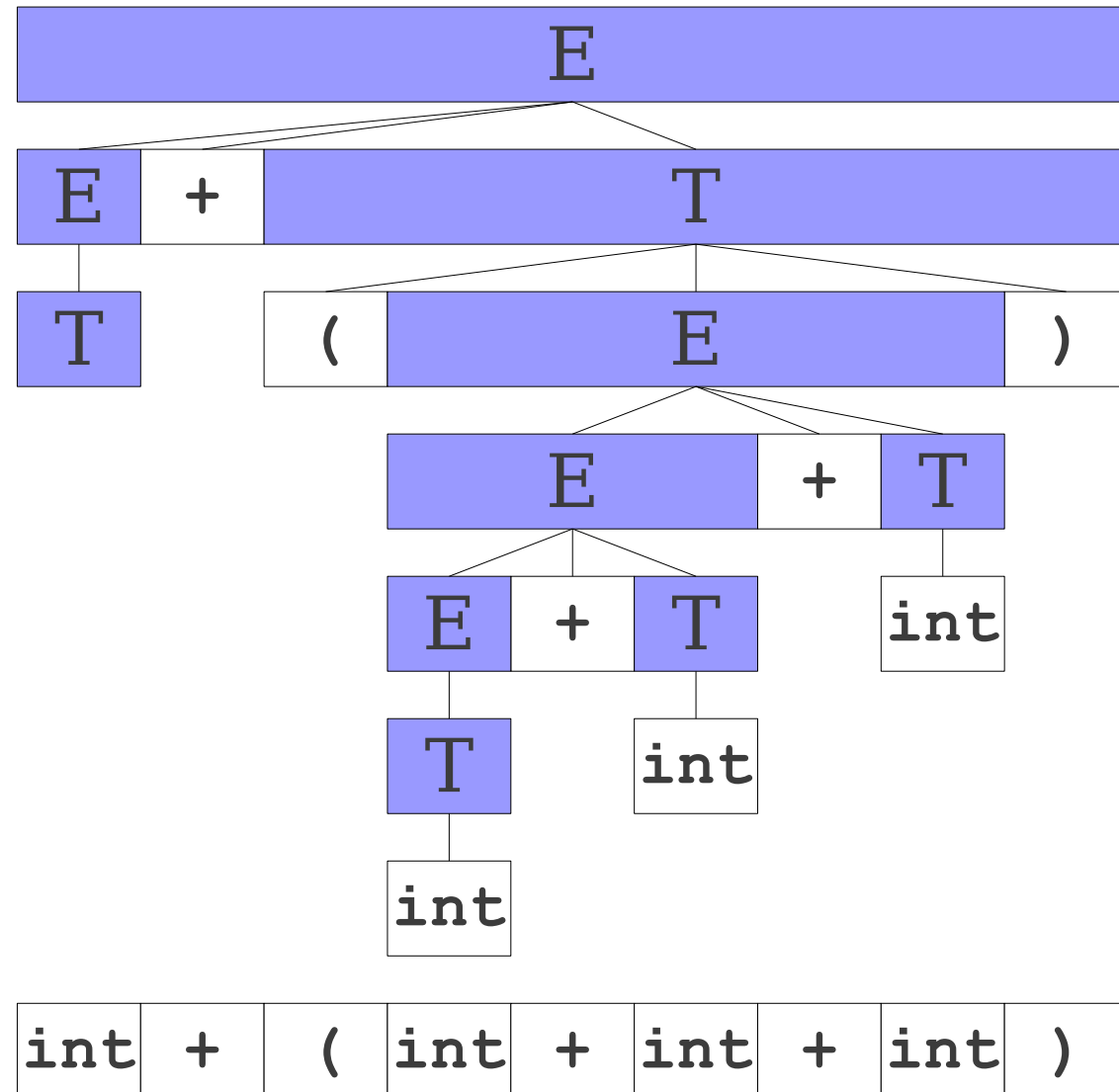
# A Third View of a Bottom-Up Parse

**int** + (int + int + int)  
⇒ **T** + (int + int + int)  
⇒ **E** + (int + int + int)  
⇒ **E** + (**T** + int + int)  
⇒ **E** + (**E** + int + int)  
⇒ **E** + (**E** + **T** + int)  
⇒ **E** + (**E** + int)  
⇒ **E** + (**E** + **T**)  
⇒ **E** + (**E**)  
⇒ **E** + **T**  
⇒ **E**



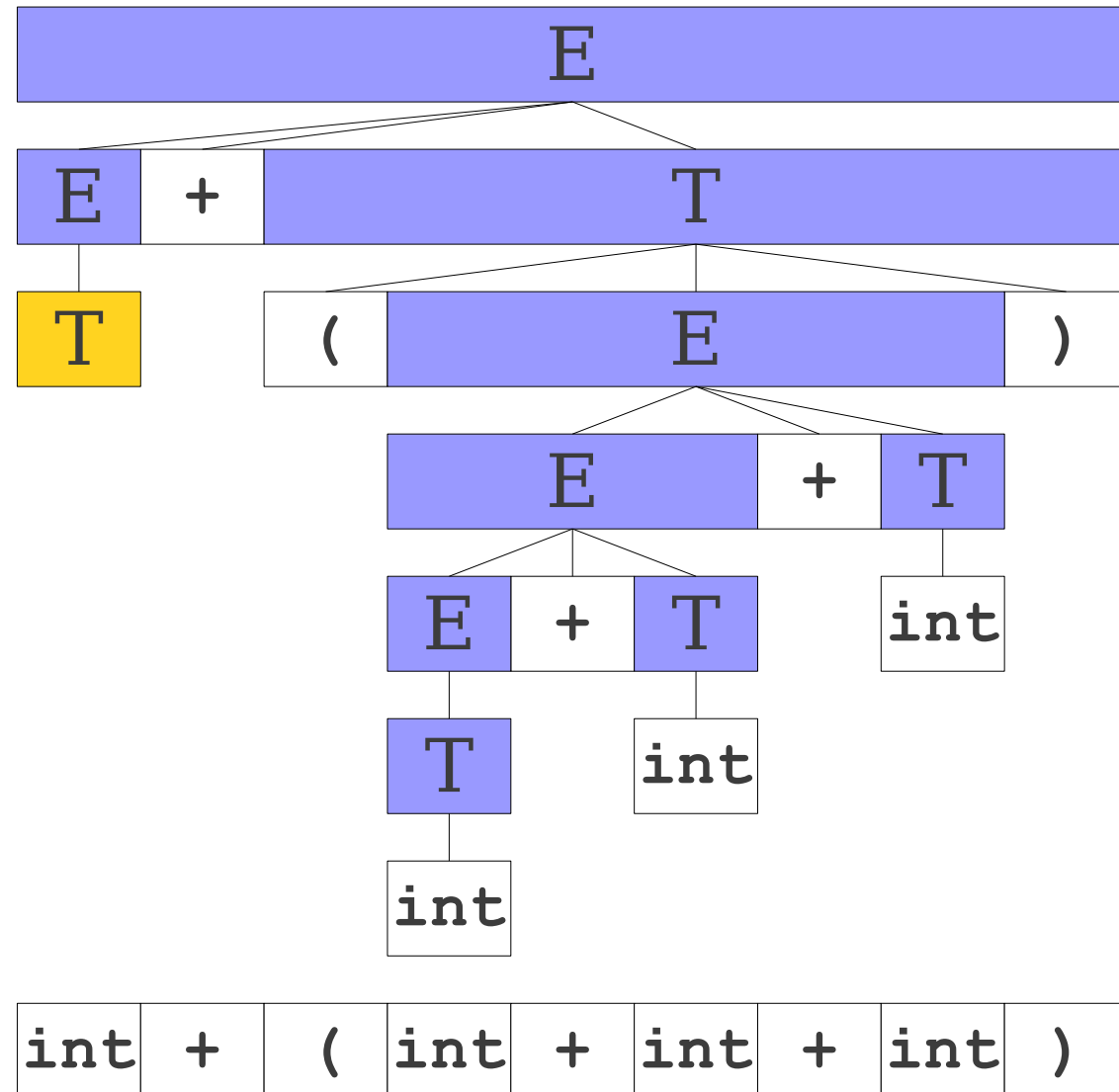
# A Third View of a Bottom-Up Parse

$\Rightarrow T + (int + int + int)$   
 $\Rightarrow E + (int + int + int)$   
 $\Rightarrow E + (T + int + int)$   
 $\Rightarrow E + (E + int + int)$   
 $\Rightarrow E + (E + T + int)$   
 $\Rightarrow E + (E + int)$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$



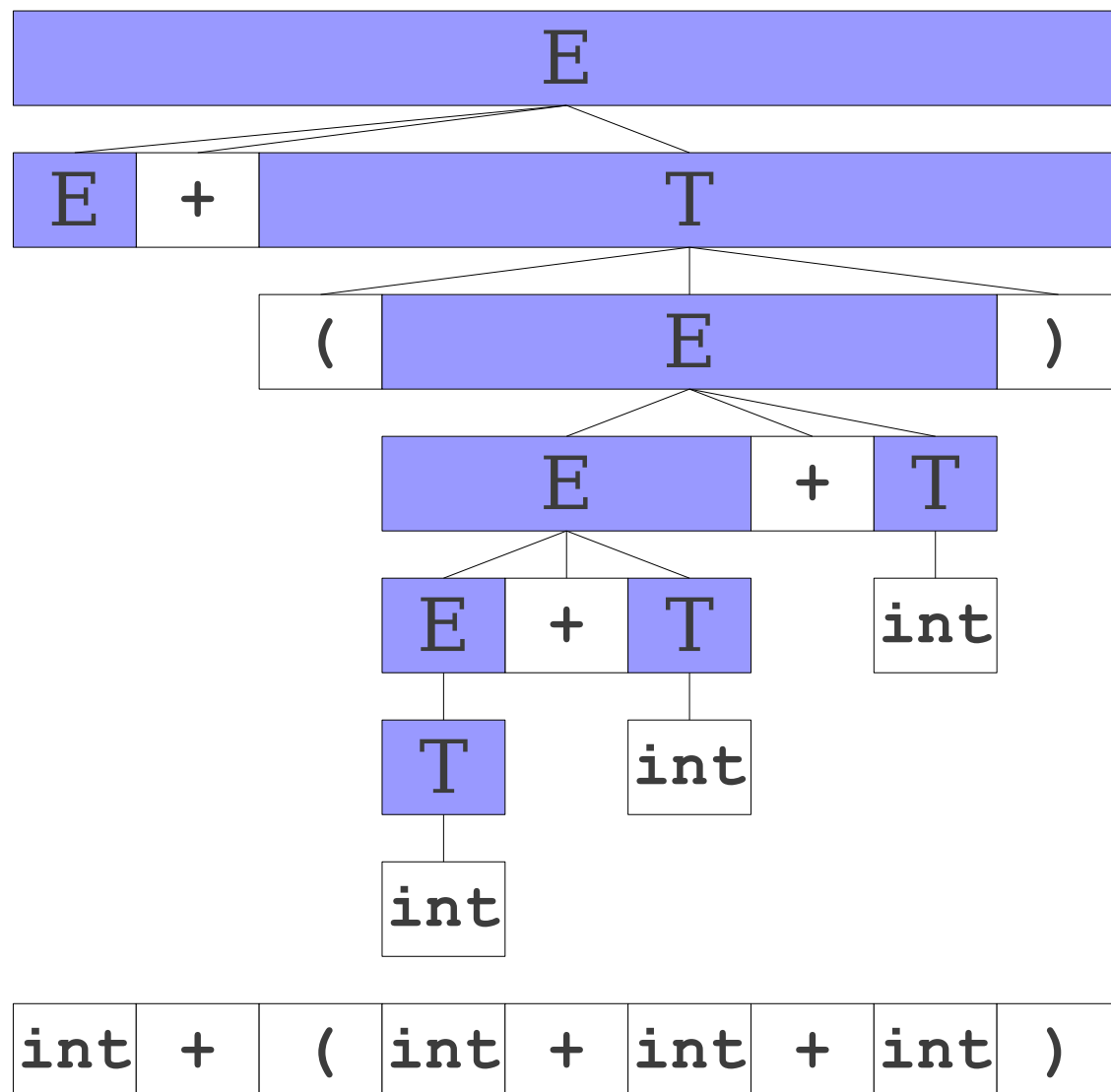
# A Third View of a Bottom-Up Parse

$\Rightarrow$  **T** + (int + int + int)  
 $\Rightarrow$  **E** + (int + int + int)  
 $\Rightarrow$  **E** + (**T** + int + int)  
 $\Rightarrow$  **E** + (**E** + int + int)  
 $\Rightarrow$  **E** + (**E** + **T** + int)  
 $\Rightarrow$  **E** + (**E** + int)  
 $\Rightarrow$  **E** + (**E** + **T**)  
 $\Rightarrow$  **E** + (**E**)  
 $\Rightarrow$  **E** + **T**  
 $\Rightarrow$  **E**



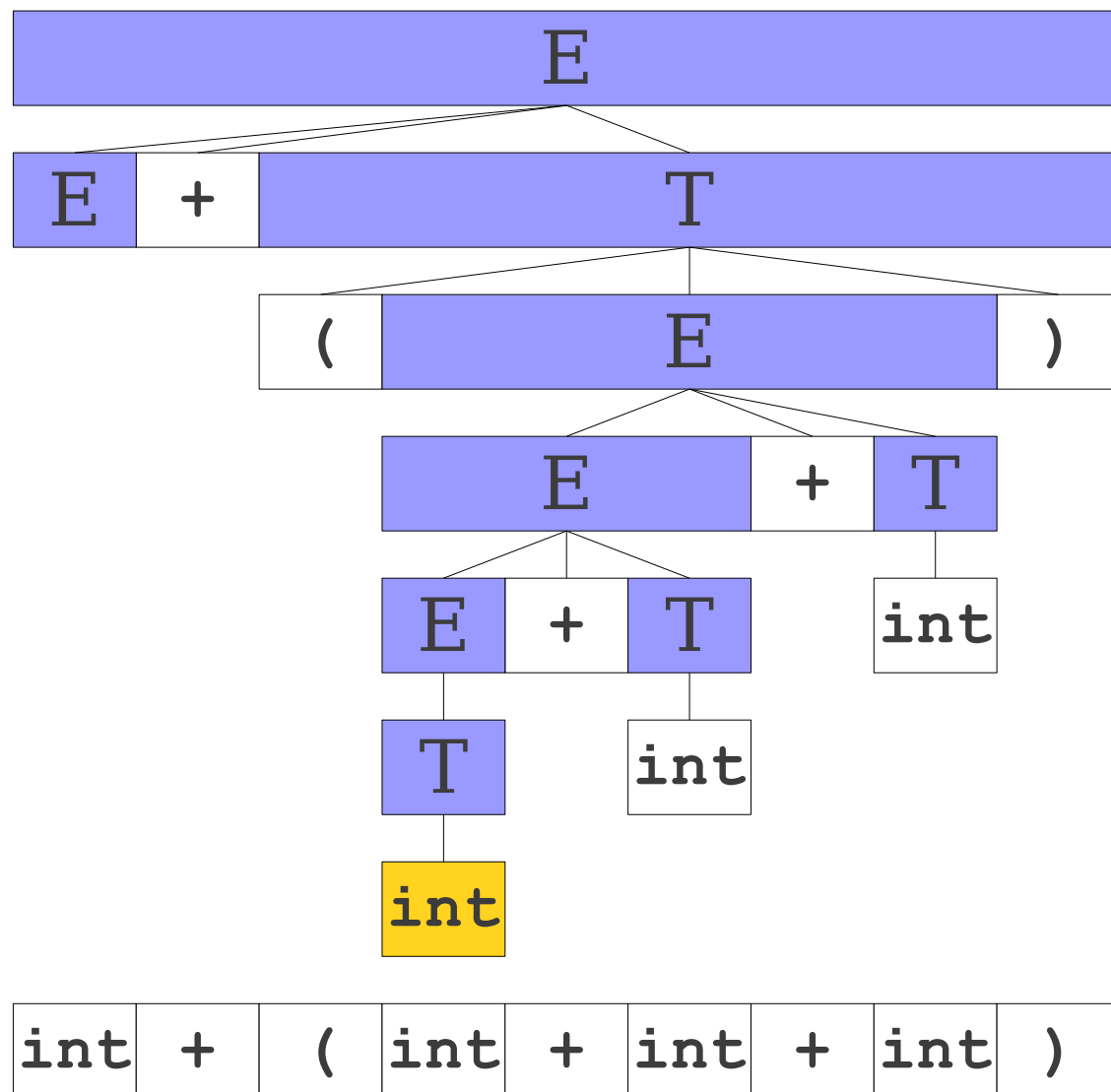
# A Third View of a Bottom-Up Parse

$\Rightarrow E + (int + int + int)$   
 $\Rightarrow E + (T + int + int)$   
 $\Rightarrow E + (E + int + int)$   
 $\Rightarrow E + (E + T + int)$   
 $\Rightarrow E + (E + int)$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$



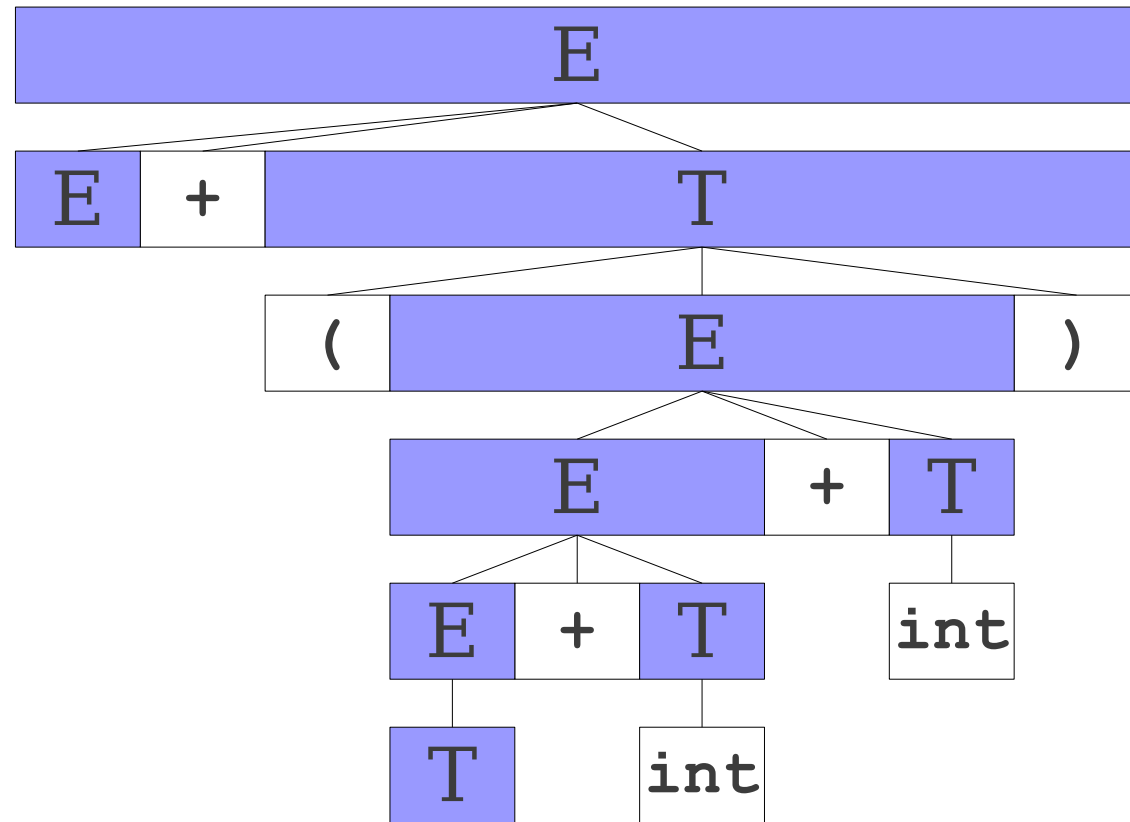
# A Third View of a Bottom-Up Parse

$\Rightarrow E + (\text{int} + \text{int} + \text{int})$   
 $\Rightarrow E + (T + \text{int} + \text{int})$   
 $\Rightarrow E + (E + \text{int} + \text{int})$   
 $\Rightarrow E + (E + T + \text{int})$   
 $\Rightarrow E + (E + \text{int})$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$



# A Third View of a Bottom-Up Parse

$\Rightarrow E + (T + \text{int} + \text{int})$   
 $\Rightarrow E + (E + \text{int} + \text{int})$   
 $\Rightarrow E + (E + T + \text{int})$   
 $\Rightarrow E + (E + \text{int})$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$

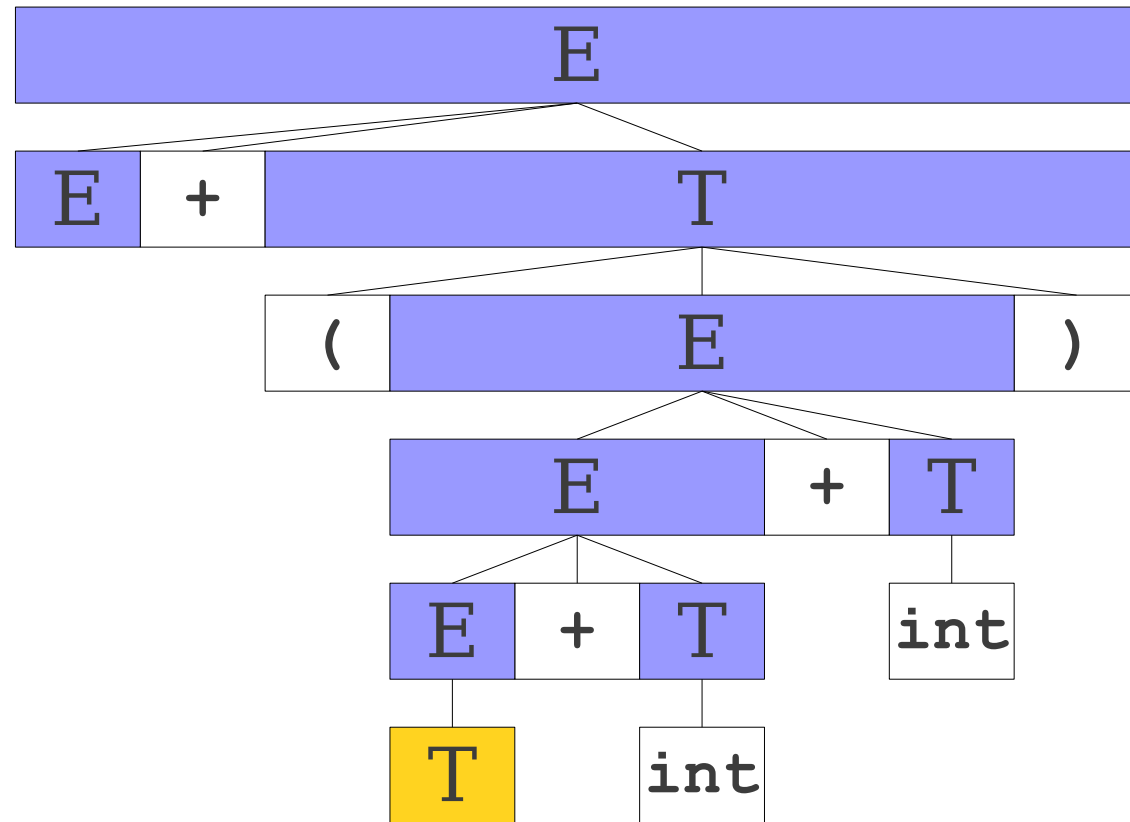


|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|



# A Third View of a Bottom-Up Parse

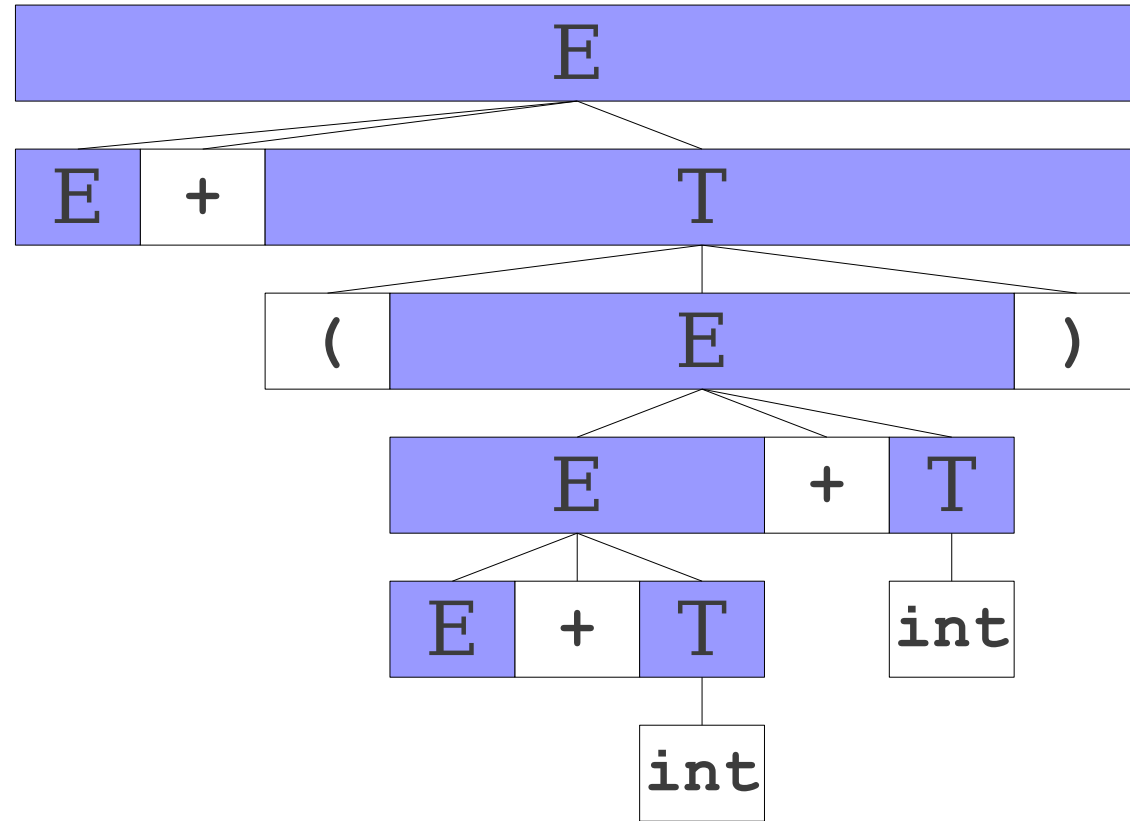
$\Rightarrow E + (T + \text{int} + \text{int})$   
 $\Rightarrow E + (E + \text{int} + \text{int})$   
 $\Rightarrow E + (E + T + \text{int})$   
 $\Rightarrow E + (E + \text{int})$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$



|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

# A Third View of a Bottom-Up Parse

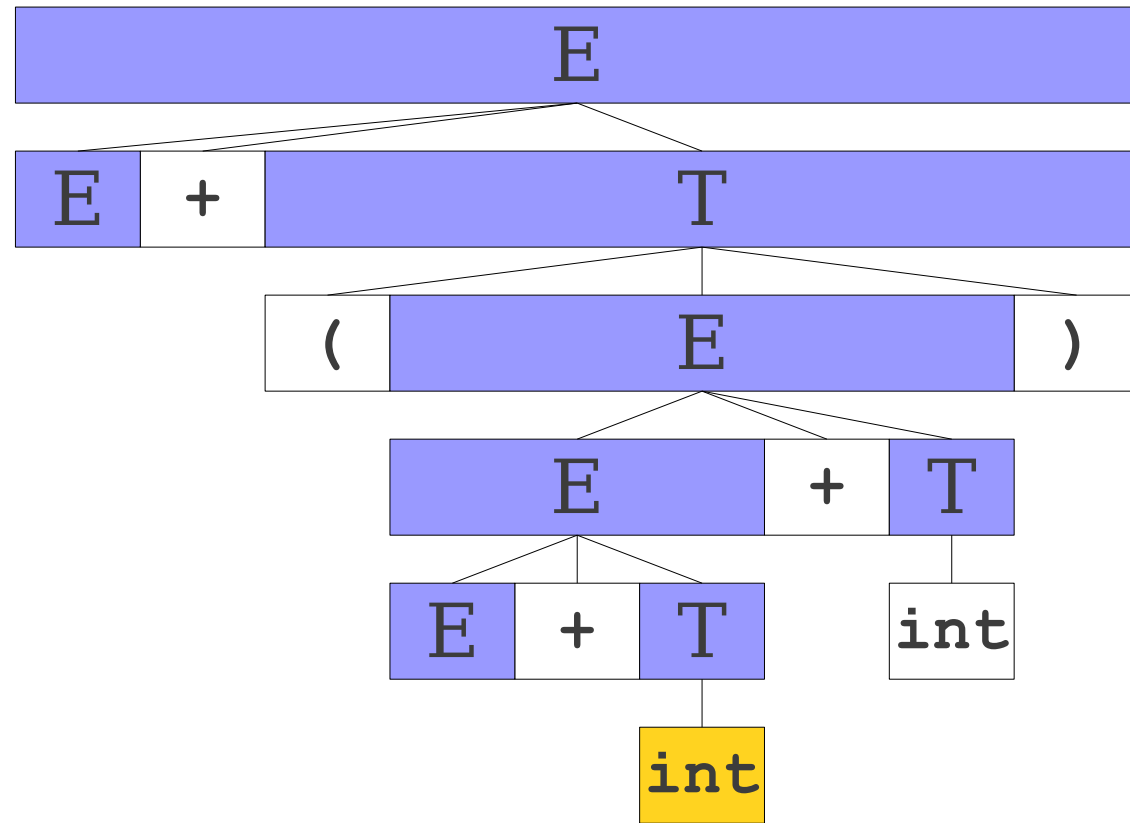
$\Rightarrow E + (E + \text{int} + \text{int})$   
 $\Rightarrow E + (E + T + \text{int})$   
 $\Rightarrow E + (E + \text{int})$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$



int + ( int + int + int )

# A Third View of a Bottom-Up Parse

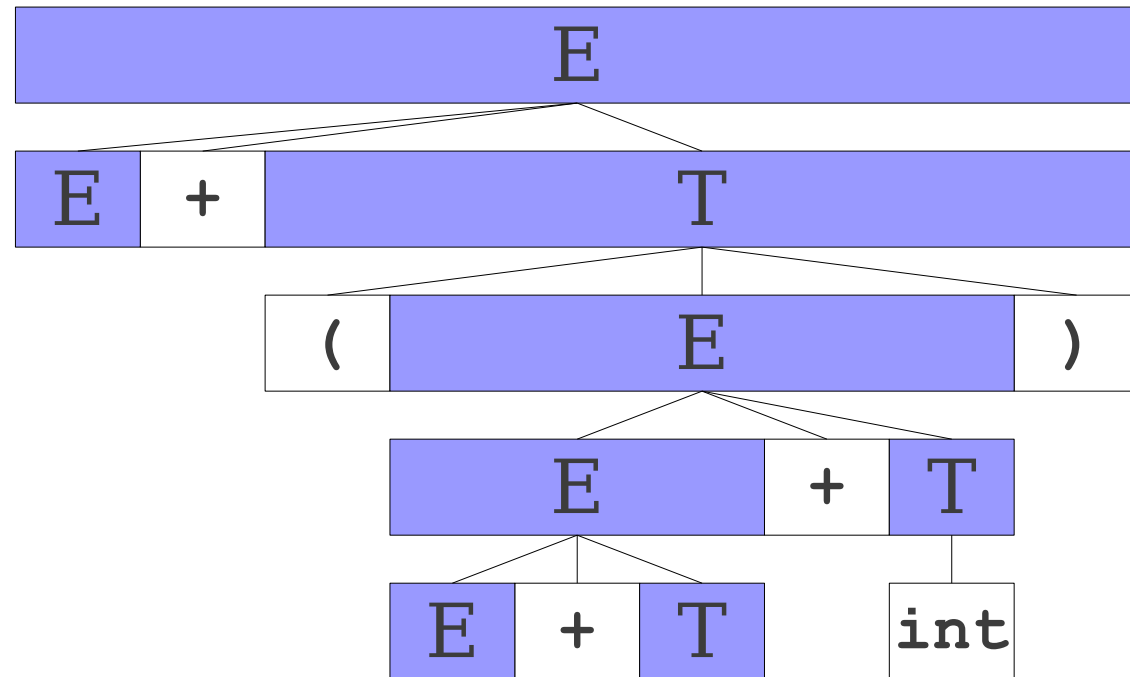
$\Rightarrow E + (E + \text{int} + \text{int})$   
 $\Rightarrow E + (E + T + \text{int})$   
 $\Rightarrow E + (E + \text{int})$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$



|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

# A Third View of a Bottom-Up Parse

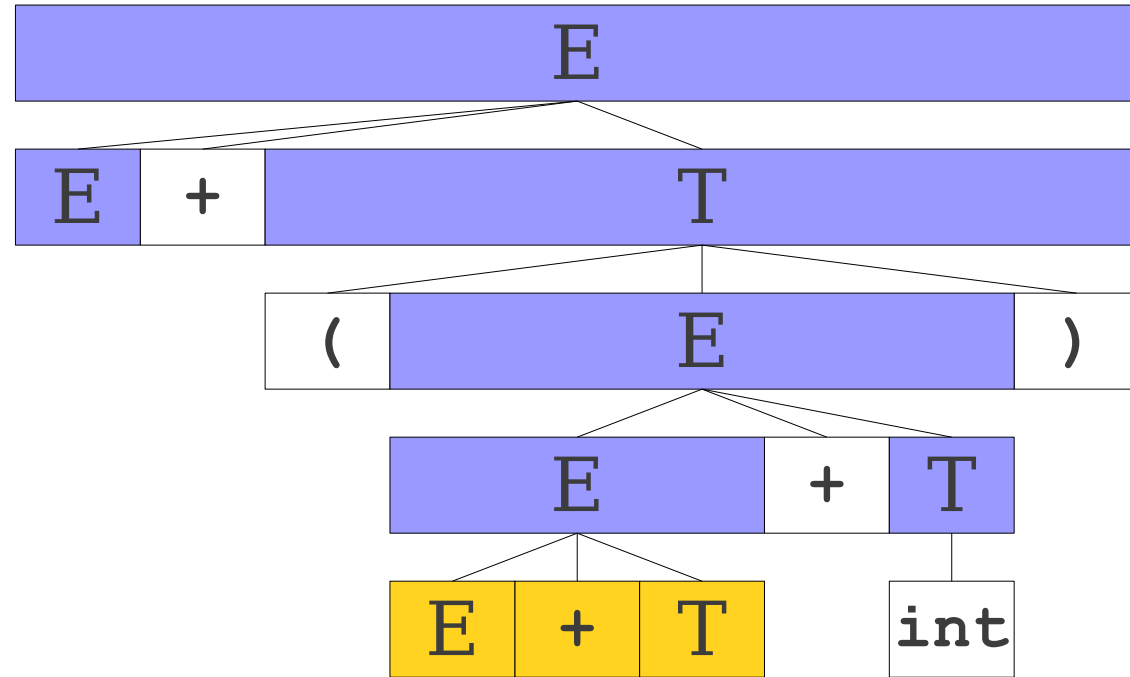
$\Rightarrow E + (E + T + \text{int})$   
 $\Rightarrow E + (E + \text{int})$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$



|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

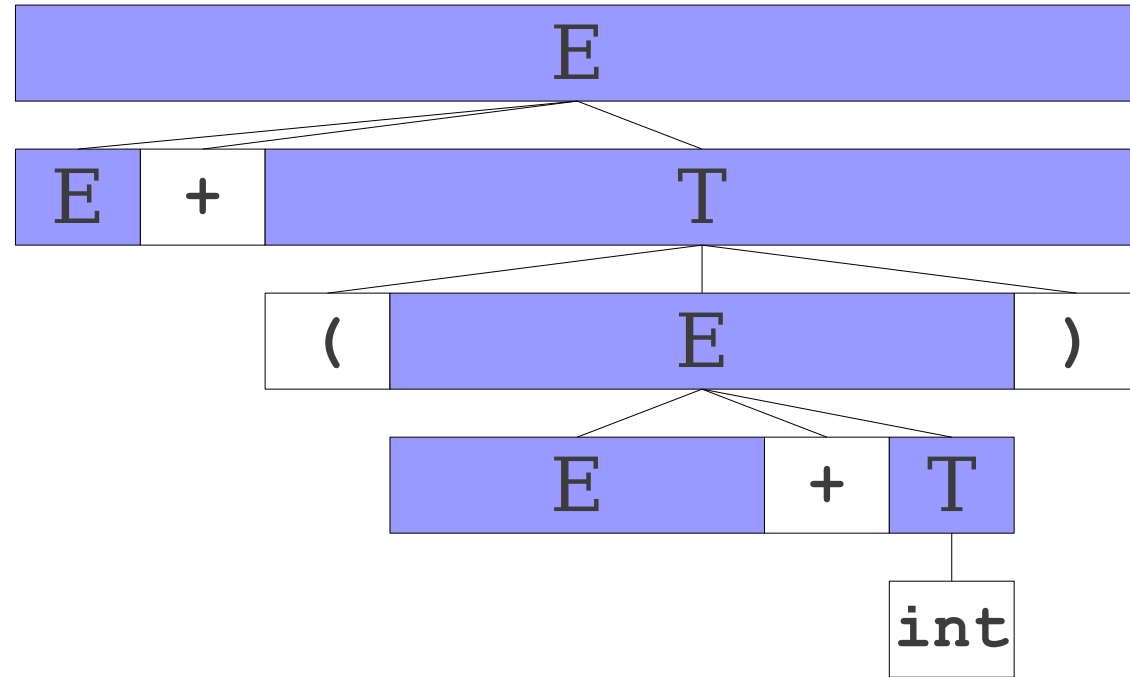
# A Third View of a Bottom-Up Parse

$\Rightarrow E + (E + T + \text{int})$   
 $\Rightarrow E + (E + \text{int})$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$



|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

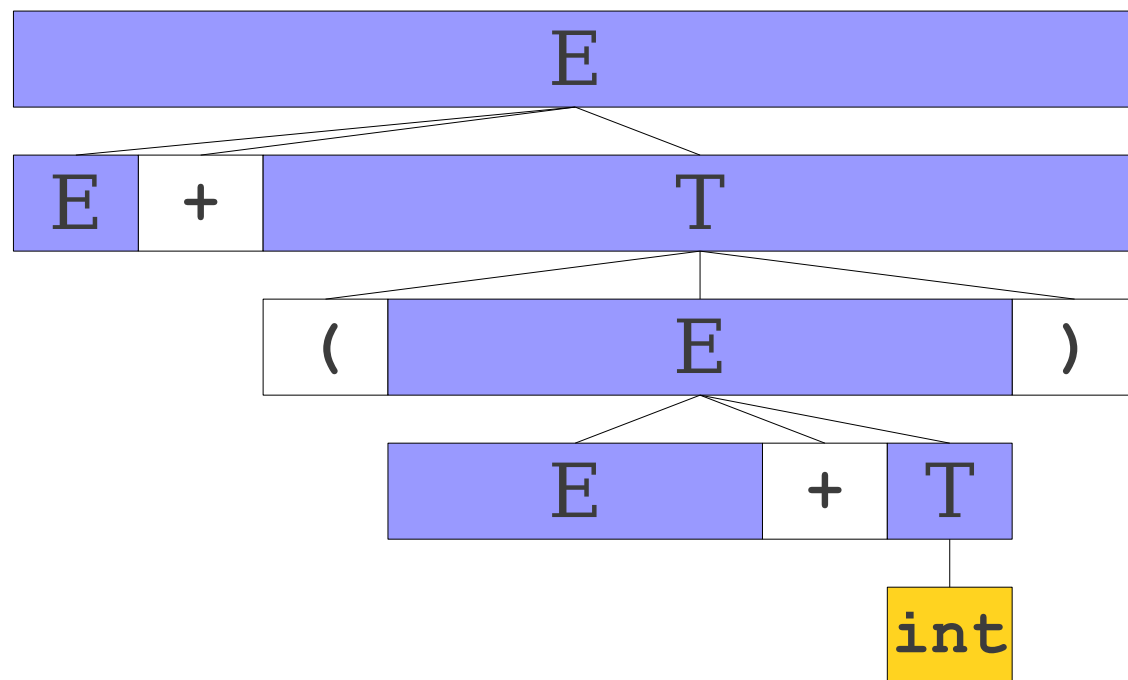
# A Third View of a Bottom-Up Parse



$\Rightarrow \mathbf{E} + (\mathbf{E} + \mathbf{int})$   
 $\Rightarrow \mathbf{E} + (\mathbf{E} + \mathbf{T})$   
 $\Rightarrow \mathbf{E} + (\mathbf{E})$   
 $\Rightarrow \mathbf{E} + \mathbf{T}$   
 $\Rightarrow \mathbf{E}$

|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

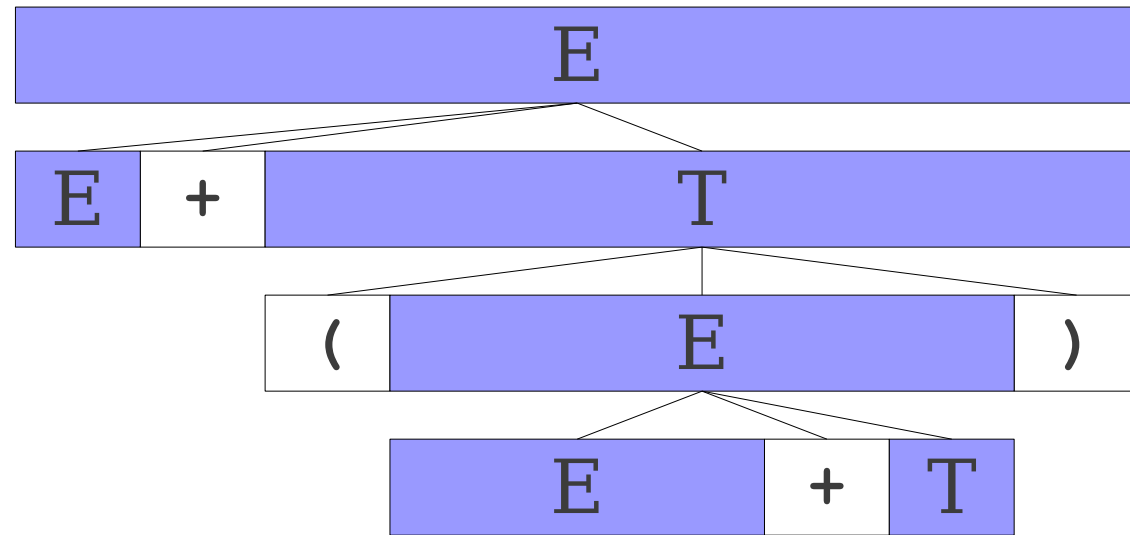
# A Third View of a Bottom-Up Parse



$\Rightarrow E + (E + \text{int})$   
 $\Rightarrow E + (E + T)$   
 $\Rightarrow E + (E)$   
 $\Rightarrow E + T$   
 $\Rightarrow E$

|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

# A Third View of a Bottom-Up Parse

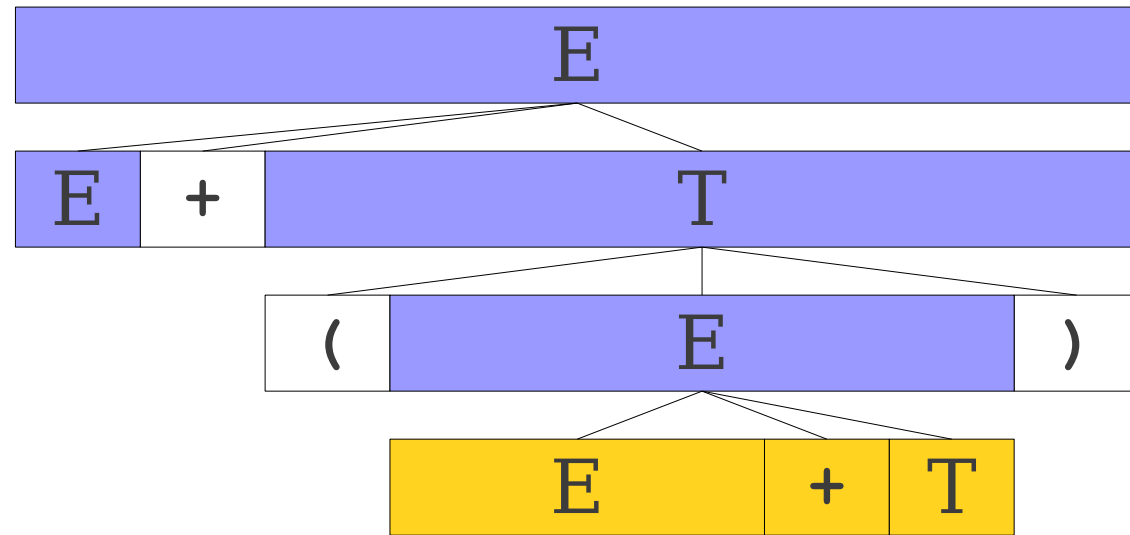


$\Rightarrow \mathbf{E} + (\mathbf{E} + \mathbf{T})$   
 $\Rightarrow \mathbf{E} + (\mathbf{E})$   
 $\Rightarrow \mathbf{E} + \mathbf{T}$   
 $\Rightarrow \mathbf{E}$

|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|



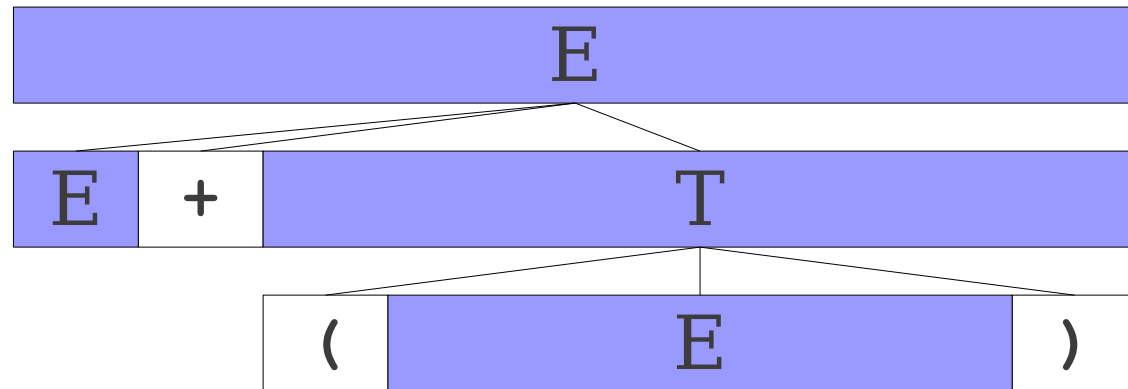
# A Third View of a Bottom-Up Parse



$\Rightarrow \mathbf{E} + (\mathbf{E} + \mathbf{T})$   
 $\Rightarrow \mathbf{E} + (\mathbf{E})$   
 $\Rightarrow \mathbf{E} + \mathbf{T}$   
 $\Rightarrow \mathbf{E}$

|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

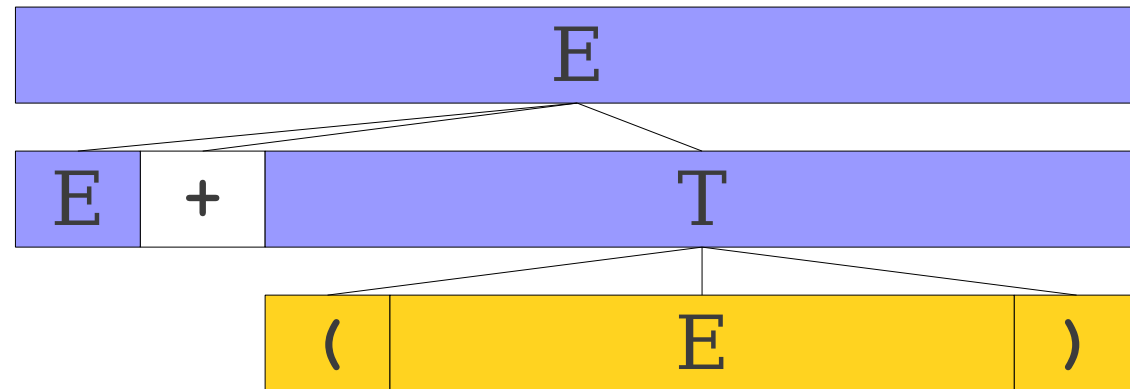
# A Third View of a Bottom-Up Parse



$\Rightarrow \mathbf{E} + (\mathbf{E})$   
 $\Rightarrow \mathbf{E} + \mathbf{T}$   
 $\Rightarrow \mathbf{E}$

|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

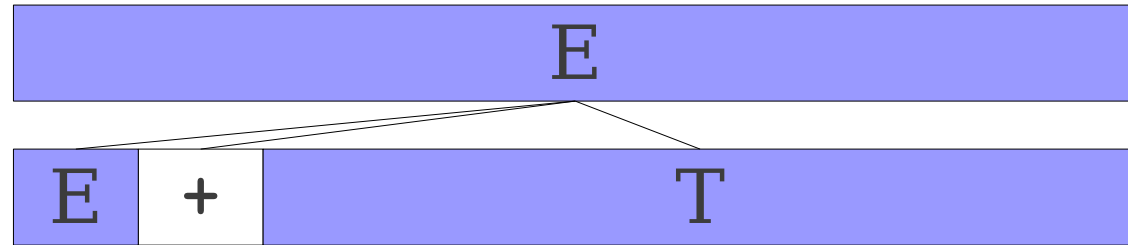
# A Third View of a Bottom-Up Parse



$\Rightarrow \mathbf{E} + \mathbf{(E)}$   
 $\Rightarrow \mathbf{E} + \mathbf{T}$   
 $\Rightarrow \mathbf{E}$

|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

# A Third View of a Bottom-Up Parse

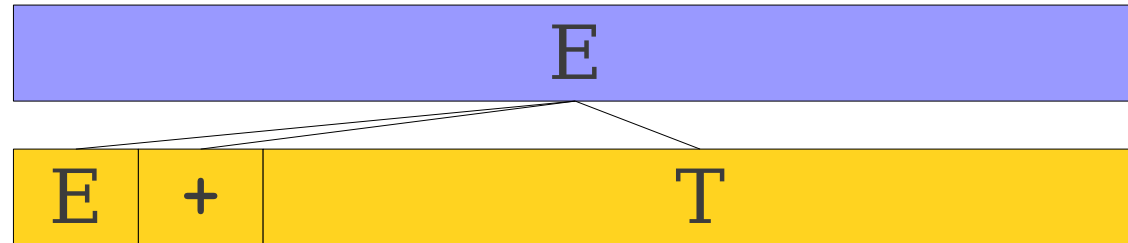


$\Rightarrow \mathbf{E} + \mathbf{T}$

$\Rightarrow \mathbf{E}$

|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

# A Third View of a Bottom-Up Parse



$\Rightarrow \mathbf{E} + \mathbf{T}$

$\Rightarrow \mathbf{E}$

|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

# A Third View of a Bottom-Up Parse

E

⇒ E

|     |   |   |     |   |     |   |     |   |
|-----|---|---|-----|---|-----|---|-----|---|
| int | + | ( | int | + | int | + | int | ) |
|-----|---|---|-----|---|-----|---|-----|---|

# Handles

- The **handle** of a parse tree  $T$  is the leftmost complete cluster of leaf nodes.
- A left-to-right, bottom-up parse works by iteratively searching for a handle, then reducing the handle.

# Summarizing Our Intuition

- Our first intuition (reconstructing the parse tree bottom-up) motivates how the parsing should work.
- Our second intuition (rightmost derivation in reverse) describes the order in which we should build the parse tree.
- Our third intuition (handle pruning) is the basis for the bottom-up parsing algorithms we will explore.



# A Detail about Handles

**E** → **F**

**E** → **E** + **F**

**F** → **F** \* **T**

**F** → **T**

**T** → **int**

**T** → (**E**)

|     |   |     |   |     |
|-----|---|-----|---|-----|
| int | + | int | * | int |
|-----|---|-----|---|-----|

# A Detail about Handles

**E** → **F**

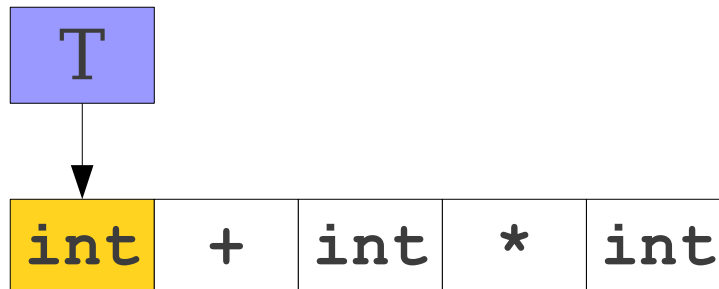
**E** → **E** + **F**

**F** → **F** \* **T**

**F** → **T**

**T** → **int**

**T** → (**E**)



# A Detail about Handles

**E** → **F**

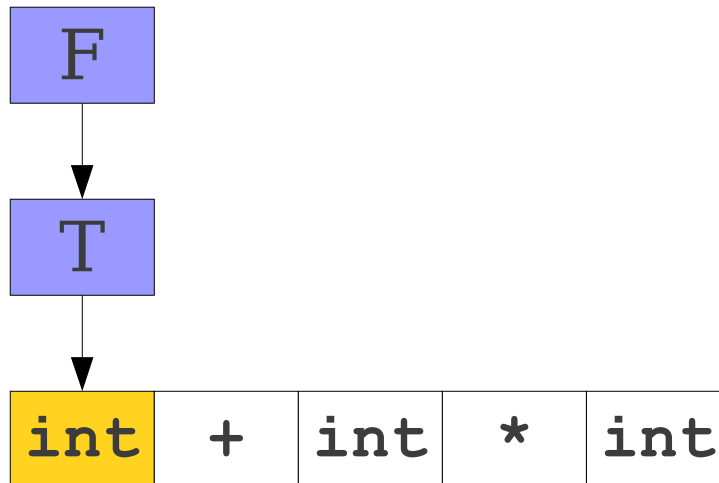
**E** → **E** + **F**

**F** → **F** \* **T**

**F** → **T**

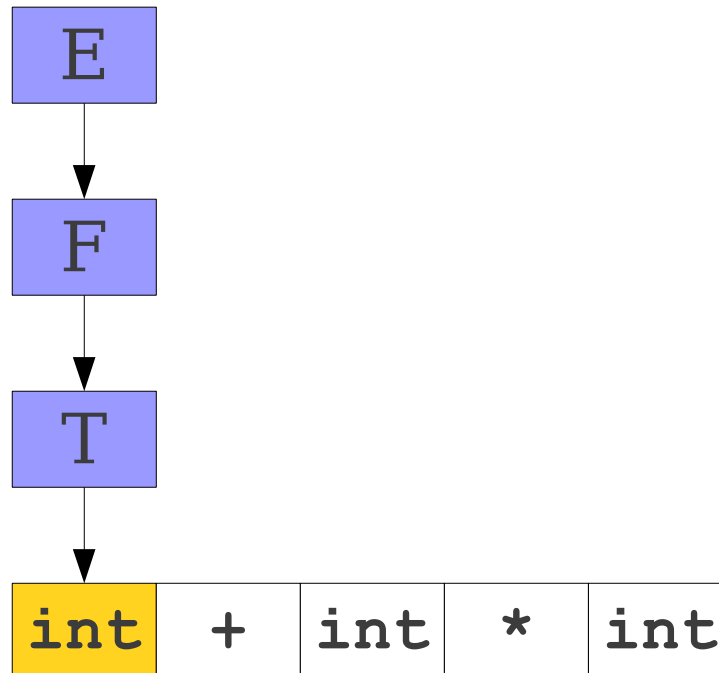
**T** → **int**

**T** → (**E**)



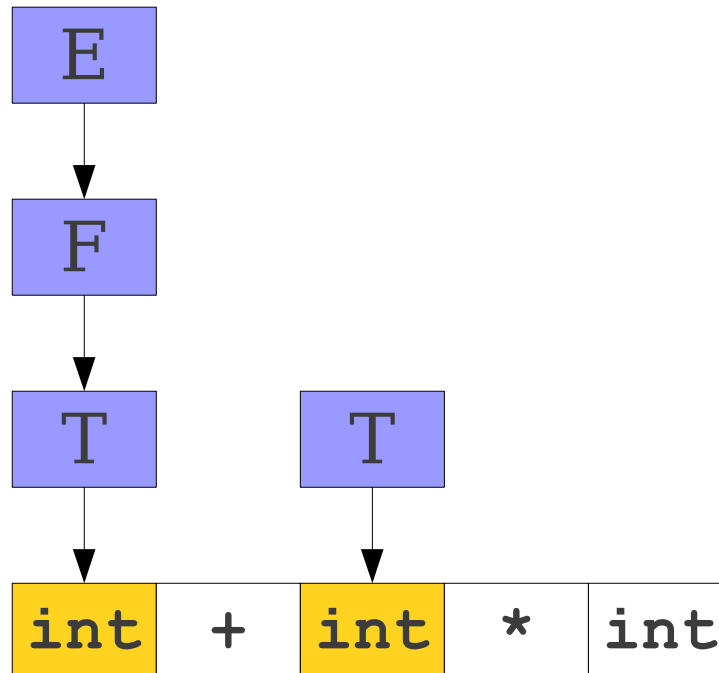
# A Detail about Handles

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



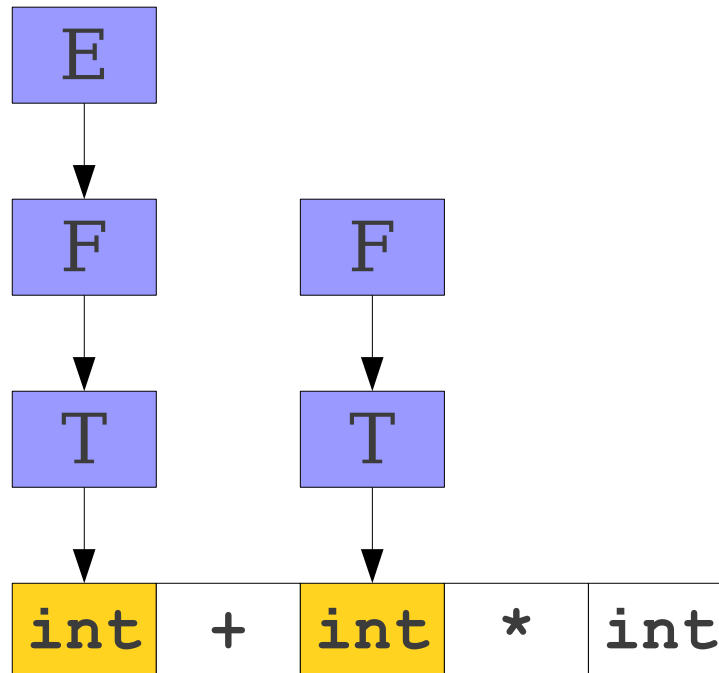
# A Detail about Handles

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



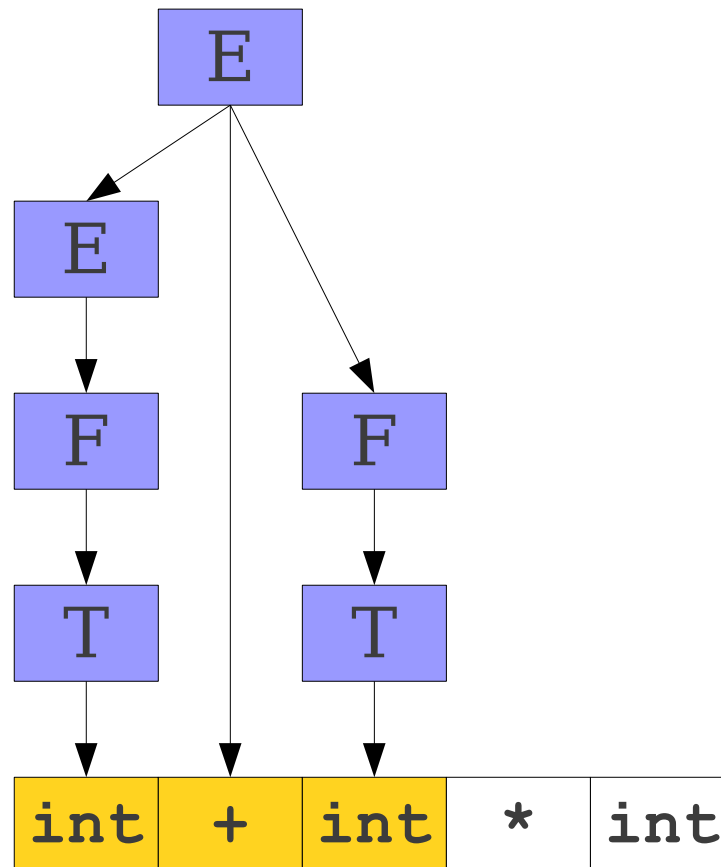
# A Detail about Handles

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



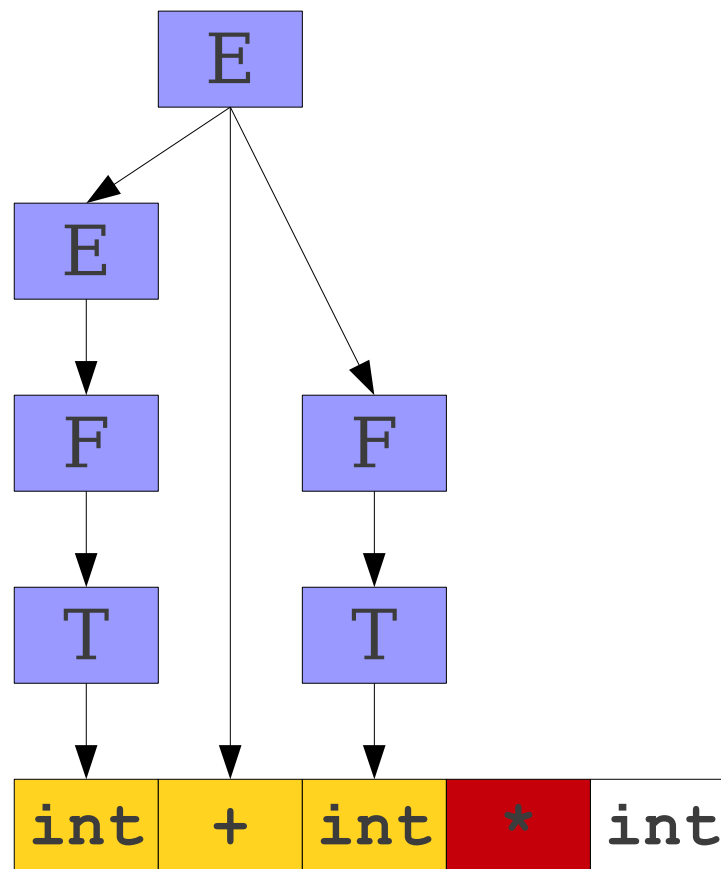
# A Detail about Handles

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



# A Detail about Handles

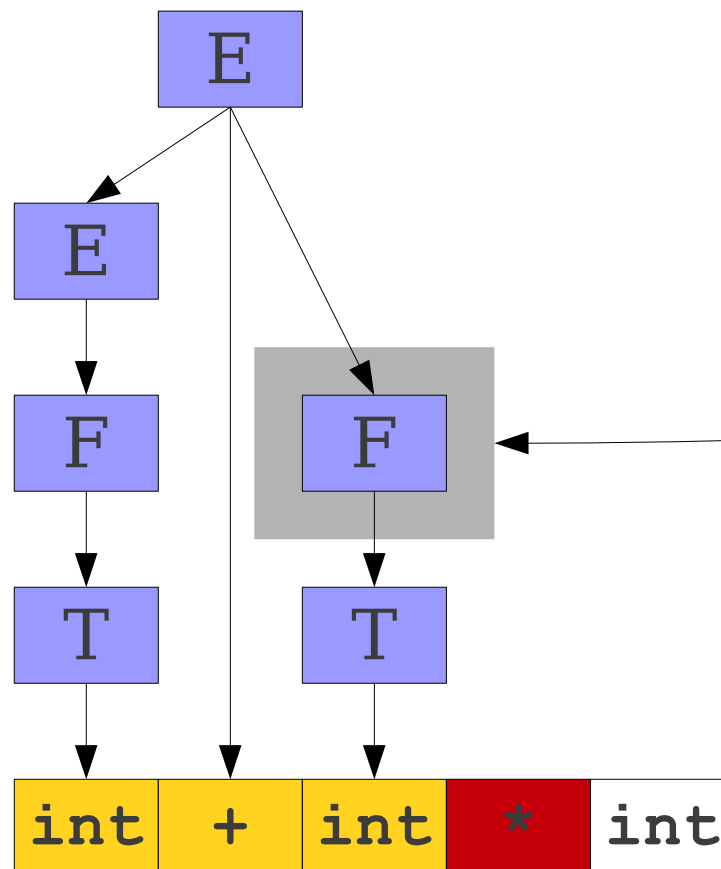
$E \rightarrow F$   
 $E \rightarrow E + F$   
 $F \rightarrow F * T$   
 $F \rightarrow T$   
 $T \rightarrow \text{int}$   
 $T \rightarrow (E)$





# A Detail about Handles

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



This reduction  
wasn't a handle!

The leftmost reduction  
isn't always the handle.

# Finding Handles

- Where do we look for handles?
  - Where in the string might the handle be?
- How do we search for possible handles?
  - Once we know where to search, how do we identify candidate handles?
- How do we recognize handles?
  - Once we've found a candidate handle, how do we check that it really is the handle?

# Question One:

Where are handles?

# Where are Handles?

- Recall: A left-to-right, bottom-up parse traces a rightmost derivation in reverse.
- Each time we do a reduction, we are reversing a production applied to the *rightmost* nonterminal symbol.
- Suppose that our current sentential form is  $\alpha\gamma\omega$ , where  $\gamma$  is the handle and  $A \rightarrow \gamma$  is a production rule.
- After reducing  $\gamma$  back to  $A$ , we have the string  $\alpha A \omega$ .
- Thus  $\omega$  must consist purely of terminals, since otherwise the reduction we just did was not for the rightmost terminal.

# Why This Matters

- Suppose we want to parse the string  $y$ .
- We will break  $y$  into two parts,  $\alpha$  and  $\omega$ , where
  - $\alpha$  consists of both terminals and nonterminals, and
  - $\omega$  consists purely of terminals.
- Our search for handles will concentrate purely in  $\alpha$ .
- As necessary, we will start moving terminals from  $\omega$  over into  $\alpha$ .

# Shift/Reduce Parsing

- The bottom-up parsers we will consider are called **shift/reduce** parsers.
  - Contrast with the LL(1) **predict/match** parser.
- Idea: Split the input into two parts:
  - Left substring is our work area; all handles must be here.
  - Right substring is input we have not yet processed; consists purely of terminals.
- At each point, decide whether to:
  - Move a terminal across the split (**shift**)
  - Reduce a handle (**reduce**)

# A Sample Shift/Reduce Parse

**E** → **F**

**E** → **E** + **F**

**F** → **F** \* **T**

**F** → **T**

**T** → **int**

**T** → (**E**)

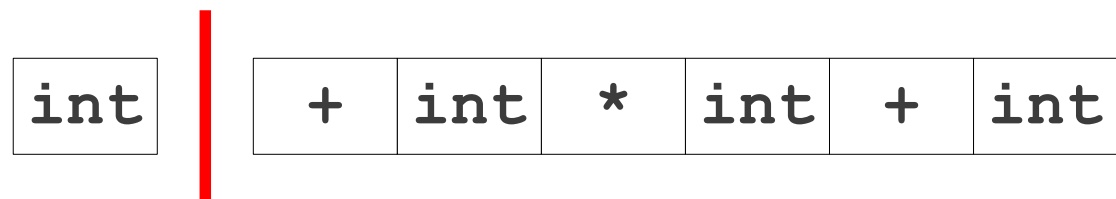


|     |   |     |   |     |   |     |
|-----|---|-----|---|-----|---|-----|
| int | + | int | * | int | + | int |
|-----|---|-----|---|-----|---|-----|



# A Sample Shift/Reduce Parse

**E** → **F**  
**E** → **E** + **F**  
**F** → **F** \* **T**  
**F** → **T**  
**T** → *int*  
**T** → (**E**)



# A Sample Shift/Reduce Parse

**E** → **F**

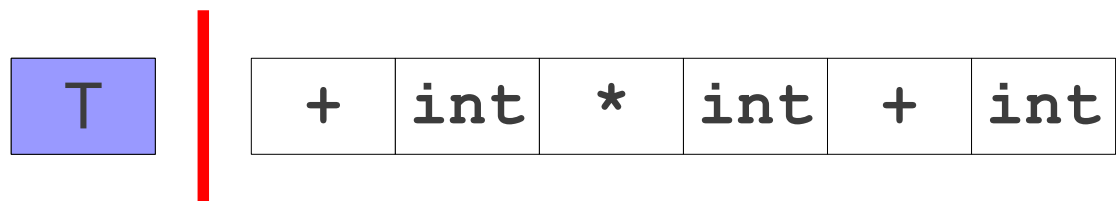
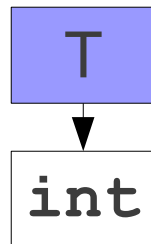
**E** → **E** + **F**

**F** → **F** \* **T**

**F** → **T**

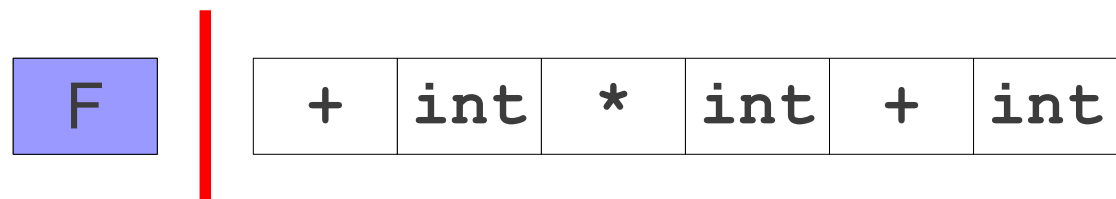
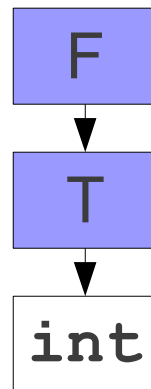
**T** → *int*

**T** → (**E**)



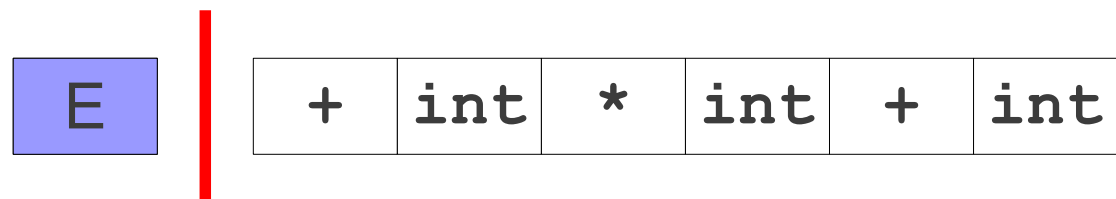
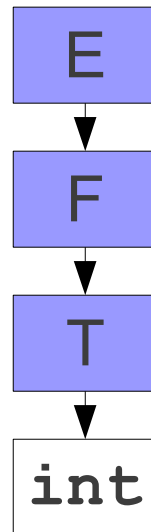
# A Sample Shift/Reduce Parse

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



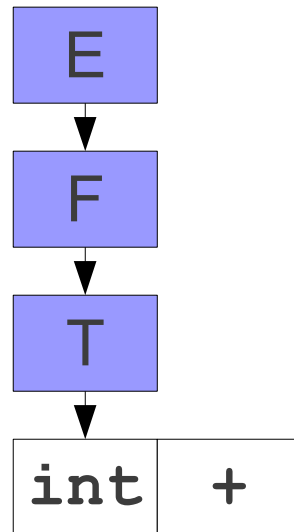
# A Sample Shift/Reduce Parse

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



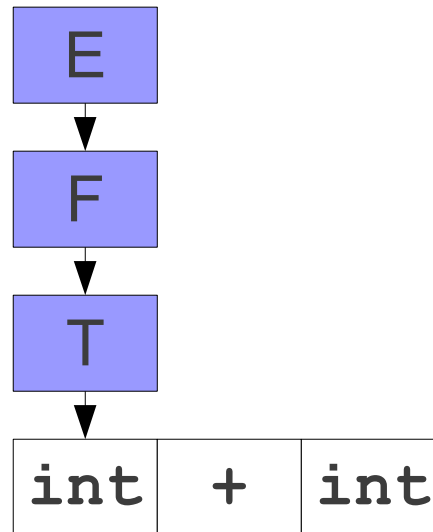
# A Sample Shift/Reduce Parse

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



# A Sample Shift/Reduce Parse

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

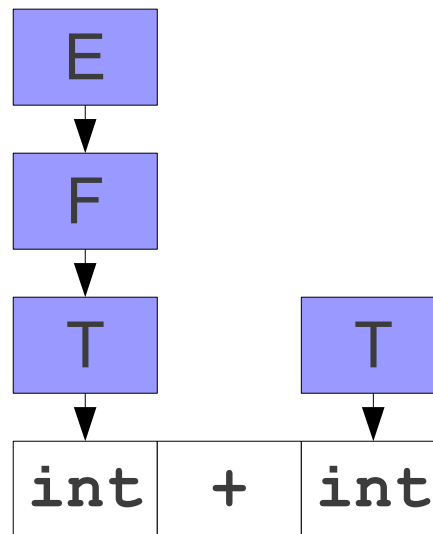


|          |   |     |
|----------|---|-----|
| <b>E</b> | + | int |
|----------|---|-----|

|   |     |   |     |
|---|-----|---|-----|
| * | int | + | int |
|---|-----|---|-----|

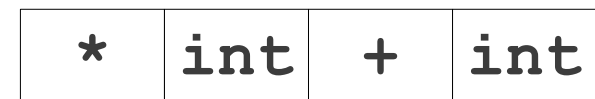
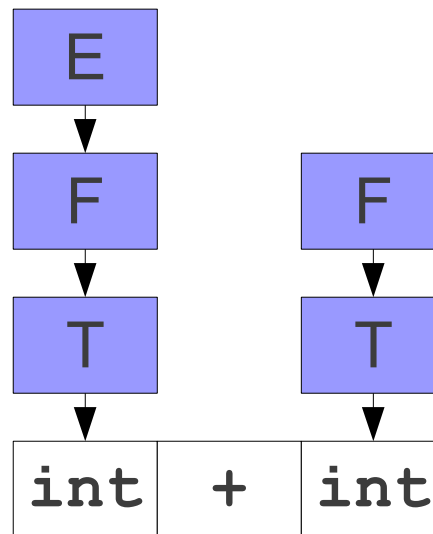
# A Sample Shift/Reduce Parse

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



# A Sample Shift/Reduce Parse

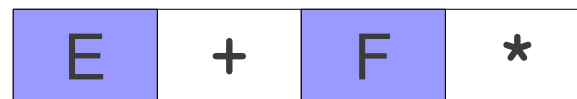
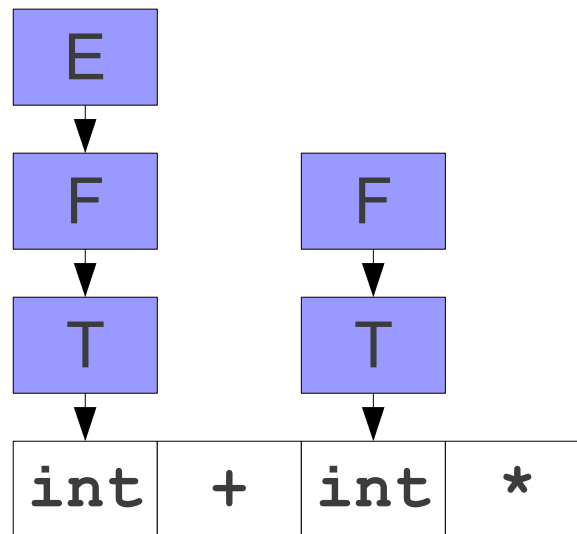
$E \rightarrow F$   
 $E \rightarrow E + F$   
 $F \rightarrow F * T$   
 $F \rightarrow T$   
 $T \rightarrow \text{int}$   
 $T \rightarrow (E)$





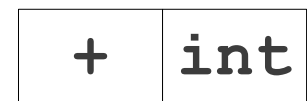
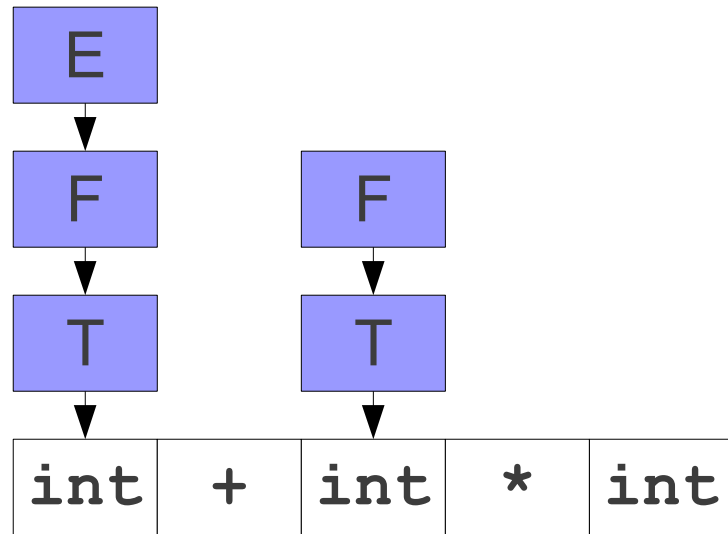
# A Sample Shift/Reduce Parse

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



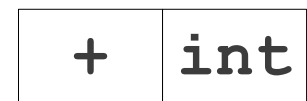
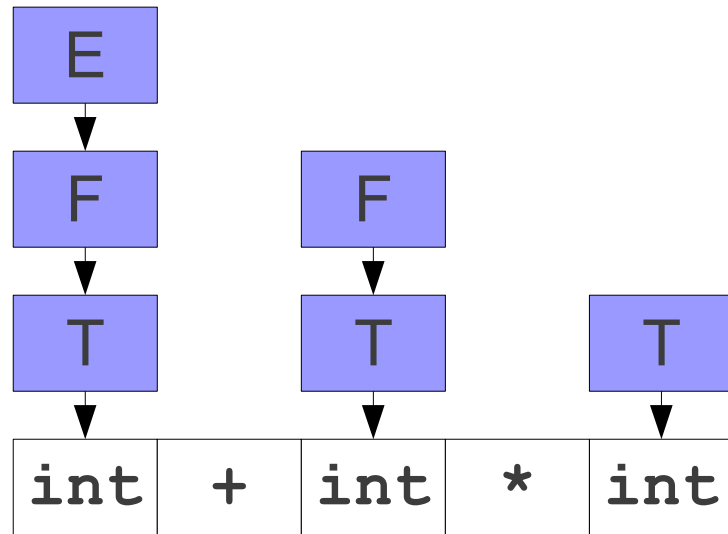
# A Sample Shift/Reduce Parse

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



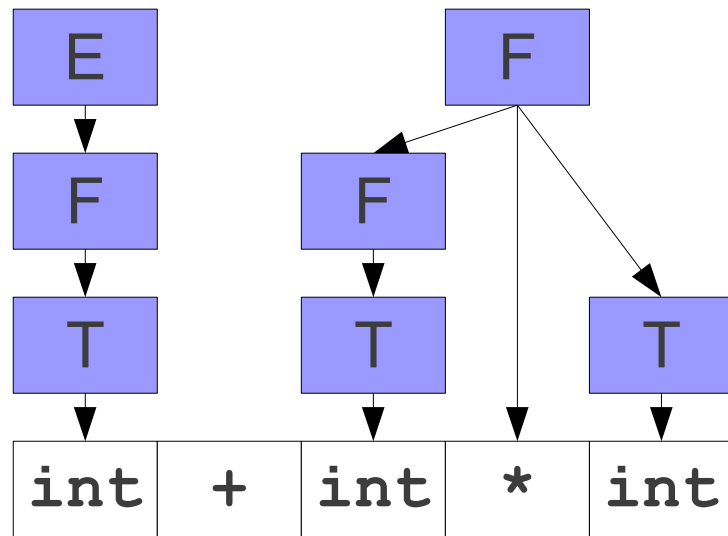
# A Sample Shift/Reduce Parse

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



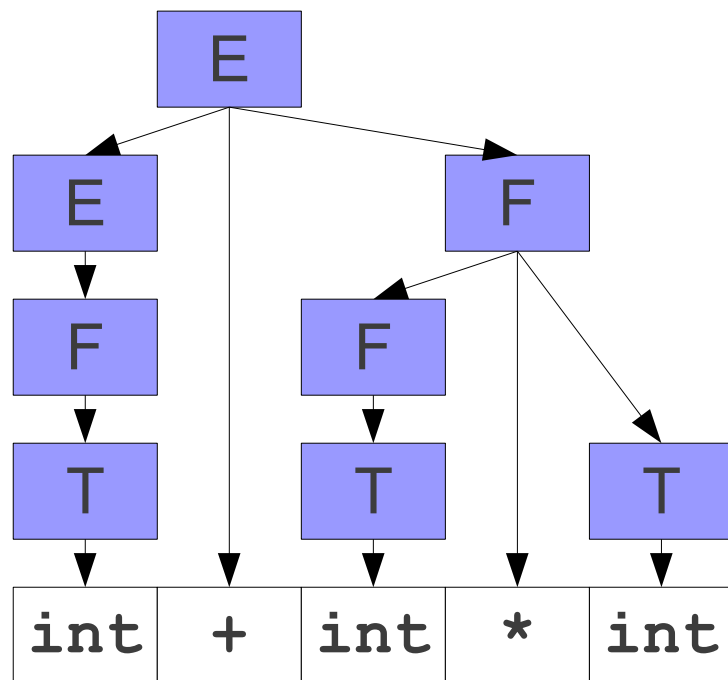
# A Sample Shift/Reduce Parse

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



# A Sample Shift/Reduce Parse

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

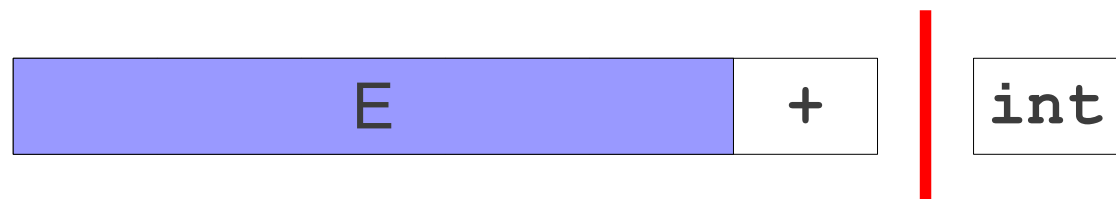
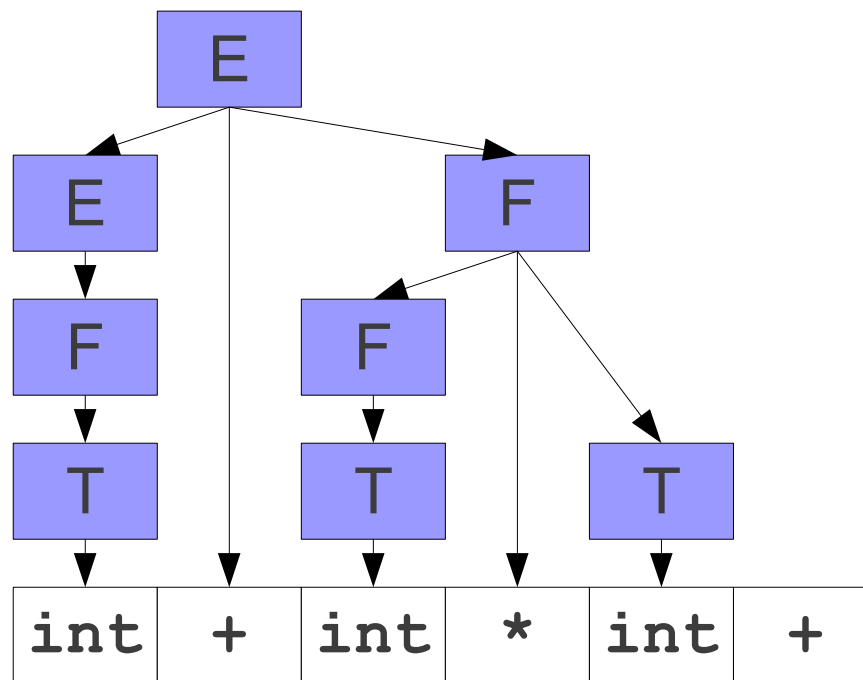


E

+ int

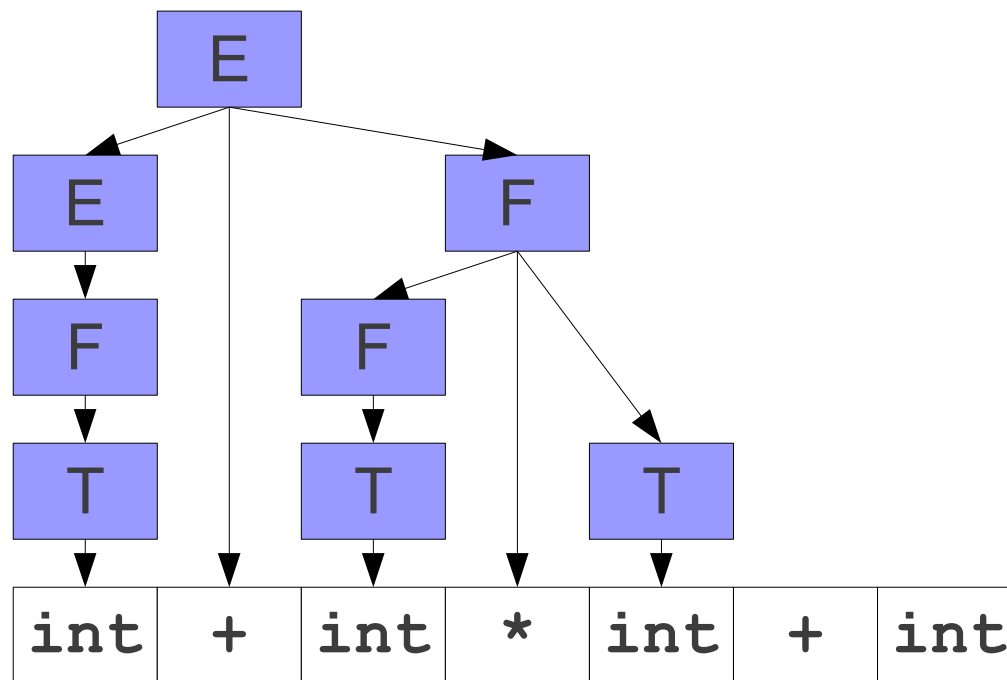
# A Sample Shift/Reduce Parse

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



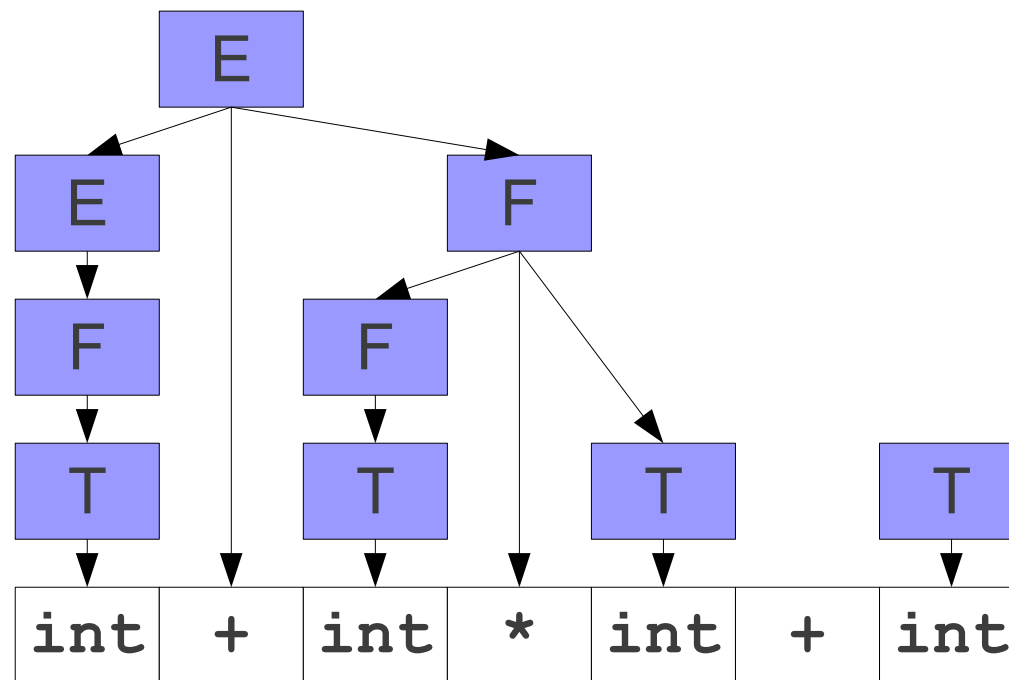
# A Sample Shift/Reduce Parse

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



# A Sample Shift/Reduce Parse

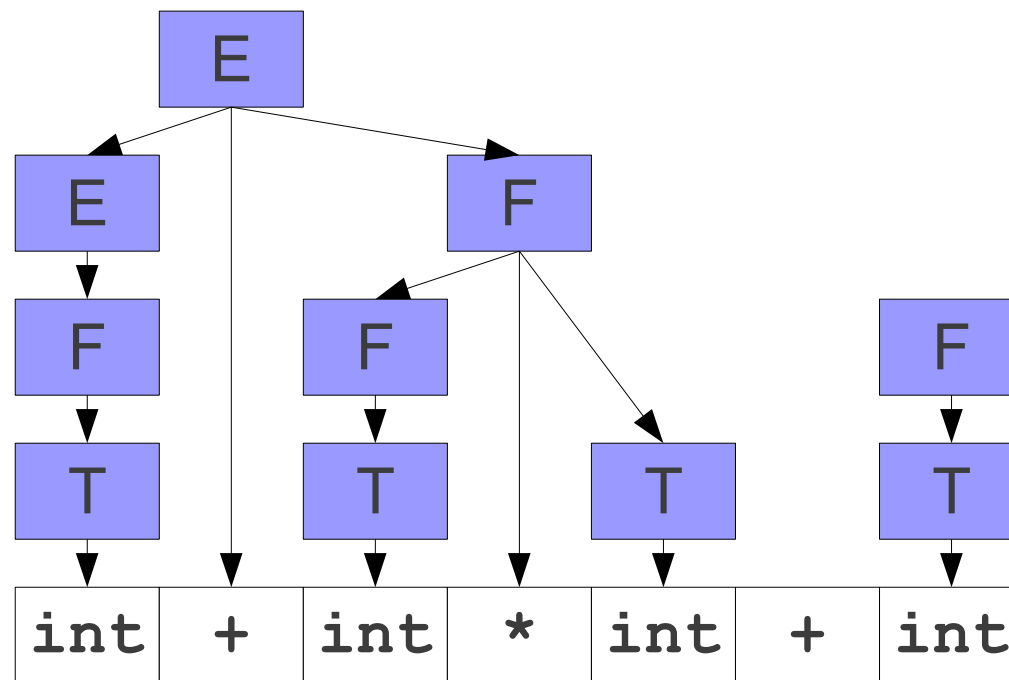
$E \rightarrow F$   
 $E \rightarrow E + F$   
 $F \rightarrow F * T$   
 $F \rightarrow T$   
 $T \rightarrow \text{int}$   
 $T \rightarrow (E)$





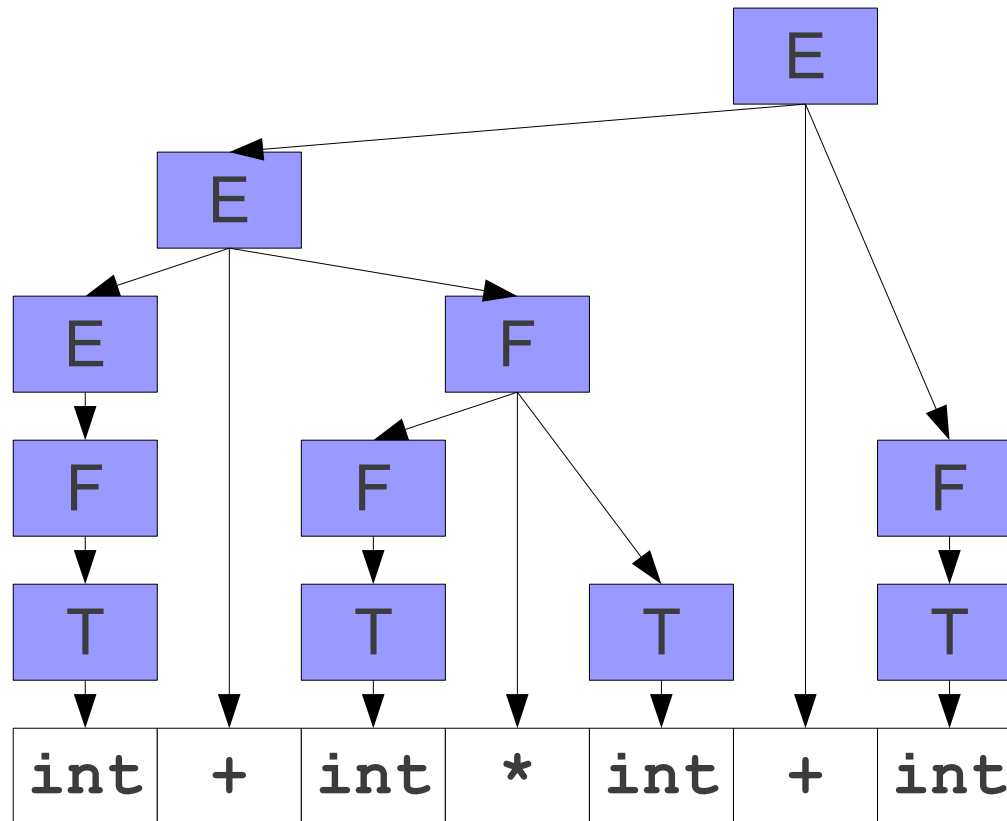
# A Sample Shift/Reduce Parse

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



# A Sample Shift/Reduce Parse

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



E

# An Important Observation

- All of the reductions we applied were to the far right end of the left area.
- This is not a coincidence; all reductions are always applied all the way to the end of the left area.
- Inductive proof sketch:
  - After no reduces, the first reduction can be done at the right end of the left area.
  - After at least one reduce, the very right of the left area is a nonterminal. This nonterminal must be part of the next reduction, since we're tracing a rightmost derivation backwards.

# An Important Corollary

- Since reductions are always at the right side of the left area, we never need to shift from the left to the right.
- No need to “uncover” something to do a reduction.
- Consequently, shift/reduce parsing means
  - **Shift**: Move a terminal from the right to the left area.
  - **Reduce**: Replace some number of symbols at the right side of the left area.

# Simplifying our Terminology

- All activity in a shift/reduce parser is at the far right end of the left area.
- Idea: Represent the left area as a stack.
- Shift: Push the next terminal onto the stack.
- Reduce: Pop some number of symbols from the stack, then push the appropriate nonterminal.

# Finding Handles

- Where do we look for handles?
  - **At the top of the stack.**
- How do we search for handles?
  - What algorithm do we use to try to discover a handle?
- How do we recognize handles?
  - Once we've found a possible handle, how do we confirm that it's correct?