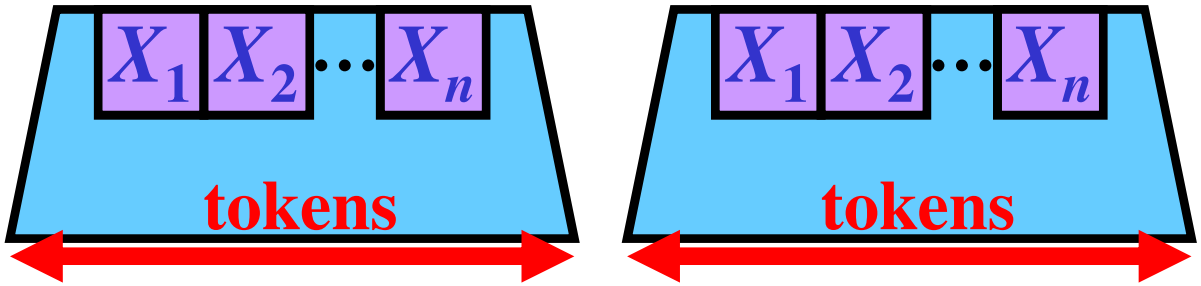
Part VIII.
Bottom-Up Parsing

Bottom-Up Parsing: Problems

1) Two or more rules have the same *handle*

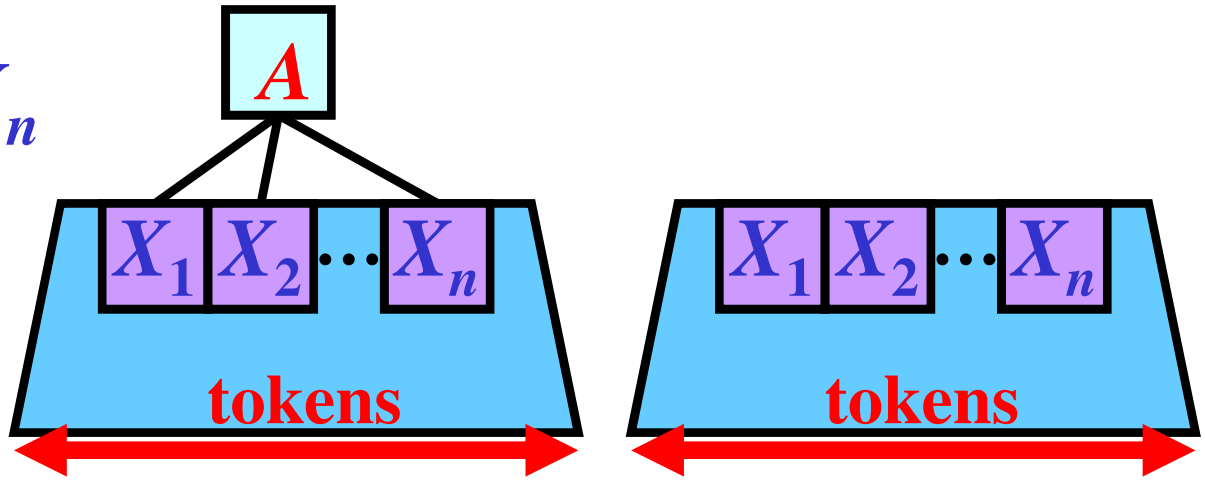


Note: A *handle* is the right-hand side of a rule.

Bottom-Up Parsing: Problems

1) Two or more rules have the same *handle*

$r_1: A \rightarrow X_1 X_2 \ldots X_n$



Note: A *handle* is the right-hand side of a rule.

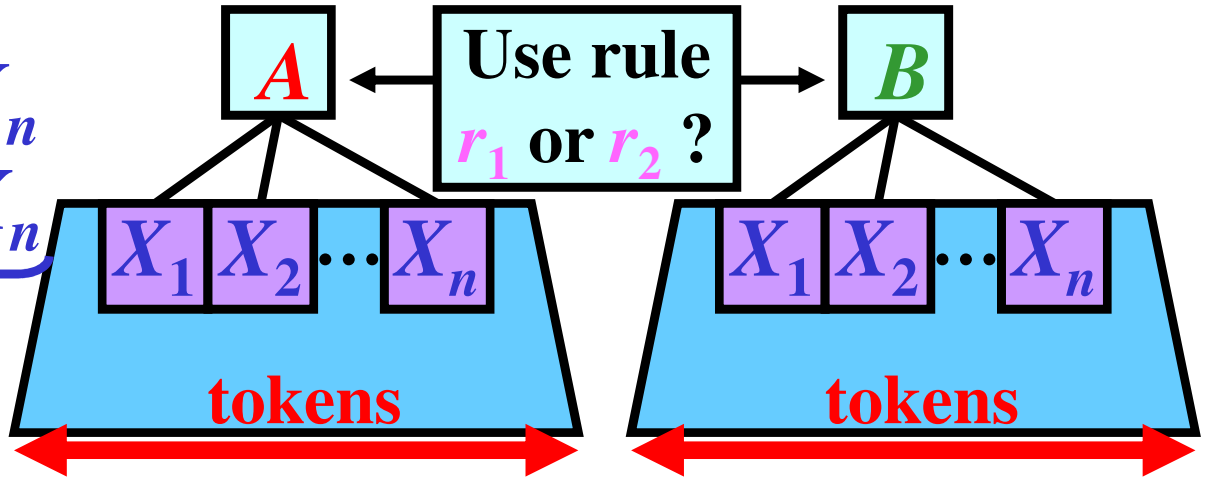
Bottom-Up Parsing: Problems

1) Two or more rules have the same *handle*

r_1: $A \rightarrow X_1 X_2 \ldots X_n$

r_2: $B \rightarrow X_1 X_2 \ldots X_n$

handle



Note: A *handle* is the right-hand side of a rule.

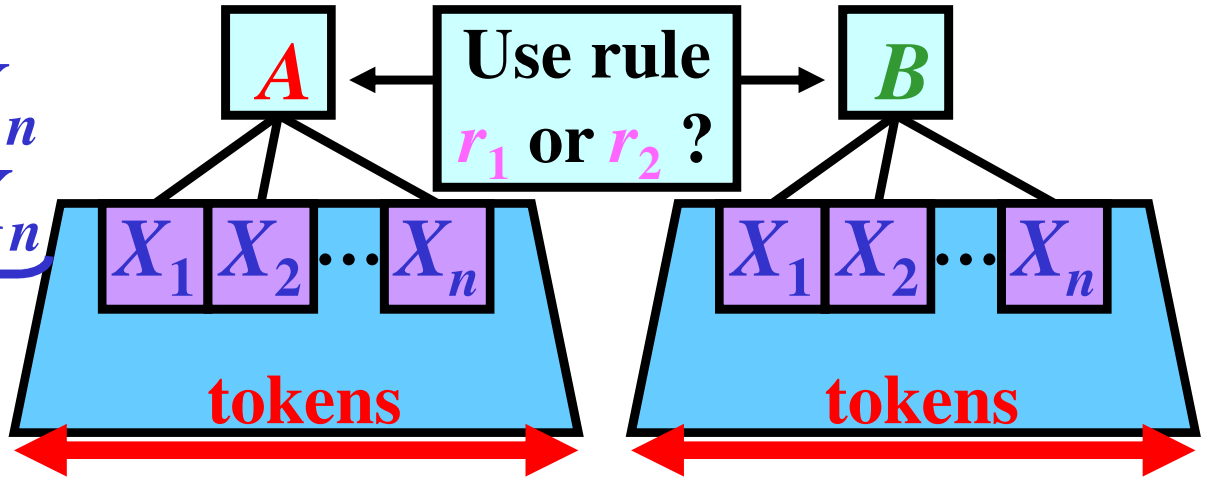
Bottom-Up Parsing: Problems

1) Two or more rules have the same *handle*

$r_1: A \rightarrow X_1 X_2 \ldots X_n$

$r_2: B \rightarrow X_1 X_2 \ldots X_n$

handle



Note: A *handle* is the right-hand side of a rule.

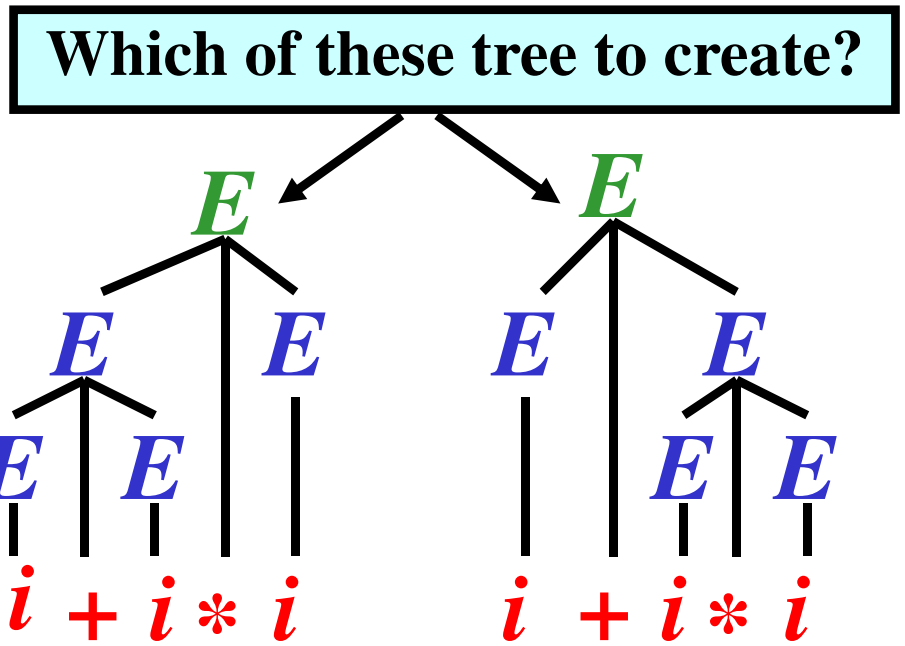
2) Ambiguous grammars

$G_{expr2} = (N, T, P, E)$, where

$N = \{E\}$, $T = \{i, +, *, (,)\}$,

$P = \{$ 1: $E \rightarrow E+E$, 2: $E \rightarrow E*E$,

3: $E \rightarrow (E)$, 4: $E \rightarrow i$ $\}$

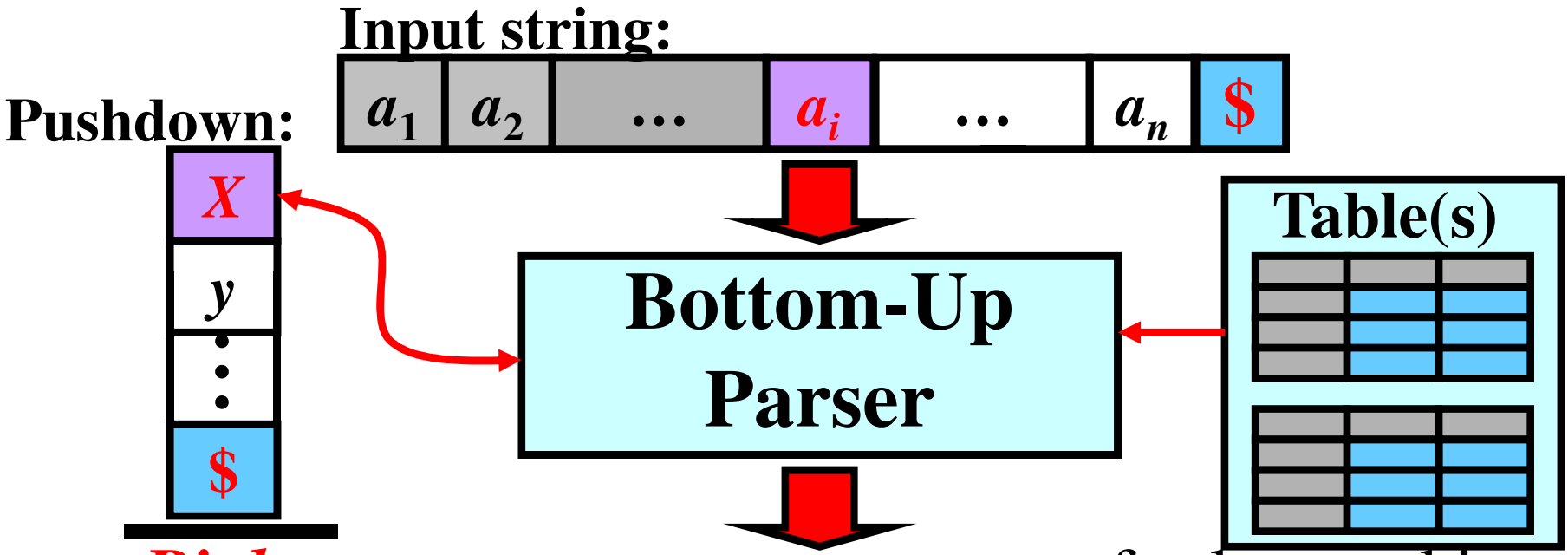


Bottom-Up Parsers

1) Operator-precedence parser
- the least powerful, but simple & easy-to-make

2) LR parser
- the most powerful

- **Model of Bottom-Up parser:**



Right parse = **reverse** sequence of rules used in the **rightmost derivation** of the tokenized source program

Operator-Precedence Parser

- No two distinct nonterminals have the same handle
- No ε-rules.

- Let $G = (N, T, P, S)$ be CFG, where $T = \{a_1, a_2, \ldots, a_n\}$

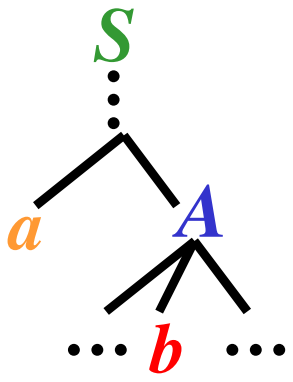
Precedence-table:

	a_1	...	a_j	...	a_n	$
a_1						
...						
a_i						
...						
a_n						
$						

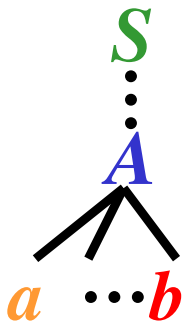
Table[a_i , a_j] ∈ {<, =, >, *blank*}

Illustration of meaning of <, =, >:

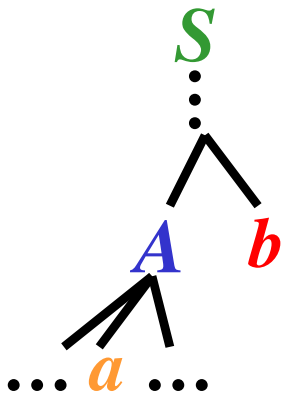
$a < b$



$a = b$



$a > b$



Operator-Precedence Parser: Algorithm

- **Input:** Precedence-table for $G = (N, T, P, S)$; $x \in T^*$
- **Output:** Right parse of x if $x \in L(G)$; otherwise, **error**

- **Method:**
- Push $ onto the pushdown;
- **repeat**
  - let *a* = the topmost **terminal** on the pushdown and
    b = the current token
  - **case** Table[*a*, *b*] **of:**
    - **=** : push(*b*) & read next *b* from input string
    - **<** : replace *a* with *a*< on the pushdown &
      push(*b*) & read next *b* from input string
    - **>** : **if** <*y* is the pushdown top string **and** *r*: $A \rightarrow y \in P$
      then replace <*y* with *A* & write *r* to output
      else error
    - **blank** : **if** *a* = $ **and** *b* = $ **then success**
      else error

until success or error

Operator-Precedence Parser: Example

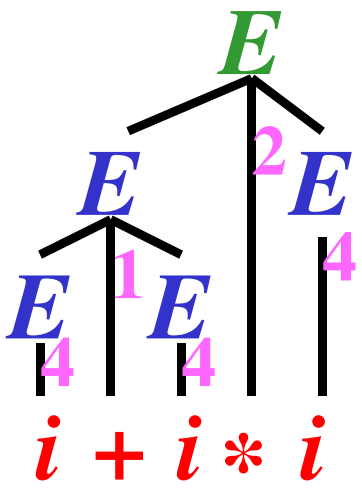
$G_{expr2} = (N, T, P, E)$, where $N = \{E\}$, $T = \{i, +, *, (,)\}$,
$P = \{$ 1: $E \rightarrow E+E$, 2: $E \rightarrow E*E$, 3: $E \rightarrow (E)$, 4: $E \rightarrow i\}$

Precedence-table for G_{expr2}:

Pushdown topmost terminal \ Input token	+	*	(	)	i	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Note: Operator associativity and precedence rules underlie the precedence table:

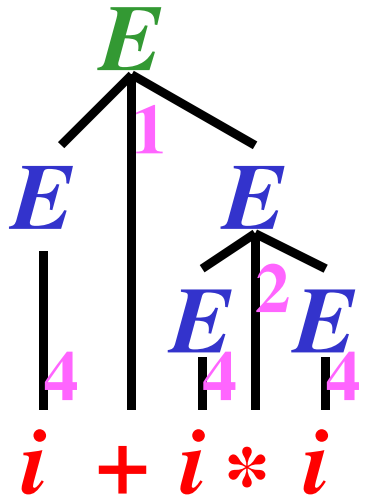
☹ **Wrong tree:**



Right parse:

44142 ←

☺ **Right tree:**



Right parse:

44421 ←

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Input string: *i* + *i* * *i* $

Pushdown	Op	Input	Rule

Rules:

1: $E \rightarrow E+E$

2: $E \rightarrow E*E$

3: $E \rightarrow (E)$

4: $E \rightarrow i$

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$

2: $E \rightarrow E*E$

3: $E \rightarrow (E)$

4: $E \rightarrow i$

Input string: *i* + *i* * *i* $

Pushdown	Op	Input	Rule
$	<	*i*+*i***i*$	

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$
2: $E \rightarrow E*E$
3: $E \rightarrow (E)$
4: $E \rightarrow i$

Input string: *i* + *i* * *i* $

Pushdown	Op	Input	Rule
$	<	*i*+*i***i*$	
$<*i*	>	+*i***i*$	4: $E \rightarrow i$

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$

2: $E \rightarrow E*E$

3: $E \rightarrow (E)$

4: $E \rightarrow i$

Input string: *i* + *i* * *i* $

Pushdown	Op	Input	Rule
$	<	*i*+*i**i$	
$<*i*	>	+*i**i$	4: $E \rightarrow i$
$*E*	<	+*i**i$	

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$
2: $E \rightarrow E*E$
3: $E \rightarrow (E)$
4: $E \rightarrow i$

Input string: *i* + *i* * *i* $

Pushdown	Op	Input	Rule
$	<	*i*+*i**i$	
$<*i*	>	+*i**i$	4: $E \rightarrow i$
$*E*	<	+*i**i$	
$<*E*+	<	*i**i$	

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$

2: $E \rightarrow E*E$

3: $E \rightarrow (E)$

4: $E \rightarrow i$

Input string: *i* + *i* * *i* $

Pushdown	Op	Input	Rule
$	<	*i*+*i**i$	
$<*i*	>	+*i**i$	4: $E \rightarrow i$
$*E*	<	+*i**i$	
$<*E*+	<	*i**i$	
$<*E*+<*i*	>	**i*$	4: $E \rightarrow i$

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$
2: $E \rightarrow E*E$
3: $E \rightarrow (E)$
4: $E \rightarrow i$

Input string: *i* + *i* * *i* $

Pushdown	Op	Input	Rule
$	<	*i*+*i**i$	
$<*i*	>	+*i**i$	4: $E \rightarrow i$
$*E*	<	+*i**i$	
$<*E*+	<	*i**i$	
$<*E*+<*i*	>	**i*$	4: $E \rightarrow i$
$<*E*+*E*	<	**i*$	

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$
2: $E \rightarrow E*E$
3: $E \rightarrow (E)$
4: $E \rightarrow i$

Input string: *i* + *i* * *i* $

Pushdown	Op	Input	Rule
$	<	*i*+*i**i$	
$<*i*	>	+*i**i$	4: $E \rightarrow i$
$*E*	<	+*i**i$	
$<*E*+	<	*i**i$	
$<*E*+<*i*	>	**i*$	4: $E \rightarrow i$
$<*E*+*E*	<	**i*$	
$<*E*+<*E**	<	*i*$	

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$

2: $E \rightarrow E*E$

3: $E \rightarrow (E)$

4: $E \rightarrow i$

Input string: $i + i * i \$$

Pushdown	Op	Input	Rule
$\$$	<	$i+i*i\$$	
$\$<i$	>	$+i*i\$$	4: $E \rightarrow i$
$\$E$	<	$+i*i\$$	
$\$<E+$	<	$i*i\$$	
$\$<E+<i$	>	$*i\$$	4: $E \rightarrow i$
$\$<E+E$	<	$*i\$$	
$\$<E+<E*$	<	$i\$$	
$\$<E+<E*<i$	>	$\$$	4: $E \rightarrow i$

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$
2: $E \rightarrow E*E$
3: $E \rightarrow (E)$
4: $E \rightarrow i$

Input string: $i + i * i \$$

Pushdown	Op	Input	Rule
$	<	$i+i*i\$$	
$<i	>	$+i*i\$$	4: $E \rightarrow i$
$E	<	$+i*i\$$	
$<E+	<	$i*i\$$	
$<E+<i	>	$*i\$$	4: $E \rightarrow i$
$<E+E	<	$*i\$$	
$<E+<E*	<	$i\$$	
$<E+<E*<i	>	$\$$	4: $E \rightarrow i$
$<E+<E*E	>	$\$$	2: $E \rightarrow E*E$

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$
2: $E \rightarrow E*E$
3: $E \rightarrow (E)$
4: $E \rightarrow i$

Input string: $i + i * i \$$

Pushdown	Op	Input	Rule
$	<	$i+i*i\$$	
$<i	>	$+i*i\$$	4: $E \rightarrow i$
$E	<	$+i*i\$$	
$<E+	<	$i*i\$$	
$<E+<i	>	$*i\$$	4: $E \rightarrow i$
$<E+E	<	$*i\$$	
$<E+<E*	<	$i\$$	
$<E+<E*<i	>	$\$$	4: $E \rightarrow i$
$<E+<E*E	>	$\$$	2: $E \rightarrow E*E$
$<E+E	>	$\$$	1: $E \rightarrow E+E$

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Rules:

1: $E \rightarrow E+E$
2: $E \rightarrow E*E$
3: $E \rightarrow (E)$
4: $E \rightarrow i$

Input string: $i + i * i \$$

Pushdown	Op	Input	Rule
$	<	$i+i*i\$$	
$<i	>	$+i*i\$$	4: $E \rightarrow i$
$E	<	$+i*i\$$	
$<E+	<	$i*i\$$	
$<E+<i	>	$*i\$$	4: $E \rightarrow i$
$<E+E	<	$*i\$$	
$<E+<E*	<	$i\$$	
$<E+<E*<i	>	$\$$	4: $E \rightarrow i$
$<E+<E*E	>	$\$$	2: $E \rightarrow E*E$
$<E+E	>	$\$$	1: $E \rightarrow E+E$
$E		$\$$	

Operator-Precedence Parsing: Example

	+	*	(	)	*i*	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	

Input string: *i* + *i* * *i* $

Pushdown	Op	Input	Rule
$	<	*i*+*i**i$	
$<*i*	>	+*i**i$	4: *E* → *i*
$*E*	<	+*i**i$	
$<*E*+	<	*i**i$	
$<*E*+<*i*	>	**i*$	4: *E* → *i*
$<*E*+*E*	<	**i*$	
$<*E*+<*E**	<	*i*$	
$<*E*+<*E**<*i*	>	$	4: *E* → *i*
$<*E*+<*E***E*	>	$	2: *E* → *E***E*
$<*E*+*E*	>	$	1: *E* → *E*+*E*
$*E*		$	

Success

Right parse: 44421

Rules:

1: *E* → *E*+*E*

2: *E* → *E***E*

3: *E* → (*E*)

4: *E* → *i*

Construction of Precedence Table 1/5

• Let $G_{expr} = (N, T, P, E)$, where $N = \{E\}$,

$T = \{(,), \mathbf{id}_1, \mathbf{id}_2, \ldots, \mathbf{id}_m, \mathbf{op}_1, \mathbf{op}_2, \ldots \mathbf{op}_n\}$,

$P = \{ E \rightarrow (E), E \rightarrow \mathbf{id}_1, E \rightarrow \mathbf{id}_2, \ldots, E \rightarrow \mathbf{id}_m,$

$E \rightarrow E\ \mathbf{op}_1\ E, E \rightarrow E\ \mathbf{op}_2\ E, \ldots, E \rightarrow E\ \mathbf{op}_n\ E\ \}$

Note: $\mathbf{id}_1, \mathbf{id}_2, \ldots, \mathbf{id}_m$ are identifiers,

$\mathbf{op}_1, \mathbf{op}_2, \ldots \mathbf{op}_n$ are different operators

1) Precedence of operators:

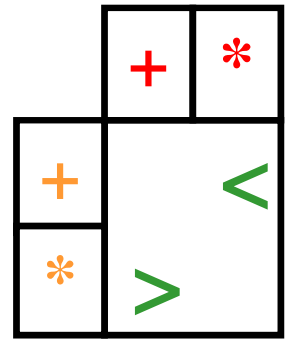
• If $\mathbf{op}_i$ has **higher precedence** than $\mathbf{op}_j$ then

$\mathbf{op}_i > \mathbf{op}_j$ and $\mathbf{op}_j < \mathbf{op}_i$

Example: Precedence-table part derived from the precedence of operators in G_{expr2}

$* > +$

$+ < *$



Construction of Precedence Table 2/5

2) Associativity:

Note:

- $\mathbf{op}_i$ is left-associative $\Leftrightarrow a\ \mathbf{op}_i\ b\ \mathbf{op}_i\ c = (a\ \mathbf{op}_i\ b)\ \mathbf{op}_i\ c$
- $\mathbf{op}_i$ is right-associative $\Leftrightarrow a\ \mathbf{op}_i\ b\ \mathbf{op}_i\ c = a\ \mathbf{op}_i\ (b\ \mathbf{op}_i\ c)$

- Let $\mathbf{op}_i$ and $\mathbf{op}_j$ have **equal precedence**
  - If $\mathbf{op}_i$ and $\mathbf{op}_j$ are **left associative** then

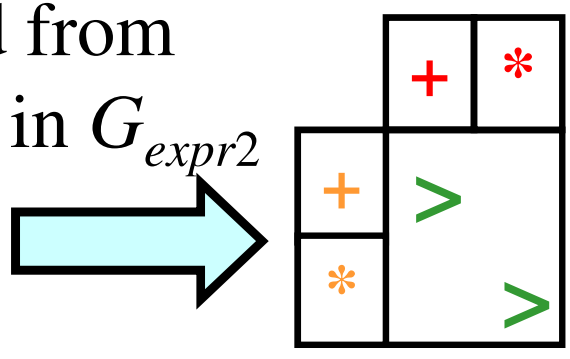
$$\mathbf{op}_i > \mathbf{op}_j \text{ and } \mathbf{op}_j > \mathbf{op}_i$$

  - If $\mathbf{op}_i$ and $\mathbf{op}_j$ are **right associative** then

$$\mathbf{op}_i < \mathbf{op}_j \text{ and } \mathbf{op}_j < \mathbf{op}_i$$

Example: Precedence-table part derived from the associativity of operators in G_{expr2}

+ is left-associative
*** is left-associative**



Construction of Precedence Table 3/5

3) Identifiers:

- If $a \in T$ may **precede** $\mathbf{id}_i$, then $a < \mathbf{id}_i$
- If $a \in T$ may **follow** $\mathbf{id}_i$, then $\mathbf{id}_i > a$

Example: Precedence-table part for identifiers

$$\$ i * (i + i) * i$$

$\$$, (, +, * may precede i

$$i * (i + i) * i \$$$

*, +,), $ may follow i

	+	*	(	)	i	$
+					<	
*					<	
(					<	
)						
i	>	>		>		>
$					<	

Construction of Precedence Table 4/5

4) Parentheses:

- A pair of parentheses:
- Let $a \in T - \{), \$\}$. Then,
- Let $a \in T - \{(, \$\}$. Then,
- Let $a \in T$ and a may **precede** (. Then,
- Let $a \in T$ and a may **follow**). Then,

(=)
(< a
a >)

a < (
) > a

Example: Precedence-table part for parentheses.

$(i + ((i * (i + (i + i)))))$

$, (, * , + may precede (

$((((i + i) + i) * i)) + i)$

+, *,), $ may follow)

	+	*	(	)	i	$
+			<	>		
*			<	>		
(	<	<	<	=	<	
)	>	>		>		>
i				>		
$			<			

Construction of Precedence Table 5/5

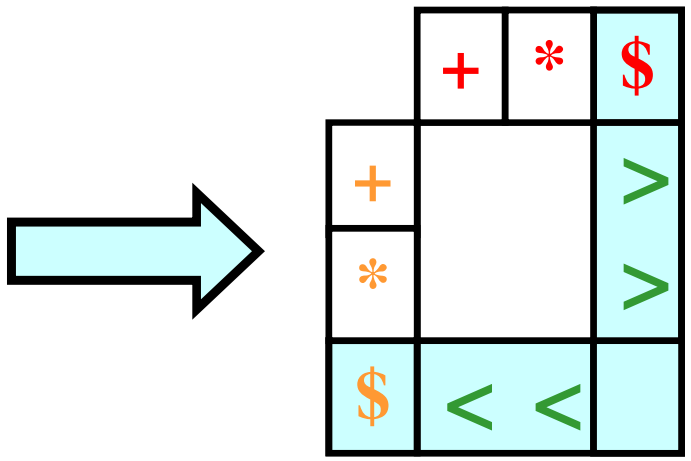
5) End Marker $

- Let $\mathbf{op}_i$ be any operator. Then:

$$\$ < \mathbf{op}_i \text{ and } \mathbf{op}_i > \$$$

Example: Precedence-table part for end-markers.

$ < +
$ < *
+ > $
* > $



Construction of Precedence Table 5/5

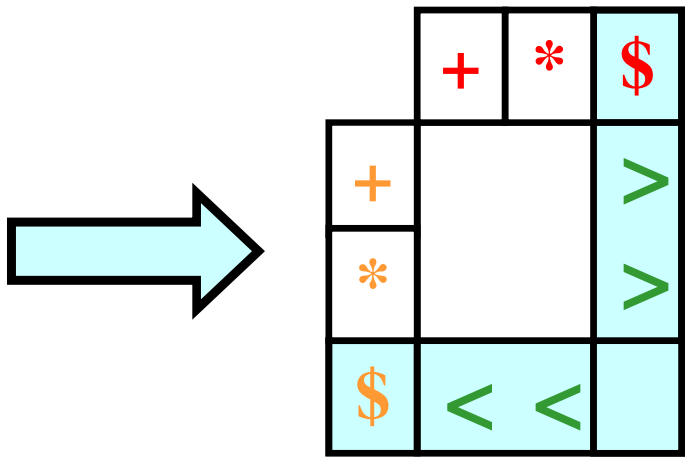
5) End Marker $

- Let op_i be any operator. Then:

$$\$ < op_i \text{ and } op_i > \$$$

Example: Precedence-table part for end-markers.

$ < +
$ < *
+ > $
* > $



Summary:

	+	*	(	)	i	$
+	>	<	<	>	<	>
*	>	>	<	>	<	>
(	<	<	<	=	<	
)	>	>		>		>
i	>	>		>		>
$	<	<	<		<	