OPERATOR PRECEDENCE PARSING

CSE 340 FALL 2021

Rida Bazzi

Notes based on the "Dragon Book"

Parsing Operator Grammars

The grammar we have see for expressions does not include the operator minus ('-').

This is not an oversight!

We can write the following grammar

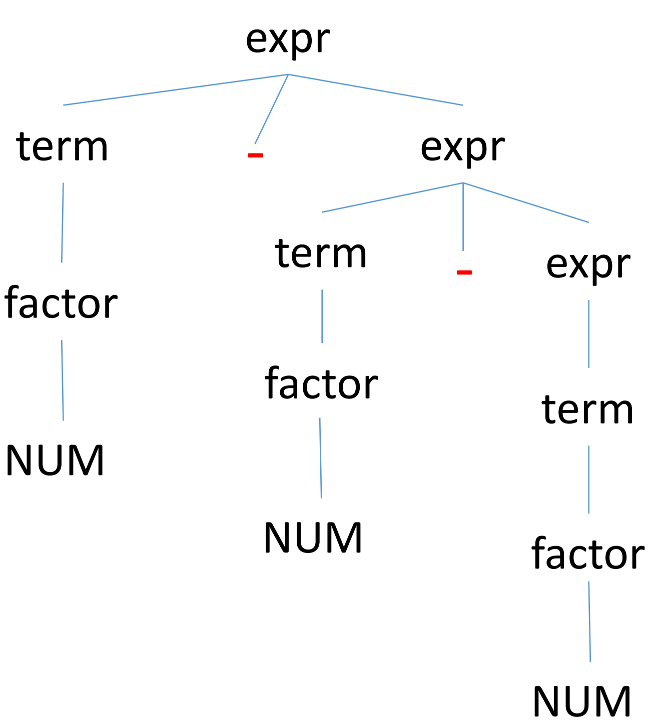
    Expr -> term - Expr
    Expr -> term + Expr
    Expr -> term

but that would not work!

How do we parse the following?

1 - 2 - 3

According to the grammar above, we get



According to this tree,

1 - 2 - 3 = 2 !!!

Parsing expressions with minus

The issue is that minus is left associative and the grammar treats minus as right-associative

Left associative grouping (correct)

1 – 2 – 3

(1 – 2) – 3

((1-2) – 3)

Right associative grouping (wrong)

1 – 2 – 3

1 – (2 – 3)

(1 – (2 - 3))

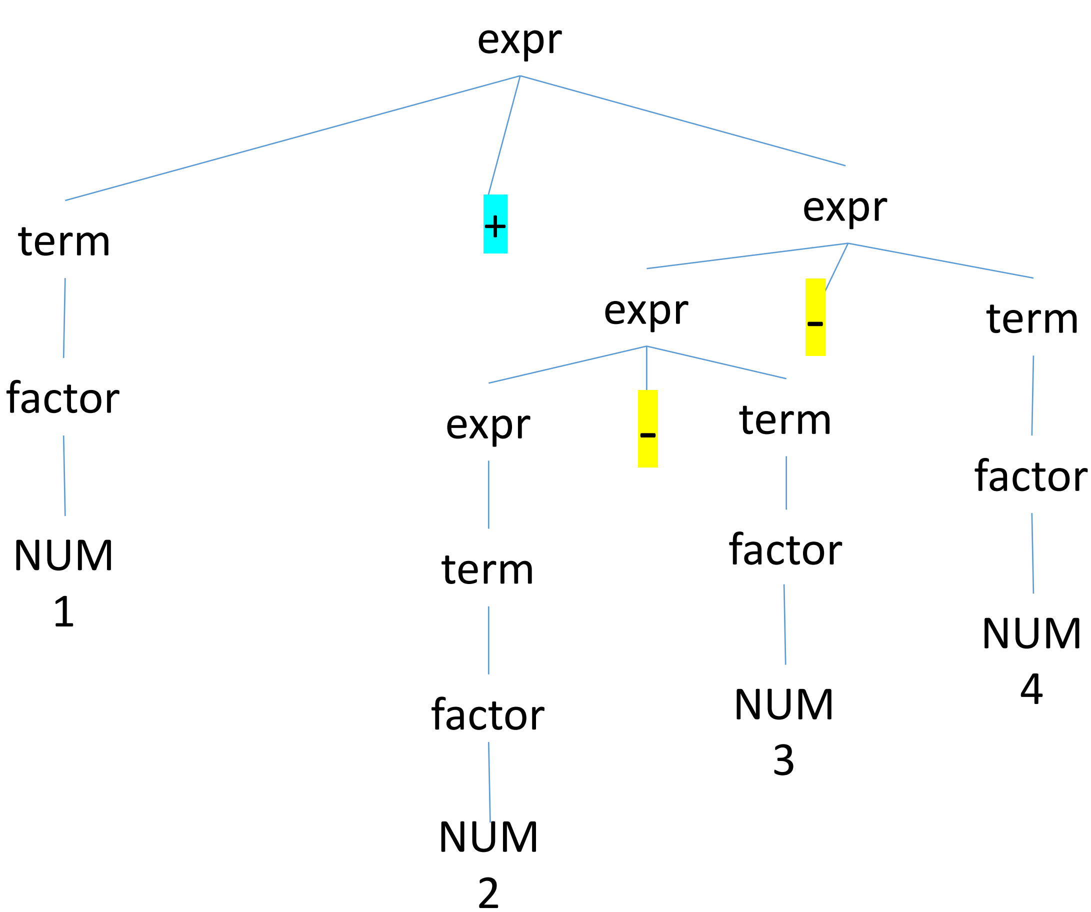
Parsing expressions with minus

We can attempt to fix the problem by using the following grammar

expr -> expr - term
expr -> term + expr
expr -> term

This grammar would give the following parsing for

1 + 2 - 3 - 4



Parsing expressions with minus

We can attempt to fix the problem by using the following grammar

```
expr -> expr - term
expr -> term + expr
expr -> term
```

We cannot parse this grammar with a recursive descent parser!

```
parse_expr()
{
    // expr -> expr - term

    ....

    parse_expr()   // infinite loop !!
```

We need another way to parse such expressions!

A NEAT TRICK FROM FORTRAN COMPILER!

a + b * c − d

add (((at the beginning
replace every + with))) + (((
replace every − with))) - (((
replace every * with)) * ((
replace every ^ with) ^ (
add))) at the end

We get

(((a))) + (((b)) * ((c))) − (((d)))

(((a))) + (((b)) * ((c))) − (((d)))

Always works !

We can then parse with a simple parser that only has to worry about matching parentheses

Operator Grammar

A grammar is called an operator grammar if

1. there is no righthand side of a rule which has two adjacent non-terminal
2. there is no rule of the form A -> ε

Example 1 E -> E A E | (E) | -E | ID

A -> + | - | * | / | ^

is not an operator grammar because of E A E has three adjacent non-terminals

Example 2 E -> E + E | E - E | E * E | E / E

| E ^ E | (E) | - E | ID

is an operator grammar

OPERATOR PRECEDENCE RELATIONSHIPS

To parse operator grammar, we first define parsing precedence relationships between the terminals of the grammar

We also introduce a new symbol $

parsing precedence relationships

- $\lessdot$ yields precedence to
- $\gtrdot$ takes precedence over
- $\doteq$ has the same precedence as

These are not the same as the operator precedence levels.

These are used in guiding the parsing

You can think of $\lessdot$ $\gtrdot$ as matching parentheses that group what appears between them (this should become clearer with the examples)

There is a theory to determine these relationship for an unambiguous operator grammar. We will only look at heuristics for common expressions.

The parsing algorithm assumes that we already have a table that defines these relationships

EXAMPLE

	+	**-**	*****	**/**	**^**	**id**	**(**	**)**	**$\$$**
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$\$$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

Parsing Algorithm

Input w $
Output parse tree with E in all internal nodes
Initially stack contains $, scanning starts at the start of w

```
repeat
        if $ is on top of the stack and lexer.peek() = $   // EOF
                return;
        else
        {
                t = lexer.peek(); b = t.type;           // next token from w
                a = stack.terminalpeek().type;          // terminal at the top of stack
                                                        // or just below if top is non-terminal

                if (table[a][b] == `<·') | ( table[a][b] = `≐')      // shift
                        t = lexer.getToken();
                        stack.push(t)
                else if (table[a][b] == `·>')                        // reduce
                {
                        RHS = an empty stack
                        repeat
                                s = stack.pop()          // pop terminals and
                                                         // non-terminals
                                if s is a terminal
                                    last_popped_term = s
                                RHS.push(s)
                        until ( ( is_a_terminal(stack.peek() ) and
                              ( table[stack.terminalpeek()][last_popped_term] ==  `<·' ))

                        if E -> RHS  rule exists// RHS calculated above
                        {
                                reduce E -> RHS
                                push E                   // we can think of E as the
                                                         // root of subtree for E -> RHS
                        }
                        else
                                syntax_error()l
                }
                else
                        syntax_error();
        }
```

Note:

- stack.peek() peeks at the symbol at the top of the stack, which could be a terminal or a non-terminal.
- stack.terminalpeek() peeks at the terminal closest to the top of the stack.

EXAMPLE

	+	-	*	/	^	id	(	)	$
+	$\cdot>$	$\cdot>$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$\cdot>$	$\cdot>$
-	$\cdot>$	$\cdot>$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$\cdot>$	$\cdot>$
*****	$\cdot>$	$\cdot>$	$\cdot>$	$\cdot>$	$<\cdot$	$<\cdot$	$<\cdot$	$\cdot>$	$\cdot>$
/	$\cdot>$	$\cdot>$	$\cdot>$	$\cdot>$	$<\cdot$	$<\cdot$	$<\cdot$	$\cdot>$	$\cdot>$
^	$\cdot>$	$\cdot>$	$\cdot>$	$\cdot>$	$<\cdot$	$<\cdot$	$<\cdot$	$\cdot>$	$\cdot>$
id	$\cdot>$	$\cdot>$	$\cdot>$	$\cdot>$	$\cdot>$			$\cdot>$	$\cdot>$
(	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$\doteq$	
)	$\cdot>$	$\cdot>$	$\cdot>$	$\cdot>$	$\cdot>$			$\cdot>$	$\cdot>$
$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$	$<\cdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	

$\lessdot$

input

stack

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) - a	shift

$\lessdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
\$	a + b * (c + d) − a	shift
\$a	+ b * (c + d) − a	

	+	-	*	/	^	id	(	)	\$
+	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
-	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
*	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
/	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
^	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\cdot\!>$	$\cdot\!>$
id	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$			$\cdot\!>$	$\cdot\!>$
(	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$\doteq$	
)	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$	$\cdot\!>$			$\cdot\!>$	$\cdot\!>$
\$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$	$<\!\cdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	

·>

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID

·>

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID

$ <. a >.   + b * (c + d) – a

$E   + b * (c + d) – a

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E	+ b * (c + d) – a	

id

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E	+ b * (c + d) − a	

id

$\lessdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E	+ b * (c + d) – a	shift

id

<·

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E	+ b * (c + d) – a	shift
$E+	b * (c + d) – a	

id

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E	+ b * (c + d) – a	shift
$E+	b * (c + d) – a	

id

⋖

	+	-	*	/	^	id	(	)	$
+	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
-	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
*****	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
/	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
^	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
id	⋗	⋗	⋗	⋗	⋗			⋗	⋗
(	⋖	⋖	⋖	⋖	⋖	⋖	⋖	≐	
)	⋗	⋗	⋗	⋗	⋗			⋗	⋗
$	⋖	⋖	⋖	⋖	⋖	⋖	⋖		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E	+ b * (c + d) − a	shift
$E+	b * (c + d) − a	shift

id

$\lessdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E+	b * (c + d) − a	shift
$E+b	* (c + d) − a	

id

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	

id

$\gtrdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E+	b * (c + d) − a	shift
$E+b	* (c + d) − a	reduce E -> ID

id

·>

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E+	b * (c + d) − a	shift
$E+b	* (c + d) − a	reduce E -> ID
$E+E	* (c + d) − a	

id id

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E+	b * (c + d) − a	shift
$E+b	* (c + d) − a	reduce E -> ID
$E+E	* (c + d) − a	

id id

⋖

	+	-	*	/	^	id	(	)	$
+	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
-	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
*****	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
/	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
^	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
id	⋗	⋗	⋗	⋗	⋗			⋗	⋗
(	⋖	⋖	⋖	⋖	⋖	⋖	⋖	≐	
)	⋗	⋗	⋗	⋗	⋗			⋗	⋗
$	⋖	⋖	⋖	⋖	⋖	⋖	⋖		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift

id id

⋖

	+	-	*	/	^	id	(	)	$
+	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
-	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
*	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
/	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
^	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
id	⋗	⋗	⋗	⋗	⋗			⋗	⋗
(	⋖	⋖	⋖	⋖	⋖	⋖	⋖	≐	
)	⋗	⋗	⋗	⋗	⋗			⋗	⋗
$	⋖	⋖	⋖	⋖	⋖	⋖	⋖		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	

id id

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	

id id

$\lessdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
\$	a + b * (c + d) - a	shift
\$a	+ b * (c + d) - a	reduce E -> ID
\$E+	b * (c + d) - a	shift
\$E+b	* (c + d) - a	reduce E -> ID
\$E+E	* (c + d) - a	shift
\$E+E*	(c + d) - a	shift

id id

$\lessdot$

	+	-	*	/	^	id	(	)	\$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
\$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift

id id

⋖

	+	-	*	/	^	id	(	)	$
+	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
-	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
*	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
/	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
^	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
id	⋗	⋗	⋗	⋗	⋗			⋗	⋗
(	⋖	⋖	⋖	⋖	⋖	⋖	⋖	≐	
)	⋗	⋗	⋗	⋗	⋗			⋗	⋗
$	⋖	⋖	⋖	⋖	⋖	⋖	⋖		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	

id id

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) - a	shift
$a	+ b * (c + d) - a	reduce E -> ID
$E+	b * (c + d) - a	shift
$E+b	* (c + d) - a	reduce E -> ID
$E+E	* (c + d) - a	shift
$E+E*	(c + d) - a	shift
$E+E*(	c + d) - a	

id id

$\lessdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E+	b * (c + d) − a	shift
$E+b	* (c + d) − a	reduce E -> ID
$E+E	* (c + d) − a	shift
$E+E*	(c + d) − a	shift
$E+E*(	c + d) − a	shift

id id

$\lessdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	

id id

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	

id id

·>

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID

id id

·>

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	

id id id

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	

id id id

$\lessdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift

id id id

$\lessdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	

id id id

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	

id id id

<·

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift

id id id

$\lessdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	

id id id

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	

id id id

·>

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID

id id id

$\gtrdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	

id id id id

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	

id id id id

$\gtrdot$

	+	-	*	/	^	id	(	)	$
+	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
-	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
*****	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
/	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
^	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$	$\gtrdot$
id	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
(	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\doteq$	
)	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$			$\gtrdot$	$\gtrdot$
$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E

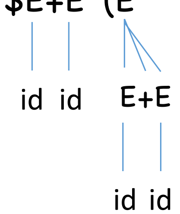
·>

id id id

	+	-	*	/	^	id	(	)	$
+	·>	·>	<·	<·	<·	<·	<·	·>	·>
-	·>	·>	<·	<·	<·	<·	<·	·>	·>
*****	·>	·>	·>	·>	<·	<·	<·	·>	·>
/	·>	·>	·>	·>	<·	<·	<·	·>	·>
^	·>	·>	·>	·>	<·	<·	<·	·>	·>
id	·>	·>	·>	·>	·>			·>	·>
(	<·	<·	<·	<·	<·	<·	<·	≐	
)	·>	·>	·>	·>	·>			·>	·>
$	<·	<·	<·	<·	<·	<·	<·		

EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	



When do we stop popping the stack?

$ E + E * (E + E)

$ ⋖ + ⋖ * ⋖ (E + E)

⋖ ⋗

when the E + E is popped, the following hold

1. the top of the stack is a terminal which is (
2. the last popped terminal is +
3. (⋖ +

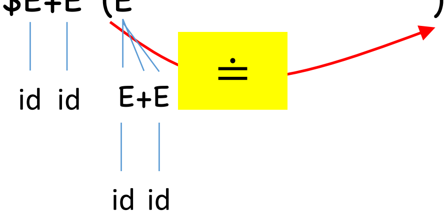
so we stop

what is popped is between a pair ⋖ and ⋗: ⋖ RHS of reduction ⋗

	+	-	*	/	^	id	(	)	$
+	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
-	⋗	⋗	⋖	⋖	⋖	⋖	⋖	⋗	⋗
*****	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
/	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
^	⋗	⋗	⋗	⋗	⋖	⋖	⋖	⋗	⋗
id	⋗	⋗	⋗	⋗	⋗			⋗	⋗
(	⋖	⋖	⋖	⋖	⋖	⋖	⋖	≐	
)	⋗	⋗	⋗	⋗	⋗			⋗	⋗
$	⋖	⋖	⋖	⋖	⋖	⋖	⋖		

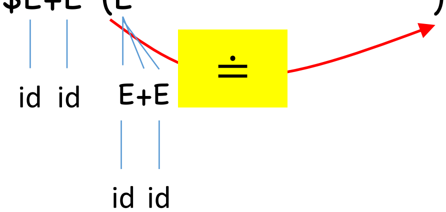
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	



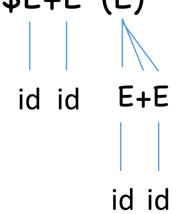
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift



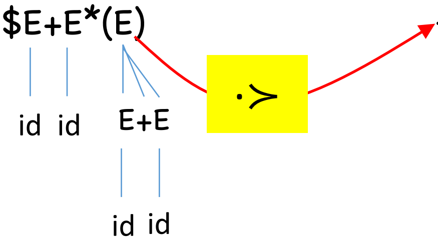
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	



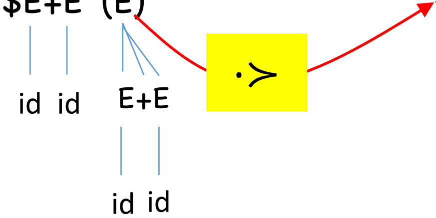
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	



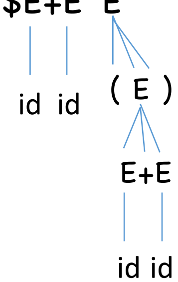
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)



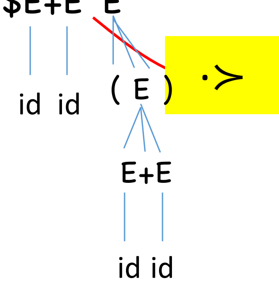
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E+	b * (c + d) − a	shift
$E+b	* (c + d) − a	reduce E -> ID
$E+E	* (c + d) − a	shift
$E+E*	(c + d) − a	shift
$E+E*(	c + d) − a	shift
$E+E*(c	+ d) − a	reduce E -> ID
$E+E*(E	+ d) − a	shift
$E+E*(E+	d) − a	shift
$E+E*(E+d	) − a	reduce E -> ID
$E+E*(E+E	) − a	reduce E -> E+E
$E+E*(E	) − a	shift
$E+E*(E)	− a	reduce E -> (E)
$E+E*E	− a	



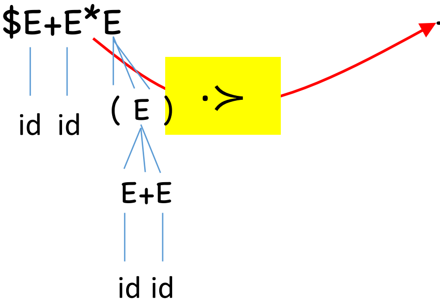
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E+	b * (c + d) − a	shift
$E+b	* (c + d) − a	reduce E -> ID
$E+E	* (c + d) − a	shift
$E+E*	(c + d) − a	shift
$E+E*(	c + d) − a	shift
$E+E*(c	+ d) − a	reduce E -> ID
$E+E*(E	+ d) − a	shift
$E+E*(E+	d) − a	shift
$E+E*(E+d	) − a	reduce E -> ID
$E+E*(E+E	) − a	reduce E -> E+E
$E+E*(E	) − a	shift
$E+E*(E)	− a	reduce E -> (E)
$E+E*E	− a	



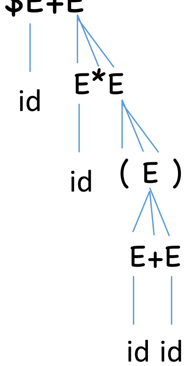
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E+	b * (c + d) − a	shift
$E+b	* (c + d) − a	reduce E -> ID
$E+E	* (c + d) − a	shift
$E+E*	(c + d) − a	shift
$E+E*(	c + d) − a	shift
$E+E*(c	+ d) − a	reduce E -> ID
$E+E*(E	+ d) − a	shift
$E+E*(E+	d) − a	shift
$E+E*(E+d	) − a	reduce E -> ID
$E+E*(E+E	) − a	reduce E -> E+E
$E+E*(E	) − a	shift
$E+E*(E)	− a	reduce E -> (E)
$E+E*E	− a	reduce E -> E*E



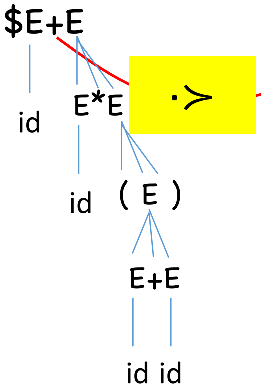
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	



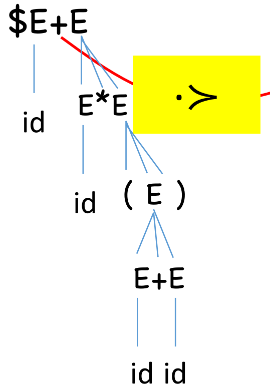
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) − a	shift
$a	+ b * (c + d) − a	reduce E -> ID
$E+	b * (c + d) − a	shift
$E+b	* (c + d) − a	reduce E -> ID
$E+E	* (c + d) − a	shift
$E+E*	(c + d) − a	shift
$E+E*(	c + d) − a	shift
$E+E*(c	+ d) − a	reduce E -> ID
$E+E*(E	+ d) − a	shift
$E+E*(E+	d) − a	shift
$E+E*(E+d	) − a	reduce E -> ID
$E+E*(E+E	) − a	reduce E -> E+E
$E+E*(E	) − a	shift
$E+E*(E)	− a	reduce E -> (E)
$E+E*E	− a	reduce E -> E*E
$E+E	− a	



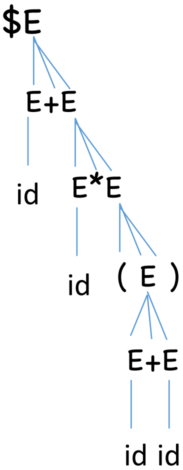
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E



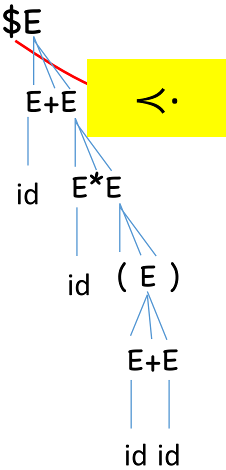
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	



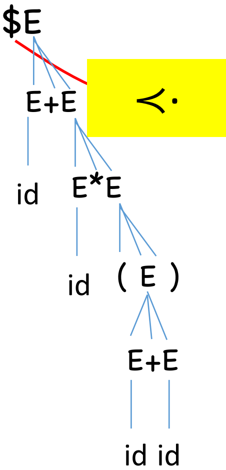
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	



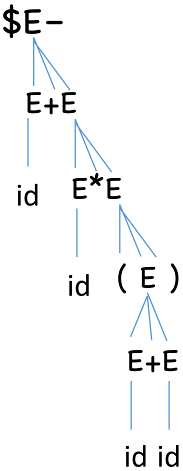
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift



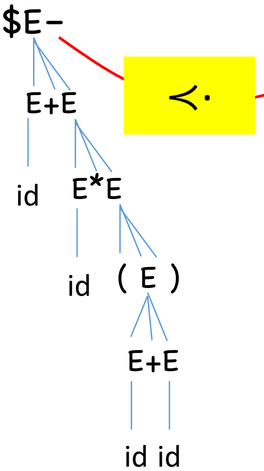
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	



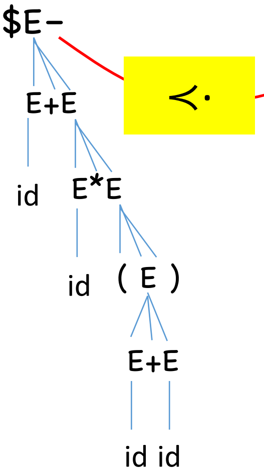
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	



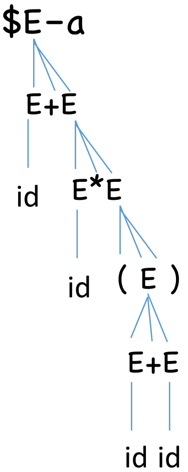
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	shift



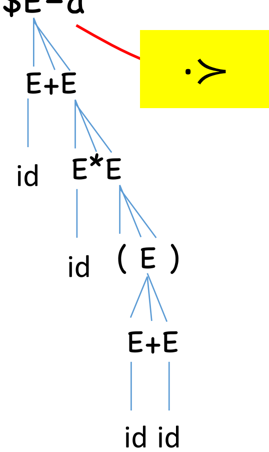
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	shift
$E–a		



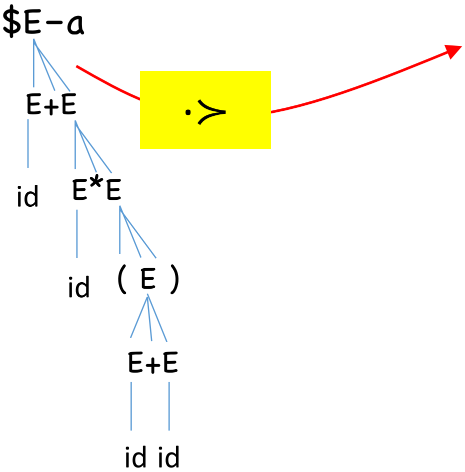
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	shift
$E–a	$	



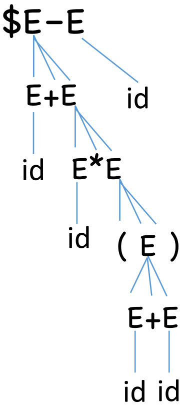
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	shift
$E–a	$	reduce E -> ID



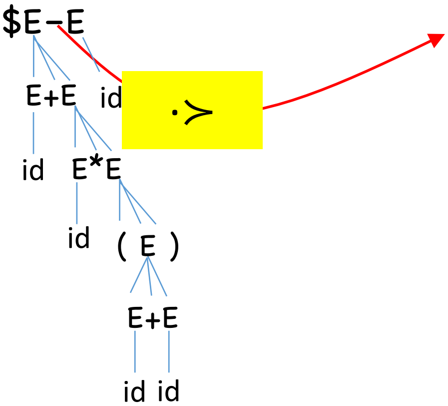
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	shift
$E–a	$	reduce E -> ID
$E–E	$	



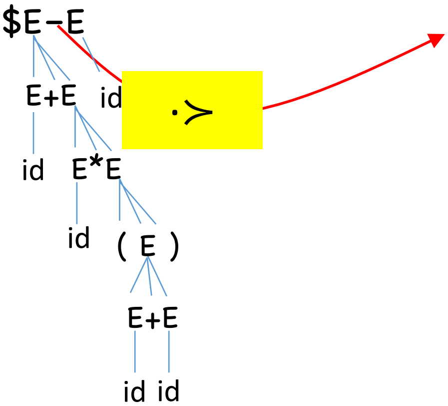
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	shift
$E–a	$	reduce E -> ID
$E–E	$	



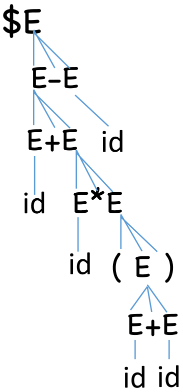
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	shift
$E–a	$	reduce E -> ID
$E–E	$	reduce E -> E–E



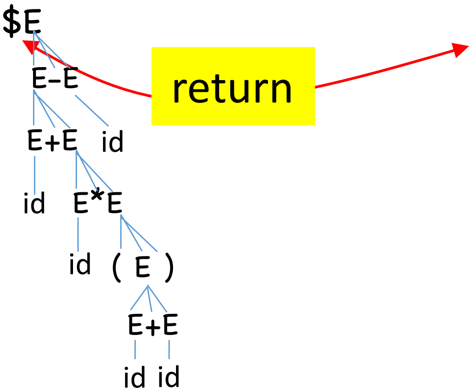
EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	shift
$E–a	$	reduce E -> ID
$E–E	$	reduce E -> E–E
$E	$	



EXAMPLE

Stack	Input	Action
$	a + b * (c + d) – a	shift
$a	+ b * (c + d) – a	reduce E -> ID
$E+	b * (c + d) – a	shift
$E+b	* (c + d) – a	reduce E -> ID
$E+E	* (c + d) – a	shift
$E+E*	(c + d) – a	shift
$E+E*(	c + d) – a	shift
$E+E*(c	+ d) – a	reduce E -> ID
$E+E*(E	+ d) – a	shift
$E+E*(E+	d) – a	shift
$E+E*(E+d	) – a	reduce E -> ID
$E+E*(E+E	) – a	reduce E -> E+E
$E+E*(E	) – a	shift
$E+E*(E)	– a	reduce E -> (E)
$E+E*E	– a	reduce E -> E*E
$E+E	– a	reduce E -> E+E
$E	– a	shift
$E–	a	shift
$E–a	$	reduce E -> ID
$E–E	$	reduce E -> E–E
$E	$	



Dealing with non-terminals

We have the non-terminals on the stack as they are pushed when a reduction occurs

Given that the grammar is an operator grammar, we can have at most one non-terminal on the top of the stack. There is always a terminal on the top of the stack or just below

In the algorithm we assume that stack.terminalpeek() ignores non-terminals and returns the terminal symbol at the top of the stack or just below

HEURISTIC FOR DETERMINING PRECDENCE RELATIONSHIPS

We assume we have a set of operators with

- precedence levels
- associativity (left or right)
- operators at the same level have the same associativity

We assume the input is of the form w $

We have the following heuristics for determining $\lessdot$, $\gtrdot$, and $\doteq$ relationships between operators

1. if op1 has higher precedence level than op2, then
   - op1 $\gtrdot$ op2
   - op2 $\lessdot$ op1

   Example: $* \gtrdot +$

   $+ \lessdot *$

   $\hat{} \gtrdot +$

   $+ \lessdot \hat{}$

HEURISTIC FOR DETERMINING PRECDENCE RELATIONSHIPS

2. if op1 and op2 are operators of the same operator precedence, possibly the same operator, then
   - **If they are left associative**:
     - op1 $\gtrdot$ op2
     - op2 $\gtrdot$ op1

   Example. + and − are left associative, so we have

   $$+ \gtrdot +$$
   $$+ \gtrdot -$$
   $$- \gtrdot +$$
   $$- \gtrdot -$$

   - **If they are right associative**:
     - op1 $\lessdot$ op2
     - op2 $\lessdot$ op1

   Example. ^ is right associative, so we have

   $$\hat{}\ \cdot\lessdot\ \hat{}$$
   $$\hat{}\ \lessdot\ \hat{}$$

HEURISTIC FOR DETERMINING PRECDENCE RELATIONSHIPS

3. Also, we have the following

1.	op	$\lessdot$	ID
2.	ID	$\gtrdot$	op
3.	op	$\lessdot$	(
4.	(	$\lessdot$	op
5.	op	$\gtrdot$	)
6.	)	$\gtrdot$	op
7.	$	$\lessdot$	op
8.	op	$\gtrdot$	$
9.	(	$\doteq$	)
10.	$	$\lessdot$	id
11.	id	$\gtrdot$	$
12.	$	$\lessdot$	(
13.	(	$\lessdot$	(
14.	(	$\lessdot$	id
15.	)	$\gtrdot$	$
16.	)	$\gtrdot$	)
17.	id	$\gtrdot$	)

Unary Operators (one operand)

If we have a unary operator uop that is not a binary operator, we can support it as follows

- op $\lessdot$ uop for every other operator op. op can be unary or binary

- uop $\lessdot$ op if uop has lower operator precedence level than op
- uop $\gtrdot$ op if uop has higher operator precedence level than op

If we have a unary operator that is also a binary operator, like MINUS, we cannot incorporate it in the scheme!

Example id*-id is not easily parsed

One solution is to have the getToken() function make the distinction by looking at the context in which the operator appears.

Example In FORTRAN a minus sign is unary if the previous token is an operator, LPAREN, COMMA, or EQUAL

It is better to handle this in the lexer than it is to make the parser more complicated

Unary Operators (one operand)

If we have a unary operator uop that is not a binary operator, we can support it as follows

- op $\lessdot$ uop for every other operator op. op can be unary or binary

- uop $\lessdot$ op if uop has lower operator precedence level than op
- uop $\gtrdot$ op if uop has higher operator precedence level than op

If we have a unary operator that is also a binary operator, like MINUS, we cannot incorporate it in the scheme!

Example id*-id is not easily parsed

One solution is to have the getToken() function make the distinction by looking at the context in which the operator appears.

Example In FORTRAN a minus sign is unary if the previous token is an operator, LPAREN, COMMA, or EQUAL

It is better to handle this in the lexer than it is to make the parser more complicated

Unary Operators (one operand)

If we have a unary operator uop <u>that is not a binary operator,</u> we can support it as follows

- op $\lessdot$ uop for every other operator op. op can be unary or binary
- uop $\lessdot$ op if uop has lower operator precedence level than op
- uop $\gtrdot$ op if uop has higher operator precedence level than op

If we have a unary operator that is also a binary operator, like MINUS, we cannot incorporate it in the scheme!

Example id*-id is not easily parsed

One solution is to have the getToken() function make the distinction by looking at the context in which the operator appears.

Example In FORTRAN a minus sign is unary if the previous token is an operator, LPAREN, COMMA, or EQUAL

It is better to handle this in the lexer than it is to make the parser more complicated

Unary Operators (one operand)

If we have a unary operator uop <u>that is not a binary operator</u> 😀, we can support it as follows

- op $\lessdot$ uop for every other operator op. op can be unary or binary
- uop $\lessdot$ op if uop has lower operator precedence level than op
- uop $\gtrdot$ op if uop has higher operator precedence level than op

If we have a unary operator that is also a binary operator 😫, like MINUS, we cannot incorporate it in the scheme!

Example id*-id is not easily parsed

One solution is to have the getToken() function make the distinction by looking at the context in which the operator appears.

Example In FORTRAN a minus sign is unary if the previous token is an operator, LPAREN, COMMA, or EQUAL

It is better to handle this in the lexer than it is to make the parser more complicated