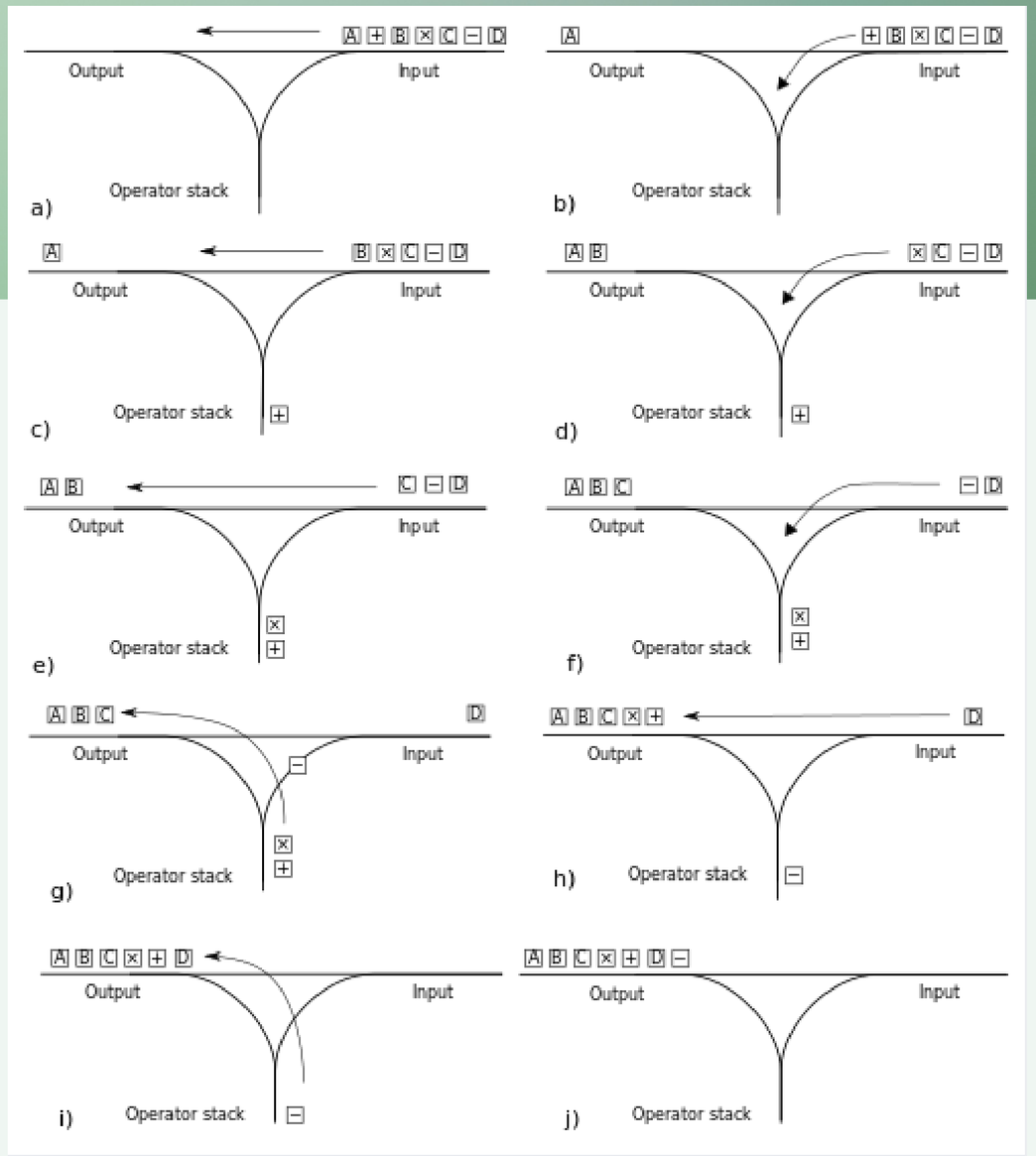
LOGOS

ZABAWKOWY JĘZYK BEZ PARSERA

Szymon Jędras


A + B × C − D
Output
Input
Operator stack
a)
A
+ B × C − D
Output
Input
Operator stack
b)
A
B × C − D
Output
Input
Operator stack
+
c)
A B
× C − D
Output
Input
Operator stack
+
d)
A B
C − D
Output
Input
Operator stack
×
+
e)
A B C
− D
Output
Input
Operator stack
×
+
f)
A B C
D
Output
Input
Operator stack
−
×
+
g)
A B C × +
D
Output
Input
Operator stack
−
h)
A B C × + D
Output
Input
Operator stack
−
i)
A B C × + D −
Output
Input
Operator stack
j)

GRAMATYKA BEZKONTEKSTOWA

```
prog := list(expr)

expr :=
  | expr ";"
  | assign
  | lambda
  | if-expr
  | pair-expr
  | lambda-app
  | arthm-expr
  | builtin
  | "{" expr "}"
  | bool-expr
```

```
assign := var "=" expr

lambda := var "->" expr

var := id

if-expr := "if" expr expr expr

pair-decl := expr "," expr

lambda-app := expr "$" expr
```

GRAMATYKA BEZKONTEKSTOWA cd.

```
arthm-expr :=
  | expr "+" expr
  | expr "-" expr
  | expr "*" expr
  | expr "/" expr
  | expr "%" expr
  | var
  | char
  | number
  | "()"
  | "(" arthm-expr ")"
```

```
builtin :=
  | "fst" expr
  | "snd" expr
  | "readc"
  | "writec" expr
  | "at" expr
  | "is_number" expr
  | "is_unit" expr
  | "is_bool" expr
  | "is_pair" expr
  | "true"
  | "false"
```

```
bool-expr :=
  | expr comp expr
  | bool-expr && bool-expr
  | bool-expr || bool-expr
  | !bool-expr
```

TYPY WARTOŚCI

- Number (int)
- Bool
- Unit
- Lambda 'a -> 'b
- Pair ('a * 'b)

PRZYKŁADOWY PROGRAM

```
map = fn -> list ->
    if is_pair list
        fn $ (fst $ list) , map $ fn $ (snd $ list)
        ()

inc = i -> i + 1

map $ inc $ (1,2,3,4,())
```

DZIĘKUJĘ

ZA UWAGĘ