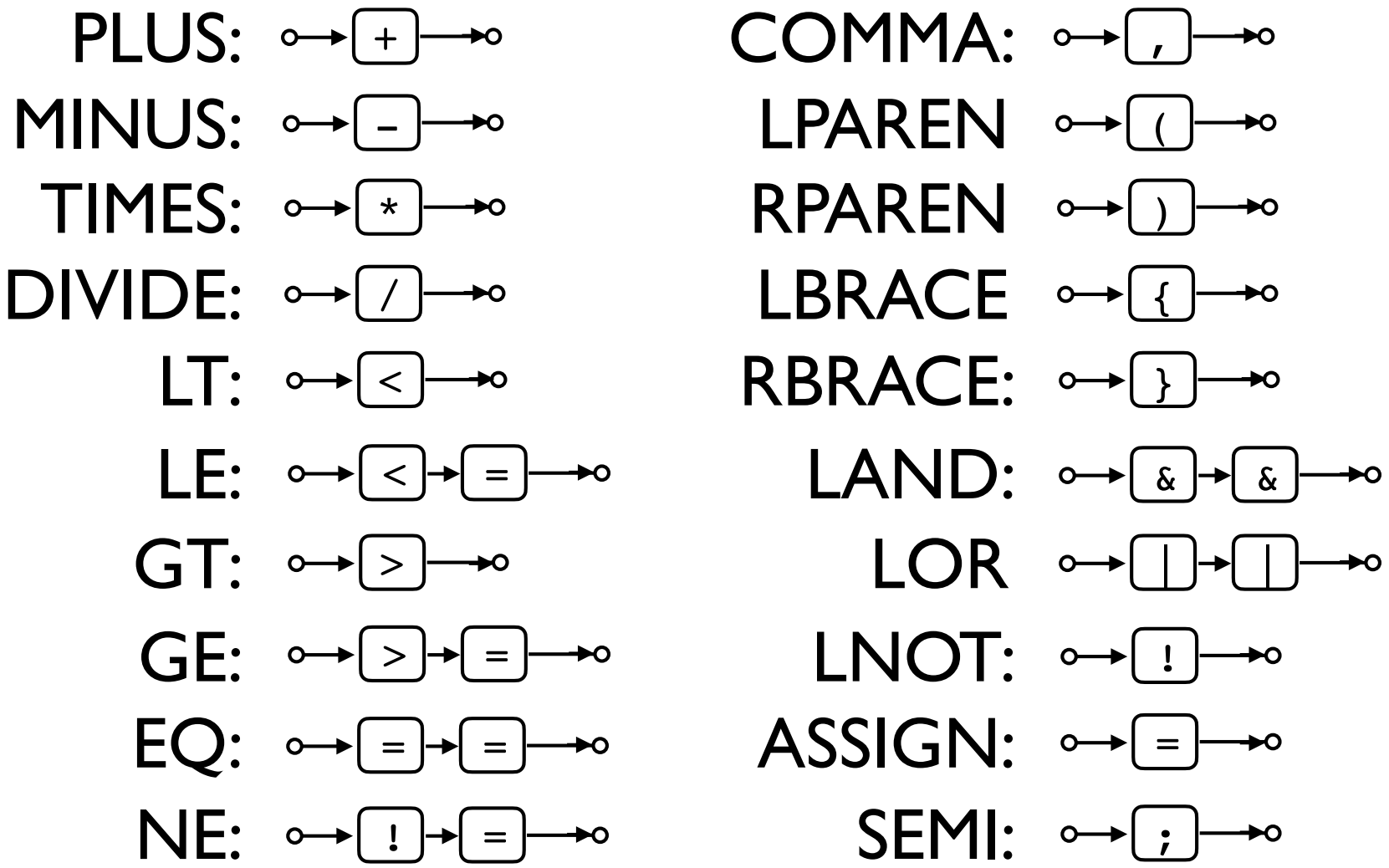
Wabbit Syntax Diagrams

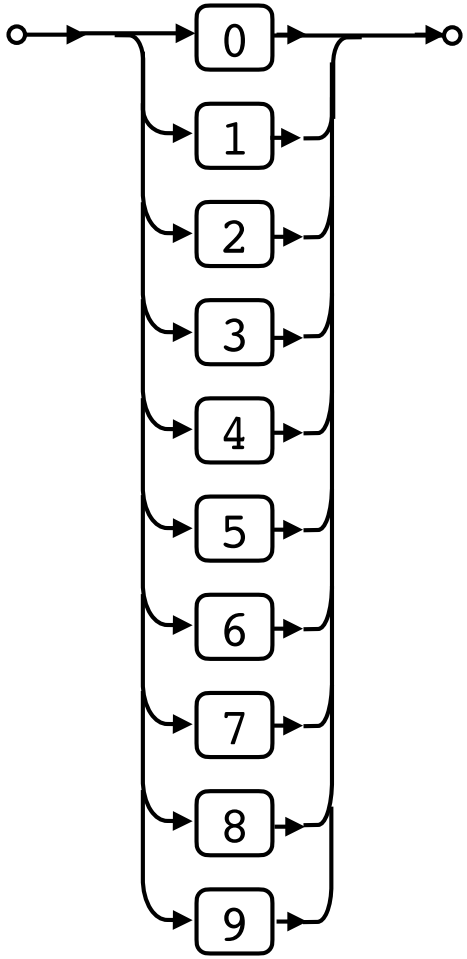
This document describes the Wabbit language syntax using syntax diagrams. These might be useful in understanding the implementation of a hand-written parser.

Wabbit: Symbols

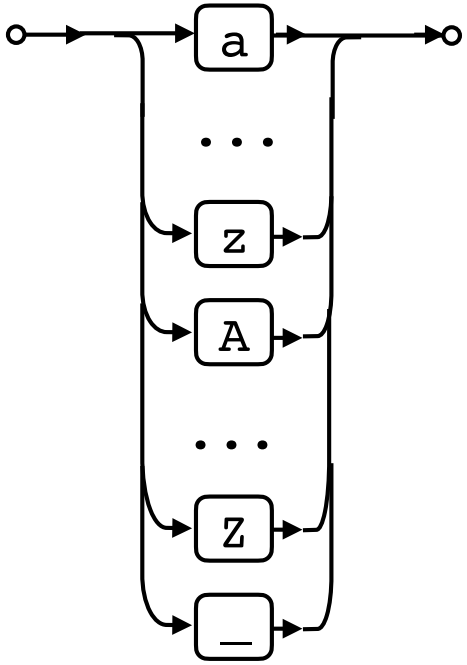


Wabbit: Character Classes

digit:



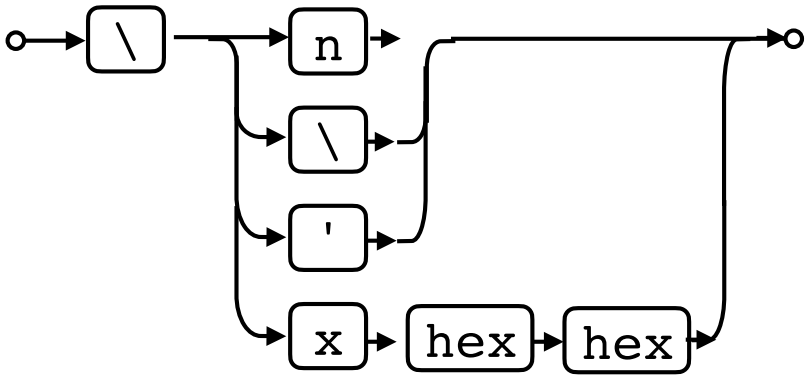
alpha:



char:

all characters
except ' and \

escape:

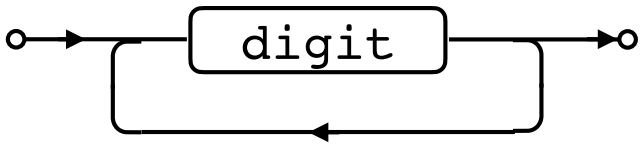


alnum:

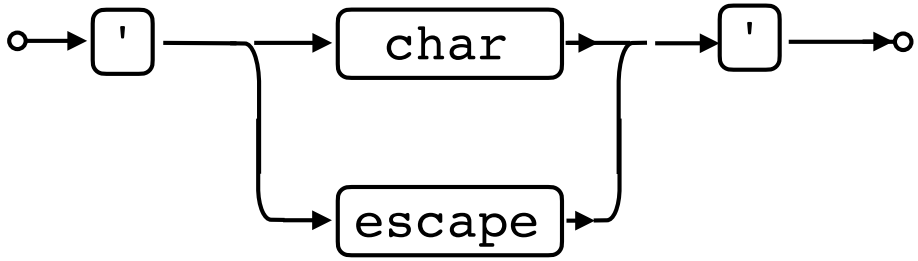
Note: These are used internally by the tokenizer

Wabbit: Literals, Names, Keywords

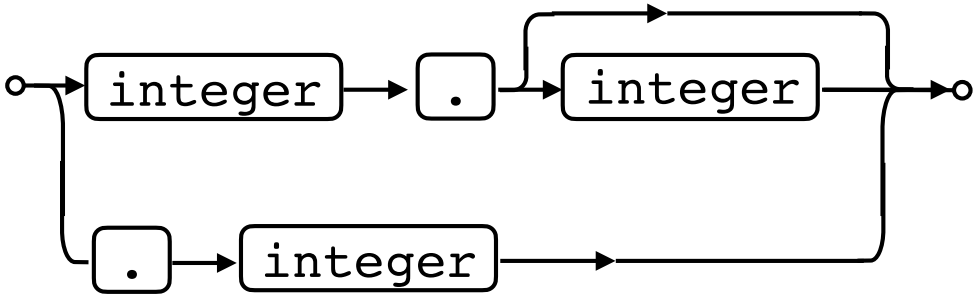
INTEGER:



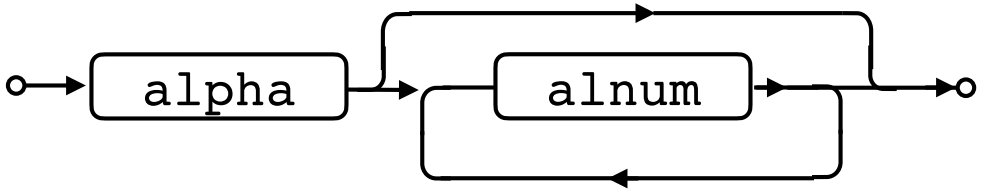
CHAR:



FLOAT:



NAME:



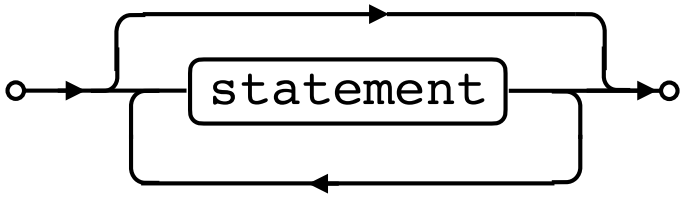
Keywords:

```
PRINT: "print"
  VAR: "var"
CONST: "const"
   IF: "if"
WHILE: "while"
 ELSE: "else"
BREAK: "break"
CONTINUE: "continue"
 FUNC: "func"
RETURN: "return"
 TRUE: "true"
FALSE: "false"
```

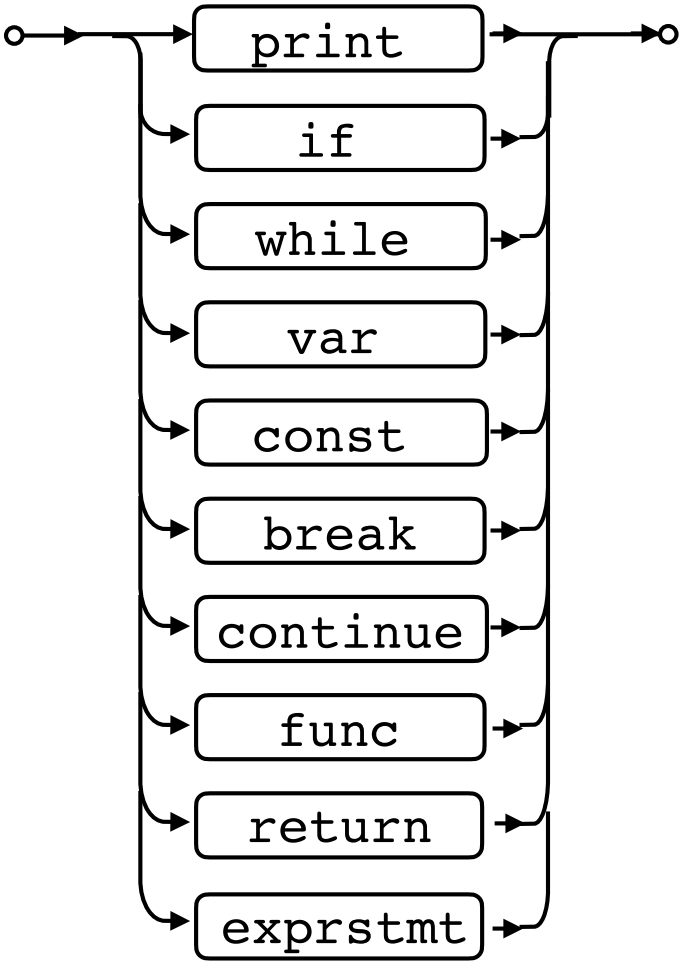
Wabbit: Program Structure

program:

statements:



statement:



Wabbit: Basic Statements

print:

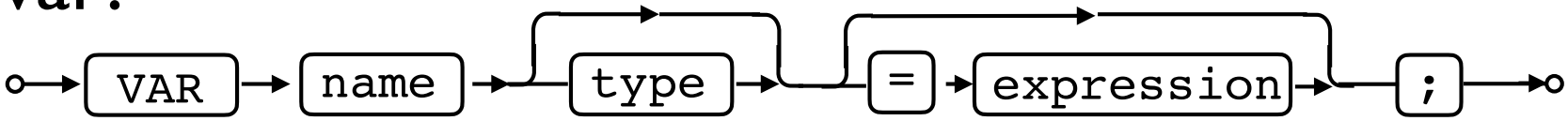
exprstmt:

example:

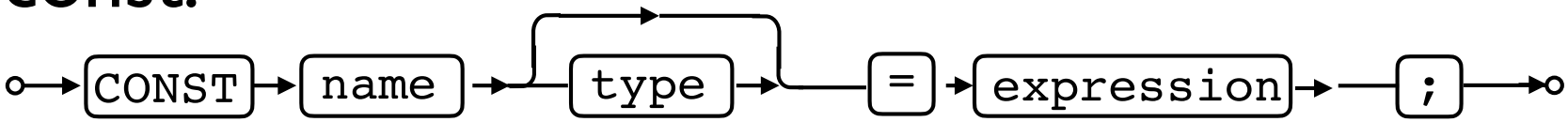
```
print a * 10;
a * 10;
```

Wabbit: Declarations

var:



const:



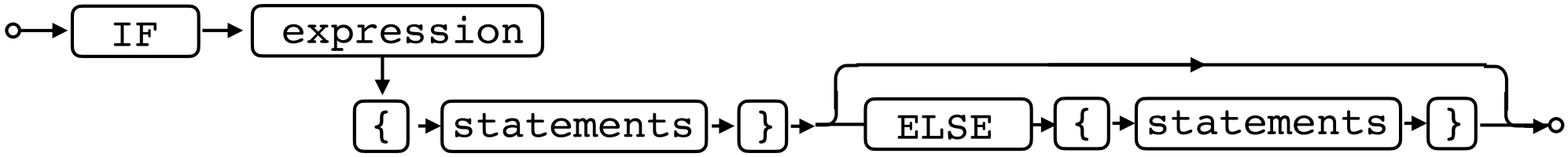
type:

example:

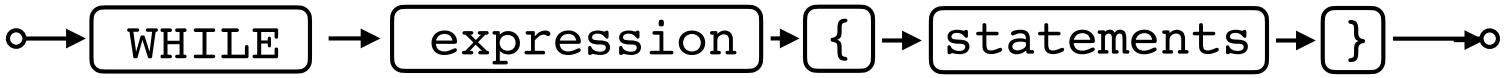
```
var a int;
var b = 42;
const c = b + 2;
```

Wabbit: Control Flow

if:



while:



break:

continue:

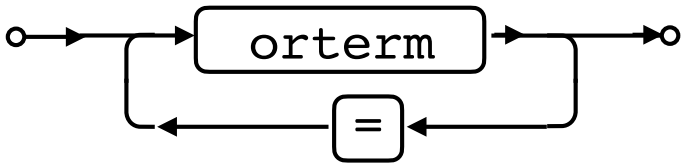
example:

```
if n < 10 {
    ...
} else {
    ...
}
```

Wabbit: Expressions

expression:

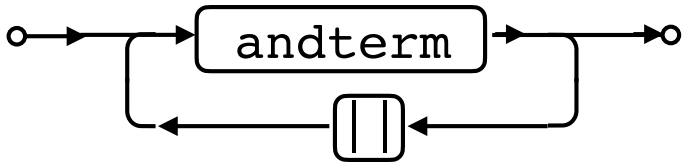
assign:



example:

```
a = b = 4;
```

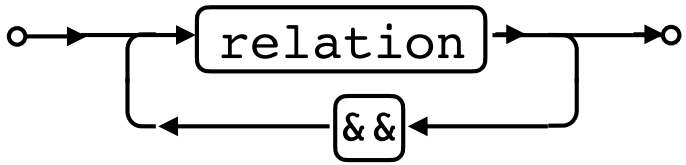
orterm:



example:

```
(...) || (...) || (...)
```

andterm:

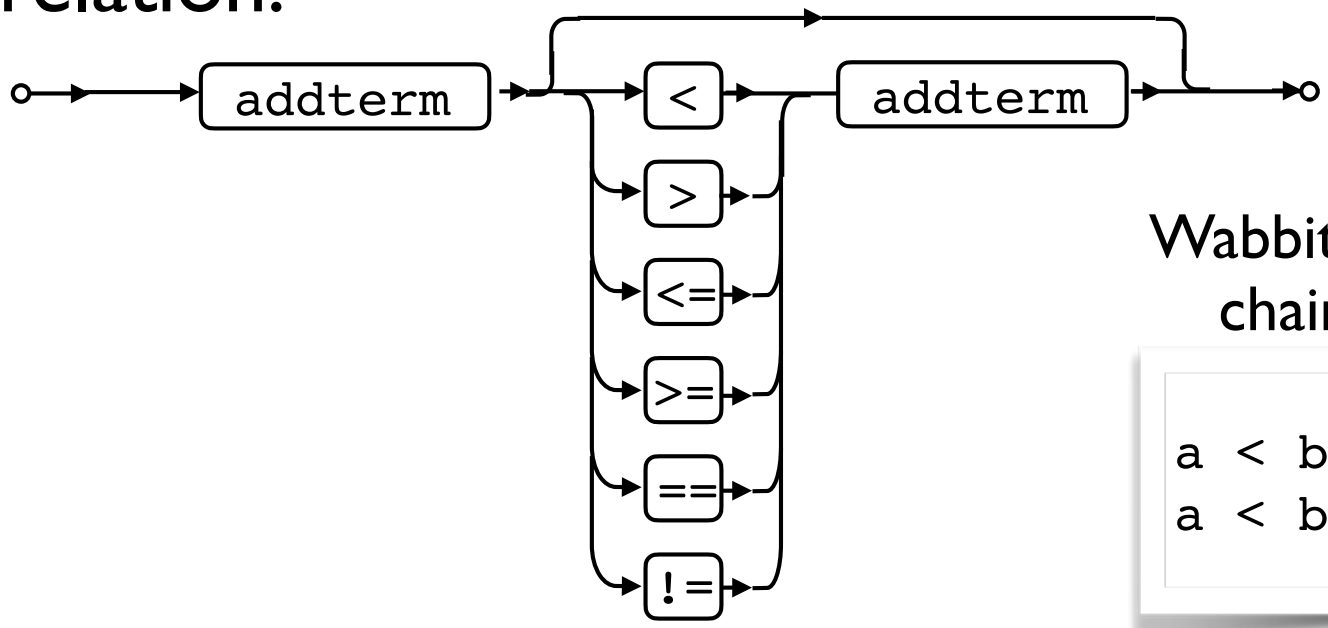


example:

```
(...) && (...) && (...)
```

Wabbit: Expressions (cont)

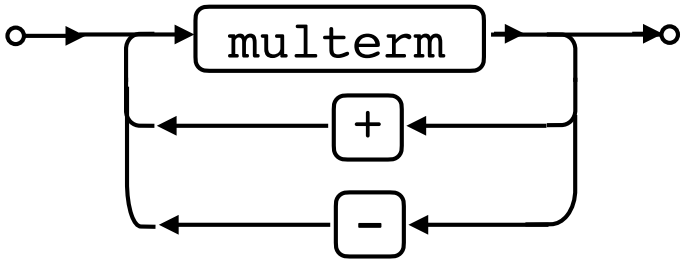
relation:



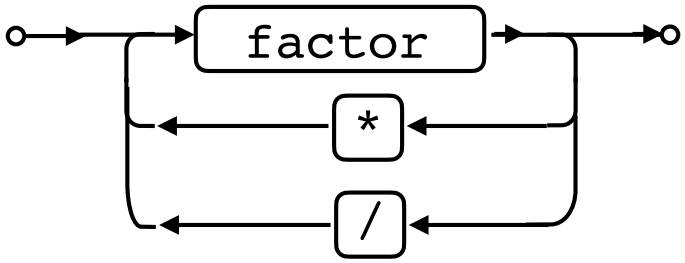
Wabbit does not allow chained relations.

```
a < b          // OK
a < b > c      // ERROR
```

addterm:

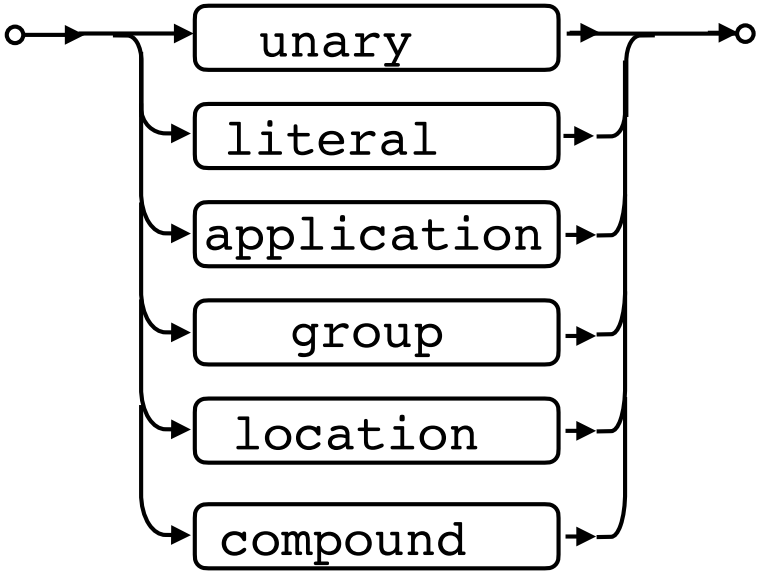


multerm:

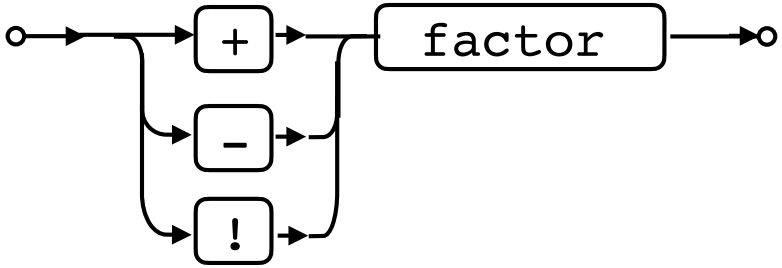


Wabbit: Expressions (cont)

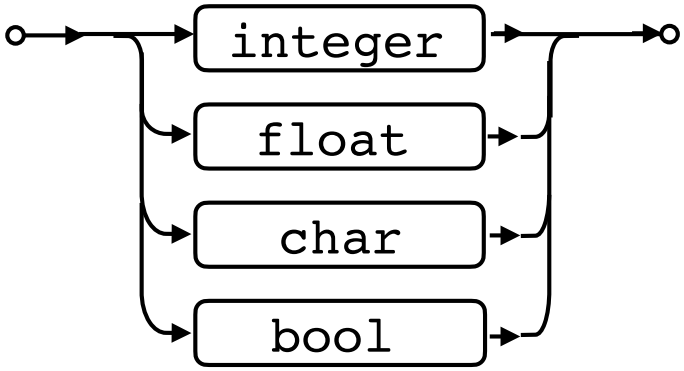
factor:



unary:

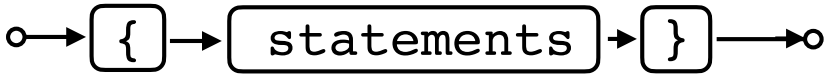


literal:



group:

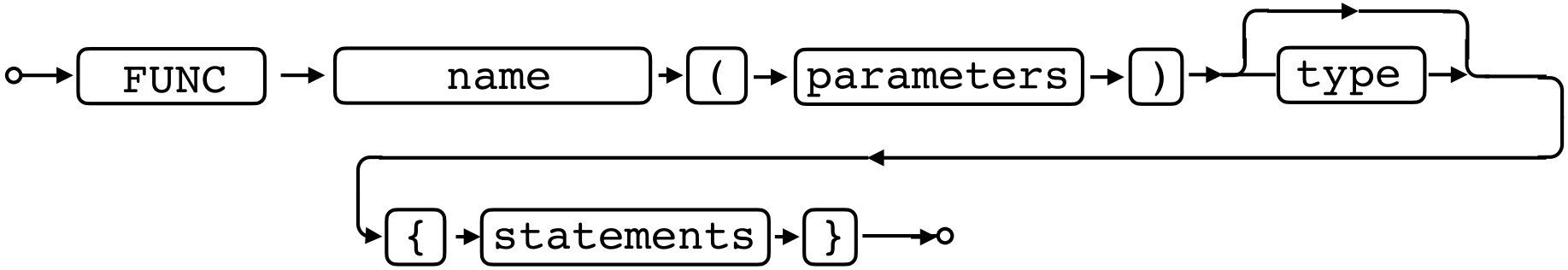
compound:



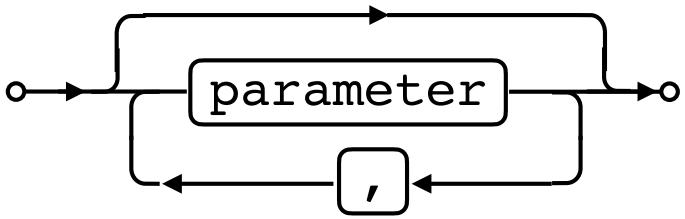
location:

Wabbit: Function Definition

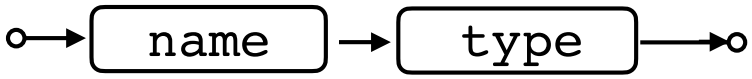
func:



parameters:



parameter:



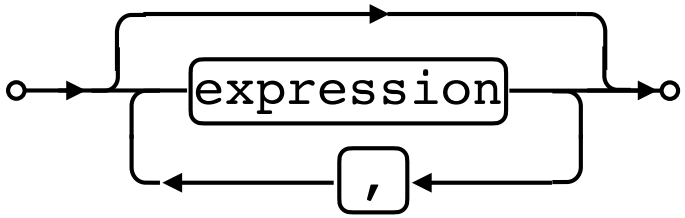
example:

```
func g(x int, y int) int {
    ...
}
```

Wabbit: Function Application and Return

application:

arguments:



return:

example:

```
r = g(2, 3+x);
return r * 10;
```