# Solutions for Exercise 1 (1)

**1)**

- *A = c { a | b }.*
- *A = c { a b }.*
- *A = { a } c { b }.*
- *A = c { b } b c { b } { a }.*

**2)**

*Float = [ "+" | "-" ] Digit { Digit } "." Digit { Digit }*
*[ "e" [ "+" | "-" ] Digit { Digit } ].*

**5)**

*Result: 42*
*typical mistakes:*

- *ts[i-1] different i (local i always 1, global i always 2)*
- *bar() not called*

# Solutions for Exercise 1 (2)

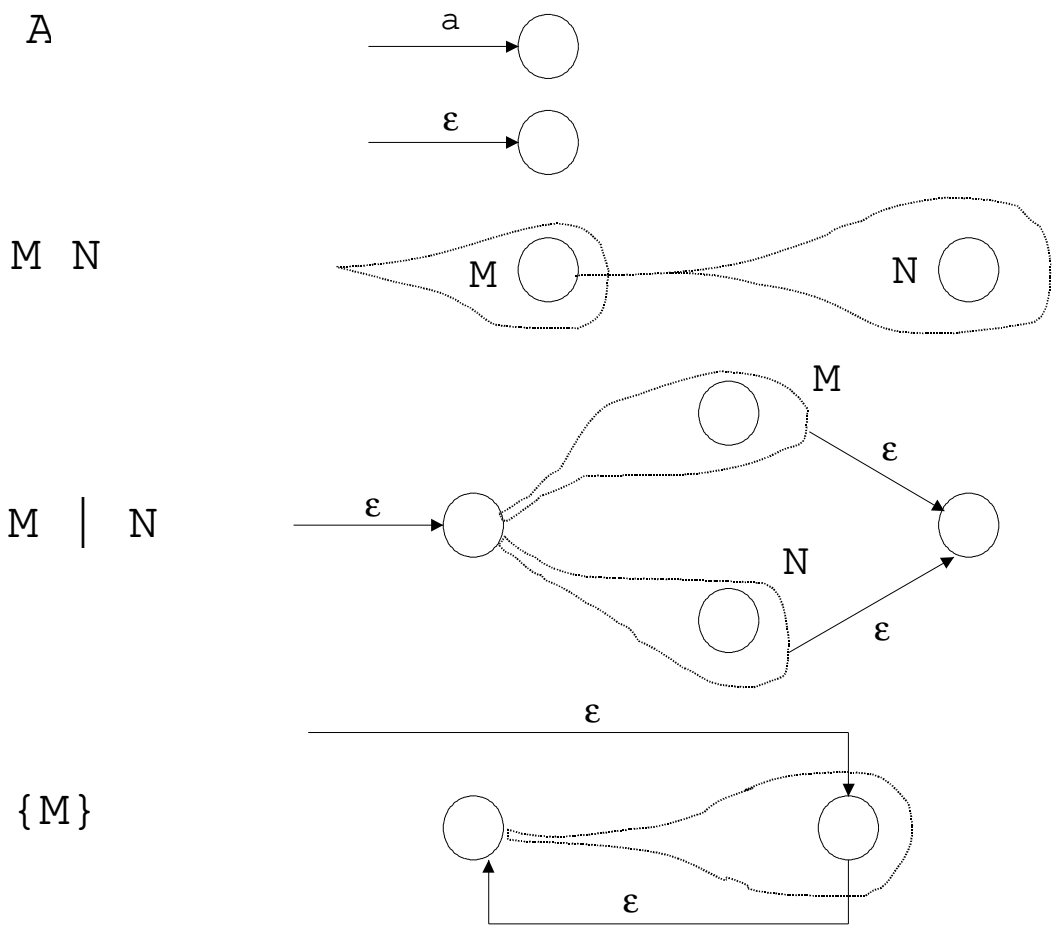
```
module Star {
    Star[][] trek;

    int[][] enterprise() {
        while (false);
        { return newarray int[]("trek"); }
    }

    int voyager(int x, y) {
        if x > y
            int z = 2;
        else
            return enterprise()[-!y];
        return !-5;
    }
}

module LinearAlgebra {
    int scalarProduct(int[] xs, int[] ys, int n) {
        if (n == 0)
            return 0;
        else
            return xs[n-1] * ys[n-1] + scalarProduct(xs,ys,n-1);
    }
    int scalarProduct2(int[] xs, int[] ys, int n) {
        int result;
        result = 0;
        while (n > 0) {
            n = n - 1;
            result = result + xs[n] * ys[n];
        }
        return result;
}
```

# Regular Expressions ➔ NFA




Example: construct a NFA for the given regular expression using the systematic scheme:

***a { ( b | { a } c ) x } | { x } a***

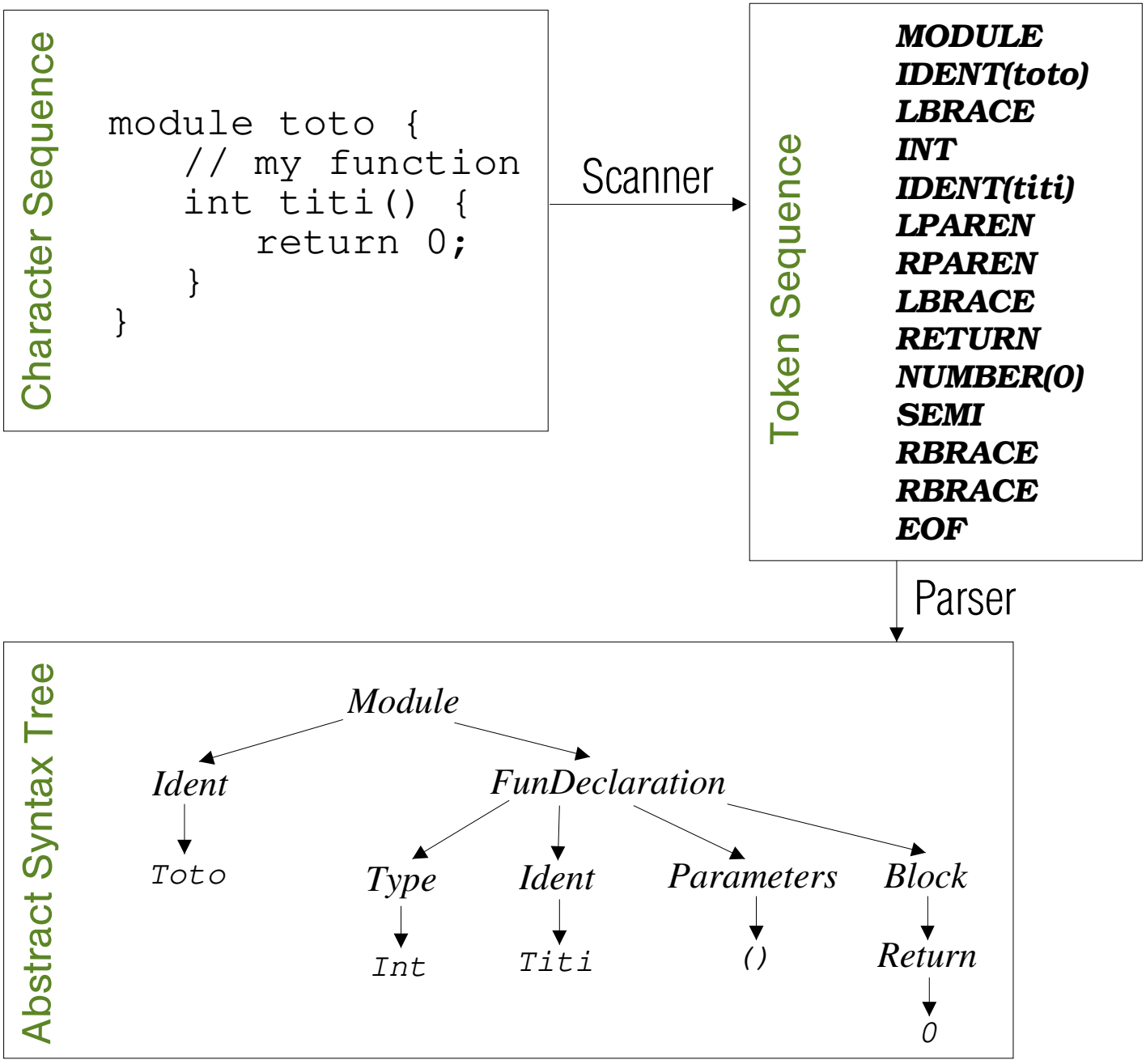
# Syntax Description

The syntax of programming languages is usually given in two stages:

- the *micro syntax* describes the form of individual words or tokens (structure of tokens)
- the *macro syntax* describes how programs are formed out of tokens (structure of sentences)

This principle is reflected within a compiler in the following way:

# Syntax Analysis



# Example: Declarations

```
Decl       = VarDecl
           | FunDecl.

FunDecl    = "void" ident "(" FormalOpt ")" ";"
           | Type ident "(" FormalOpt ")" ";".

VarDecl    = Type ident ";".

Type       = "int" | "string" | Type "[" "]".

FormalOpt = Type ident | ε.
```

- What are the token classes?
- What is the task of the scanner?
- What is the task of the parser?
- Which non-terminals are nullable?
- What are the first and follow sets?
- Is this Grammar LL(1)?

# Declarations in LL(1)

```
Decl        = VoidFunDecl
            | Type ident TypedDecl.
VoidFunDecl = "void" ident "(" FormalOpt ")" ";".
TypedDecl   = ";"
            | "(" FormalOpt ")" ";".
Type        = "int" Brackets
            | "string" Brackets.
Brackets    = "[" "]" Brackets | ε.
FormalOpt   = Type ident | ε.
```