Context-Free Grammars
Parser

Assumed Knowledge

- Binary trees
  - traversing in-order, post-order
  - term trees
- Stack
  - programming of stack

Definition: Context-Free Grammars

Grammar $G = (N, T, \Pi, S)$

Set of non-terminals (variables)

not yet completely released

Set of terminals

`*` `(` `)` `9` `if`

Start variable

`begin`

Set of productions of the form:

$$N \rightarrow x$$

x is string of variables and terminals

Definition: Parser

= a **computer program** that analyses the grammatical structure of an **input**, with respect to a given **formal grammar**.

- Parsers can be made both for **natural languages** and for **programming languages**.
- Programming language parsers tend to be based on **context free grammars**.

VSL – Very Simple Language

(Language of Arithmetical Expressions)

Language Elements:

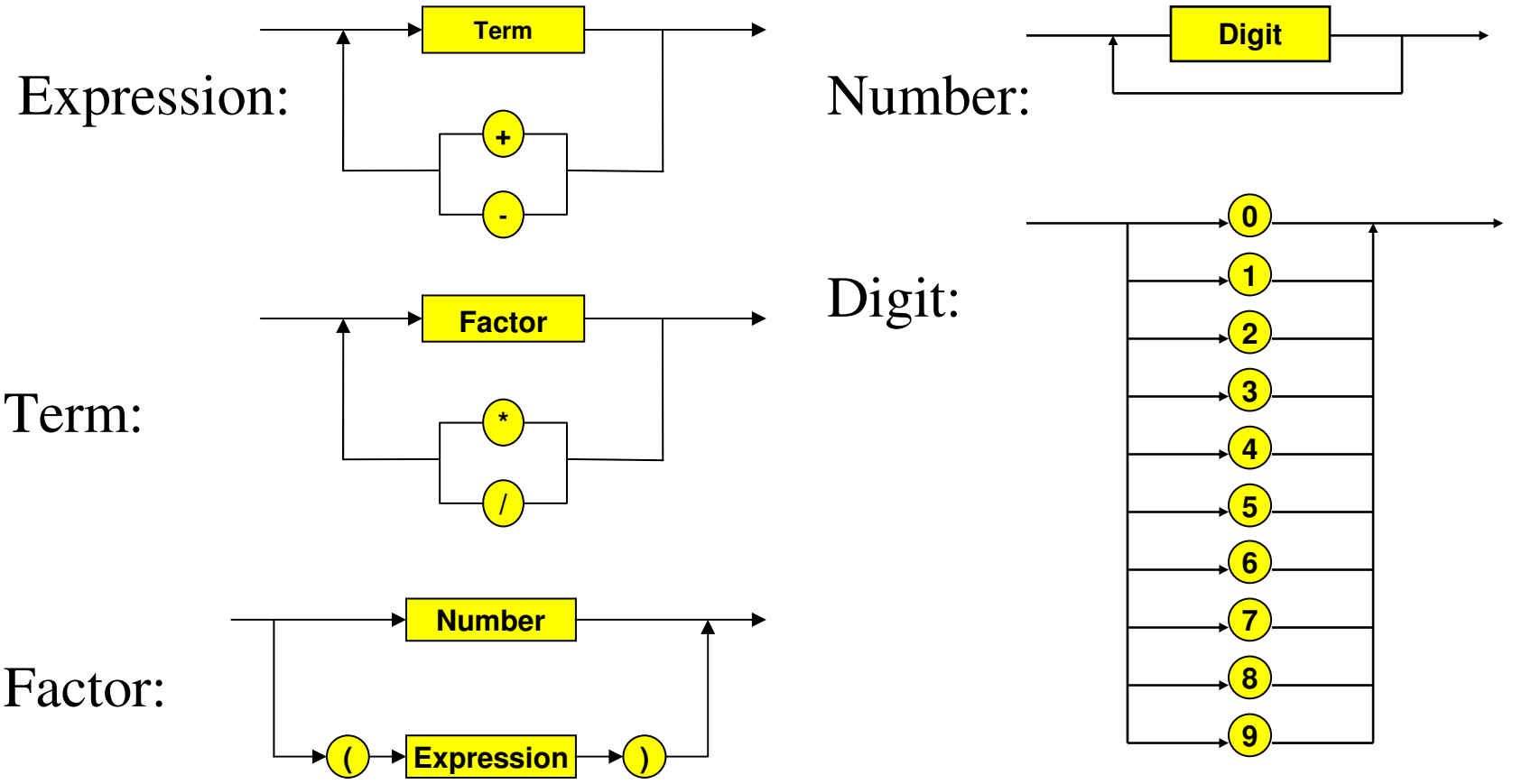
1. Binary Operators
2. Parentheses for building groups
3. Operands in the form of numbers

Basis: a number is an expression

Induction: E is an expression => the following constructs are expressions as well

- (E)
- E + E
- E - E
- E * E
- E / E

Syntax graphs



Production rules

(1) <Expression> → <Expression> + <Term> | <Expression> - <Term> | <Term>

(2) <Term> → < Term > * <Factor> | < Term > / <Factor> | <Factor>

(3) < Factor > → (<Expression>) | <Number>

(4) <Number> → < Number > <Digit> | < Digit >

(5) < Digit > → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

(6) <Program> → { <Expression> }

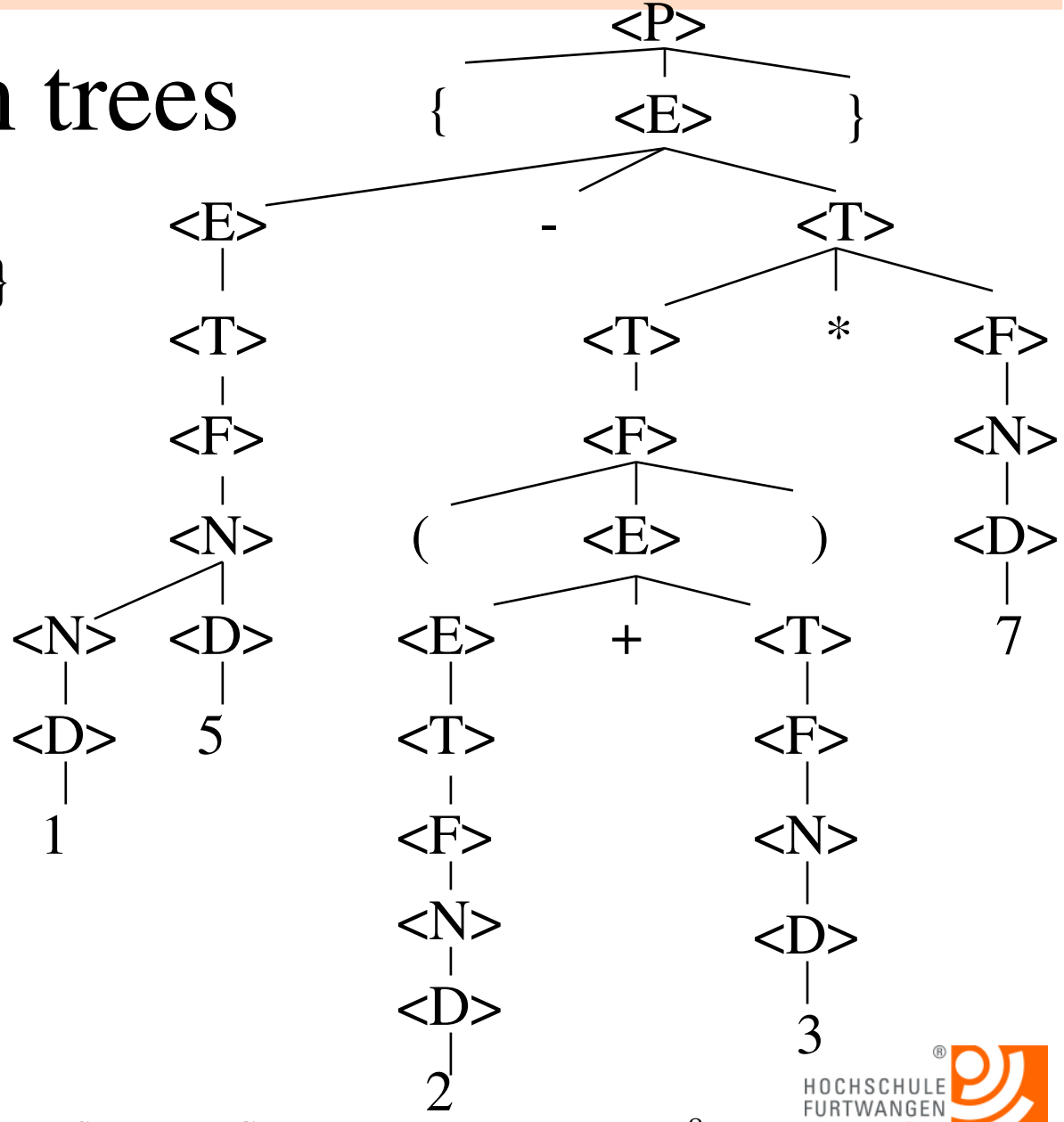
Notation:

<xyz>:	Syntax category
→	can be built from
0,1,2,...,9	atomic operands, so called **terminal symbols**

Feedback in the *syntax graphs* accords to
a **recursive definition** in *production rules*

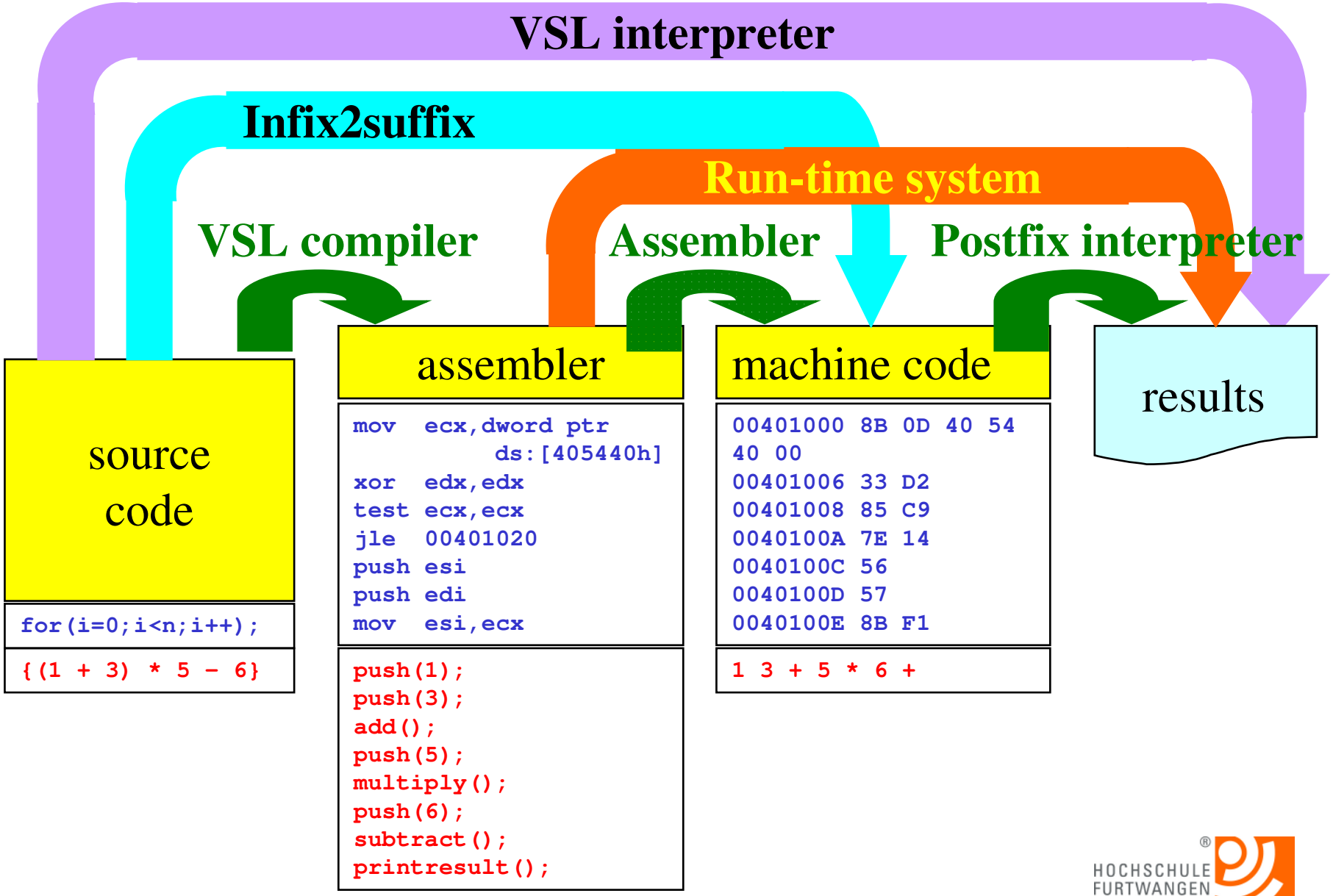
Derivation trees

{ 15 - (2 + 3) * 7 }



Example programs

- **VSL interpreter**
  interprets VSL-source
- **Infix2suffix**
  compiles VSL-source to VSL-machine language (EXE)
- **Suffix interpreter**
  Executes VSL-machine language
- **VSL compiler**
  compiles VSL-source to VSL-assembler
  The VSL-Assembler file is a C++ or Java-source
- **Run-time system**
  Links the VSL assembler with a pushdown automaton file to a run-time system


VSL interpreter
Infix2suffix
Run-time system
VSL compiler
Assembler
Postfix interpreter
source code
assembler
machine code
results
for(i=0;i<n;i++);
{(1 + 3) * 5 - 6}
mov ecx,dword ptr ds:[405440h]
xor edx,edx
test ecx,ecx
jle 00401020
push esi
push edi
mov esi,ecx
00401000 8B 0D 40 54 40 00
00401006 33 D2
00401008 85 C9
0040100A 7E 14
0040100C 56
0040100D 57
0040100E 8B F1
push(1);
push(3);
add();
push(5);
multiply();
push(6);
subtract();
printresult();
1 3 + 5 * 6 +