

Introduction to Program Synthesis (WS 2024/25)

Chapter 2.1 - Foundations (Program Representation: Tree)

Dr. rer. nat. Roman Kalkreuth

Chair for AI Methodology (**AIM**), Department of Computer Science,
RWTH Aachen University, Germany



Center for
Artificial Intelligence

RWTHAACHEN
UNIVERSITY

Computer Programs: Representations

- ▶ Data structures can be used to represent programs on a syntactical level:
 - ▶ Non-linear:
 - ▶ Tree
 - ▶ Graph
 - ▶ Linear:
 - ▶ List
 - ▶ Stack
- ▶ Another way are grammars:
 - ▶ Backus-Naur form

Computer Programs: Representations

Expression Tree

- ▶ Trees are commonly used to represent programs and expression for different purposes:
 - ▶ Symbolic expressions → **expression tree**
 - ▶ Syntactic structures → **program tree**
 - ▶ Context-free grammars → **parse tree**

Computer Programs: Representations

Expression Tree

Definition (Computer Program)

A computer program \mathcal{P} can be defined as a finite **sequence** of **operations** (or instructions) executed with respective **arguments** and in specific order. Let

- ▶ \mathcal{F} a finite non-empty set of operations
- ▶ \mathcal{T} a finite non-empty set of arguments
- ▶ Ω an order of operations (of the programming language)

\mathcal{P} is then a sequence of elements from $\mathcal{F} \times \mathcal{T}$ w.r.t. Ω

Computer Programs: Representations

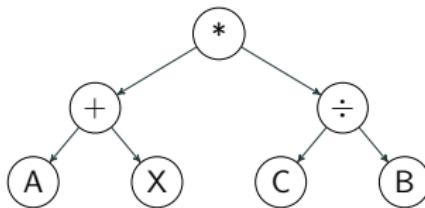
Expression Tree

- ▶ Trees are a suitable way to represent **symbolic expressions**
- ▶ Each **node** is annotated either with a **function/operator** or **terminal** symbol
 - ▶ Inner **nodes** represent **operators**
 - ▶ Each **leaf node** represents an **operand**
- ▶ Concept is also known in as **syntax tree** in literature

Computer Programs: Representations

Expression Tree

Example (Expression Tree)



$$\mathcal{F} = \{ +, -, *, \div \}$$

$$\mathcal{T} = \{ A, X, C, B \}$$

\mathcal{E} = Edges

$$\Psi = (A + X) * (C \div B)$$

Computer Programs: Representations

Expression Tree

Definition (Expression Tree)

An expression tree \mathcal{P} is an element of $\mathcal{T} \times \mathcal{F} \times \mathcal{E}$:

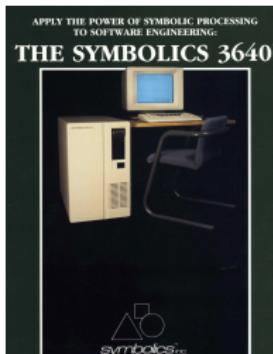
- ▶ \mathcal{F} is a finite non-empty set of operators
- ▶ \mathcal{T} is a finite non-empty set of terminals
- ▶ \mathcal{E} is a finite non-empty set of edges

Each leaf node represents an **operand** and each inner node represents an **operator** or **function**. Let $\phi : \mathcal{P} \mapsto \Psi$ be a **decode function** which maps \mathcal{P} to a expression Ψ

Computer Programs: Representations

Expression Tree

- ▶ LISP (*List Processor*) → family of programming languages
 - ▶ Developed in the late 1950s
 - ▶ Second oldest high-level programming language (besides Fortran)
 - ▶ Follows functional and procedural paradigm
- ▶ Influenced by λ -calculus
- ▶ Popular representative of the Lisp family → Common Lisp
 - ▶ Often applied for symbolic computation → *symbolic programming language*
 - ▶ Originally designed for **symbolic data processing**



Computer Programs: Representations

Expression Tree

```
1 ; Function which iteratively calculates the GCD by using Euclidean algorithm
2 (defun euclidean (a, b)
3   (if (< a b)
4     (let* ((tmp a))
5       (setq ((a b)
6             (b tmp))))
7     (let ((r 0) (d 0))
8       (loop
9         (setq (r (mod a b)))
10        (when (= r 0) (return a))
11        (setq (a (floor (/ a b))))
12        (setq (b r))))))
```

Listing: Iterative implementation of Euclidean's algorithm in LISP

```
1 ; Function which recursively calculates the GCD by using Euclidean algorithm
2 (defun euclidean (a b)
3   (if (< a b)
4     (progn
5       (let* ((tmp a)
6             (setq a b b tmp))))
7     (if (= b 0)
8         (return a))
9     euclidean(b, a mod b)))
```

Listing: Recursive implementation of Euclidean's algorithm in LISP

Computer Programs: Representations

Expression Tree

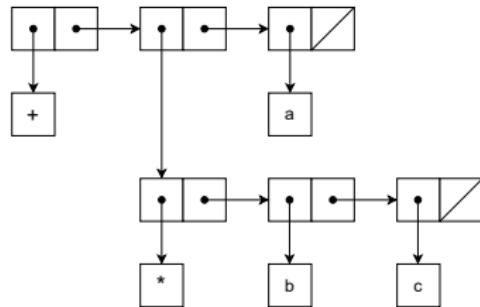
- ▶ LISP programs can be composed entirely of expressions with no use of statements
- ▶ Programs are made of three major building blocks:
 - ▶ Atoms
 - ▶ Lists
 - ▶ Symbolic expressions
- ▶ Lists → **sequence of atoms** or **nested sublists** encapsulated in the parentheses

Computer Programs: Representations

Expression Tree

Example (Nested List)

```
1 (defun foobar (a b c) (+ a (* b c)))
```



Computer Programs: Representations

Expression Tree

- ▶ Representation of LISP code and data → **S-expression** (symbolic expression)
 - ▶ Also known as sexpr or s-exp
 - ▶ Storage unit for LISP programmes
- ▶ LISP statements are represented in **prefix notation** (Polish notation)
- ▶ S-expressions are **nested lists** of symbols → can be represented as **tree-like structures**

Computer Programs: Representations

Expression Tree

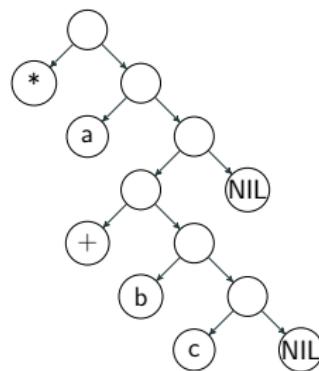
- ▶ A S-expression can be considered a composition of:
 - ▶ Atoms → Symbols, numbers or strings
 - ▶ Cons cell → Connection of subexpressions: $(S_1 . S_2)$
 - ▶ List termination symbol → NIL

Computer Programs: Representations

Expression Tree

Example (S-Expression)

(* a (+ b c))

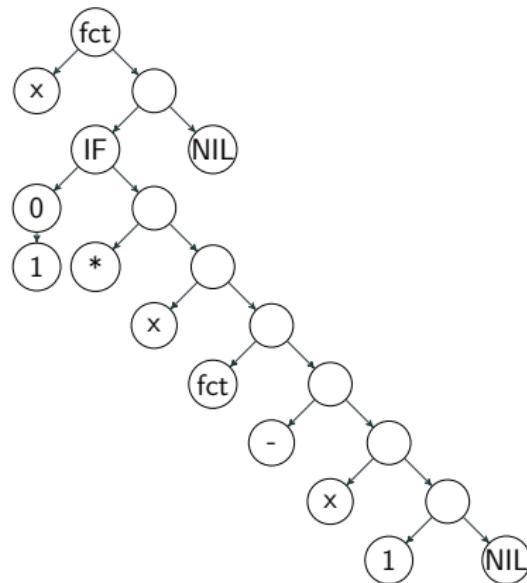


Computer Programs: Representations

Expression Tree

Example (S-Expression)

```
(defun fct (x)
  (if (zerop x)
      1
      (* x (fct (- x 1)))))
```



Computer Programs: Representations

Parse Tree

- ▶ **Abstract Syntax Tree (AST)** → abstract representation of a program's syntactic structure
 - ▶ Expression and program trees can be considered AST
- ▶ **Parse Tree** → syntactic structure that relates to some context-free grammar
- ▶ Compilers transform the **high-level code** of the respective language into a **parse tree**
 - ▶ Sometimes called a concrete syntax tree (CST)

Computer Programs: Representations

Parse Tree

Definition (Parse Tree)

A parse tree is a **tree-like hierarchical representation** of the derivation of a character string in accordance with a **formal grammar**.

Computer Programs: Representations

Parse Tree

- ▶ **Parsing** → Breaking down a syntactic element into its component parts
 - ▶ Describing their **syntactic components**
 - ▶ Performing syntactic analysis
 - ▶ Establishing hierarchical structures
- ▶ **Tree** → Suitable format for representing hierarchical structures
- ▶ **Grammar** → Set of rules that defines how to write valid statements (for the respective programming language)

Computer Programs: Representations

Parse Tree

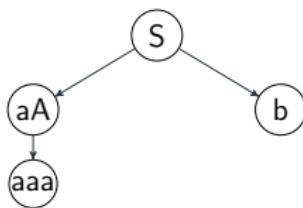
Example (Parse Tree)

Let G be a grammar with vocabulary $V = \{ A, B, a, b \}$, set of terminal symbols $T = \{a, b\}$, start symbol S and production rules $P = \{S \rightarrow aA, S \rightarrow b, A \rightarrow aa\}$

Computer Programs: Representations

Parse Tree

Example (Parse Tree)



$$S \rightarrow aA$$

$$S \rightarrow b$$

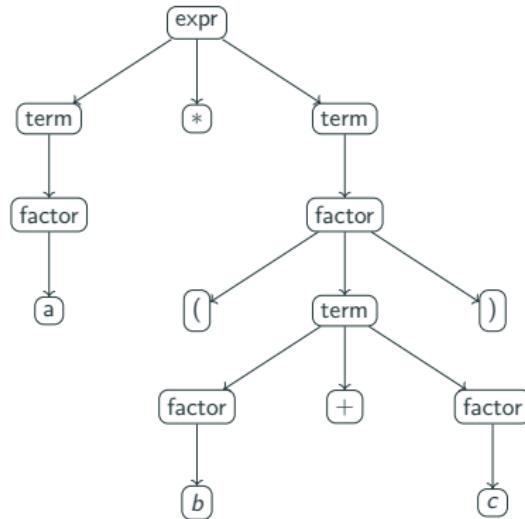
$$A \rightarrow aa$$

Computer Programs: Representations

Parse Tree

Example (Parse Tree)

a * (b + c)



Computer Programs: Representations

Parse Tree

▶ AST

- ▶ Do not represent every detail from the real syntax (abstract)
- ▶ Represents an **abstract syntactic structure** of a symbolic or language construct

▶ Parse Tree

- ▶ Respect the **syntactical** and **grammatical definitions** of the respective language
- ▶ Represents an **concrete syntactic** construct