CS 520

Theory of Programming Language

03/03 – 03/10, 2021

1. Motivation

1) Syntax

2) Semantics (Operational Semantics, Denotational Semantics)

3) Inference rules.

4) Binding.

Study them in predicate logic

integer expression.

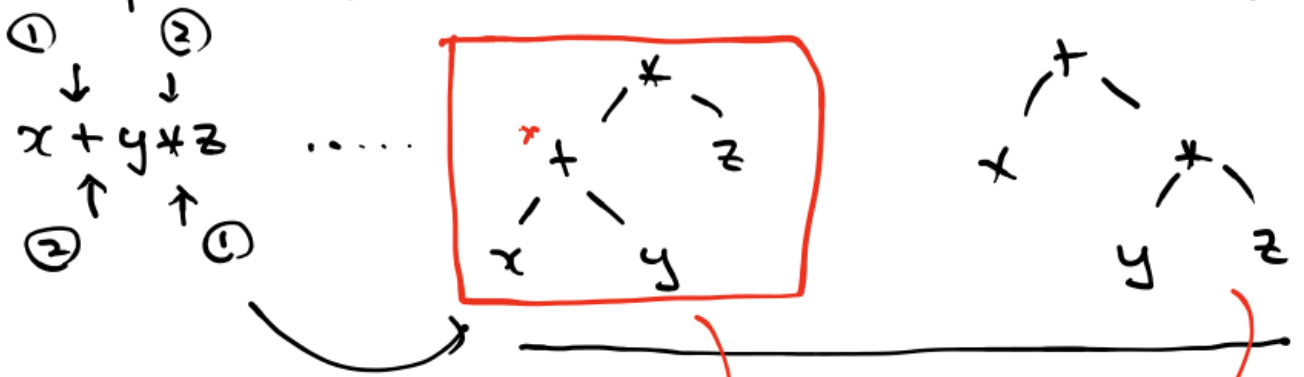
2. Integer expression

$x+3*y$ - $z \div 4 + 7$

integer division.

3. Syntax ···· abstract syntax , abstract phrase , abstract grammar.

abs. grammar. <intexp> ::= 1 | 2 | ···· | <var> | -<intexp> | <intexp> + <intexp>
- x
- ÷
- rem



abs phrase

abstract syntax.

4. Initial algebra view of abstract syntax

① algebra A ..... sets with operations & constants

$$\begin{bmatrix} (\mathbb{Z}, 0, +, -(\_)) \\ (\mathbb{R}_{>0}, 1, \times, (\_)^{-1}) \end{bmatrix}$$

(unary −)

signature S ..... type of an algebra.

$(u, t, c_1 : u, c_2 : t, c_3 : u, op_1 : u \times u \to u, op_2 : t \times t \to t)$

(≒ Java interface)

$A : S$

$A = (Set_u, Set_t, \underline{c_1} \in Set_u, \underline{c_2} : Set_t, \cdots, \underline{op_1} \in [Set_u \times Set_u \to Set_u], \cdots)$

(Impl of Java interface)

e.g. (a) $S_{group} = (u, e : u, \circ : u \times u \to u, (\_)^{-1} : u \to u)$

algebras of the signature S_{group}.

(b) $S_{intexp} = (u, 0 : u, 1 : u, \dots$ → constant integers.

$x : u, y : u, \dots$ → variables

$- : u \to u, + : u \times u \to u$).

abstract grammar.

$A_{FinTrees} = (FinTrees, 0, 1, new, \dots$

abstract syntax

abstract phrase.

$x, y, \dots$

$- : FinTrees \to FinTrees, \quad + : FinTrees^2 \to FinTrees$

$-(\Delta) = \overline{\Delta}$ $\quad +(\Delta, \Delta') = \Delta + \Delta'$

$\dots)$

What is so special about the $A_{FinTrees}$? → it is an initial algebra.

$A' = (2^{Var}, \{\}, \{\}, \dots$

$\{V_0 \mid V_0 \subseteq Var\}, \{x\}, \{y\}, \dots$

$, -(V_0) = V_0, \quad +(V_0, V_1) = V_0 \cup V_1)$

algebra for S_{intexp}.

FV.

Ex 1

$f : FinTree \to 2^{Var}$.

$f(\Delta) =$ {all variables appearing in Δ}.

Ex 2.

A, B : S initial.

$f : A \to B$ homo.

$g : B \to A$ homo.

By the initiality of B.

$\therefore f \circ g : B \to B$ homo.

$g \circ f : A \to A$ homo

But $id_B : B \to B$ homo.

$id_A : A \to A$ homo.

By the initiality of A.

(2) homomorphism ..... map bt two algebras of the same sig. s.t. the map preserves constants and operations.

$S = (u, c_1 : u, c_2 : u, \cdots$

$op_1 : u \times \cdots u \to u,\ op_2 : u \times \cdots \times u \to u, \cdots)$

$A_1, A_2 : S.$

$A_1 = (U_1, c_1^1 \in U_1, c_2^1 \in U_1, \cdots$

$op_1^1 \in [U_1 \times \cdots \times U_1 \to U_1], \cdots)$

$A_2 = (U_2, c_1^2 \in U_2, c_2^2 \in U_2, \cdots$

$op_1^2 \in [U_2 \times \cdots \times U_2 \to U_2], \cdots)$

A function $f \in [U_1 \to U_2]$ ($U_1 = |A_1|$, $U_2 = |A_2|$) is a homomorphism if:

preserve constants

1) for all constants c_j, $f(c_j^1) = c_j^2$.

preserve operations

2) for all operations op_i,

$$f(op_i^1(x_1, \dots, x_n)) = op_i^2(f(x_1), \dots, f(x_n))$$

($x_1, \dots, x_n \in U_1$; $f(x_1), \dots, f(x_n) \in U_2$)

③ An algebra A of a signature S is initial if:

for any algebra B of the same signature S,

there exists a unique homomorphism $f: A \to B$

(structural induction
syntax-directed definition for int. exp)

Claim. $A_{FmTrees} : Sintexp$ is initial.

Exercise ① $A' = (2^{Var}, \ldots) : Sintexp$.

Construct the unique hom. from $A_{FmTrees}$ to A'.

② A, B initial algebras of S.

$\Rightarrow \exists$ homomorphisms $f : A \to B$, $g : B \to A$ s.t

$g \circ f = id_A$ and $f \circ g = id_B$.

(A, B are isomorphic)

5. Syntax-directed definition. F (abstract phrase) = ...

FV (abstract phrase) = the set of free variables in Δ

Δ

map over abstract phrases. defined inductively. ← Inductive/recursive calls should be only over immediate subphrases.

$F(e) = \ldots$

$$\begin{cases} e \equiv n \\ e \equiv x \\ e \equiv -e' \\ e \equiv e' + e'' \\ e \equiv e' \times e'' \end{cases} \quad \ldots \text{ return } (\ldots F(e') \ldots F(e'') \ldots)$$

$$FV(e) \equiv \begin{cases} \{\} & \text{if } e \equiv n \\ \{x\} & \text{if } e \equiv x \\ FV(e') & \text{if } e \equiv -e' \\ FV(e') \cup FV(e'') & \text{if } e \equiv e' + e'' \\ FV(e') \cup FV(e'') & \text{if } e \equiv e' \times e'' \end{cases}$$

← well-defined because the syntax (abstract) is initial. (the syntax consists of finite trees)

$FV : FmTrees \to 2^{Var}$