1 Lecture

1.1 Foundations

In mathematics, the meaning of the equals sign is that the object on its left-hand side is actually the same object as the one on its right-hand side. As a result, any condition that holds true for the left-hand side object also holds true for the right-hand side one (and vice versa).

In particular:

$$\frac{\lambda x.y = \lambda z.y \quad \lambda z.y[y \mapsto x] = \lambda z.x}{\lambda x.y[y \mapsto x] = \lambda z.x} \tag{1}$$

1.2 Operational Semantics

- Usually well suited for reasoning about whole programs, less than ideal for reasoning about program fragments.
- Sometimes tends to overspecify the implementation of certain language features (e.g. evaluation order).
- Tends to put emphasis on syntax (rather than semantics) of the language.

1.3 Denotational Semantics of Expression Language

1.3.1 Syntax

$$\begin{aligned} \mathit{Expr} ::= {} & \mathit{Num} \mid \\ & \triangle \mathit{Expr} \mid \\ & \mathit{Expr} \odot \mathit{Expr} \end{aligned} \tag{2}$$

1.3.2 Semantics

Semantic domain: N.

$$[\![n]\!] = n \tag{3}$$

$$[\![\triangle e]\!] = [\![\triangle]\!]([\![e]\!]) \tag{4}$$

$$[\![\triangle]\!] = \lambda x. - x \text{ (i.e. unary minus)} \tag{5}$$

$$[\![e_1 \odot e_2]\!] = [\![\odot]\!]([\![e_1]\!], [\![e_2]\!]) \tag{6}$$

$$[\![\odot]\!] = \lambda x, y.x + y \text{ (i.e. plus)} \tag{7}$$

1.4 Denotational Semantics of Logic Formulae

1.4.1 Syntax

$$\begin{aligned} Formula ::= &\ true \mid \\ &\ false \mid \\ &\ \neg Formula \mid \\ &\ Formula\ BinaryConnective\ Formula \\ BinaryConnective ::= &\ \wedge \mid \vee \end{aligned} \tag{8}$$

1.4.2 Semantics

Semantic domain: $\{0, 1\}$.

$$[\![true]\!] = 1 \tag{9}$$

$$[\![false]\!] = 0 \tag{10}$$

$$[\![\neg f]\!] = [\![\neg]\!]([\![f]\!]) \tag{11}$$

$$[\![\neg]\!] = \lambda x.1 - x \tag{12}$$

$$[\![f_1\ c\ f_2]\!] = [\![c]\!]([\![f_1]\!], [\![f_2]\!]) \quad (c \in BinaryConnective) \tag{13}$$

$$[\![\wedge]\!] = \lambda x, y.x \cdot y \tag{14}$$

$$[\![\vee]\!] = \lambda x, y.(x + y) - (x \cdot y) \tag{15}$$

1.5 Denotational Semantics of Regular Expressions

1.5.1 Syntax

$$\begin{aligned} RegExp ::= &\ \emptyset \mid \\ &\ \epsilon \mid \\ &\ A \mid \\ &\ RegExp^* \mid \\ &\ RegExp\ BinOp\ RegExp \\ BinOp ::= &\ + \mid \cdot \end{aligned} \tag{16}$$

where A is a predefined set of characters (alphabet).

1.5.2 Semantics

Semantic domain: A^*.

$$[\![\emptyset]\!] = \{\} \tag{17}$$

$$[\![\epsilon]\!] = \{\epsilon\} \tag{18}$$

$$[\![a]\!] = \{a\} \tag{19}$$

$$[\![e^*]\!] = [\![^*]\!]([\![e]\!]) \tag{20}$$

$$[\![^*]\!] = \lambda L.\{l_1 \cdot \ldots \cdot l_n | n \in N \wedge l_i \in L\} \quad \text{(note: including } \epsilon) \tag{21}$$

$$[\![e_1\ o\ e_2]\!] = [\![o]\!]([\![e_1]\!], [\![e_2]\!]) \quad (o \in BinOp) \tag{22}$$

$$[\![+]\!] = \cup \tag{23}$$

$$[\![\cdot]\!] = \lambda A, B.\{a \cdot b | a \in A \wedge b \in B\} \tag{24}$$

1.6 Denotational Semantics of Lambda Calculus

1.6.1 Syntax

$$\begin{aligned} Expr ::= {} & X \mid \\ & \lambda X.Expr \mid \\ & Expr\ Expr \end{aligned} \tag{25}$$

1.6.2 Semantics

Semantic domains: $env = string \to function$, $fcn = fcn \to fcn$; notational conventions $e \in env, f, f' \in fcn, E, E' \in Expr$

$$[\![x]\!] = \lambda e.e(x) \tag{26}$$

$$[\![\lambda x.E]\!] = \lambda e.\lambda p.[\![E]\!](e[x \mapsto p]) \tag{27}$$

$$[\![E_1\ E_2]\!] = \lambda e.([\![E_1]\!](e))([\![E_2]\!](e)) \tag{28}$$

1.7 Relational Algebra

Semantic domain: n-ary relations; key operations:

- selection σ: $\sigma_{age>=18}$,
- projection π: $\pi_{name,age}$,
- union $\cup$, intersection $\cap$ and difference $\setminus$ and
- cross-product $\times$.

2 Seminar

1. Define a denotational semantics of the language of expressions with variables.

———

Semantic domain: $varName \to Num$.

$$[\![n]\!] = \lambda env.n \tag{29}$$

$$[\![v]\!] = \lambda env.env(v) \tag{30}$$

$$[\![\triangle]\!] = \lambda e.\lambda env. - e(env) \tag{31}$$

$$[\![\odot]\!] = \lambda e_1, e_2.\lambda env.e_1(env) + e_2(env) \tag{32}$$

2. Extend the semantics of regular expressions with the subtraction operator $(-)$: $r_1 - r_2$ denotes the set of words generated by r_1 and not generated by r_2.

———

$$[\![-]\!] = \setminus \tag{33}$$

3. For OI graduates: what could be an alternative semantic algebra for regular expressions?

———

Finite automata.

4. What is the denotation of $\lambda x.x$?

———

$$\begin{aligned} [\![\lambda x.x]\!] = \lambda env.\lambda p.[\![x]\!](env[x \mapsto p]) = \\ \lambda env.\lambda p.(\lambda env'.env'(x))(env[x \mapsto p]) = \\ \lambda env.\lambda p.(env[x \mapsto p])(x) = \lambda env.\lambda p.p \end{aligned} \tag{34}$$

5. Consider the following situation: relation $Student(name, age, schoolId)$ has n_1 records, relation $School(id, schoolName, location)$ has n_2 records. What is the cost of executing

$$\pi_{name,schoolName}(\sigma_{age>18 \wedge schoolId=id}(Student \times School))?$$

The cost of fetching one record from the permanent storage is k_1, the cost of processing one record is k_2. Are there any ways of speeding up the query?

$$\begin{aligned} k_1 \cdot n_1 \text{ (for fetching } Student) &+ \\ k_1 \cdot n_2 \text{ (for fetching } School) &+ \\ k_2 \cdot n_1 \cdot n_2 \text{ (for computing the cross product)} &+ \\ k_2 \cdot n_1 \cdot n_2 \text{ (for selection)} &+ \\ k_2 \cdot \alpha \cdot n_1 \cdot n_2 \text{ (for projection, } \alpha \text{ is the percentage of records retained by the selection)} \end{aligned} \tag{35}$$

Alternatively, the same result can be obtained by executing

$$\pi_{name,schoolName}(\sigma_{schoolId=id}(\sigma_{age>18}(Student) \times School)), \tag{36}$$

in which case, the cost of the query is

$$\begin{aligned} k_1 \cdot n_1 \text{ (for fetching } Student) &+ \\ k_1 \cdot \beta \cdot n_1 \text{ (for inner selection)} &+ \\ k_1 \cdot n_2 \text{ (for fetching } School) &+ \\ k_2 \cdot \beta \cdot n_1 \cdot n_2 \text{ (for computing the cross product)} &+ \\ k_2 \cdot \beta \cdot n_1 \cdot n_2 \text{ (for outer selection)} &+ \\ k_2 \cdot \alpha \cdot n_1 \cdot n_2 \text{ (for projection)}. \end{aligned} \tag{37}$$