Fundamentals of Programming Languages

Assignment 1

Statics and Dynamics

Mestrado em (Engenharia) Informática
Faculdade de Ciências da Universidade de Lisboa

2022/2023

1 Natural numbers and induction (20%)

We have introduced natural numbers by means of a grammar such as the one below.

$\mathsf{a} ::=$	*naturals:*
$\quad \mathsf{0}$	*constant zero*
$\quad \mathsf{succ(a)}$	*successor*

An alternative presentation uses rules featuring judgements of the form $\mathsf{a\ nat}$. The rules are as follows.

$$\textsc{N-Zero}\ \frac{}{\mathsf{0\ nat}} \qquad \textsc{N-Succ}\ \frac{\mathsf{a\ nat}}{\mathsf{succ(a)\ nat}}$$

A. Show that $\mathsf{succ(succ(succ(succ(succ(0)))))\ nat}$ by exhibiting an appropriate derivation.

Now consider the even and the odd predicate, also defined in rule format

$$\textsc{E-Zero}\ \frac{}{\mathsf{0\ even}} \qquad \textsc{E-Succ}\ \frac{\mathsf{a\ odd}}{\mathsf{succ(a)\ even}} \qquad \textsc{O-Succ}\ \frac{\mathsf{a\ even}}{\mathsf{succ(a)\ odd}}$$

B. Show that the successor of the successor of an odd number is odd. Then show that the successor of the successor of an odd even number is even.

After terms (natural numbers) and predicates (odd and even), also n-ary relations can be written in rule format. Consider the addition function on natural numbers. There are only two rules. One for the base case ($\mathsf{0}$) and the other for step ($\mathsf{succ(a)}$). The rule for the base case says that if $\mathtt{b}$ is a natural number, then $0 + \mathtt{b} = \mathtt{b}$.

That for the successor is for you to derive. To say that $\mathtt{a} + \mathtt{b} = \mathtt{c}$ we may use a judgement of the form $\mathsf{sum}(\mathtt{a}, \mathtt{b}, \mathtt{c})$.

C. Define a ternary relation $\mathsf{sum}(\mathtt{a}, \mathtt{b}, \mathtt{c})$ by means of two rules. The relation should be defined on natural numbers only: we do not want to be able to conclude, for example, that $\mathsf{sum}(\mathtt{succ}(0), \mathtt{fish}, \mathtt{succ}(\mathtt{fish}))$.

D. Using rule induction show that

1. For every $\mathtt{a}$ nat and $\mathtt{b}$ nat, there is a $\mathtt{c}$ nat such that $\mathsf{sum}(\mathtt{a}, \mathtt{b}, \mathtt{c})$;
2. $\mathtt{c}$ nat is unique.

Together these results say that sum is a total function.

E. Give an inductive definition of the judgement $\mathsf{max}(\mathtt{a}, \mathtt{b}, \mathtt{c})$ where $\mathtt{a}$ nat, $\mathtt{b}$ nat and $\mathtt{c}$ nat, with the meaning that $\mathtt{c}$ is the largest of $\mathtt{a}$ and $\mathtt{b}$.

2 Natural numbers, strings and typing (20%)

We now define a language of natural numbers, strings and operations on these.

$\mathtt{e} ::=$	*terms:*
0	*constant zero*
$\mathtt{succ}(\mathtt{e})$	*successor*
$\mathtt{e} + \mathtt{e}$	*addition*
ε	*empty string*
$\mathtt{Ae}$	*non-empty string*
$\mathtt{Be}$	*non-empty string*
$\mathsf{len}(\mathtt{e})$	*string length*

Natural numbers and their only operation $(+)$ are as defined in Section 1. Strings are built from ε, the empty string, and by prefixing a given string by letters $\mathtt{A}$ and $\mathtt{B}$. For simplicity strings are built from letters A and B alone. Examples of strings are ε, $\mathtt{A}\varepsilon$, $\mathtt{B}\varepsilon$, $\mathtt{BABA}\varepsilon$ and $\mathtt{ABBABABABAABABABAB}\varepsilon$. The length of a string, len, denotes a natural number. For example, $\mathsf{len}(\mathtt{BABA}\varepsilon)$ denotes the natural number $\mathtt{succ}(\mathtt{succ}(\mathtt{succ}(\mathtt{succ}(0))))$.

A. Suggest a grammar for types appropriate to type terms. Use metavariable $\mathtt{T}$ to denote arbitrary types.

B. Present a type system assigning types to terms. Use judgements of the from $\mathtt{e} : \mathtt{T}$.

C. Exhibit a term that is typable according to the type system just defined and another that is untypable. Show that the terms you have selected are indeed typable and untypable.

D. Show the Unicity of Typing lemma: For every term $\mathtt{e}$ there is at most one type $\mathtt{T}$ such that $\mathtt{e} : \mathtt{T}$.

3 Evaluation and type safety (30%)

The dynamics of the term language is given by a transition system whose states are terms. All states are initial. Final states are *values*. Values form a subset of terms and include the natural numbers and the strings.

A. Define the predicate is-value by means of rules. Use a judgement of the form $\mathsf{e}\ \mathsf{val}$.

B. Define a *one-step* evaluation relation that computes the addition and length operations. Use a judgement of the form $\mathtt{e} \to \mathtt{e}$. Take call-by-value for your reduction strategy: first evaluate the parameter(s), and only then apply the operator. For example, if $\mathtt{e}$ is a value, then $\mathtt{0} + \mathtt{e}$ should evaluate to $\mathtt{e}$. But if $\mathtt{e}$ is *not* a value (and $\mathtt{e}$ is typable), then $\mathtt{e} \to \mathtt{e}'$, for some $\mathtt{e}'$. In this case $\mathtt{0} + \mathtt{e}$ should evaluate to $\mathtt{0} + \mathtt{e}'$.

C. Use your rules to show the following transitions

$$\begin{aligned}
\mathtt{succ(0)} + \mathtt{succ(0)} &\to \mathtt{succ(0} + \mathtt{succ(0))} \\
\mathtt{succ(0} + \mathtt{succ(0))} &\to \mathtt{succ(succ(0))} \\
\mathtt{len(AB}\varepsilon\mathtt{)} &\to \mathtt{succ(len(B}\varepsilon\mathtt{))} \\
\mathtt{succ(len(B}\varepsilon\mathtt{))} &\to \mathtt{succ(succ(len(}\varepsilon\mathtt{)))} \\
\mathtt{len(A}\varepsilon\mathtt{)} + \mathtt{succ(succ(0))} &\to \mathtt{succ(len(A}\varepsilon\mathtt{))} + \mathtt{succ(succ(0))}
\end{aligned}$$

D. Show that if $\mathtt{e} : \mathtt{T}$ and no evaluation rule applies to $\mathtt{e}$, then $\mathsf{e}\ \mathsf{val}$.

E. Show the Progress theorem. If $\mathtt{e} : \mathtt{T}$, then either $\mathsf{e}\ \mathsf{val}$ or there exists $\mathtt{e}'$ such that $\mathtt{e} \to \mathtt{e}'$.

F. Show the Preservation theorem. If $\mathtt{e} : \mathtt{T}$ and $\mathtt{e} \to \mathtt{e}'$, then $\mathtt{e}' : \mathtt{T}$.

4 Implementation (30%)

A. Write a data declaration to describe terms. Write Haskell values for the five terms at the left of the arrow in Section 3.**C**.

B. Write predicate val and function eval1. Use these functions to define a function eval that computes the normal form of a term. Predicates must be complete. Functions may yield exceptions when in presence of non-typable terms.

B. Write function typeof that computes the type of a given term if this exists, and raises an exception otherwise.

Due date: October 24, 2022, 23:59