Teorija programskih jezikov: 1. izpit

31. januar 2020

1. naloga (20 točk)

V λ-računu lahko vsako naravno število n predstavimo s *Churchevim naravnim številom*

$$e_n = \lambda s.\lambda z.\underbrace{s(s(\cdots(s}_{n}\ z)\cdots))$$

a) Zapišite vse korake v evalvaciji izraza $(e_2\,(\lambda n.2 * n))\,1$.

b) Napišite funkcijo f, ki vsako Churchevo naravno število pretvori v običajno naravno število. Veljati mora torej $f\ e_n \rightsquigarrow^* n$.

c) Izračunajte najbolj splošen tip izraza e_2.

2. naloga (20 točk)

Operacijsko semantiko programskega jezika IMP z ukazi

$$c ::= \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \texttt{while}\ b\ \texttt{do}\ c \mid c_1; c_2 \mid \ell := e \mid \texttt{skip}$$

smo običajno podali z velikimi koraki, torej z relacijami

$$s, e \Downarrow n, \qquad s, b \Downarrow p, \qquad s, c \Downarrow s',$$

vendar bi lahko podobno storili tudi z malimi koraki

$$s, e \rightsquigarrow e', \qquad s, b \rightsquigarrow b', \qquad s, c \rightsquigarrow s', c'$$

Pri tem aritmetični izrazi s koraki končajo, ko dosežejo število n, Booleovi izrazi takrat, ko dosežejo logično vrednost p, ukazi pa takrat, ko dosežejo $s, \texttt{skip}$.

a) Podajte pravila, ki določajo relacijo $s, c \rightsquigarrow s', c'$. Pravil za aritmetične in Booleove izraze vam ni treba podajati.

b) Jezik IMP razširimo s prepletenim izvajanjem $c_1 \leftrightharpoons c_2$, ki najprej izvede en korak ukaza c_1, nato en korak ukaza c_2, nato naslednji korak ukaza c_1 in tako naprej. Če je ukaz, ki je na vrsti, z izvajanjem končal, do konca izvedemo preostali ukaz. Zapišite dodatna pravila, s katerimi morate razširiti operacijsko semantiko.

3. naloga (20 točk)

Dana naj bo domena D in naj bo $\mathit{fix} : [D \to D] \to D$ preslikava, ki vsaki zvezni preslikavi $f : D \to D$ priredi njeno najmanjšo fiksno točko $\mathit{fix}(f) \in D$. Dokažite, da je preslikava fix zvezna.

4. naloga (20 točk)

Imejmo λ-račun samo z Booleovimi vrednostmi in funkcijami

$$e ::= x \mid \texttt{true} \mid \texttt{false} \mid \texttt{if}\, e\, \texttt{then}\, e_1\, \texttt{else}\, e_2 \mid \lambda x.e \mid e_1\, e_2$$

ter z običajnimi pravili za neučakano operacijsko semantiko malih korakov in določanje tipov. Ker v jeziku nimamo rekurzije, vsak program, ki ima tip, tudi konvergira, vendar preprosta indukcija žal ne zadošča za dokaz.[1]

Namesto tega za vsak tip A definiramo množico *dobrih izrazov* $\mathscr{D}_A$ kot

$$\mathscr{D}_{\texttt{bool}} = \{e \mid (e \rightsquigarrow^* \texttt{true}) \vee (e \rightsquigarrow^* \texttt{false})\} \qquad \mathscr{D}_{A_1 \to A_2} = \{e \mid \forall e' \in \mathscr{D}_{A_1}.e\, e' \in \mathscr{D}_{A_2}\}$$

in dokaz razbijemo na tri dele...

a) Dokažite, da vsi dobri izrazi konvergirajo, torej da za vsak tip A in vsak $e \in \mathscr{D}_A$ obstaja vrednost v, da velja $e \rightsquigarrow^* v$.

b) Dokažite, da za vsak tip A iz $e \rightsquigarrow e'$ in $e' \in \mathscr{D}_A$ sledi $e \in \mathscr{D}_A$.

c) Dokažite, da za vsak izraz $x_1 : A_1, \ldots, x_n : A_n \vdash e : A$ in vse vrednosti $v_1 \in \mathscr{D}_{A_1}, \ldots, v_n \in \mathscr{D}_{A_n}$ velja tudi $e[v_1/x_1, \ldots, v_n/x_n] \in \mathscr{D}_A$. Torej: vsak izraz, ki ima tip, je dober, če vse proste spremenljivke zamenjamo z dobrimi vrednostmi.

[1] Za čast in slavo lahko poiščete izraz e, ki divergira, torej da obstaja neskončno zaporedje $e \rightsquigarrow e_1 \rightsquigarrow e_2 \rightsquigarrow \cdots$

Theory of programming languages: first exam

31 January 2020

Question 1 (20 marks)

In λ-calculus, each natural number n can be represented by a *Church natural number*

$$e_n = \lambda s.\lambda z.\underbrace{s(s(\cdots(s}_{n}\ z)\cdots))$$

a) Write down all the steps in the evaluation of $(e_2\,(\lambda n.2 * n))\,1$.

b) Write a function f, which converts every Church natural number to an ordinary natural number, ie. $f\ e_n \rightsquigarrow^* n$ must hold.

c) Compute the most general type of the expression e_2.

Question 2 (20 marks)

Operational semantics of the IMP language with commands

$$c ::= \texttt{if}\ b\ \texttt{then}\ c_1\ \texttt{else}\ c_2 \mid \texttt{while}\ b\ \texttt{do}\ c \mid c_1; c_2 \mid \ell := e \mid \texttt{skip}$$

is usually given with big steps, ie. with relations

$$s, e \Downarrow n, \qquad s, b \Downarrow p, \qquad s, c \Downarrow s',$$

though we could do the same with small steps

$$s, e \rightsquigarrow e', \qquad s, b \rightsquigarrow b', \qquad s, c \rightsquigarrow s', c'$$

In this case, arithmetic expressions end evaluation when they reach a number n, Boolean expressions when they reach a truth value p, and the commands when they reach $s, \texttt{skip}$.

a) Give the rules that determine the relation $s, c \rightsquigarrow s', c'$. You do not need to give the rules for arithmetic and Boolean expressions.

b) We extend IMP with interleaving $c_1 \leftrightharpoons c_2$, which first executes one step of c_1, then one step of c_2, then the next step of c_1 and so on. If the command in turn has finished the execution, we evaluate the remaining command. Write the additional rules with which you need to extend the operational semantics.

Question 3 (20 marks)

Let D be a domain and let $\mathit{fix}\colon [D \to D] \to D$ map each continuous function $f : D \to D$ into its least fixed point $\mathit{fix}(f) \in D$. Prove that fix is continuous.

Question 4 (20 marks)

Take λ-calculus with booleans and functions

$$e ::= x \mid \texttt{true} \mid \texttt{false} \mid \texttt{if}\, e\, \texttt{then}\, e_1\, \texttt{else}\, e_2 \mid \lambda x.e \mid e_1\, e_2$$

and usual rules for eager small-step operational semantics and typing judgements. Since the language does not feature recursion, each well-typed expression converges, though simple induction is not sufficient for the proof,.[2]

Instead, we define a set of *good expressions* $\mathscr{D}_A$ for each type A as

$$\mathscr{D}_{\texttt{bool}} = \{e \mid (e \rightsquigarrow^* \texttt{true}) \vee (e \rightsquigarrow^* \texttt{false})\} \qquad \mathscr{D}_{A_1 \to A_2} = \{e \mid \forall e' \in \mathscr{D}_{A_1}. e\, e' \in \mathscr{D}_{A_2}\}$$

and split the proof into three parts...

a) Prove that all good expressions converge, ie. for each type A and each $e \in \mathscr{D}_A$ there exists a value v, such that $e \rightsquigarrow^* v$.

b) Prove that $e \rightsquigarrow e'$ and $e' \in \mathscr{D}_A$ imply $e \in \mathscr{D}_A$ for each type A.

c) Prove that for each expression $x_1 : A_1, \ldots, x_n : A_n \vdash e : A$ and all values $v_1 \in \mathscr{D}_{A_1}, \ldots, v_n \in \mathscr{D}_{A_n}$ we have $e[v_1/x_1, \ldots, v_n/x_n] \in \mathscr{D}_A$. Ie. each well-typed expression is good, if we replace all free variables with good values.

[2] For honor and glory you can find an expression e, which diverges, ie. there exists an infinite sequence $e \rightsquigarrow e_1 \rightsquigarrow e_2 \rightsquigarrow \cdots$