# Systemy Typów 2023

Lista zadań nr 1

Na zajęcia 11 października 2023

**Uwaga:** zadania 5, 8, 9 i 10 można zrobić w Coqu/Agdzie relatywnie niewielkim (jak na walkę z asystentem dowodzenia) nakładem pracy. Za takie rozwiązanie można dostać podwójną liczbę punktów!

**Zadanie 1.** Na wykładzie zdefiniowaliśmy (częściowe) podstawienie dla pratermów. Pokaż, że można podnieść tą definicję do podstawienia dla termów. W tym celu pokaż, że:

- wynik nie zależy od wyboru reprezentanta klasy abstrakcji, tzn. dla dowolnych $t_1 \equiv_\alpha t_2$ zachodzi $t_1\{t/x\} \equiv_\alpha t_2\{t/x\}$, o ile te operacje są zdefiniowane;
- w każdej każdej klasie abstrakcji jest reprezentant, dla którego podstawienie jest zdefiniowane, tzn. dla dowolnych $t_1$, $t$ oraz $x$ istnieje $t_2 \equiv_\alpha t_1$, takie, że $t_2\{t/x\}$ jest zdefiniowane.

Czy jak usuniemy klauzulę $(\lambda x.t')\{t/x\} = t'$ z definicji podstawienia pratermów, to powyższe własności dalej będą zachodzić?

**Zadanie 2.** Jednym ze sposobów formalnego ujęcia konwencji Barendregta jest zasada indukcji bazująca na kwantyfikacji koskończonej. Nie $\mathcal{P}_{\text{fin}}(\mathrm{Var})$ oznacza zbiór skończonych podzbiorów zbioru $\mathrm{Var}$. Niech $Q$ będzie predykatem termów (unarną relacją na termach) takim, że:

- $\forall x \in \mathrm{Var}.\, Q(x)$;
- $\forall x \in \mathrm{Var}.\, \forall t \in \mathrm{PreTerm}.$
  $(\forall L \in \mathcal{P}_{\text{fin}}(\mathrm{Var}).\, \exists x' \in \mathrm{Var} \setminus L.\, \exists t' \in \mathrm{PreTerm}.\, \lambda x.t \equiv_\alpha \lambda x'.t' \wedge Q(t\{x'/x\}))$
  $\Rightarrow P(\lambda x.t)$;
- $\forall t_1, t_2 \in \mathrm{PreTerm}.\, Q(t_1) \wedge Q(t_2) \Rightarrow Q(t_1\ t_2)$.

Korzystając ze zwykłej zasady indukcji dla pratermów, pokaż, że dla dowolnego termu $t$ zachodzi $Q(t)$.

**Zadanie 3.** Pokaż, że $t\{t_1/x\}\{t_2/y\} = t\{t_2/y\}\{t_1\{t_2/y\}/x\}$ o ile $x \neq y$ oraz $x \notin \mathrm{fv}(t_2)$. W tym zadaniu pracujemy już na termach. Postaraj się użyć zasady indukcji z poprzedniego zadania.

**Zadanie 4.** Zdefiniuj w ulubionym silnie typowanym języku programowania (preferowane: OCaml, Coq, SML, Haskell, Agda, Idris) typ reprezentujący składnię rachunku lambda. Twoja definicja powinna odzwierciedlać pokazaną na wykładzie konstrukcję indeksowanych rodzin zbiorów, więc w szczególności typ termów powinien mieć jeden parametr typowy opisujący zmienne potencjalnie wolne. Zdefiniuj podstawienie ($t\{t'\}$) oraz wszystkie potrzebne do tego operacje (np. `fmap` i `bind`).

**Uwaga:** operacje `fmap` i `bind` wymagają rekursji polimorficznej, co w OCamlu oznacza napisanie jakiś okropnych anotacji typowych.

**Zadanie 5.** Pokaż następujące własności składni abstrakcyjnej reprezentowanej przy pomocy indeksowanych rodzin zbiorów ($f$ to przemianowania, natomiast $g$ to podstawienia).

1. $\mathsf{id}^{\uparrow} = \mathsf{id}$,
2. $(f_1 \circ f_2)^{\uparrow} = f_1^{\uparrow} \circ f_2^{\uparrow}$,
3. $\mathsf{id}^{\dagger} = \mathsf{id}$,
4. $(f_1 \circ f_2)^{\dagger} = f_1^{\dagger} \circ f_2^{\dagger}$,
5. $f^{\uparrow} \circ \mathsf{s} = \mathsf{s} \circ g$,
6. $g^{\Uparrow} \circ \mathsf{s} = \mathsf{s}^{\dagger} \circ f$,
7. $(f^{\dagger} \circ g)^{\Uparrow} = f^{\uparrow\dagger} \circ g^{\Uparrow}$,
8. $\eta^{\Uparrow} = \eta$,
9. $(g \circ f)^{\Uparrow} = g^{\Uparrow} \circ f^{\uparrow}$,
10. $(g_1^* \circ g_2)^{\Uparrow} = g_1^{\Uparrow *} \circ g_2^{\Uparrow}$,
11. jeśli $g_1 \circ f_1 = f_2^{\dagger} \circ g_2$ to $g_1^* \circ f_1^{\dagger} = f_2^{\dagger} \circ g_2^*$,
12. $\eta^* = \mathsf{id}$,
13. $g_1^* \circ g_2^* = (g_1^* \circ g_2)^*$,
14. $(t \triangleleft g)^* \circ \mathsf{s}^{\dagger} = g^*$,
15. $(\mathsf{s}^{\dagger}(t))\{t'\} = t$,
16. $g^{\Uparrow} = \eta(0) \triangleleft (\mathsf{s}^{\dagger} \circ g)$,
17. $f^{\dagger} \circ (t \triangleleft g) = f^{\dagger}(t) \triangleleft (f^{\uparrow\dagger} \circ g)$,
18. $g_1^* \circ (t \triangleleft g_2) = g_1^*(t) \triangleleft (g_1^{\Uparrow *} \circ g_2)$.

Jeśli decydujesz się rozwiązać to zadanie w Coqu lub Agdzie, to będziesz musiał mówić o równości funkcji. Możesz założyć ekstensjonalność funkcji[1], lub (preferowane) zawsze dawać funkcjom wszystkie parametry. Np. punkt 11 przyjmie postać: jeśli dla dowolnego $x$ zachodzi $g_1(f_1(x)) = f^{\dagger}(g_2(x))$ to dla dowolnego $t$ mamy $g_1^*(f_1^{\dagger}(t)) = f_2^{\dagger}(g_2^*(t))$. Punkt 3 i 12 może wymagać drobnego uogólnienia na dowolną funkcję, która dla każdego parametru zachowuje się jak odpowiednio identyczność i $\eta$.

**Zadanie 6.** Pokaż, równoważność definicji zbioru termów poprzez klasy abstrakcji $\alpha$-równoważności (zbiór $\mathrm{Term}$) oraz poprzez indeksowane rodziny zbiorów (zbiór $\mathrm{Term}_{\mathrm{Var}}$, gdzie $\mathrm{Var}$ to zbiór wszystkich zmiennych) są równoważne. W tym celu zdefiniuj odpowiedni izomorfizm (bijekcję) między tymi zbiorami. Twój izomorfizm powinien działać homomorficznie na wszystkich konstrukcjach z języka. Czy umiesz taką homomorficzność zdefiniować? A czy umiesz udowodnić?

**Uwaga:** zostawiłem to zadanie, bo nie jestem pewny, czy umiem je elegancko rozwiązać. Chętnie o tym zadaniu porozmawiam, ale też nie oczekuję, że ktoś je zrobi.

**Zadanie 7.** Zdefiniuj semantykę dużych kroków, czyli relację $e \Downarrow v$ mówiącą o tym, że wyrażenie $e$ oblicza się do wartości $v$, lub inaczej $e \longrightarrow^* v$. Twoja definicja nie powinna odwoływać się ani do semantyki małych kroków, ani do przechodniego domknięcia relacji.

**Zadanie 8.** Pokaż, że Twoje semantyka z poprzedniego zadania jest równoważna semantyce małych kroków. W tym celu pokaż, że $e \Downarrow v$ wtedy i tylko wtedy gdy $e \longrightarrow^* v$, gdzie $\longrightarrow^*$ oznacza zwrotne i przechodnie domknięcie relacji $\longrightarrow$.

**Zadanie 9.** Pokaż równoważność strukturalnej i redukcyjnej semantyki operacyjnej rachunku lambda CBV.

**Zadanie 10.** Pokaż, że semantyka redukcyjna rachunku lambda CBV jest deterministyczna, tzn. jeżeli $e \longrightarrow e_1$ oraz $e \longrightarrow e_2$ to $e_1 = e_2$. By to pokazać, zdefiniuj *potencjalne redeksy* (które będziemy oznaczać metazmienną $r$) i pokaż następujące własności:

1. żadna wartość nie jest postaci $E[r]$;
2. każde wyrażenie jest postaci $E[r]$ lub jest wartością;
3. jeśli $E_1[r_1] = E_2[r_2]$ to $E_1 = E_2$ oraz $r_1 = r_2$ (własność ta nazywana jest *jednoznacznością rozkładu*).

Jaka reprezentacja kontekstów ewaluacyjnych (*outside-in* czy *inside-out*) jest wygodniejsza do pokazania tych własności?

[1] `Coq.Logic.FunctionalExtensionality`