

1 Punktacja

Najważniejszym elementem zadania było równanie dla deklaracji funkcji i pasujące do niego równania dla wywołania funkcji oraz instrukcji `return` i `again`. Za tę część można było uzyskać **6 pkt**. Po **1 pkt** można było uzyskać za poprawne klauzule dla standardowych wyrażeń, standardowych instrukcji, programu i deklaracji. Ponieważ jednak te równania były raczej standardowe i pojawiały się wielokrotnie na ćwiczeniach, więc wszystkie pomyłki/błędy/literówki były w tej części bezlitośnie karane. Szczegóły poniżej.

1.1 Standardowe klauzule dla deklaracji

Za poprawne równanie deklaracji stałej **oraz** poprawną strukturę deklaracji funkcji można było uzyskać łącznie **1 punkt**. Równania dla złożenia deklaracji i deklaracji zmiennych miały dość standardową postać:

$$\mathcal{D}[d_1; d_2] \varrho c_d = \mathcal{D}[d_1] \varrho (\lambda \varrho'. \mathcal{D}[d_2] \varrho' c_d)$$

$$\mathcal{D}[\text{int } x := e] \varrho c_d = \mathcal{E}[e] \varrho (\lambda n. \lambda s. \text{let } l = \text{newloc } s \text{ in } c_d \varrho[x \mapsto l] s[l \mapsto n])$$

Częstym błędem w powyższym równaniu było przeniesienie λs przed wyliczenie wyrażenia, co prowadzi do niepoprawnego pod względem typów równania:

$$\mathcal{D}[\text{int } x := e] \varrho c_d = \lambda s. \mathcal{E}[e] \varrho (\lambda n. \text{let } l = \text{newloc } s \text{ in } c_d \varrho[x \mapsto l] s[l \mapsto n])$$

Klauzula dla deklaracji funkcji powinna mieć strukturę następującej postaci:

$$\begin{aligned} \mathcal{D}[\text{func } f(x) d; i \text{ endfunc}] \varrho c_d = \\ \text{let} \\ \quad F = \dots \\ \text{in} \\ \quad c_d (\varrho[f \mapsto F]) \end{aligned}$$

Częstym błędem było używanie kontynuacji c_d w charakterze kontynuacji ciała funkcji lub w ogóle pominięcie części po `in`.

1.2 Deklaracja funkcji

Za w pełni poprawną klauzulę deklaracji funkcji i pasujące do niej równania dla wywołania, `return` i `again` można było otrzymać łącznie **6 punktów**. Na te punkty składają się następujące elementy.

1.2.1 Przepływ sterowania

Funkcje w zadaniu miały mieć statyczne wiązanie identyfikatorów, parametr przekazywany przez referencję, a `return e` powinien kończyć wykonanie funkcji przekazując w wyniku wartość wyrażenia e . Domyślną wartością przekazywaną w wyniku miało być 0.

Równania opisujące takie działanie funkcji są w miarę standardowe:

$$\mathcal{I}[\text{return } e] \varrho c = \mathcal{E}[e] \varrho (\varrho \text{ret})$$

$$\mathcal{E}[\text{call } f(x)] \varrho c_e = (\varrho f) (\varrho x) c_e$$

oraz równanie dla deklaracji (na razie pominięto w nim niektóre istotne elementy):

$$\begin{array}{l}
\mathcal{D}[\mathbf{func} \ f(x) \ d; i \ \mathbf{endfunc}] \ \varrho \ c_d = \\
\text{let} \\
\quad F = \lambda loc. \ \lambda c_e. \ \mathcal{D}[d] \ \varrho(\lambda \varrho'. \ \mathcal{I}[i] \ \varrho'[ret \mapsto c_e] \ (c_e \ 0)) \\
\text{in} \\
\quad c_d \ (\varrho[f \mapsto F])
\end{array}$$

Za powyższy schemat można było uzyskać łącznie **1 punkt**. Typowym błędem było pominięcie deklaracji obiektów lokalnych (wyliczenia wartości $\mathcal{D}[d]$). Inny typowy błąd, to traktowanie funkcji dla deklaracji jak funkcji w semantyce bezpośredniej:

$$\begin{array}{l}
\dots \\
\quad F = \lambda loc. \ \lambda c_e. \ \text{let} \ \varrho' = \mathcal{D}[d] \ \varrho \ \text{in} \ \mathcal{I}[i] \ \varrho'[ret \mapsto c_e] \ (c_e \ 0) \\
\dots
\end{array}$$

1.2.2 Właściwa widoczność parametru formalnego

W powyższym równaniu nie związane jeszcze właściwej lokacji loc z parametrem formalnym x . Parametr ten musi być już widoczny przy przetwarzaniu deklaracji (jak w przykładzie 3 z treści: deklaracja `int d := x` wewnątrz `func f(x)`). Należy więc uzupełnić równanie tak:

$$\begin{array}{l}
\mathcal{D}[\mathbf{func} \ f(x) \ d; i \ \mathbf{endfunc}] \ \varrho \ c_d = \\
\text{let} \\
\quad F = \lambda loc. \ \lambda c_e. \ \mathcal{D}[d] \ \varrho[x \mapsto loc](\lambda \varrho'. \ \mathcal{I}[i] \ \varrho'[ret \mapsto c_e] \ (c_e \ 0)) \\
\text{in} \\
\quad c_d \ (\varrho[f \mapsto F])
\end{array}$$

a nie tak:

$$\begin{array}{l}
\mathcal{D}[\mathbf{func} \ f(x) \ d; i \ \mathbf{endfunc}] \ \varrho \ c_d = \\
\text{let} \\
\quad F = \lambda loc. \ \lambda c_e. \ \mathcal{D}[d] \ \varrho(\lambda \varrho'. \ \mathcal{I}[i] \ \varrho'[x \mapsto loc, ret \mapsto c_e] \ (c_e \ 0)) \\
\text{in} \\
\quad c_d \ (\varrho[f \mapsto F])
\end{array}$$

Zapewnienie właściwej widoczności parametru formalnego było warte **1 punkt**

1.2.3 Wywołania rekurencyjne

W treści funkcji należało zapewnić możliwość jej rekurencyjnego wywołania. Powyższe równania nie dają rekurencji, bo przy wykonywaniu treści funkcji (instrukcji i) identyfikator funkcji f nie znajduje się w środowisku.

Zapewnienie rekurencji dawało **1 punkt**.

1.2.4 Właściwa widoczność identyfikatora funkcji

Tę rekurencję należało także właściwie zrealizować. Pokazywał to przykład 2 z treści, w którym dochodziło do przesłonięcia identyfikatora zewnętrznej funkcji, funkcją zdefiniowaną lokalnie (i w konsekwencji wywołanie funkcji lokalnej a nie rekurencję). Właściwe równanie wygląda więc tak:

$$\begin{array}{l}
\mathcal{D}[\mathbf{func} \ f(x) \ d; i \ \mathbf{endfunc}] \ \varrho \ c_d = \\
\text{let} \ F = \lambda loc. \ \lambda c_e. \ \mathcal{D}[d] \ \varrho[x \mapsto loc, f \mapsto F](\lambda \varrho'. \ \mathcal{I}[i] \ \varrho'[ret \mapsto c_e] \ (c_e \ 0)) \\
\text{in} \ c_d \ (\varrho[f \mapsto F])
\end{array}$$

a nie tak:

$$\begin{array}{l}
\mathcal{D}[\mathbf{func} \ f(x) \ d; i \ \mathbf{endfunc}] \ \varrho \ c_d = \\
\text{let} \\
\quad F = \lambda loc. \ \lambda c_e. \ \mathcal{D}[[d]] \ \varrho(\lambda \varrho'. \ \mathcal{I}[[i]] \ \varrho'[x \mapsto loc, f \mapsto F, ret \mapsto c_e] \ (c_e \ 0)) \\
\text{in} \\
\quad c_d \ (\varrho[f \mapsto F])
\end{array}$$

W tym drugim przypadku wprowadzenie wiązania $f \mapsto F$ do ϱ' przesłania (błędnie) ewentualną redeklarację f w d .

Poprawne rozwiązanie tego problemu było warte **1 punkt**.

1.2.5 again

I wreszcie **again**, warty **2 punkty**. Wprowadzając do środowiska odpowiednie wiązanie należało zadbać, aby przy **again** nie były przetwarzane ponownie deklaracje. Zatem rozwiązanie:

$$\begin{array}{l}
\mathcal{D}[\mathbf{func} \ f(x) \ d; i \ \mathbf{endfunc}] \ \varrho \ c_d = \\
\text{let} \\
\quad F = \lambda loc. \ \lambda c_e. \ \mathcal{D}[[d]] \ \varrho[x \mapsto loc, f \mapsto F](\lambda \varrho'. \ \mathcal{I}[[i]] \ \varrho'[ret \mapsto c_e, \mathbf{again} \mapsto \\
\quad \quad \quad F \ loc \ c_e] \ (c_e \ 0)) \\
\text{in} \\
\quad c_d \ (\varrho[f \mapsto F])
\end{array}$$

jest błędne. Prawidłowa deklaracja wygląda tak:

$$\begin{array}{l}
\mathcal{D}[\mathbf{func} \ f(x) \ d; i \ \mathbf{endfunc}] \ \varrho \ c_d = \\
\text{let} \\
\quad F = \lambda loc. \ \lambda c_e. \ \mathcal{D}[[d]] \ \varrho[f \mapsto F, x \mapsto loc] \ (\lambda \varrho'. \\
\quad \quad \text{let} \\
\quad \quad \quad c = \mathcal{I}[[i]] \ \varrho'[\mathbf{again} \mapsto c, ret \mapsto c_e] \ (c_e \ 0) \\
\quad \quad \text{in} \\
\quad \quad \quad c) \\
\text{in} \\
\quad c_d \ (\varrho[f \mapsto F])
\end{array}$$

I jeszcze wersja dla „twardzieli” z punktami stałymi:

$$\begin{array}{l}
\mathcal{D}[\mathbf{func} \ f(x) \ d; i \ \mathbf{endfunc}] \ \varrho \ c_d = \\
\quad c_d \ (\varrho[f \mapsto \text{Fix}(\lambda g. \ \lambda loc. \ \lambda c_e. \ \mathcal{D}[[d]] \ \varrho[f \mapsto g, x \mapsto loc] \ (\lambda \varrho'. \\
\quad \quad \quad \text{Fix}(\lambda c. \ \mathcal{I}[[i]] \ \varrho'[\mathbf{again} \mapsto c, ret \mapsto c_e] \ (c_e \ 0))))))
\end{array}$$

1.3 Równanie dla programu

Równanie dla programu (**1 punkt**) powinno uwidaczniać fakt, że po wykonaniu instrukcji nic się już nie dzieje i że jest wykonywane w specyficznym środowisku (a nie w dowolnym). Zatem właściwe definicje wyglądają tak:

$$\begin{array}{l}
\mathcal{P} : \text{Prg} \rightarrow \text{Cont} \\
\mathcal{P}[[d; i]] = \mathcal{D}[[d]] \ (\lambda x. \ \mathbf{undef}) \ (\lambda \varrho. \ \mathcal{I}[[i]] \ \varrho \ (\lambda s. \ s))
\end{array}$$

Oczywiście *undef* powinno być w przeciwdziedzinie środowiska. Nie można napisać, że bierzemy puste środowisko, bo semantyka denotacyjnie operuje funkcjami, które nie są częściowe, więc funkcja pusta nie jest poprawnym środowiskiem.

1.4 Równania dla pozostałych instrukcji

Te równania były standardowe i były warte łącznie **1 punkt**. Najczęstszy błąd to zapominanie o środowisku w instrukcji przypisania.

1.5 Równania dla pozostałych wyrażeń

Te równania były standardowe i były warte łącznie **1 punkt**. Najczęstszy błąd to złe równanie dla zmiennej, które powinno wyglądać tak:

$$\mathcal{E}[x] \rho c_e = \lambda s. c_e(s(\rho x))s$$