SEMWER zadanie 2, am418402

December 27, 2021

widoczność funkcji jest statyczna - tzn jeśli mamy funkcję f która wywołuje funkcję g, i w momencie deklaracji f jest widoczna g, a potem w jakimś bloku mamy przesłonioną definicję g i wywołamy f, to f skorzysta z tego g, które zobaczyła w momencie deklaracji, a nie wywołania. innymi słowy funkcje pamiętają środowisko funkcji z momentu deklaracji (wywołanie nie przyjmuje środowiska funkcji jako argumentu - no w każdym razie, na slajdach było jak zrobić widoczność statyczną)

w momencie wywołania funkcji przekazujemy jej dwie kontynuacje - domyślną i returnową, przy czym obie są kontynuacjami wyrażeń, oczekującymi argumentu w postaci albo zwróconego przez return wyrażenia, albo wartości domyślnej

Dziedziny semantyczne

Stan

Ponieważ wszystkie zmienne są globalne, oraz traktujemy je jako zadeklarowane i z nadaną wartością, to nie potrzebujemy środowiska zmiennych, a stan ma po prostu postać

$$\text{State} = \text{Var} \longrightarrow \text{Int}$$

i skoro wszystkie zmienne mają wartość, nie jest to funkcja częściowa. Powstaje tutaj pytanie, jaką wartość mają zmienne, którym nie nadano jeszcze w programie żadnej wartości - wszak można się do nich odwołać. To jednak nie zależy od denotacji instrukcji i wyrażeń, więc nie jest częścią zadania. Można to ustalić np. podając stan "początkowy" w denotacji programów.

Typy kontynuacji

Skoro wynikiem działania instrukcji ma być stan końcowy, to kontynuacje mają postać

$$\text{Cont} = \text{State} \rightharpoonup \text{State}$$

Ale dla rozróżnienia wprowadzę typ $\text{Ans} = \text{State}$ i będę pisał

$$\text{Cont} = \text{State} \rightharpoonup \text{Ans}$$

Dalej, dla kontynuacji wyrażeń arytmetycznych:

$$\text{Cont}_E = \text{Num} \longrightarrow \text{State} \rightharpoonup \text{Ans}$$

Tutaj istotne jest, że wyrażenia mogą zmieniać stan, więc nie może być $\text{Cont}_E = \text{Num} \rightharpoonup \text{Ans}$. Dla wyrażeń boolowskich:

$$\text{Cont}_B = \text{Bool} \longrightarrow \text{State} \rightharpoonup \text{Ans}$$

a dla deklaracji:

$$\text{Cont}_D = \text{FEnv} \rightharpoonup \text{Ans}$$

Gdzie FEnv to opisane niżej środowisko funkcji. Tym razem jest istotne, że deklaracje nie zmieniają stanu.

Środowisko funkcji

Środowisko funkcji ma postać

$$\text{FEnv} = \text{FName} \rightharpoonup \text{Fun}$$

Gdzie FName to nazwy funkcji, natomiast funkcje reprezentuje typ Fun:

$$\text{Fun} = \text{Cont}_E \longrightarrow \text{State} \rightharpoonup \text{Ans}$$

todo uważać na funkcje deklarowane w ciele innej funkcji

Typy funkcji semantycznych

Dla instrukcji:

$$\mathcal{I}[\![\,]\!] : \text{Instr} \longrightarrow \text{FEnv} \longrightarrow \text{Cont} \longrightarrow \text{Cont}_E \longrightarrow \text{State} \rightharpoonup \text{Ans}$$

Jak warto zauważyć, podajemy tutaj jako argumenty dwie różne kontynuacje. Ma to na celu umożliwienie zarówno wykonania całości ciała funkcji, jak i wyjście z niej wcześniej za pomocą instrukcji `return`. Dla wyrażeń arytmetycznych jest:

$$\mathcal{E}[\![\,]\!] : \text{Expr} \longrightarrow \text{FEnv} \longrightarrow \text{Cont}_E \longrightarrow \text{State} \rightharpoonup \text{Ans}$$

i tym razem oczywiście nie potrzebujemy dwóch różnych kontynuacji. Dla wyrażeń boolowskich analogicznie:

$$\mathcal{B}[\![\,]\!] : \text{BExpr} \longrightarrow \text{FEnv} \longrightarrow \text{Cont}_B \longrightarrow \text{State} \rightharpoonup \text{Ans}$$

Dla deklaracji:

$$\mathcal{D}[\![\,]\!] : \text{FDecl} \longrightarrow \text{FEnv} \longrightarrow \text{Cont}_D \longrightarrow \text{State} \rightharpoonup \text{Ans}$$

Denotacje

Denotacje wyrażeń

$$\mathcal{E}[\![n]\!]\rho_F\ \kappa_E\ s = \kappa_E(\mathcal{N}[\![n]\!])$$

tylko zaraz. w takim razie jak przekazujemy domyślny wynik funkcji? bo jego też powinna oczekiwać jakaś kontynuacja wyrażenia arytmetycznego, a w denotacji instrukcji jest tylko jeden argument będący nią, który odpowiada za wyjście przez return. odpowiedź: to jest chyba opędzlowane w taki sposób, że domyślnego wyniku wcale nie oczekuje kontynuacja wyrażenia arytmetycznego, tylko kontynuacja po prostu - ona ma tę domyślną wartość już wstawioną. ale to się zobaczy.

denotacja odwołania się do zmiennej
denotacja plusa
wzmianka że denotacje minusa i mnożenia są takie same

$$\mathcal{E}[\![f()]\!]\rho_F\ \kappa_E\ s =$$

denotacja literału bulionowego
denotacja negacji
denotacja koniunkcji sfer
denotacja równości dwóch wyrażeń, ojej
wzmianka że denotacja znaku mniejszości działa tak samo

Denotacje instrukcji

denotacja przypisania
denotacja średnika
denotacja ifa
denotacja while'a
denotacja bloku
denotacja returna?

Denotacje deklaracji

funkcje są rekurencyjne, więc definicja deklaracji funkcji musi być stałopunktowa.
denotacja deklaracji funkcji
denotacja złożenia deklaracji funkcji też