

SEMWER zadanie 2, am418402

December 27, 2021

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

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

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

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

Dalej, dla kontynuacji wyrażeń arytmetycznych:

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

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

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

a dla deklaracji:

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

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

Środowisko funkcji

Środowisko funkcji ma postać

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

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

$$\text{Fun} = (\text{Cont} \rightarrow \text{Cont}_E \rightarrow \text{State} \rightarrow \text{Ans}) \times \text{Expr}$$

Jak warto zauważyć, w pierwszej części produktu podajemy 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**. Ta pierwsza część intuicyjnie jest odpowiedzialna za wykonanie ciała funkcji. Natomiast druga część produktu to wyrażenie odpowiedzialne za domyślny wynik działania funkcji. W środowisku należy pamiętać całe to wyrażenie, gdyż będzie ono wyliczane przed każdym wywołaniem funkcji.

Typy funkcji semantycznych

Dla instrukcji:

$$\mathbb{J} : \text{Instr} \rightarrow \text{FEnv} \rightarrow \text{Cont} \rightarrow \text{Cont}_E \rightarrow \text{State} \rightarrow \text{Ans}$$

Ponownie, i z analogicznych powodów, mamy tu dwie różne kontynuacje. Dla wyrażeń arytmetycznych jest:

$$\mathbb{E} : \text{Expr} \rightarrow \text{FEnv} \rightarrow \text{Cont}_E \rightarrow \text{State} \rightarrow \text{Ans}$$

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

$$\mathbb{B} : \text{BExpr} \rightarrow \text{FEnv} \rightarrow \text{Cont}_B \rightarrow \text{State} \rightarrow \text{Ans}$$

Dla deklaracji:

$$\mathbb{D} : \text{FDecl} \rightarrow \text{FEnv} \rightarrow \text{Cont}_D \rightarrow \text{State} \rightarrow \text{Ans}$$

Denotacje

Denotacje wyrażeń

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

denotacja odwołania się do zmiennej

denotacja plusa

wzmianka że denotacje minusa i mnożenia są takie same

$$\mathbb{E}[f()]_{\rho_F \ \kappa_E} s = \mathbb{E}[e_d]_{\rho_F}(\lambda n. \beta \ \kappa_E(n) \ \kappa_E)s$$

gdzie

$$\rho_F(f) = (\beta, e_d)$$

Innymi słowy, wyciągamy ze środowiska funkcji krotkę oznaczającą ciało funkcji i jej wyrażenie domyślne, i przekazujemy kontrolę wyrażeniu domyślnemu, mówiąc, że tym, co się stanie z jego wynikiem n , będzie nowo skonstruowana kontynuacja zależna od n . Ta kontynuacja z kolei jest wykonaniem ciała funkcji, biorąc jako kontynuację domyślną $\kappa_E(n)$, a jako kontynuację dla instrukcji **return** - κ_E .

- 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

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 przesłoniętą 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)

- funkcje są rekurencyjne, więc definicja deklaracji funkcji musi być stałopunktowa.

- denotacja deklaracji funkcji

- denotacja złożenia deklaracji funkcji też