Semantyka i Weryfikacja Programów
Praca domowa # 2

MIMUW 2018/19

Michał Szafraniuk
219673

3 stycznia 2019

Idea rozwiązania

Clue zadania sprowadza się do poprawnego zdefiniowania deklaracji procedur, które muszą sensownie „połączyć" przekazywanie parametrów do procedur w wersjach przez zmienną i przez wartość.

Niech `z` będzie identyfikatorem zmiennej, z którą wywołujemy procedurę a `x` identyfikatorem zmiennej w deklaracji procedury (tj. `proc(x) I` oraz `call(z)`). W swoim rozwiązaniu chcę osiągnąć następujący mechanizm:

- w momencie wywołania procedury tworzona jest nowa komórka pamięci, której lokację utożsamiamy z `x` i do której wpisywana jest wartość zmiennej `z` z chwili wywołania
- w środku procedury przy odwołaniach do `x` działamy na tej nowej komórce pamięci, tak jak przy wołaniu przez wartość
- na koniec nadpisujemy wartość `z` wartością `x` a zatem deklaracja procedury musi mieć informację o lokacji zmiennej `z`

Rozwiązanie

Dziedziny semantyczne

Wprowadzam następujące dziedziny semantyczne:

- $\mathsf{Val} = \mathbb{Z}$: zbiór wartości wyrażeń arytmetycznych
- $\mathsf{Bool} = \{\mathbf{tt}, \mathbf{ff}\}$: zbiór wartości wyrażeń boolowskich
- $\mathsf{Loc} = \{l_0, l_1, ...\}$: (abstrakcyjny) zbiór lokacji
- $\mathsf{Store} = \mathsf{Loc} \to \mathsf{Val}$: zbiór składów
- $\mathsf{VEnv} = Var \to \mathsf{Loc}$: środowisko zmiennych
- $\mathsf{Proc} = \mathsf{Loc} \to \mathsf{Store} \rightharpoonup \mathsf{Store}$: reprezentacja procedur
- $\mathsf{PEnv} = PName \to \mathsf{Proc}$: środowisko procedur

Uwagi

Motywowany chęcią niezaciemniania obrazu:

- zakładam *implicite* dostępność operacji $newloc : \mathsf{Store} \to \mathsf{Loc}$, która generuje nową, nieużywaną lokację (np. z implementacją jak na wykładzie ale pomijam szczegóły, aby nie zaciemniać obrazu)
- zakładam, że nieokreślenie się odpowiednio propaguje
- przy λ-notacji - tam gdzie jest to w miarę oczywiste - unikam specyfikacji zbiorów poszczególnych abstrakcji uznając, iż następująca notacja (z ewentualnymi primami, subskryptami etc) jednoznacznie określa o jakich abstrakcjach mowa:
  - $\rho_V \in \mathsf{VEnv}$
  - $\rho_P \in \mathsf{PEnv}$
  - $s \in \mathsf{Store}$
  - $l \in \mathsf{Loc}$

Typy funkcji semantycznych

Wprowadzam następujące typy funkcji semantycznych:

- $\mathcal{N}\colon Num \to \mathsf{Val}$
- $\mathcal{E}\colon Expr \to \mathsf{VEnv} \to \mathsf{Store} \rightharpoonup \mathsf{Val}$
- $\mathcal{B}\colon BExpr \to \mathsf{VEnv} \to \mathsf{Store} \rightharpoonup \mathsf{Bool}$
- $\mathcal{I}\colon Instr \to \mathsf{VEnv} \to \mathsf{PEnv} \to (\mathsf{Store} \rightharpoonup \mathsf{Store})$
- $\mathcal{D}\colon Decl \to (\mathsf{VEnv} \times \mathsf{PEnv} \times \mathsf{Store} \rightharpoonup \mathsf{VEnv} \times \mathsf{PEnv} \times \mathsf{Store})$

Denotacja dla numerałów i wyrażeń

Standardowa jak na wykładzie/ćwiczeniach.

Denotacja dla instrukcji

- **instrukcja `skip`**

  Denotacją jest funkcja identycznościowa na zbiorze składów:

  $$\mathcal{I}[\![\texttt{skip}]\!]\rho_V\rho_P = \lambda s.s$$

- **instrukcja przypisania**

  Denotacją jest odpowiednia modyfikacja składu:

  $$\mathcal{I}[\![x := e]\!]\rho_V\rho_P = \lambda s.s[l \mapsto v]$$

  where

  $$l = \rho_V x, \quad v = \mathcal{E}[\![e]\!]\rho_V s$$

- **instrukcja złożenia**

  Denotacją złożenia dwóch instrukcji jest złożenie denotacji składowych (pamiętając o założeniu dot. propagacji nieoznaczoności):

$$\mathcal{I}[\![I_1; I_2]\!]\rho_V\rho_P = \mathcal{I}[\![I_2]\!]\rho_V\rho_P \circ \mathcal{I}[\![I_1]\!]\rho_V\rho_P$$

- **instrukcja warunkowa**

  Standardowa denotacja:

$$\mathcal{I}[\![\texttt{if}\ b\ \texttt{then}\ I_1\ \texttt{else}\ I_2]\!]\rho_V\rho_P = \lambda s.\begin{cases} \mathcal{I}[\![I_1]\!]\rho_V\rho_P s & \text{if} \quad \mathcal{B}[\![\texttt{b}]\!]\rho_V s = \mathbf{tt} \\ \mathcal{I}[\![I_2]\!]\rho_V\rho_P s & \text{oth} \end{cases}$$

- **intrukcja pętli `while`**

  Denotacją pętli `while` jest punkt stały odpowiedniego funkcjonału, który odpowiadaja operacyjnej intuicji dla tej pętli:

$$\mathcal{I}[\![\texttt{while}\ b\ \texttt{do}\ I]\!]\rho_V\rho_P = \text{fix}\ \Phi$$

  where

$$\Phi\colon (\mathsf{Store} \rightharpoonup \mathsf{Store}) \to (\mathsf{Store} \rightharpoonup \mathsf{Store})$$

$$\Phi = \lambda\phi \in \mathsf{Store} \rightharpoonup \mathsf{Store}.\lambda s.\begin{cases} \phi(\mathcal{I}[\![I]\!]\rho_V\rho_P s) & \text{if} \quad \mathcal{B}[\![\texttt{b}]\!]\rho_V s = \mathbf{tt} \\ s & \text{oth} \end{cases}$$

- **intrukcja bloku**

  Denotacją bloku jest denotacja instrukcji tego bloku wykonanej w środowiskach i składzie zmodyfikowanych przez deklarację bloku:

$$\mathcal{I}[\![\texttt{begin}\ D; I\ \texttt{end}]\!]\rho_V\rho_P = \lambda s.\mathcal{I}[\![I]\!]\rho'_V\rho'_P s'$$

  where

$$\langle\rho'_V, \rho'_P, s'\rangle = \mathcal{D}[\![D]\!]\langle\rho_V, \rho_P, s\rangle$$

- **instrukcja wywołania procedury**

  Denotacją jest wywołanie odpowiedniego „obliczenia" reprezentującego wołaną procedurę z przekazaniem jej lokacji zmiennej będącej parametrem aktualnym w tym wołaniu:

$$\mathcal{I}[\![\texttt{call}\ p(x)]\!]\rho_V\rho_P = \lambda s.Pls$$

  where

$$P = \rho_P p, \quad l = \rho_V x$$

Denotacja dla deklaracji

- **deklaracja zmiennej**

  Tworzymy nową komórkę pamięci i wpisujemy do niej odpowiednią wartość:

$$\mathcal{D}[\![\texttt{var}\, x = e]\!]\langle \rho_V, \rho_P, s\rangle = \langle \rho_V[x \mapsto l], \rho_P, s[l \mapsto v]\rangle$$

  where

$$l = \mathit{newloc}(s), \quad v = \mathcal{E}[\![e]\!]\rho_V s$$

- **deklaracja procedury**

  Przyjmując na moment, że operujemy w świecie bez rekurencji, denotacją deklaracji procedury jest aktualizacja środowiska procedur o „quasi-obliczenie", które reprezentuje deklarowaną procedurę. W momencie wywołania, to quasi-obliczenie

  - tworzy nową komórkę pamięci
  - ładuje do niej wartość zmiennej, z którą została wywołana procedura
  - na końcu tejże zmiennej nadaje wartość znajdującą się w utworzonej na początku komórce

  Przenosząc się do świata z rekurencją, podobnie jak przy pętli `while`, aby zachować kompozycjonalność musimy zdefiniować obliczenie jako punkt stały pewnego standardowego funkcjonału:

$$\mathcal{D}[\![\texttt{proc}\, p(x)\ I]\!]\rho_V \rho_P = \lambda\tau \in \mathsf{Store}.\langle \rho_V, \rho_P[p \mapsto \mathrm{fix}\,\Psi], \tau\rangle$$

  where

$$\Psi \colon \mathsf{Proc} \to \mathsf{Proc}$$

$$\Psi = \lambda\psi \in \mathsf{Proc}.\lambda l.\lambda s.s'[l \mapsto s'l']$$

  where

$$l' = \mathit{newloc}(s), \quad s' = \mathcal{I}[\![I]\!]\rho_V[x \mapsto l']\rho_P[p \mapsto \psi]s[l' \mapsto sl]$$

- **deklaracja pusta**

  Denotacją jest funkcja identycznościowa:

$$\mathcal{D}[\![\epsilon]\!] = \lambda\tau \in \mathsf{VEnv} \times \mathsf{PEnv} \times \mathsf{Store}.\tau$$

- **złożenie deklaracji**

  Denotacją jest złożenie denotacji składowych:

$$\mathcal{D}[\![D_1; D_2]\!] = \mathcal{D}[\![D_2]\!] \circ \mathcal{D}[\![D_1]\!]$$