Semantyka Denotacyjna

MIMUW 2018/19

Michał Szafraniuk

2 lutego 2019

1 Semantyka denotacyjna języka TINY

Dziedziny semantyczne

- $\mathsf{Val} = \mathbb{Z}$: zbiór wartości wyrażeń arytmetycznych
- $\mathsf{Bool} = \{\mathbf{tt}, \mathbf{ff}\}$: zbiór wartości wyrażeń boolowskich
- $\mathsf{State} = \mathit{Var} \to \mathsf{Val}$: zbiór stanów

Semantyka instrukcji

Typ:

$$[\![\cdot]\!] : \mathit{Instr} \to \mathsf{State} \rightharpoonup \mathsf{State}$$

Definicje kompozycjonalne:

- proste

$$\begin{aligned} [\![x := e]\!] &= \lambda s.s[x \mapsto [\![e]\!]s] \\ [\![\texttt{skip}]\!] &= \lambda s.s \\ [\![i_1;i_2]\!] &= \lambda s.[\![i_2]\!]([\![i_1]\!]s) \\ [\![\texttt{if}\ b\ \texttt{then}\ i_1\ \texttt{else}\ i_2]\!] &= \lambda s.ifte([\![b]\!]s, [\![i_1]\!]s, [\![i_2]\!]s) \end{aligned}$$

- pętle
  - `while`

$$[\![\texttt{while}\ b\ \texttt{do}\ I]\!] = \mathrm{fix}\ \Phi$$

where

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

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

czyli krócej:

$$\Phi = \lambda\phi.\lambda s.ifte([\![b]\!]s, \phi([\![I]\!]s), s)$$

– `repeat`

$$[\![\texttt{repeat}\ i\ \texttt{until}\ b]\!] = \text{fix}\ \Phi$$

where

$$\Phi = \lambda\phi.\lambda s.(\text{let}\ s' = [\![i]\!]s\ \text{in}\ ifte([\![b]\!]s', s', \phi([\![i]\!]s'),))$$

– `for`

$$[\![\texttt{for}\ x := e_1\ \texttt{to}\ e_2\ \texttt{do}\ i]\!] = \lambda s.(\text{fix}\ \Phi)s[x \mapsto [\![e_1]\!]s]$$

where

$$\Phi = \lambda\phi.\lambda s.ifte(s(x) \leqslant [\![e_2]\!]s, \phi((\lambda\tau.\tau[x \mapsto \tau(x)+1])([\![i]\!]s)), s)$$

2 TINY + bloki z deklaracjami zmiennych

$$\mathit{Instr} \ni i ::= ... \mid \texttt{begin}\ d\ \texttt{in}\ i$$

$$\mathit{Decl} \ni d ::= \texttt{var}x := e \mid d_1; d_2$$

Wprowadzamy środowisko zmiennych oraz podział na lokacje oraz składy:

- $\mathsf{Loc} = \{l_0, l_1, ...\}$: (abstrakcyjny) zbiór lokacji
- $\mathsf{Store} = \mathsf{Loc} \to \mathsf{Val}$: zbiór składów
- $\mathsf{VEnv} = \mathit{Var} \to \mathsf{Loc}$: środowisko zmiennych

Typy funkcji semantycznych

- $[\![\cdot]\!]_{\mathcal{N}} \colon \mathit{Num} \to \mathsf{Val}$
- $[\![\cdot]\!]_{\mathcal{E}} \colon \mathit{Expr} \to \mathsf{VEnv} \to \mathsf{Store} \rightharpoonup \mathsf{Val}$
- $[\![\cdot]\!]_{\mathcal{B}} \colon \mathit{BExpr} \to \mathsf{VEnv} \to \mathsf{Store} \rightharpoonup \mathsf{Bool}$
- $[\![\cdot]\!]_{\mathcal{I}} \colon \mathit{Instr} \to \mathsf{VEnv} \to (\mathsf{Store} \rightharpoonup \mathsf{Store})$
- $[\![\cdot]\!]_{\mathcal{D}} \colon \mathit{Decl} \to (\mathsf{VEnv} \times \mathsf{Store} \rightharpoonup \mathsf{VEnv} \times \mathsf{Store})$

Denotacja dla instrukcji

- **instrukcja `skip`**

  Denotacją jest funkcja identycznościowa na zbiorze składów:

$$[\![\texttt{skip}]\!] = \lambda\rho.\lambda s.s$$

- **instrukcja przypisania**

  Denotacją jest odpowiednia modyfikacja składu:

$$[\![x := e]\!] = \lambda\rho.\lambda s.s[\rho x \mapsto [\![e]\!]\rho s]$$

- **instrukcja złożenia**

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

$$[\![i_1; i_2]\!] = [\![i_2]\!] \circ [\![i_1]\!]$$

- **instrukcja warunkowa**

  Standardowa denotacja:

$$[\![\texttt{if}\ b\ \texttt{then}\ I_1\ \texttt{else}\ I_2]\!] = \lambda\rho.\lambda s. \begin{cases} [\![i_1]\!]\rho s & \text{if} \quad [\![b]\!]\rho s = \mathbf{tt} \\ [\![i_2]\!]\rho s & \text{oth} \end{cases}$$

- **intrukcja pętli `while`**

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

$$[\![\texttt{while}\ b\ \texttt{do}\ i]\!] = \lambda\rho.\ \mathrm{fix}\ \Phi$$

  where

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

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

- **intrukcja bloku z deklaracjami zmiennych**

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

$$[\![\texttt{begin}\ d; i\ \texttt{end}]\!] = \lambda\rho.\lambda s.(\mathsf{let}\ \langle\rho', s'\rangle = [\![d]\!]\rho s\ \mathsf{in}\ [\![i]\!]\rho' s')$$

Denotacja dla deklaracji

- **deklaracja zmiennej**

  Tworzymy nową komórkę pamięci i wpisujemy do niej odpowiednią wartość:

$$[\![\texttt{var}\ x = e]\!] = \lambda\langle\rho, s\rangle.(\mathsf{let}\ l = newloc(s)\ \mathsf{in}\ \langle\rho[x \mapsto l], s[l \mapsto [\![e]\!]\rho s]\rangle)$$

- **złożenie deklaracji**

  Denotacją jest złożenie denotacji składowych:

$$[\![d_1; d_2]\!] = [\![d_2]\!] \circ [\![d_1]\!]$$

3 TINY + bloki z deklaracjami zmiennych oraz procedur

Rozszerzamy język o procedury:

$$\mathit{Instr} \ni i ::= \ldots \mid \texttt{begin}\, d\, \texttt{in}\, i \mid \texttt{call}\, p(x)$$

$$\mathit{Decl} \ni d ::= \ldots \mid \texttt{proc}\, p(x)\, \texttt{is}\, i$$

3.1 Widoczność statyczna, parametry przez zmienną

Dziedziny semantyczne

Dodajemy:

- $\mathsf{Proc} = \mathsf{Loc} \to \mathsf{Store} \rightharpoonup \mathsf{Store}$: reprezentacja procedur
- $\mathsf{PEnv} = \mathit{PName} \to \mathsf{Proc}$: środowisko procedur

W semantyce denotacyjnej nie możemy składować sobie "napisów", gdyż ogranicza nas wymóg kompozycjonalności. Oznacza to, że procedurę będziemy składować jako obliczenie typu $\mathsf{Store} \rightharpoonup \mathsf{Store}$. Ponieważ przekazujemy do procedur zmienne więc potrzebujemy też lokacji w typie Proc.

Typy funkcji denotacyjnych

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

Denotacje

- dla instrukcji wywołania procedury

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

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

- dla deklaracji procedury

  $$\begin{aligned}[\![\texttt{proc}\, p(x)\, \texttt{is}\, I]\!] \quad = \quad & \lambda\langle\rho_V, \rho_P, s\rangle. \\ & (\mathsf{let}\ Q = \lambda l.\lambda\tau.([\![i]\!]\rho_P\rho_V[x \mapsto l]\tau)\ \mathsf{in}\ \langle\rho_V, \rho_P[p \mapsto Q], s\rangle)\end{aligned}$$

  Q jest obliczeniem, które wiążemy z identyfikatorem procedury. Wyliczamy je w środowiskach z chwili deklaracji co skutkuje statycznym wiązaniem.

3.2 Widoczność statyczna, parametry przez wartość

$$\mathit{Instr} \ni i ::= \ldots \mid \texttt{call}\ p(e)$$