

Семантика — отображение из программ в семантический домен. Программы обычно представляются как абстрактные синтаксические деревья. Семантический домен — то, что описывает смысл программы в контексте решаемой задачи.

Семантика бывает разной. Классические примеры: динамическая семантика (вычисление программ) и статическая семантика (вывод типов).

Доменом динамической семантики арифметических выражений является множество функций из функции, сопоставляющей переменным целочисленные значения, в целые числа.

Семантика часто обозначается скобками $\llbracket \cdot \rrbracket$ и $\llbracket \cdot \rrbracket$.

Семантика арифметических выражений:

$$\llbracket \cdot \rrbracket :: (X \rightarrow \mathbb{N}) \rightarrow \mathbb{N}$$

$$\llbracket n \rrbracket(\sigma) = n, n \in \mathbb{N}$$

$$\llbracket v \rrbracket(\sigma) = \sigma(v)$$

$$\llbracket e_1 \oplus e_2 \rrbracket(\sigma) = \llbracket e_1 \rrbracket(\sigma) \otimes \llbracket e_2 \rrbracket(\sigma)$$

$$\llbracket !e \rrbracket(\sigma) = 0, \llbracket e \rrbracket(\sigma) \neq 0$$

$$\llbracket !e \rrbracket(\sigma) = 1, \llbracket e \rrbracket(\sigma) \equiv 0$$

$$\llbracket -e \rrbracket(\sigma) = \llbracket 0 - e \rrbracket(\sigma)$$

В языке L есть переменные, операции чтения из входного потока и записи в него. Поэтому один из возможных семантических доменов — множество функций из тройки (значения переменных, входной поток, выходной поток) в целые числа.

Будем описывать операционную семантику большого шага в терминах правил вывода. Стрелка \xrightarrow{p} связывает состояние входного i и выходного o потоков и значений переменных σ до выполнения инструкции p и после нее.

Абстрактный синтаксис языка L:

$$\begin{aligned} \text{data } LAst = & \text{If}\{cond :: Expr, thn :: LAst, els :: LAst\} \\ & | \text{While}\{cond :: Expr, body :: LAst\} \\ & | \text{Assign}\{var :: Var, expr :: Expr\} \\ & | \text{Read}\{var :: Var\} \\ & | \text{Write}\{expr :: Expr\} \\ & | \text{Seq}\{statements :: [LAst]\} \end{aligned}$$

При обработке инструкции чтения снимаем значение со входного потока и сопоставляем его переменной x .

$$\langle \sigma, (h : t), o \rangle \xrightarrow{\text{read}(x)} \langle \sigma[x \mapsto h], t, o \rangle$$

При обработке инструкции записи вычисляем значение выражения и кладем его в выходной поток.

$$\frac{}{\langle \sigma, i, o \rangle \xrightarrow{\text{write}(e)} \langle \sigma, i, ([e](\sigma)) : o \rangle}$$

При обработке инструкции присвоения вычисляем значение выражения и сопоставляем его переменной.

$$\frac{}{\langle \sigma, i, o \rangle \xrightarrow{\text{assign } x \ (e)} \langle \sigma[x \mapsto n], i, o \rangle}, \llbracket e \rrbracket(\sigma) \equiv n$$

При обработке условного перехода вычисляем значение условия, в зависимости от его истинности выполняем соответствующую ветку.

$$\frac{c \xrightarrow{\text{thn}} c'}{c \xrightarrow{\text{if } (cond) \ (thn) \ (els)} c'}, \llbracket cond \rrbracket(\sigma) \neq 0$$

$$\frac{c \xrightarrow{\text{els}} c'}{c \xrightarrow{\text{if } (cond) \ (thn) \ (els)} c'}, \llbracket cond \rrbracket(\sigma) \equiv 0$$

Если значение условия в цикле ложно, то конфигурация не меняется

$$\frac{}{c \xrightarrow{\text{while } (cond) \ body} c}, \llbracket cond \rrbracket(\sigma) \equiv 0$$

Если значение условия в цикле истинно, то выполняем тело один раз, затем запускаем цикл заново.

$$\frac{c \xrightarrow{\text{body}} c', \quad c' \xrightarrow{\text{while } (cond) \ body} c''}{c \xrightarrow{\text{while } (cond) \ body} c''}, \llbracket cond \rrbracket(\sigma) \neq 0$$

При обработке пустой последовательности инструкций конфигурация не меняется.

$$\frac{}{c \xrightarrow{\text{seq } []} c}$$

При обработке непустой последовательности инструкций сначала вычисляем результат выполнения первой инструкции, затем на нем — всех остальных.

$$\frac{c \xrightarrow{h} c', \quad c' \xrightarrow{\text{seq } t} c''}{c \xrightarrow{\text{seq } (h:t)} c''}$$