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 $[\![$ и $]\!]$.

Семаника арифметических выражений:

$$[\![\cdot]\!] :: (X \to \mathbb{N}) \to \mathbb{N}$$

$$\begin{gathered}
[\![n]\!](\sigma) = n, n \in \mathbb{N} \\
[\![v]\!](\sigma) = \sigma(v) \\
[\![e_1 \oplus e_2]\!](\sigma) = [\![e_1]\!](\sigma) \otimes [\![e_2]\!](\sigma)
\end{gathered}$$

$$\begin{gathered}
[\![!e]\!](\sigma) = 0, [\![e]\!](\sigma) \neq 0 \\
[\![!e]\!](\sigma) = 1, [\![e]\!](\sigma) \equiv 0
\end{gathered}$$

$$[\![-e]\!](\sigma) = [\![0 - e]\!](\sigma)$$

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:

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

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

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

При обработке инструкции записи вычисляем значение выражения и кладем его во входной поток.

$$\frac{}{\langle \sigma, i, o\rangle \xrightarrow{write(e)} \langle \sigma, i, (\llbracket e \rrbracket(\sigma)) : o\rangle}$$

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

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

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

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

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

Если значение условия в цикле ложно, то конфигурация не меняется

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

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

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

При обработке пустой последовательности инструкций конфигурация не меняется.

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

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

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