ДЕНОТАЦИОННАЯ СЕМАНТИКА ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

Рассмотрим стандартную денотационную семантика языков программирования императивного типа.

Семантические определения

Вначале определим простейший язык программирования и зададим его семантику с помощью семантических функций, отображающих конструкции языка в некоторую семантическую область.

В дальнейшем будем расширять язык, вводя в него новые конструкции, добавляя новые семантические определения.

На начальном этапе не будем обращать внимания на детали изменения состояния. Кроме того, в языке не будет определяемых программистом имен.

СЕМАНТИЧЕСКИЕ ОПРЕДЕЛЕНИЯ

Прежде, чем определить язык программирования **L**, введем следующие синтаксические области:

C ∈ Cmd	(команды)
E ∈ Exp	(выражения)
c ∈ Com	(примитивные команды)
P ∈ Pre	(примитивные предикаты)

СЕМАНТИЧЕСКИЕ ОПРЕДЕЛЕНИЯ

Синтаксис языка **L** может быть задан следующим образом:

$$C ::= c \mid e \mid \underline{\text{if}}\ E\ \underline{\text{then}}\ C_0\ \underline{\text{else}}\ C_1\ \underline{\text{fi}} \mid C_0; C_1$$

$$E ::= p \mid \underline{\text{true}} \mid \underline{\text{false}} \mid \underline{\text{if}}\ E_0\ \underline{\text{then}}\ E_1\ \underline{\text{else}}\ E_2\ \underline{\text{fi}}$$

Результатом выполнения команды является изменение состояния и в качестве подходящего значения команды будут выбираться функции изменения состояния.

СЕМАНТИЧЕСКИЕ ОПРЕДЕЛЕНИЯ

Таким образом, если **S** – множество состояний, то значением команды будет элемент из области **S → S**.

Предположим, что выражение не имеет побочного эффекта, т.е. вычисление выражения не производит изменения состояния.

Ограничим также область значений выражений областью истинностных значений **T = {true, false, ⊥}**.

Результат выражения зависит от состояния. Таким образом, значением выражения будет элемент из области **S → T**.

СЕМАНТИЧЕСКИЕ ОПРЕДЕЛЕНИЯ

Из вышесказанного следует, что семантическими областями (областями значений) для языка **L** являются:

$t \in T$ (истинностные значения)

$\sigma \in S$ (машинные состояния)

$\gamma \in (S \to S)$ (значения команд)

$\omega \in (S \to T)$ (значения выражений)

УСЛОВНЫЙ ОПЕРАТОР И ПОНЯТИЕ СТРОГОЙ ФУНКЦИИ

Прежде, чем задать семантические функции, определим **условный оператор** и введем **понятие строгой функции**.

Определение 1. Для $\mathbf{x} \in \mathbf{P}$, где $\mathbf{P}$ – область, содержащая $\mathbf{T}$ как слагаемое ($\mathbf{P} = \mathbf{P_1}+\mathbf{P_2}+\ldots+\mathbf{P_n}$), и $\mathbf{y}, \mathbf{z} \in \mathbf{D}$.

$$\mathbf{x} \to \mathbf{y}, \mathbf{z} = \begin{cases} \mathbf{y}, & \text{если } (\mathbf{x}|\mathbf{T}) \equiv \mathbf{true} \\ \mathbf{z}, & \text{если } (\mathbf{x}|\mathbf{T}) \equiv \mathbf{false} \\ \perp_{\mathbf{D}}, & \text{если } (\mathbf{x}|\mathbf{T}) \equiv \perp_{\mathbf{T}} \end{cases}$$

Определение 2. Функция $\mathbf{f}: \mathbf{A} \to \mathbf{B}$ называется строгой (записывается **strict f**), если для $\perp \in \mathbf{A}$ $\mathbf{f}(\perp) = \perp \in \mathbf{B}$, т.е.

$$\mathbf{strict\ f(x)} = \begin{cases} \perp, \text{ если } \mathbf{x} = \perp, \\ \mathbf{f(x)}, \text{ в противном случае.} \end{cases}$$

СЕМАНТИЧЕСКИЕ ФУНКЦИИ КОМАНД

Семантические функции команд **C: Cmd → (S→S)** определяются следующим образом:

C⟦c⟧ = strict γ, где **γ** – функция, ассоциируемая с командой "**c**";

C⟦e⟧ = I – тождественная функция (т.е. **I ≡ λσ.σ**);

C⟦if E then C_0 else C_1 fi⟧ = λσ.E⟦E⟧σ → C⟦C_0⟧σ, C⟦C_1⟧σ;

C⟦C_0;C_1⟧ = C⟦C_1⟧·C⟦C_0⟧.

Здесь и далее двойные скобки "⟦" и "⟧" используются для выделения синтаксических конструкций, к которым применяется семантическая функция.

Семантическая функция **E**, определяющая значения выражений языка **L**, будет определена далее.

СЕМАНТИЧЕСКИЕ ФУНКЦИИ КОМАНД

Для лучшего понимания пункта, определяющего семантику **условной команды**, зададим состояние $\tilde{\sigma}$, которому применяется соответствующая функция.

Тогда:

$$C[\![\underline{\text{if}}\ E\ \underline{\text{then}}\ C_0\ \underline{\text{else}}\ C_1\ \underline{\text{fi}}]\!]\tilde{\sigma} = E[\![E]\!]\tilde{\sigma} \rightarrow C[\![C_0]\!]\tilde{\sigma}, C[\![C_1]\!]\tilde{\sigma},$$

т.е. функция $E[\![E]\!]$ применяется к состоянию $\tilde{\sigma}$ и результат используется для выбора альтернативы.

$C[\![C_i]\!]\tilde{\sigma}$, $i = 0, 1$ означает применение соответствующей функции $C[\![C_i]\!]$ к состоянию $\tilde{\sigma}$.

Запись вида $\gamma_1 \cdot \gamma_0$ в последнем пункте семантического определения означает функциональную композицию, т.е. $(\gamma_1 \cdot \gamma_0)\sigma = \gamma_1(\gamma_0\sigma)$.

СЕМАНТИЧЕСКИЕ ФУНКЦИИ ДЛЯ ВЫРАЖЕНИЙ ЯЗЫКА L

Семантические функции для выражений языка **L** имеют аналогичное функциональное определение.

E : Exp → (S→T) и задаются следующим образом:

E⟦p⟧ = strict ω, где **ω** – функция, ассоциированная с "**p**";

E⟦true⟧ = strict(λσ.true);

E⟦false⟧ = strict(λσ.false);

E⟦if E_0 then E_1 else E_2 fi⟧ = λσ.E⟦E_0⟧σ → E⟦E_1⟧σ, E⟦E_2⟧σ;

Пусть например, **σ** – подходящее состояние, т.е. **σ≢⊥**. Тогда **E⟦true⟧σ = true**. Если же **σ≡⊥**, то **E⟦true⟧σ =⊥**.

ТЕОРЕМА

Теорема 1. Значения всех команд и выражений языка L являются строгими функциями.

Доказательство. Тривиально, методом структурной индукции.

Смысл этой, теоремы в том, что если однажды что-то не завершилось при выполнении программы, то ничего уже сделать нельзя.

В дальнейшем при расширении языка **L** будем полагать, что теорема 1 доказана для соответствующего расширения.

Отметим, что в приведенных определениях используются семантические области, образованные из базисных областей значений, таких как целостные, истинностные и т.д., являющиеся п.ч.п.

ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ И ИХ СЕМАНТИКА

Прежде всего добавим в язык **L** команду вида:

while E do C od

Необходимо, чтобы эта команда означала то же самое, что и конструкция:

if E then C; while E do C od else e fi

Поэтому, если обозначить **C⟦while E do C od⟧** через **γ'**, то необходимо, чтобы

$$\gamma' = \lambda\sigma.E[\![E]\!]\sigma \rightarrow C[\![C;\ \text{while}\ E\ \text{do}\ C\ \text{od}]\!]\sigma, I\sigma = \lambda\sigma.E[\![E]\!]\sigma \rightarrow (\gamma' \cdot C[\![C]\!])\sigma, \sigma$$

ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ И ИХ СЕМАНТИКА

Рассматривая правую часть равенства как функцию **H** от **γ** такую, что $H(\gamma) = \lambda\sigma.E[\![E]\!]\sigma \to (\gamma \cdot C[\![C]\!])\sigma, \sigma$, т.е.

$H = \lambda\gamma.\lambda\sigma.E[\![E]\!]\sigma \to (\gamma \cdot C[\![C]\!])\sigma, \sigma$, необходимо показать, что $\gamma' = H(\gamma')$.

Таким образом, **γ'** должна быть **фиксированной точкой H**.

Кроме того, как было показано ранее, требуется также, чтобы это была **минимальная фиксированная точка H**.

Поэтому, требуется применение оператора минимальной фиксированной точки **fix**:

$$\gamma' = \mathrm{fix}\ H$$

ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ И ИХ СЕМАНТИКА

Итак, вводим дополнительно в язык **L** синтаксическую конструкцию

$$\mathbf{C ::= \underline{while}\ E\ \underline{do}\ C\ \underline{od}}$$

и новый пункт определения семантической функции **C**:

$$\mathbf{C[\![\underline{while}\ E\ \underline{do}\ C\ \underline{od}]\!] = fix\ (\lambda\gamma.\lambda\sigma.E[\![E]\!]\sigma \rightarrow (\gamma \cdot C[\![C]\!])\sigma, \sigma)}$$

Часто требуется добавить новые черты в язык не для увеличения его семантической мощности, а просто для упрощения использования уже имеющихся в языке возможностей.

СПОСОБЫ ДОБАВЛЕНИЯ НОВЫХ СИНТАКСИЧЕСКИХ КОНСТРУКЦИЙ

Существуют два способа добавления новых синтаксических конструкций в язык.

Первый способ состоит в добавлении новых конструкций и соответствующих им семантических определений. Этот путь формально расширяет язык и усложняет доказательства, которые используют структурную индукцию. Этот способ использовался при добавлении конструкции **while E do C od** в язык **L**.

Альтернативным способом является сохранение ядра языка без изменений и введение дополнительных конструкций с помощью синтаксических равенств. Этот путь показывает, как программа может быть сведена к основным конструкциям ядра языка чисто синтаксическими средствами.

ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ И ИХ СЕМАНТИКА

На практике используются оба способа, причем иногда оба сразу, давая определения обоих видов и доказывая их эквивалентность.

Рассмотрим в качестве примера использования **второго способа** добавление оператора **отрицания** "¬":

$$\neg E = \underline{\text{if}}\ E\ \underline{\text{then}}\ \underline{\text{false}}\ \underline{\text{else}}\ \underline{\text{true}}\ \underline{\text{fi}}$$

Используя это равенство нетрудно доказать справедливость следующих утверждений:

$$\underline{\text{if}}\ \neg E_0\ \underline{\text{then}}\ E_1\ \underline{\text{else}}\ E_2\ \underline{\text{fi}} = \underline{\text{if}}\ E_0\ \underline{\text{then}}\ E_2\ \underline{\text{else}}\ E_1\ \underline{\text{fi}}$$

$$\neg(\neg E) = E$$

ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ И ИХ СЕМАНТИКА

Докажем первое утверждение. Пусть $\omega = E[\![\underline{\text{if}}\ \neg E_0\ \underline{\text{then}}\ E_1\ \underline{\text{else}}\ E_2\ \underline{\text{fi}}]\!]$ и $\omega_r = E[\![E_r]\!]$ для $r = 0, 1, 2$.

Требуется доказать, что $\omega = \lambda\sigma.\omega_0\sigma \to \omega_2\sigma, \omega_1\sigma$ (*)

Из семантического определения имеем:

$\omega = \lambda\sigma.E[\![\neg E_0]\!]\sigma \to E[\![E_1]\!]\sigma, E[\![E_2]\!]\sigma =$

$= \lambda\sigma.E[\![\neg E_0]\!]\sigma \to \omega_1\sigma, \omega_2\sigma =$

$= \lambda\sigma.E[\![\underline{\text{if}}\ E_0\ \underline{\text{then}}\ \underline{\text{false}}\ \underline{\text{else}}\ \underline{\text{true}}\ \underline{\text{fi}}]\!]\sigma \to \omega_1\sigma, \omega_2\sigma =$

$= \lambda\sigma.(\omega_0\sigma \to \text{strict}(\lambda\sigma.\text{false})\sigma, \text{strict}(\lambda\sigma.\text{true})\sigma) \to \omega_1\sigma, \omega_2\sigma.$

Применив обе части доказываемого равенства (*) к некоторому произвольному σ. Если $\sigma=\perp$, то легко проверить, что равенство (*) имеет место. Если σ – подходящее состояние, то рассмотрим различные случаи, зависящие от значения $\omega_0\sigma$. Например, пусть $\omega_0\sigma = \text{true}$. Тогда:

$\omega\sigma = (\text{true} \to \text{strict}(\lambda\sigma.\text{false})\sigma, \text{strict}(\lambda\sigma.\text{true})\sigma) \to \omega_1\sigma, \omega_2\sigma =$

$= \text{false} \to \omega_1\sigma, \omega_2\sigma = \omega_2\sigma = \omega_0\sigma \to \omega_2\sigma, \omega_1\sigma$

Аналогично доказываются и другие случаи, ч.т.д.

ЦИКЛИЧЕСКИЕ КОНСТРУКЦИИ И ИХ СЕМАНТИКА

Введем в язык **L** дополнительно команду **until E do C od**, определив ее с помощью следующего синтаксического равенства: **until E do C od = while ¬E do C od**.

Доказательство того, что это определение эквивалентно альтернативному определению команды **until E do C od**:

$$C[\![\textbf{until E do C od}]\!] = \mathrm{fix}\,(\lambda\gamma.\lambda\sigma.E[\![E]\!]\sigma \to \sigma, (\gamma \cdot C[\![C]\!]\sigma))$$.

Расширим язык **L** еще двумя циклическими: конструкциями **(repeat C while E** и **repeat C until E)** с добавлением соответствующих семантических равенств:

$$C[\![\textbf{repeat C while E}]\!] = \mathrm{fix}\,(\lambda\gamma.(\lambda\sigma.E[\![E]\!]\sigma \to \gamma\sigma, \sigma)\cdot C[\![C]\!])$$

$$C[\![\textbf{repeat C until E}]\!] = \mathrm{fix}\,(\lambda\gamma.(\lambda\sigma.E[\![E]\!]\sigma \to \sigma, \gamma\sigma)\cdot C[\![C]\!])$$

ТЕОРЕМЫ ДЛЯ ОПРЕДЕЛЕНИЯ ДОПОЛНИТЕЛЬНЫХ КОНСТРУКЦИЙ

Следующие две теоремы могут служить как альтернативные определения дополнительных конструкций.

Теорема 2. **repeat C while E = C; while E do C od**

Теорема 3. **repeat C until E = C; until E do C od**

Сформулируем еще одну теорему.

Теорема 4. **while E do C od = if E then repeat C while E else e fi**

Для доказательства этих теорем, а также для доказательства других подобных утверждений используются методы, базирующиеся на понятии **фиксированных точек**.