СЕМАНТИЧЕСКИЕ ФУНКЦИИ, ВКЛЮЧАЮЩИЕ СРЕДУ ВЫЧИСЛЕНИЙ

Идентификаторы и среда вычислений

Введем синтаксическую категорию **Ide** (множество идентификаторов) и понятие среды вычисления **ρ**, являющейся элементом области функций **U: Ide→D**, где **D** – область значений идентификаторов.

Важным требованием является разрешимость вопроса о равенстве двух идентификаторов.

Отметим, что единственные значения, которые мы можем обозначать – это истинностные значения, а следовательно **D** совпадает с областью **T**.

СЕМАНТИЧЕСКИЕ ФУНКЦИИ, ВКЛЮЧАЮЩИЕ СРЕДУ ВЫЧИСЛЕНИЙ

Обозначим через $\mathcal{E}$ множество результатов выражений. Отметим, что $\mathcal{E}$ также совпадает пока с областью **T**.

Поскольку было введено понятие среды **ρ**, то необходимо обновить все предыдущие семантические функции.

Значение команды или выражения зависит теперь также и от среды. Таким образом, если ранее функциональность семантической функции команд имела вид **C: Cmd → (S → S)**, то теперь она будет **C: Cmd → (U → (S → S))**.

Дале будем в большинстве случаев опускать скобки в описании областей непрерывных функций, полагая, что они ассоциируются справа налево, т.е. запись **Cmd → U → S → S** означает область **(Cmd → (U → (S → S)))**.

СЕМАНТИЧЕСКИЕ ФУНКЦИИ, ВКЛЮЧАЮЩИЕ СРЕДУ ВЫЧИСЛЕНИЙ

Семантическая функция выражений, имеющая ранее вид **E: Exp → S → (ℰ×S)**, будет иметь вид **E: Exp → U → S → (ℰ×S)**.

Все предыдущие семантические равенства обновляются просто введением параметра среды в подходящее место.

Например, новое определение семантической функции для условной команды будет иметь следующий вид:

$$\mathbf{C[\![\underline{if}\ E\ \underline{then}\ C_0\ \underline{else}\ C_1\ \underline{fi}]\!]\rho = (\lambda\beta.\beta \rightarrow C[\![C_0]\!]\rho, C[\![C_1]\!]\rho)^* E[\![E]\!]\rho}$$

Таким образом, среда, заданная для всей конструкции, используется для вычисления каждого компонента.

СЕМАНТИЧЕСКИЕ ФУНКЦИИ, ВКЛЮЧАЮЩИЕ СРЕДУ ВЫЧИСЛЕНИЙ

Введем новые синтаксические конструкции, использующие понятие идентификатора:

$$\mathbf{E ::= I \mid \underline{let}\ I = E_0\ in\ E_1}$$

$$\mathbf{C ::= \underline{let}\ I = E\ in\ C}$$

В этих конструкциях через, **I** обозначается элемент множества **Ide**.

Зададим теперь семантические функции, соответствующие новым конструкциям языка.

Для выражения **E≡I** имеем:

$$\mathbf{E[\![I]\!]\rho = strict(\lambda\sigma.<\rho[I],\sigma>)}$$

Как видно из этого определения, значение идентификатора определяется только средой и не зависит от состояния.

СЕМАНТИЧЕСКИЕ ФУНКЦИИ, ВКЛЮЧАЮЩИЕ СРЕДУ ВЫЧИСЛЕНИЙ

Для выражения **let** $I=E_0$ **in** E_1 имеем:

$$E[\![\textbf{let } I=E_0 \textbf{ in } E_1]\!]\rho = (\lambda\beta.E[\![E_1]\!](\rho[\beta/I]))*E[\![E_0]\!]\rho$$

Для того, чтобы выяснить смысл правой части написанного равенства, необходимо применить ее к состоянию **σ**. Выражение E_0 вычисляется в среде **ρ** и начальном состоянии **σ**.

$E[\![E_0]\!]\rho\sigma$ дает значение **<β',σ'>**. Результирующая часть **β'** ассоциируется с идентификатором **I** в новой среде $\rho[\beta'/I]$, и тело выражения E_1 вычисляется в этой новой среде и состоянии, получающемся в результате вычисления E_0.

Окончательным значением выражения будет: $E[\![E_1]\!](\rho[\beta'/I])\sigma'$.

Аналогичным образом задается новый семантический пункт для соответствующей команды.

СЕМАНТИЧЕСКИЕ ФУНКЦИИ, ВКЛЮЧАЮЩИЕ СРЕДУ ВЫЧИСЛЕНИЙ

Заметим, что выражение **let I=E_0 in E_1** очень похоже по результату на выражение $(\lambda I.E_1)E_0$ в λ-нотации. Однако, если допускается побочный эффект, то нельзя просто заменить все свободные вхождения **I** в E_1 выражением E_0, так как многократные вычисления E_0 могут вызвать многократный побочный эффект.

Если расширить язык программирования оператором “=” для проверки равенства двух выражений. Из-за наличия побочного эффекта нельзя сказать, что значение выражения **E=E** всегда истинно, т.к. два вычисления выражения **E** могут дать различные результаты. В то же время значение выражения **let I=E in (I=I)** – всегда истинно, если **E** имеет допустимое значение, т.к. **E** вычисляется только один раз и подставляется вместо всех вхождений идентификатора **I**.

РАСШИРЕНИЕ СЕМАНТИЧЕСКИХ ОБЛАСТЕЙ

Расширим области $\mathcal{E}$ (значения выражений) и **D** (значения идентификаторов), добавив в них, помимо истинностных значений области **T**, целые значения области **N**, т.е. $\mathcal{E} = D = T+N$.

Расширив области, необходимо проверить, что это не нарушает полученных ранее результатов. Необходимо, также ввести некоторые дополнительные операторы для работы со значениями новых типов. В частности, среди таких операторов будут операторы сложения (**+**) и сравнения (**>**).

Введя оператор сложения над целыми числами, необходимо добавить соответствующий пункт семантического определения, например: $E[\![E_0+E_1]\!]\rho = Add^*le<E[\![E_0]\!]\rho,E[\![E_1]\!]\rho>$.

РАСШИРЕНИЕ СЕМАНТИЧЕСКИХ ОБЛАСТЕЙ

le – оператор для последовательного вычисления двух выражений от их аргументов, определяемый следующим образом:

$$\mathbf{le}\langle\omega_0, \omega_1\rangle\sigma = \langle\langle\beta_0,\beta_1\rangle,\sigma_2\rangle,$$

где:

$$\langle\beta_0,\sigma_1\rangle = \omega_0\sigma,$$

$$\langle\beta_1,\sigma_2\rangle = \omega_1\sigma_1.$$

Таким образом, вычисления осуществляются слева направо (с учетом побочного эффекта).

РАСШИРЕНИЕ СЕМАНТИЧЕСКИХ ОБЛАСТЕЙ

По аналогии с оператором **le** от двух аргументов можно определить **le** от **n** аргументов:

$$\mathbf{le}\langle\omega_0, \omega_1, \dots, \omega_{n-1}\rangle\sigma = \langle\langle\beta_0,\beta_1, \dots, \beta_{n-1}\rangle,\sigma_n\rangle,$$

где: $\langle\beta_0,\sigma_1\rangle = \omega_0\sigma,$

$\langle\beta_1,\sigma_2\rangle = \omega_1\sigma_1,$

...

$\langle\beta_{n-1},\sigma_n\rangle = \omega_{n-1}\sigma_{n-1}.$

оператор **le** задает последовательный порядок вычислений выражений слева направо.

Оператор **Add** в приведенном ранее семантическом равенстве имеет функциональность вида: $\mathbf{Add} : (\mathcal{E}\times\mathcal{E}) \to \mathbf{S} \to (\mathcal{E}\times\mathbf{S})$.

РАСШИРЕНИЕ СЕМАНТИЧЕСКИХ ОБЛАСТЕЙ

Семантический пункт для оператора сравнения может быть записан аналогичным образом, например:

$E[\![E_0>E_1]\!]\rho = Great*le<E[\![E_0]\!]\rho,E[\![E_1]\!]\rho>$, где оператор **Great** имеет функциональность вида: **$Great : (\mathcal{E}\times\mathcal{E}) \to S \to (T\times S)$**.

Отметим, что, если семантическая область в качестве подобластей включает в себя области различного типа, то результаты выполнения операторов **Add** и **Great** могут быть определены для аргументов, принадлежащих одним областям, и не определены для аргументов, принадлежащих другим.

В частности, значение **Add<x,y>** определено, если **(x|N)** и **(y|N)** – допустимые значения ($\neq\perp$), и неопределено – в противном случае.

ФУНКЦИИ И ПОДПРОГРАММЫ

Введем новые синтаксические конструкции в язык **L**:

E ::= fn I.E | E(E) | rt C

C ::= call E

Предположим, что рассматриваемые функции и подпрограммы не являются рекурсивными. Выражение **rt C** обозначает значение команды C, а **call E**, где **E** обозначает значение некоторой команды, имеет эффект вычисления команды.

Для задания семантики новых конструкций расширим области **D** и $\mathcal{E}$ включением в них области $\sum = (S \to S)$, являющейся областью значений команд и области функций $D \to W$, где $W = S \to (\mathcal{E} \times S)$. Таким образом, $\mathcal{E} = D = T + N + \sum + (D \to W)$.

Заметим, что область $\mathcal{E}$ стала рефлексивной.

ФУНКЦИИ И ПОДПРОГРАММЫ

Новые семантические функции задаются следующим образом:

$$\mathbf{E}\,[\![\underline{\mathbf{fn}}\ \mathbf{I.E}]\!]\boldsymbol{\rho} = \mathbf{strict}\ (\lambda\boldsymbol{\sigma}.\langle(\lambda\boldsymbol{\beta}.\mathbf{E}[\![\mathbf{E}]\!](\boldsymbol{\rho}[\boldsymbol{\beta}/\mathbf{I}])\ \mathbf{in}\ \boldsymbol{\varepsilon}),\boldsymbol{\sigma}\rangle)$$

Результат принадлежит области $\mathbf{D} \to \mathbf{W}$. Отметим также, что вычисление этого выражения не изменяет состояния $\boldsymbol{\sigma}$ и результирующая часть не зависит от состояния.

$$\mathbf{E}\,[\![\mathbf{E}_0(\mathbf{E}_1)]\!]\boldsymbol{\rho} = \mathbf{Apply{*}le}\langle\mathbf{E}[\![\mathbf{E}_0]\!]\boldsymbol{\rho},\mathbf{E}[\![\mathbf{E}_1]\!]\boldsymbol{\rho}\rangle,$$ где

$$\mathbf{Apply}\ \langle\boldsymbol{\beta}_0,\boldsymbol{\beta}_1\rangle = \mathbf{strict}\ (\boldsymbol{\beta}_0|(\mathbf{D} \to \mathbf{W}))(\boldsymbol{\beta}_1).$$

Apply $\langle\boldsymbol{\beta}_0,\boldsymbol{\beta}_1\rangle$ есть элемент области **W**, который применяется к состоянию, являющемуся результатом вычисления "**le**".

$$\mathbf{E}\,[\![\underline{\mathbf{rt}}\ \mathbf{C}]\!]\boldsymbol{\rho} = \mathbf{strict}\ (\lambda\boldsymbol{\sigma}.\langle(\mathbf{C}[\![\mathbf{C}]\!]\boldsymbol{\rho}\ \mathbf{in}\ \mathcal{E}),\boldsymbol{\sigma}\rangle)$$

Результат принадлежит компоненту **S** области $\mathcal{E}$.

ФУНКЦИИ И ПОДПРОГРАММЫ

C⟦call E⟧ρ = Run*E⟦E⟧ρ, где **Run(β) = strict(β|∑)**.

Run(β) есть элемент области **∑**, который применяется к состоянию, получающемуся при выполнении **E**.

Введем теперь, помимо параметризованных функций (выражений), параметризованные подпрограммы (команды).

Новые синтаксические конструкции имеют вид:

E ::= fn I.rt C **C ::= call $E_0(E_1)$**

Соответствующие этим конструкциям семантические функции имеют вид:

E⟦fn I.rt C⟧ρ = strict (λσ.<λβ.(strict (λσ.<C⟦C⟧(ρ[β/I]) in ℰ,σ>) in ℰ,σ>)

C⟦call call $E_0(E_1)$⟧ρ = Run*Apply*le<E⟦E_0⟧ρ, E⟦E_1⟧ρ>

ФУНКЦИИ И ПОДПРОГРАММЫ

Теорема:

$$(\text{fn } I.E_0)(E_1) = (\underline{\text{let}}\ I=E_1\ \underline{\text{in}}\ E_0)$$

Утверждение:

$$\underline{\text{call}}\ (\underline{\text{rt}}\ C) = C$$

Утверждение:

$$\underline{\text{call}}\ (\underline{\text{fn}}\ I.\underline{\text{rt}}\ C)(E) = (\underline{\text{let}}\ I=E\ \underline{\text{in}}\ C)$$