ОСНОВЫ ДЕНОТАЦИОННОЙ СЕМАНТИКИ

Денотационная семантика обеспечивает наиболее удобный стандарт для формулирования вопросов о корректности реализаций языков программирования и программ, написанных на этих языках.

Операционные определения содержат дополнительные детали реализации, необходимость точного определения которых возникает лишь на этапе реализации языка.

Аксиоматическая семантика не дает полного определение языка, но это позволяет программисту концентрировать внимание на тех свойствах, которые его интересуют. В то же время множество аксиом и правил вывода не определяет язык удовлетворительно, поскольку совсем не очевидно существование языка, для которого эти аксиомы и правила вывода удовлетворяются.

Демонстрация существования такого языка осуществляется заданием его денотационной семантики.

ОСНОВНЫЕ КОНЦЕПЦИИ ДЕНОТАЦИОННОГО ПОДХОДА

Денотационный подход базируется на трех основных концепциях:

1. **Аппроксимации**
2. **Понятие фиксированной точки**
3. **Рефлексивные области**

ФУНКЦИИ, АЛГОРИТМЫ И АППРОКСИМАЦИИ

С математической точки зрения **функция – это однозначное отображение аргументов в результаты.**

Любая функция **f** может быть представлена множеством пар вида **(x, y)**, где **x** аргумент, а **y** – результат отображения аргумента **x.**

Это множество пар называется графиком функции **f.**

Большинство функций представляют собой бесконечные отображения, т.е. отображения, определенные на бесконечных областях.

Очевидно, что бесконечные отображения не могут быть представлены внутри конечной машины явным образом.
Лишь некоторые функции могут быть представлены явным образом конечными отображениями.

ФУНКЦИИ, АЛГОРИТМЫ И АППРОКСИМАЦИИ

Алгоритм не задает явным образом никаких функциональных отображений, но позволяет получать любые конечные части требуемого отображения, применяя алгоритм к соответствующим аргументам.

Таким образом, применяя алгоритм к большему числу аргументов, мы получим новые **аппроксимации** отображения, являющиеся приемлемыми для решения конкретной задачи.

Следует различать функции и алгоритмы, которые их представляют. С одной стороны, функция может иметь несколько алгоритмов. С другой стороны, поскольку количество алгоритмов является конечным (и даже счетным), а количество функций является несчетным множеством, то многие функции не имеют алгоритмов, позволяющих их вычислять.

Таким образом, возникает необходимость представлять функции алгоритмами (или конечными аппроксимациями явного отображения) вследствие бесконечной сущности самой функции.

ПОНЯТИЕ ФИКСИРОВАННОЙ ТОЧКИ

Для любой функции **F: D→D** равенство вида **x = F(x)** называется **равенством фиксированной точки (р.ф.т.)**, и любое решение этого равенства называется фиксированной точкой функции **F**.

Рассмотрим несколько примеров **р.ф.т.**

Пример 1. В области функций Nat → Nat:

а) р.ф.т. **g = λx.g(x)+1**
не имеет решения;

б) р.ф.т. **f = λx.if x=0 then 1 else x*f(x–1)**
имеет единственное решение **f(x) = x!**;

в) р.ф.т. **q = λx.if x=0 then 1 else q(x+1)**
имеет много решений:

$$q'(x) = \begin{cases} 1, & \text{если } x = 0, \\ \bot, & \text{если } x > 0. \end{cases}$$

$$q_n(x) = \begin{cases} 1, & \text{если } x = 0, \\ n, & \text{если } x > 0. \end{cases}$$

для любого $n \geq 0$.

ПОНЯТИЕ ФИКСИРОВАННОЙ ТОЧКИ

Пример 2. В области функций S → S:
р.ф.т. **M(<u>while</u> B <u>do</u> C <u>od</u>)σ = Eval(B)σ → M(<u>while</u> B <u>do</u> C <u>od</u>)(M(C)σ),σ**
может рассматриваться как семантическое определение команды **while-do**.

Следует отметить, что с математической точки зрения совсем не очевидно, что это равенство имеет решения.

Первый пример показывает, что р.ф.т. может:
1) не иметь решения;
2) иметь только одно решение;
3) иметь более, чем одно решение.

Если р.ф.т. используется как определение (пример 2), то наиболее интересным является второй случай (единственность решения). Однако, это бывает лишь в исключительных случаях.

ПОНЯТИЕ ФИКСИРОВАННОЙ ТОЧКИ

В то же время, в случае наличия многих решений, можно выбрать минимальное решение (меньшее, чем любое другое). Такое минимальное решение является уже единственным.

Например, минимальным решением в примере 1 (в) является:

$$q'(x) = \begin{cases} 1, & \text{если } x = 0, \\ \bot, & \text{если } x > 0. \end{cases}$$

В этом примере минимальное решение означает теоретико-множественное включение его в любое другое. Так как функции рассматриваются как множества пар теоретико-множественное включение имеет смысл.

Заметим, что не каждое р.ф.т., имеющее много решений, имеет единственное минимальное решение, но данный вопрос о существовании единственных фиксированных точек решается положительно в **семантике минимальной фиксированной точки** (называемой также **теорией Скотта**).

РЕФЛЕКСИВНЫЕ ОБЛАСТИ

Рефлексивная область – это область, содержащая саму себя, т.е., иными словами, циклически определяемая область.

Возьмем в качестве примера простейшее рекурсивное равенство вида: **D = D → D**.

Если **D** – множество, то это равенство не имеет решения, т.к. множество **D → D** имеет всегда большую мощность (больше элементов), чем **D**.

Аналогично этому примеру, для решения семантических равенств, задаваемых рекурсивно, необходимо определить области, изоморфные с областями функций, и описываемые равенством **D = D → D**.

Это означает, что семантические области не могут рассматриваться как обычные множества.

РЕФЛЕКСИВНЫЕ ОБЛАСТИ

В качестве областей используются **частично-упорядоченные множества**.

Частичная упорядоченность в областях обычно соответствует некоторой концепции аппроксимации.

Например, включение одной функции в другую может быть интерпретировано как аппроксимация.

Если $\mathbf{f_1} \sqsubseteq \mathbf{f_2}$, то это означает, что для всех значений аргументов, на которых функция $\mathbf{f_1}$ определена, функция $\mathbf{f_2}$ также определена и имеет то же самое значение. Можно сказать, что функция $\mathbf{f_1}$ аппроксимирует функцию $\mathbf{f_2}$.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

На простых примерах рекурсивных определений рассмотрим все основные концепции семантики минимальной фиксированной точки.

Как было показано на предыдущих примерах, рекурсивные определения в общем случае могут определять функцию неодназначно.

Например, рекурсивное определение

q(x) = if x=0 then 1 else q(x+1),

имеет бесконечно много решений, т.е. бесконечно много функций, удовлетворяющих этому определению.

При этом встает вопрос, какую из этих функций целесообразно выбрать в качестве решения р.ф.т.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Для программирования естественным представляется выбор в качестве решения функции **q'**, так как он соответствует вычислению рекурсивной функции на ЭВМ.

Графиком функции **q'** является множество **graph(q') = {(0,1)}**, записываемое так для практичности, вместо **{(0,1),(1,⊥),(2,⊥), ... }**. Пары **(n,⊥)**, для **n>1**, рассматриваются, при этом, как «призрачные» члены.

С точки зрения программиста любое другое решение (например, график **{(0,1),(1,К),(2,К), ... }**) не является естественным, т.е. его нельзя получить при выполнении данного определения как программы на ЭВМ, хотя с точки зрения математики может быть интересна функция с наибольшим графиком. Очевидно, что функция **q'** является минимальной из всех функций q_n, **n≥0**, так как **graph(q') ⊆ graph(q_n)**, **n≥0**.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Выбор минимальной фиксированной точки рекурсивного определения в качестве решения соответствует обычному операционному вычислению рекурсии.

Последнее означает, что, выполняя рекурсивное определение как программу на ЭВМ, будет получено ничто иное, как минимальная фиксированная точка.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Рассмотрим на примере рекурсивного определения функции факториала процесс получения решения для этого определения.

Пусть функция **fac** задана следующим определением:

fac(x) = if x=0 then 1 else x*fac(x–1).

Решение этого определения функции будем искать в области функций $\mathbf{Nat} \to \mathbf{Nat}_\perp$, где $\mathbf{Nat}_\perp = \mathbf{Nat} \cup \{\perp\}$. В дальнейшем для любого множества **N** запись $\mathbf{N}_\perp$ будет обозначать множество **N**, дополненное элементом ⊥.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

В отличие от **q**, этому определению удовлетворяет только одна функция (факториал), графиком которой является

{(0,1), (1,1), (2,2), (3,6), … , (i,i!), … }

Это бесконечное множество пар, т.е. график функции является бесконечным объектом.

Для понимания бесконечного объекта будем рассматривать конечные его части, строя их шаг за шагом, все более приближаясь к объекту, т.е. будем аппроксимировать бесконечные объекты их конечными представлениями.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Одним из способов получения конечных представлений является выбор множества аргументов и вычисление соответствующих им «результатов».

В этом случае вычисление, использующее определение **fac**, для заданного аргумента (например, равного 3), имеет вид:

fac(3) $\Longrightarrow$ if (3=0) then 1 else 3*fac(3–1) = 3*fac(2)

$\Longrightarrow$ 3*(if (2=0) then 1 else 2*fac(2–1))

= 3*2* fac(1)

$\Longrightarrow$ 3*2*(if (1=0) then 1 else 1*fac(1–1))

= 3*2*1*fac(0)

$\Longrightarrow$ 3*2*1*(if (0=0) then 1 else 0*fac(0–1))

= 3*2*1*1

= 6

Двойная стрелка "$\Longrightarrow$" используется здесь для обозначения шага раскрытия рекурсии.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Рассмотрим другой, более систематический способ получения конечных представлений для бесконечных объектов.

Вместо того, чтобы задавать аргументы и определять количество раскрытий рекурсии для получения результата, будем задавать ограничения на количество раскрытий рекурсии и определять множество аргументов, которые приведут к результату в этом случае.

Тогда аппроксимации для функции, задаваемой определением **fac**, можно получить следующим образом:

1. Нулевое количество раскрытий. Никакой аргумент **n∈Nat** не может привести к получению ответа, так как никакая форма **fac(n)** не может привести к результату без начального раскрытия. Соответствующий график функции будет **{ }**, т.е. пустое множество пар.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

2. Одно раскрытие. Позволяет заменить **fac** его телом (правой частью определения) только один раз.

Таким образом, получим:

$$\mathbf{fac(x) \Longrightarrow \underline{if}\ x=0\ \underline{then}\ 1\ \underline{else}\ x*fac(x-1)}$$

Полученное выражение позволяет вычислить ответ лишь для аргумента **x**, равного **0**, а все другие аргументы требуют дальнейших раскрытий рекурсии для получения ответа. Графиком функции, полученной в результате одного раскрытия, будет **{(0,1)}**.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

3. Два раскрытия. Позволяет вычислить ответ для двух значений аргумента **x** (**0** или **1**).

Для вычисления ответа при аргументе **x**, равном **0**, достаточно и одного раскрытия.

Для аргумента **x**, равного **1**, имеем:

fac(1) ⟹ if 1=0 then 1 else 1*(fac(1–1)) = 1*fac(0)

⟹ 1*(if 0=0 then 1 else 0*fac(0–1)) = 1*1 = 1

Для других аргументов ответ не будет получен.

Таким образом, график функции будет **{(0,1), (1,1)}**.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

4. (i+1) раскрытие, i≥0. В этом случае все аргументы со значением **x≤i** приведут к результату, что даст график функции **{(0,1), (1,1), (2,2), (3,6), … , (i,i!)}**.

График, получающийся на каждом шаге, определяет некоторую функцию.

Обозначим через $\mathbf{fac_i}$ функцию, определяемую на i-ом шаге.

Например, функцию $\mathbf{fac_3}$ определяет график

$$\mathbf{graph(fac_3) = \{(0,1), (1,1), (2,2)\}}.$$

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Отметим некоторые интересные свойства:

- для всех **$i \geq 0$, $graph(fac_i) \subseteq graph(fac_{i+1})$**. Это означает, что частичные функции, получаемые на каждом шаге, согласованы друг с другом в получении ответов;
- для всех **$i \geq 0$, $graph(fac_i) \subseteq graph(factorial)$**.

Это означает, что каждое **fac_i** задает поведение, согласующееся с предельным решением определения **fac**, функцией факториала.

Это влечет:

$$\bigcup_{i=0}^{\infty} graph(fac_i) \subseteq graph(factorial) \qquad (*)$$

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

С другой стороны, если некоторая пара **(a,b)∈graph(factorial)**, то должно существовать конечное **i>0**, такое что **(a,b)∈graph(fac_i)**, поскольку ответы получаются, за конечное число шагов раскрытия.

Таким образом, имеем:

$$\mathbf{graph(factorial)} \subseteq \bigcup_{i=0}^{\infty} \mathbf{graph(fac_i)} \qquad (**)$$

Из **(*)** и **(**)** получаем: $\mathbf{graph(factorial)} = \bigcup_{i=0}^{\infty} \mathbf{graph(fac_i)}$.

Это равенство соответствует операционному пониманию рекурсии и утверждает, что функция **factorial** может быть понята в терминах конечных подфункций $\mathbf{fac_i}$, **i≥0**.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Рассмотренный пример демонстрирует фундаментальный принцип семантики фиксированной точки:

«Значение любой рекурсивно определенной функции есть в точности объединение значений ее конечных подкомпонентов».

Каждая функция $\mathbf{fac_i}$, $\mathbf{i \geq 0}$, есть функция вида $\mathbf{Nat \to Nat_\perp}$.

Определим семейство функций $\mathbf{fac_i}$, $\mathbf{i \geq 0}$, следующим образом:

$\mathbf{fac_0 = \lambda x.\perp}$

$\mathbf{fac_{i+1} = \lambda x.}$ **if** $\mathbf{x=0}$ **then** $\mathbf{1}$ **else** $\mathbf{x * fac_i(x-1)}$ для всех $\mathbf{i>0}$.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

График каждой функции $\mathbf{fac}_i$ получается на i-ом шаге раскрытия **fac**.

Отсюда следует:

- каждое $\mathbf{fac}_i$ является нерекурсивным определением. Это означает, что рекурсивное определение может быть понято в терминах семейства нерекурсивных определений;
- для всех функций $\mathbf{fac}_i$ может быть получен общий формат.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Определим отображение **F** следующим образом:

$$\mathbf{F = \lambda f.\lambda x.\underline{if}\ x=0\ \underline{then}\ 1\ \underline{else}\ x*f(x-1)}$$

Тогда очевидно, что каждая функция $\mathbf{fac_{i+1}}$ может быть задана следующим образом:

$$\mathbf{fac_{i+1} = F(fac_{i+1})}$$

Нерекурсивное отображение F: $(Nat \to Nat_\perp) \to (Nat \to Nat_\perp)$ есть функционал, так как его аргументом и результатом являются функции.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Обозначая нигде не определенную функцию $\lambda x.\perp$ через $\emptyset$, получим:

$$\mathbf{graph(factorial)} = \bigcup_{i=0}^{\infty} \mathbf{graph(F^i(\emptyset))},$$

где $\mathbf{F^i(\emptyset) = F(F^{i-1}(\emptyset))}$.

Еще одним важным фактом является то, что **graph(F(factorial)) = graph(factorial)**.

Отсюда следует, что **F(factorial) = factorial**, а следовательно функция **factorial** является **фиксированной точкой** функционала **F**, поскольку, задавая функцию **factorial** в качестве аргумента, получим ту же самую функцию в качестве результата.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Применим этот способ к определению функции **q**.

Введем функционал **Q**: **(Nat → Nat$_\perp$) → (Nat → Nat$_\perp$)** следующим образом:

$$\mathbf{Q = \lambda q.\lambda x.\underline{if}\ x=0\ \underline{then}\ 1\ \underline{else}\ q(x+1)}$$

Тогда получим следующую последовательность функций:

$q_0 = \lambda x.\perp = \emptyset$,

$graph(q_0) = \{\}$;

$q_1 = Q(\emptyset) = \lambda x.\underline{if}\ x=0\ \underline{then}\ 1\ \underline{else}\ ((\lambda x.\perp)\ (x+1)) = \lambda x.\underline{if}\ x=0\ \underline{then}\ 1\ \underline{else}\ \perp$,

$graph(q_1) = \{(0,1)\}$;

$q_2 = Q^2(\emptyset) = Q(Q(\emptyset)) = \lambda x.\underline{if}\ x=0\ \underline{then}\ 1\ \underline{else}\ ((\lambda x.\underline{if}\ x=0\ \underline{then}\ 1\ \underline{else}\ \perp)(x+1)) = \lambda x.\underline{if}\ x=0\ \underline{then}\ 1\ \underline{else}\ (\underline{if}\ (x+1)=0\ \underline{then}\ 1\ \underline{else}\ \perp)$,

$graph(q_2) = \{(0,1)\}$.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Таким образом, в результате вычисления последовательности аппроксимирующих функций получаем совпадение графиков функций $\mathbf{q_1}$ и $\mathbf{q_2}$, а следовательно, и для всех функций $\mathbf{q_i}$, $\mathbf{i \geq 1}$.

Поскольку $\mathbf{graph(Q^i(\emptyset)) = graph(q_i) = \{(0,1)\}}$ для всякого $\mathbf{i \geq 1}$, то получаем

$$\bigcup_{i=0}^{\infty} \mathbf{graph(Q^i(\emptyset)) = \{(0,1)\}}$$

Пусть **q'** – функция, имеющая график **{(0,1)}**. Легко показать, что **Q(q') = q'**, т.е. **q' – фиксированная точка Q**.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

В отличие от **fac**, определение **q** имеет много решений. Как было показано ранее, каждое решение определения **q** имеет график вида **{(0,1), (1,K), …, (i,K), … }** для некоторого $K \in Nat_\perp$. Пусть q_k – одно из этих решений.

Можно легко показать, что :

1) q_k фиксированная точка Q, т.е. $Q(q_k) = q_k$;

2) $graph(q') \subseteq graph(q_k)$.

Из первого факта следует, что только фиксированные точки соответствующего функционала **Q** могут быть решениями определения **q**. Из второго факта следует, что решение, получаемое методом раскрытия рекурсии, является минимальным. По этой причине его называют **минимальной фиксированной точкой** функционала **Q**.

РЕКУРСИВНЫЕ ОПРЕДЕЛЕНИЯ ФУНКЦИЙ

Рассмотренные примеры рекурсивного определения функций позволяют сформулировать следующие три важных свойства:

1) решение рекурсивного определения существует;

2) критерий минимальности используется для выбора из возможных решений;

3) поскольку метод построения функции, являющейся решением, в точности следует обычному операционному вычислению рекурсивного определения, решение соответствует определяемому вычислительно решению.

В дальнейшем формализуем этот метод построения значения рекурсивного определения и докажем справедливость этих свойств для общего случая рекурсивных определений.