РОССИЙСКАЯ АКАДЕМИЯ НАУК

ИНСТИТУТ ПРОБЛЕМ ИНФОРМАТИКИ

А.М.МИРОНОВ

ТЕОРИЯ ФУНКЦИОНАЛЬНЫХ ПРОГРАММ

Часть 1

Москва

ИПИ РАН

2013

Теория функциональных программ. Часть 1.
/ Миронов А.М. – М.: ИПИ РАН, 2013. – 160 с.
– ISBN 978-5-91993-024-2

В книге рассматриваются математические модели и методы анализа функциональных программ. Основное внимание уделено теории функций, вычисляемых фукциональными программами (эти функции называются наименьшими неподвижными точками функциональных программ). Также излагаются основные методы верификации функциональных программ: метод вычислительной индукции и метод структурной индукции. В книге содержится большое количество задач на доказательство различных свойств функций, вычисляемых функциональными программами. Во второй части будут изложены модели функциональных программ, основанные на лямбда-исчислении и на теории типов.

Книга предназначена для студентов высших учебных заведений, обучающихся по специальностям "теоретические основы информатики"и "информационная безопасность". Также она представляет интерес для специалистов в данных областях.

ISBN 978-5-91993-024-2

Оглавление

Предисловие

Парадигма функционального программирования

Теория функциональных программ является математической основой **функционального программирования**, которое представляет собой одну из парадигм (т.е. совокупность понятий и методов) высокоуровневого программирования. Программы, написанные в парадигме функционального программирования, называются **функциональными программами**.

Главной особенностью функционального программирования является представление программы

- не в виде последовательности действий, преобразующих входные данные в выходные,
- а в виде определений функций, значения которых должны вычисляться этими программами.

Основным достоинством парадигмы функционального программирования является то, что она позволяет описывать функциональную зависимость результата выполнения программы от её входных данных наиболее простым и естественным образом. Как писал один из классиков функционального программирования Лоуренс Паульсон [1],

> Функциональное программирование ставит своей целью придать каждой программе простое математическое толкование, которое должно быть независимо от деталей исполнения.

Использование инструментов функционального программирования (в первую очередь развитых средств типизации данных) позволяет создавать компактные и легко понимаемые программы, в которых отсутствуют излишние детали, связанные с реализацией вспомогательных операций.

На протяжении всей истории своего развития функциональное программирование зарекомендовало себя как один из наиболее эффективных и надёжных инструментов программирования. Применение функционального программирования достигло наибольших успехов в задачах разработки систем искусственного интеллекта и обработки сложноструктурированных данных.

История функционального программирования

Функциональное программирование возникло в пятидесятых годах прошлого века. Основоположником функционального программирования является выдающийся американский информатик **Джон Мак-Карти** [2]. Отметим, что этот специалист является также основателем другого важного направления информатики – искусственного интеллекта.

Первым языком функционального программирования был язык Lisp [3] (его название является аббревиатурой от словосочетания “List Processing”, которое указывает на главное предназначение этого языка – обработка списков). Этот язык был разработан в 1958 г. Джоном Мак-Карти в период его работы в Массачусетском технологическом институте. Первое описание этого языка было опубликовано в 1960 г. в статье [4]. Семантика [5] языка Lisp (а также многих других языков функционального программирования) основана на λ–исчислении ([6], [7]), которое представляет собой логическую теорию, описывающую свойства вычислительных процессов.

Первые применения языка Lisp были связаны с задачами символьной обработки данных и разработкой систем принятия решений. В последующие годы Lisp и его диалекты (Common Lisp [8], Scheme [13] и др.) использовались в самых разных проектах, среди которых: разработка интеллектуальных систем, декодирова-

ние генома человека, проектирование авиалайнеров, разработка Интернет-серверов и служб, систем управления базами данных, научные расчёты и т.д.

Одним из главных направлений развития языков функционального программирования было создание различных средств типизации данных [9]. В конце 70-х — начале 80-х годов XX века были созданы различные модели типизации данных, включающие в себя поддержку таких механизмов как абстракция данных [10] и полиморфизм [11]. В это время появляется множество языков функционального программирования с различными средствами типизации данных: ML (Meta Language) [12], Scheme [13], Hope [14], Miranda [15], и многие другие.

Многообразие языков функционального программирования порождало многочисленные проблемы в деятельности программистов и исследователей в области функционального программирования. Главная из этих проблем заключалась в затруднении взаимодействия между специалистами в области функционального программирования. Это привело их к пониманию того, что для дальнейшего прогресса функционального программирования необходимо разработать универсальный язык функционального программирования, воплощающий главные достоинства наиболее популярных функциональных языков. В качестве такого универсального языка выступил язык Haskell [16], названный так в честь выдающегося американского математика Хаскелла Карри [17]. Первая версия языка Haskell была создана в начале 90-х годов. В настоящее время действителен стандарт Haskell-98. Компиляторы этого языка можно найти на сайте <http://www.haskell.org/>.

Наряду с языком Haskell в настоящее время используются такие языки функционального программирования как Objective Caml (OCaml) [18], F# [19], Standard ML [20], и др. В задачах автоматизированной обработки документов используется функциональный язык XQuery [21].

Среди отечественных разработок по функциональному программированию отметим язык РЕФАЛ ([22], [23]), разработанный в Институте прикладной математики им. М.В.Келдыша РАН группой под руководством В.Ф. Турчина [24].

Некоторые императивные [25] языки программирования под-

держивают типичные для функциональных языков конструкции, такие как функции высшего порядка и др. Это позволяет использовать функциональный стиль программирования в этих языках. Примером языков такого типа является язык Python [26].

Отметим также, что для решения сложных задач, связанных с автоматизацией логических рассуждений, используются следующие программные системы, основанные на парадигме функционального программирования:

1. Coq [27] – интерактивная программная система доказательства теорем, использующая язык функционального программирования Gallina. Coq позволяет записывать математические теоремы и их доказательства, удобно модифицировать их, проверяет их на правильность. Использование данной системы заключается в том, что пользователь интерактивно создаёт доказательство требуемого утверждения, и Coq автоматически проверяет корректность этого доказательства. Также Coq может автоматически находить доказательства в некоторых ограниченных теориях с помощью так называемых тактик. Наиболее успешно Coq применяется в решении задач верификации программ.

2. Agda [28]. Теоретической основой Agda является интуиционистская теория типов Мартин-Лёфа. Автоматическое доказательство теорем с использованием Agda заключается в том, что логические утверждения записываются как типы, а доказательствами являются программы соответствующих типов. Синтаксис языка Agda весьма близок к синтаксису Haskell, на котором система Agda и реализована.

Основные задачи теории функциональных программ

Основные задачи, для решения которых предназначена теория функциональных программ можно разделить на следующие три класса.

1. Верификация различных свойств функциональных программ (к числу которых относятся свойства корректности, безопасности, оптимальности и т.п.)

2. Оптимизирующие преобразования функциональных программ (т.е. преобразования их в такие программы, которые вычисляют те же самые функции, что и исходные программы, но при этом работают быстрее, чем исходные программы).

3. Автоматизация проектирования (синтез) функциональных программ.

В настоящем тексте мы рассматриваем лишь первый из этих классов задач. В главе 4 излагаются наиболее широко используемые методы верификации функциональных программ – вычислительная индукция и структурная индукция. Описание других подходов к верификации функциональных программ можно найти например в работах [29]–[39].

Проблематика оптимизирующих преобразований функциональных программ и автоматизации синтеза функциональных программ изучена существенно слабее чем проблематика их верификации. В качестве примера работ по оптимизирующим преобразованиям функциональных программ укажем основополагающую статью [40].

Обзор литературы и ресурсов в интернете по теории и практике функционального программирования

Среди источников по теории функциональных программ в первую очередь следует назвать статью [41] (которая является переводом пятой главы знаменитой книги [42]). Именно этот текст был положен в основу настоящей книги. Отметим, что доказательства многих ключевых утверждений в статье [41] отсутствуют (или изложены в виде краткого описания идей, на основе которых они могут быть построены), в настоящей книге все эти пробелы устранены.

Основные понятия функционального программирования и теоретические модели функциональных программ, основанные на λ–исчислении, изложены в книге [43]. Общие концепции функционального программирования изложены также в книге [44].

Основные методы верификации функциональных программ можно найти в вышеупомянутой статье [41], и книгах [45] и [46].

Описание подхода к функциональному программированию на основе теории типов можно найти в книге [47].

Теоретические модели функциональных программ, в которых присутствуют операторы взаимодействия программы с окружением, можно найти в диссертации [48].

Теоретические и практические вопросы функционального программирования изложены также в книгах и электронных учебных курсах [49], [50], [51], [52], [53], [54], [55].

Среди современных русскоязычных периодических изданий по функциональному программированию отметим журнал «Практика функционального программирования», который доступен в интернете по адресу `<http://fprog.ru/>`.

Более подробная информация о различных источниках в области теории и практики функционального программирования может быть найдена на сайте [56].

Глава 1

Введение

1.1 Метод функционального программирования

Метод **функционального программирования** заключается в построении описания вычислимых функций в виде систем функциональных уравнений, решениями которых должны быть описываемые функции. Этот метод альтернативен по отношению к методу **императивного программирования**, при котором вычислимая функция определяется путём указания действий, преобразующих входные данные в выходные.

Описания вычислимых функций, построенные на основе метода функционального программирования, называются **функциональными программами (ФП)**.

Метод функционального программирования является одним из наиболее эффективных инструментов программирования, он позволяет разрабатывать в короткие сроки легко понимаемые и надёжные программы.

Функциональное программирование можно также использовать для

- описания **спецификаций**, т.е. свойств разрабатываемых программ, к числу которых относятся, например, свойства корректности, оптимальности, безопасности, устойчивости, и т.д., а также

- быстрого создания прототипов требуемых функций, которые можно затем реализовывать более эффективно на языках программирования низкого уровня.

Во втором случае описание функции в виде ФП рассматривается

- не как описание реального процесса вычисления её значений,
- а как эталон, предназначенный для контроля правильности реализации этой функции на языке программирования низкого уровня.

Для неформальной иллюстрации понятия ФП мы рассмотрим некоторые примеры ФП, описывающих функции на символьных строках (которые мы будем называть просто строками).

1.2 Строки и функции на строках

Мы предполагаем, что задано конечное множество $\mathbf{C}$, элементы которого называются **символами**. В $\mathbf{C}$ входят буквы, цифры, скобки, и некоторые другие символы.

Строкой называется конечная последовательность символов из $\mathbf{C}$. Совокупность всех строк обозначается символом $\mathbf{S}$. Мы считаем, что $\mathbf{C} \subset \mathbf{S}$ (каждый символ можно рассматривать как строку, состоящую из одного этого символа). Существует пустая строка, она не содержит ни одного символа.

Все функции на строках, рассматриваемые в этом тексте, предполагаются частичными (напомним, что функция $f : A \to B$ называется **частичной**, если для некоторых $a \in A$ значение $f(a)$ м.б. не определено).

Для построения ФП мы будем использовать перечисляемые ниже функции на строках.

1. $head : \mathbf{S} \to \mathbf{C}$
   Эта функция определена только на непустых строках, она сопоставляет каждой непустой строке её первый символ.

2. $tail : \mathbf{S} \to \mathbf{S}$
   Эта функция тоже определена только на непустых строках. Она сопоставляет каждой непустой строке u строку, получаемую из u удалением первого символа.

3. $conc : \mathbf{C} \times \mathbf{S} \to \mathbf{S}$
   Для каждой пары $(a, u) \in \mathbf{C} \times \mathbf{S}$ строка $conc(a, u)$ представляет собой строку, получаемую приписыванием символа a спереди к строке u, т.е.

$$conc(a, u) \stackrel{\text{def}}{=} \begin{cases} aa_1 \dots a_n & \text{если } u = a_1 \dots a_n \\ a & \text{если строка } u \text{ пуста} \end{cases}$$

4. $if\_then\_else\ : \mathbf{C} \times \mathbf{S} \times \mathbf{S} \to \mathbf{S}$
   Для каждой тройки $(a, u, v) \in \mathbf{C} \times \mathbf{S} \times \mathbf{S}$

$$if\_then\_else\ (a, u, v) \stackrel{\text{def}}{=} \begin{cases} u, & \text{если } a = 1 \\ v, & \text{иначе} \end{cases}$$

   Ниже запись

$$if\_then\_else\ (a, u, v)$$

   будет обозначаться знакосочетанием

$$\textbf{if } a \textbf{ then } u \textbf{ else } v$$

5. $eq : \mathbf{S} \times \mathbf{S} \to \mathbf{C}$
   Для каждой пары $(u, v) \in \mathbf{S} \times \mathbf{S}$

$$eq(u, v) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{если } u = v \\ 0, & \text{иначе} \end{cases}$$

Для построения ФП могут использоваться и другие функции, в частности такие, которые сами описываются в виде ФП.

1.3 Примеры функциональных программ

1.3.1 Конкатенация строк

Приводимая ниже ФП описывает функцию

$$append : \mathbf{S} \times \mathbf{S} \to \mathbf{S}$$

которая имеет два аргумента. Данная функция преобразует пару строк (u, v) в их **конкатенацию**, т.е. в

- строку $a_1 \ldots a_n\, b_1 \ldots b_m$, если u и v имеют вид соответственно
$$a_1 \ldots a_n \quad \text{и} \quad b_1 \ldots b_m \quad (n, m > 0)$$
- строку v, если строка u пуста,
- строку u, если строка v пуста.

ФП, описывающая функцию *append*, имеет вид

$$\begin{aligned} \varphi(u, v) = \quad & \textbf{if } eq(u, \varepsilon) \textbf{ then } v \\ & \textbf{else } conc(head(u), \varphi(tail(u), v)) \end{aligned} \tag{1.1}$$

где символ ε обозначает пустую строку.

ФП (1.1) состоит из одного функционального уравнения, неизвестная величина в котором – функциональная переменная φ.

Левая часть уравнения (1.1) состоит из

- функциональной переменной φ, соответствующей описываемой функции, и
- списка формальных параметров этой функции (u и v).

Правая часть уравнения (1.1) представляет собой выражение, описывающее связь значения описываемой функции на паре аргументов (u, v) с её значениями на других аргументах.

Словесно эту связь можно выразить следующим образом: значением описываемой функции на паре аргументов (u, v) является

- строка v, если u пуста, и
- строка, состоящая из первого элемента строки u, за которым следует строка, являющаяся значением описываемой функции на паре $(tail(u), v)$, если u непуста.

Нетрудно доказать, что функция *append* является единственным решением уравнения (1.1).

Уравнение (1.1) можно рассматривать как рекурсивный алгоритм вычисления функции *append*.

1.3.2 Инвертирование строки

Другим примером является ФП, описывающая функцию

$$reverse : \mathbf{S} \to \mathbf{S}$$

которая преобразует каждую строку u в строку, получаемую из u её записью в обратном порядке, т.е.

$$reverse(a_1 \dots a_n) = a_n \dots a_1$$

ФП, описывающая функцию $reverse$, имеет вид

$$\begin{aligned} \varphi(u) = \quad & \textbf{if } eq(u, \varepsilon) \textbf{ then } \varepsilon \\ & \textbf{else } append\ (\varphi(tail(u)), head(u)) \end{aligned}$$

В этой ФП используется функция $append$, которую мы описали в виде ФП в предыдущем параграфе.

1.3.3 Поиск подстроки

Третьим примером является ФП, описывающая функцию

$$find : \mathbf{S} \times \mathbf{S} \to \mathbf{C}$$

Значением данной функции на паре строк u, v является символ

- 1, если u является подстрокой строки v, т.е. если v является значением выражения

  $$append(x, append(u, y))$$

  для некоторых строк x, y, и
- 0, иначе.

ФП, описывающая функцию $find$, имеет вид

$$\left\{\begin{array}{l} \varphi_1(u,v) = \quad \textbf{if } \varphi_2(u,v) \textbf{ then } 1 \\ \qquad\qquad\quad \textbf{else } \left(\begin{array}{l} \textbf{if } eq(v,\varepsilon) \textbf{ then } 0 \\ \textbf{else } \varphi_1(u, tail(v)) \end{array}\right) \\ \\ \varphi_2(u,v) = \\ \quad = \quad \textbf{if } eq(u,\varepsilon) \textbf{ then } 1 \\ \qquad\qquad \textbf{else } \left(\begin{array}{l} \textbf{if } eq(v,\varepsilon) \textbf{ then } 0 \\ \textbf{else } \left(\begin{array}{l} \textbf{if } eq\ (head(u), head(v)) \\ \textbf{then } \varphi_2(tail(u), tail(v)) \\ \textbf{else } 0 \end{array}\right) \end{array}\right) \end{array}\right.$$

Данная ФП представляет собой систему из двух функциональных уравнений. Эта система имеет единственное решение, представляющее собой пару функций

$$(find,\ prefix)$$

($find$ соответствует φ_1, а $prefix$ – φ_2), где

- функция $find$ является той функцией, для описания которой предназначена данная ФП, и
- функция $prefix$ является вспомогательной функцией, она имеет вид

  $$prefix : \mathbf{S} \times \mathbf{S} \to \mathbf{C}$$

  её значением на паре строк u, v является символ
  - 1, если u является префиксом строки v, т.е. если v является значением выражения

    $$append(u, x)$$

    для некоторой строки x, и
  - 0, иначе.

Глава 2

Функциональные программы

В этой главе мы формально определяем понятие функциональной программы. Для этого мы сначала вводим необходимые синтаксические конструкции.

2.1 Домены и функции на доменах

2.1.1 Типы и домены

Мы предполагаем, что задано множество $Types$ **типов данных** (или просто **типов**). С каждым типом t связано непустое множество D_t, называемое **доменом типа** t. Элементы домена D_t называются **значениями типа** t.

Ниже мы перечисляем некоторые типы и указываем соответствующие им домены.

- `char`, домен типа `char` состоит из символов
- `string`, домен типа `string` состоит из символьных строк
- `int`, домен типа `int` состоит из целых чисел
- `nat`, домен типа `nat` состоит из натуральных чисел (0, 1, 2, и т.д.)

- `bool`, домен типа `bool` состоит из двух элементов, которые
  - называются **истина** и **ложь**, и
  - обозначаются символами $\top$ и $\bot$ соответственно.

Мы будем обозначать символами **C**, **S**, **Z**, **N** и **B** домены типов `char`, `string`, `int`, `nat` и `bool` соответственно.

2.1.2 Пополненные домены

Для каждого домена D запись $\tilde{D}$ обозначает множество, состоящее из

- всех элементов домена D, и
- ещё одного элемента, который
  - не входит в D,
  - обозначается символом ω
    (для обозначения этого элемента используется один и тот же символ ω для всех доменов), и
  - называется **неопределённым значением**.

Для каждого типа t множество $\tilde{D}_t$ называется **пополненным доменом типа** t. Мы будем рассматривать $\tilde{D}_t$ как **частично упорядоченное множество (ЧУМ)**, отношение порядка на котором определяется следующим образом: $\forall\, d_1, d_2 \in \tilde{D}_t$

$$d_1 \le d_2 \;\;\Leftrightarrow\;\; d_1 = \omega \;\text{ или }\; d_1 = d_2 \tag{2.1}$$

Для каждого списка $\tilde{D}_1, \ldots, \tilde{D}_n$ пополненных доменов мы будем рассматривать их декартово произведение

$$\tilde{D}_1 \times \ldots \times \tilde{D}_n \tag{2.2}$$

тоже как ЧУМ, отношение порядка на котором определяется следующим образом: для любых списков $\bar{d}$ и $\bar{d}'$ из множества (2.2), где

$$\bar{d} = (d_1, \ldots, d_n), \quad \bar{d}' = (d'_1, \ldots, d'_n)$$

мы будем полагать $\bar{d} \le \bar{d}'$, если

$$d_1 \le d'_1,\; \ldots,\; d_n \le d'_n$$

2.1.3 Монотонные функции

Функция f вида

$$f : \tilde{D}_1 \times \ldots \times \tilde{D}_n \to \tilde{D} \tag{2.3}$$

называется **монотонной**, если для каждой пары списков

$$\bar{d}, \bar{d}' \in \tilde{D}_1 \times \ldots \times \tilde{D}_n$$

верна импликация

$$\bar{d} \leq \bar{d}' \quad \Rightarrow \quad f(\bar{d}) \leq f(\bar{d}')$$

Примером немонотонной функции является функция **нестрогого равенства**, которая имеет вид

$$\tilde{D} \times \tilde{D} \to \tilde{\mathbf{B}}$$

и сопоставляет каждой паре $(d_1, d_2) \in \tilde{D} \times \tilde{D}$ значение

$$(d_1 \equiv d_2) \stackrel{\text{def}}{=} \begin{cases} \top & \text{если } d_1 = d_2 \\ \bot & \text{иначе} \end{cases}$$

Немонотонность этой функции следует из того, что

- $(\omega, \omega) \leq (\omega, d)$, где $d \in D$, но
- $(\omega \equiv \omega) = \top, \quad (\omega \equiv d) = \bot, \quad \top \not\leq \bot.$

2.1.4 Естественные продолжения

Пусть задана частичная функция f вида

$$f : \ D_1 \times \ldots \times D_n \to D \tag{2.4}$$

где $D_1, \ldots, D_n, D$ – некоторые домены.

Мы будем обозначать тем же символом f тотальную (т.е. всюду определённую) функцию вида

$$f : \tilde{D}_1 \times \ldots \times \tilde{D}_n \to \tilde{D} \tag{2.5}$$

которая определяется следующим образом: для любого списка

$$\bar{d} = (d_1, \ldots, d_n) \in \tilde{D}_1 \times \ldots \times \tilde{D}_n$$

- если $\forall\, i = 1, \ldots, n \quad d_i \neq \omega$, т.е.

$$\bar{d} \in D_1 \times \ldots \times D_n$$

  то значение функции (2.5) на $\bar{d}$ равно

  - значению частичной функции (2.4) на $\bar{d}$, если это значение определено
  - элементу ω, иначе

- если $\exists\, i \in \{1, \ldots, n\} :\ d_i = \omega$, то значение функции (2.5) на $\bar{d}$ равно ω.

Функция (2.5) называется **естественным продолжением** частичной функции (2.4).

Нетрудно доказать, что естественное продолжение любой частичной функции является монотонной функцией.

2.1.5 Частично упорядоченные множества монотонных функций

Функциональный тип (ФТ) – это запись вида

$$(t_1, \ldots, t_n) \to t \tag{2.6}$$

где $n \geq 0$, и $t_1, \ldots, t_n, t \in Types$.

В случае $n = 0$ та часть ФТ (2.6), которая находится перед стрелкой, является пустой последовательностью.

Для каждого ФТ $ft = \ (t_1, \ldots, t_n) \to t$ знакосочетание D_{ft} обозначает множество всех монотонных функций f вида

$$f : \tilde{D}_{t_1} \times \ldots \times \tilde{D}_{t_n} \to \tilde{D}_t \tag{2.7}$$

В случае $n = 0$ область определения функции (2.7) является одноэлементным множеством.

Мы будем рассматривать D_{ft} как ЧУМ, отношение порядка на котором определяется следующим образом: для любых $f_1, f_2 \in D_{ft}$ мы будем полагать $f_1 \leq f_2$, если

$$\forall\, \bar{d} \in \tilde{D}_{t_1} \times \ldots \times \tilde{D}_{t_n} \quad f_1(\bar{d}) \leq f_2(\bar{d})$$

где отношение неравенства на $\tilde{D}_t$ понимается в смысле определения (2.1).

Обозначим символом $\textcircled{\omega}$ функцию из D_{ft}, принимающую на каждом своём аргументе значение ω.

Нетрудно видеть, что

$$\forall\, f \in D_{ft} \quad \textcircled{\omega} \leq f \tag{2.8}$$

Цепь в D_{ft} – это последовательность $\{f_i \mid i \geq 0\}$ элементов D_{ft}, удовлетворяющая условию:

$$f_0 \leq f_1 \leq f_2 \leq \ldots$$

Это условие эквивалентно тому, что для каждого $\bar{d}$ из множества

$$\tilde{D}_{t_1} \times \ldots \times \tilde{D}_{t_n} \tag{2.9}$$

верно соотношение

$$f_0(\bar{d}) \leq f_1(\bar{d}) \leq f_2(\bar{d}) \leq \ldots \tag{2.10}$$

Согласно определению отношения порядка на пополненных доменах (см. (2.1)), соотношение (2.10) эквивалентно следующему условию:

- либо $\forall\, i \geq 0 \quad f_i(\bar{d}) = \omega$,
- либо $\exists\, i \geq 0, \exists\, d' \in D_t$:

$$\begin{array}{ll} \forall j < i & f_j(\bar{d}) = \omega \\ \forall j \geq i & f_j(\bar{d}) = d' \end{array} \tag{2.11}$$

Обозначим записью

$$\sup\{f_i \mid i \geq 0\}$$

функцию вида (2.7), определяемую следующим образом: для каждого списка $\bar{d}$ из множества (2.9)

$$\begin{array}{l} \sup\{f_i \mid i \geq 0\}(\bar{d}) \stackrel{\text{def}}{=} \\ \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} \omega & \text{если} & \forall\, i \geq 0 \quad f_i(\bar{d}) = \omega \\ d' & \text{если} & \exists\, i \geq 0 : \text{ верно (2.11)} \end{array} \right. \end{array} \tag{2.12}$$

Нетрудно доказать, что функция $\sup\{f_i \mid i \geq 0\}$

- монотонна (т.е. принадлежит D_{ft}), и
- обладает следующими свойствами:
  - $\forall\, i \geq 0 \quad f_i \leq \sup\{f_i \mid i \geq 0\}$
  - если для некоторой функции $f \in D_{ft}$ верно соотношение
    $$\forall\, i \geq 0 \quad f_i \leq f$$
    то $\sup\{f_i \mid i \geq 0\} \leq f$.

2.1.6 Полные частично упорядоченные множества

ЧУМ P называется **полным**, если

- P содержит наименьший элемент, т.е. такой элемент $\mathbf{0}$, что
  $$\forall\, p \in P \quad \mathbf{0} \leq p$$
- для каждой цепи $p_0 \leq p_1 \leq p_2 \leq \ldots$ элементов P существует **точная верхняя грань** этой цепи, т.е. такой элемент
  $$\sup\{p_i \mid i \geq 0\} \in P \tag{2.13}$$
  который обладает следующими свойствами
  - $\forall\, i \geq 0 \quad p_i \leq \sup\{p_i \mid i \geq 0\}$
  - если элемент $p' \in P$ таков, что
    $$\forall\, i \geq 0 \quad p_i \leq p'$$
    то $\sup\{p_i \mid i \geq 0\} \leq p'$.

  Ниже мы будем обозначать элемент вида (2.13) более короткой записью $\sup p_i$.

Из рассуждений в конце параграфа 2.1.5 вытекает, что D_{ft} – полное ЧУМ. Из (2.8) следует, что наименьшим элементом D_{ft} является функция $\textcircled{\omega}$.

2.2 Термы

2.2.1 Переменные и константы

Мы будем предполагать, что для каждого типа t заданы

- счётное множество Var_t **переменных типа** t, и
- множество Con_t **констант типа** t, причём
  - каждой константе $c \in Con_t$ соответствует некоторый элемент пополненного домена $\tilde{D}_t$, называемый **значением** этой константы, и
  - любой элемент $d \in \tilde{D}_t$ является константой типа t, которой соответствует сам элемент d.

Ниже символ ε обозначает константу типа `string`, которой соответствует пустая строка.

2.2.2 Функциональные символы

Мы будем предполагать, что задано некоторое множество Fun, элементы которого называются **функциональными символами (ФС)**, и каждому ФС g сопоставлены

- ФТ $type(g)$, и
- функция из $D_{type(g)}$, обозначаемая тем же символом g.

Разные ФС могут иметь одинаковое обозначение. В частности, запись

$$if\_then\_else$$

обозначает ФС

- ФТ которого имеет вид

  $$(\texttt{bool}, t, t) \to t$$

  где t – произвольный тип, и

- функция, соответствующая которому, имеет вид

$$\tilde{\mathbf{B}} \times \tilde{D}_t \times \tilde{D}_t \to \tilde{D}_t$$

и сопоставляет каждой тройке (d_1, d_2, d_3) из своей области определения элемент

  - d_2, если $d_1 = \top$
  - d_3, если $d_1 = \bot$
  - ω, если $d_1 = \omega$

Нетрудно доказать, что данная функция монотонна.

Ниже мы перечисляем некоторые ФС, входящие в Fun. Каждому такому ФС g соответствует функция из $D_{type(g)}$, являющаяся естественным продолжением функции вида

$$D_1 \times \ldots \times D_n \to D \tag{2.14}$$

где $D_1, \ldots, D_n, D$ – соответствующие домены. Мы будем указывать для каждого из перечисляемых ФС соответствующую ему функцию вида (2.14).

1. ФС $+$, $-$, $\cdot$, им соответствуют арифметические функции на целых или натуральных числах.

2. ФС $\neg$, тип которого имеет вид

$$\texttt{bool} \to \texttt{bool}$$

   и ФС $\wedge$, $\vee$, $\to$, $\leftrightarrow$, тип которых имеет вид

$$(\texttt{bool}, \texttt{bool}) \to \texttt{bool}$$

   Этим ФС соответствуют булевы функции: отрицание, конъюнкция, дизъюнкция, импликация и эквиваленция.

3. ФС $=$, тип которого имеет вид

$$(t, t) \to \texttt{bool}$$

   где t – произвольный тип.

   Этому ФС соответствует функция отображающая каждую пару $(d_1, d_2) \in D_t \times D_t$ в элемент

- $\top$, если $d_1 = d_2$, и
- $\bot$, если $d_1 \neq d_2$.

4. ФС $<$, $\leq$, $>$, $\geq$, тип которых имеет вид

$$(t, t) \to \texttt{bool}$$

где t – произвольный тип, такой, что на домене D_t определено отношение частичного порядка.

Каждому из этих ФС соответствует функция вида

$$D_t \times D_t \to \mathbf{B}$$

отображающая каждую пару $(d_1, d_2) \in D_t \times D_t$ в элемент

- $\top$, если $d_1 < d_2$, $d_1 \leq d_2$, $d_1 > d_2$, $d_1 \geq d_2$ соответственно
- $\bot$, если соответствующее соотношение неверно.

5. ФС *conc*, *head* и *tail*, соответствующие функциям на строках. Описание этих функций приведено в параграфе 1.2.

6. ФС $\textcircled{\omega}$, которому соответствует функция из D_{ft}, где ft – произвольный ФТ, принимающая на каждом своём аргументе значение ω.

2.2.3 Функциональные переменные

Мы будем предполагать, что задано множество $FVar$, элементы которого называются **функциональными переменными**, и каждой функциональной переменной φ сопоставлен некоторый ФТ $type(\varphi)$, причём для каждого ФТ ft множество всех функциональных переменных φ, таких, что $type(\varphi) = ft$, является счётным.

2.2.4 Термы

Ниже мы определяем понятие **терма**. С каждым термом e связан некоторый тип $type(e)$.

Понятие терма определяется индуктивно следующим образом.

1. Для каждого типа t все переменные и константы типа t являются термами типа t.

2. Если g – ФС или функциональная переменная, и

$$type(g) \;=\; (t_1, \ldots, t_n) \to t$$

то для каждого списка термов $e_1, \ldots, e_n$, типы которых равны $t_1, \ldots, t_n$ соответственно, запись

$$g(e_1, \ldots, e_n)$$

является термом типа t.

Ниже мы будем использовать следующие соглашения и обозначения.

1. Термы вида

$$if\_then\_else\ (e_1, e_2, e_3)$$

будут сокращённо записываться в виде

$$e_1\ ?\ e_2 : e_3$$

2. Термы, содержащие ФС

$$+,\ -,\ \cdot,\ =,\ <,\ \leq,\ >,\ \geq,\ \wedge,\ \vee,\ \to,\ \leftrightarrow$$

будут записываться так, как это принято в математических текстах, т.е. каждый терм вида $g(e_1, e_2)$, где g – один из вышеперечисленных ФС, мы будем записывать в виде $e_1\, g\, e_2$.

3. Термы вида

$$conc(e_1, e_2), \quad head(e), \quad tail(e)$$

мы будем записывать в виде $e_1 \cdot e_2, \hat{e}$ и e' соответственно.

4. В целях большей наглядности, термы вида $e_1 \wedge e_2$ и $e_1 \vee e_2$ будут иногда записываться в виде

$$\left\{ \begin{array}{c} e_1 \\ e_2 \end{array} \right\} \quad \text{и} \quad \left[\begin{array}{c} e_1 \\ e_2 \end{array} \right]$$

соответственно.

5. Если ФС или функциональная переменнная g имеет тип $(t_1) \to t$, то в терме вида $g(h(\ldots))$, где h – ФС или функциональная переменнная, скобки слева и справа от $h(\ldots)$ могут опускаться.

6. Для каждого терма e

   - $Var(e)$ обозначает совокупность всех переменных, входящих в e, и
   - $FVar(e)$ обозначает совокупность всех функциональных переменных, входящих в e.

7. Если

   - e – некоторый терм,
   - $x_1, \ldots, x_n$ – список различных переменных, и
   - $e_1, \ldots, e_n$ – список термов, таких, что

   $$\forall\, i = 1, \ldots, n \quad type(e_i) = type(x_i)$$

   то запись

   $$e(e_1/x_1, \ldots, e_n/x_n) \tag{2.15}$$

   обозначает терм, получаемый из e заменой для каждого $i \in \{1, \ldots, n\}$ каждого вхождения переменной x_i в e на терм e_i.

   Если все термы в списке $e_1, \ldots, e_n$ являются константами, и данный список обозначается записью $\bar{d}$, то терм (2.15) можно также обозначать записью $e(\bar{d})$.

2.3 Функции, соответствующие термам

2.3.1 Значения термов

Если терм e содержит только константы и ФС, то этому терму можно сопоставить **значение**, которое

- является элементом множества $\tilde{D}_{type(e)}$, и

- определяется индуктивно следующим образом:
  - если e является константой типа t, то его значение равно тому элементу множества $\tilde{D}_t$, который сопоставлен этой константе,
  - если e имеет вид
    $$g(e_1, \ldots, e_n) \quad (g \in Fun)$$
    то его значение равно значению функции g на списке значений термов e_1, ..., e_n.

Мы будем обозначать значение терма той же записью, которой обозначается сам этот терм.

2.3.2 Функции, соответствующие термам

Пусть заданы

- терм e, не содержащий функциональных переменных, и
- список $\bar{x} = (x_1, \ldots, x_n)$ различных переменных, причём каждая переменная из $Var(e)$ входит в $\bar{x}$
  (список $\bar{x}$ может быть пустым, если e не содержит переменных).

Обозначим символом $D_{\bar{x}}$ декартово произведение

$$\tilde{D}_{type(x_1)} \times \ldots \times \tilde{D}_{type(x_n)} \tag{2.16}$$

и символом ft – ФТ

$$(type(x_1), \ldots, type(x_n)) \to type(e)$$

Запись $e(\bar{x})$ обозначает функцию из D_{ft}, которая

- называется **функцией, соответствующей терму** e, и
- определяется следующим образом: для каждого списка
  $$\bar{d} = (d_1, \ldots, d_n) \in D_{\bar{x}}$$
  значение функции $e(\bar{x})$ на списке $\bar{d}$ обозначается записью $e(\bar{d})$ и равно значению терма
  $$e(d_1/x_1, \ldots, d_n/x_n).$$

Если список $\bar{x}$ пуст, то область определения функции $e(\bar{x})$ состоит из одного элемента, и функция $e(\bar{x})$ сопоставляет этому элементу значение терма e.

Нетрудно доказать, что если функции, соответствующие тем ФС, которые входят в e, монотонны, то функция $e(\bar{x})$ тоже монотонна.

2.3.3 Означивания функциональных переменных

Пусть задано конечное множество Φ функциональных переменных:

$$\Phi = \{\varphi_1, \ldots, \varphi_n\}.$$

Означиванием функциональных переменных из множества Φ называется список пар вида

$$((\varphi_1, f_1), \ldots, (\varphi_n, f_n)) \tag{2.17}$$

где $\forall\, i = 1, \ldots, n \quad f_i \in D_{type(\varphi_i)}$.

Пусть заданы терм e и означивание вида (2.17).

Мы будем обозначать записью

$$e[f_1/\varphi_1, \ldots, f_n/\varphi_n]$$

терм, получаемый из e заменой входящих в него функциональных переменных $\varphi_1, \ldots, \varphi_n$ на новые ФС, которым сопоставлены функции $f_1, \ldots, f_n$ соответственно.

2.4 Функциональные программы

2.4.1 Понятие функциональной программы

Функциональная программа (ФП) – это совокупность Σ формальных равенств вида

$$\begin{cases} \varphi_1(\bar{x}_1) = e_1 \\ \ldots \\ \varphi_n(\bar{x}_n) = e_n \end{cases} \tag{2.18}$$

где

1. $\varphi_1, \ldots, \varphi_n$ – различные функциональные переменные, и
2. для каждого $i = 1, \ldots, n$
   (a) $\varphi_i(\bar{x}_i)$ и e_i – термы одинакового типа,
   (b) $\bar{x}_i$ – список всех переменных из $Var(e_i)$,
   (c) $FVar(e_i) \subseteq \{\varphi_1, \ldots, \varphi_n\}$, и
   (d) функции, соответствующие тем ФС, которые входят в e_i, являются монотонными.

ФП (2.18) можно рассматривать как систему функциональных уравнений, решением которой является произвольный список $\bar{f} = (f_1, \ldots, f_n)$ функций, таких, что

$$\forall\, i = 1, \ldots, n \quad f_i = e_i[f_1/\varphi_1, \ldots, f_n/\varphi_n](\bar{x}_i) \tag{2.19}$$

Если верно (2.19), то мы будем говорить, что ФП Σ является **описанием** списка функций $\bar{f}$.

Ниже запись

$$\Sigma : \left\{ \begin{array}{l} \varphi_1(\bar{x}_1) = e_1 \\ \ldots \\ \varphi_n(\bar{x}_n) = e_n \end{array} \right.$$

означает, что Σ – это ФП, состоящая из равенств, изображённых справа от фигурной скобки. Если Σ состоит из одного равенства, то фигурная скобка может отсутствовать.

2.4.2 Функционал, соответствующий функциональной программе

Пусть Σ – ФП вида (2.18).

Обозначим символом P_Σ декартово произведение

$$D_{type(\varphi_1)} \times \ldots \times D_{type(\varphi_n)}$$

Мы будем рассматривать P_Σ как ЧУМ, отношение порядка на котором определяется следующим образом: для любых списков $\bar{f}$ и $\bar{f}'$ из множества P_Σ, где

$$\bar{f} = (f_1, \ldots, f_n), \quad \bar{f}' = (f'_1, \ldots, f'_n)$$

мы будем полагать $\bar{f} \leq \bar{f}'$, если

$$f_1 \leq f'_1, \ \ldots \ , \ f_n \leq f'_n.$$

Нетрудно видеть, что P_Σ – полное ЧУМ, т.к.

- список
$$(\textcircled{\omega}, \ldots, \textcircled{\omega})$$
является наименьшим элементом в P_Σ, и
- для каждой цепи $\bar{f}_0 \leq \bar{f}_1 \leq \bar{f}_2 \leq \ldots$ элементов P_Σ существует её точная верхняя грань, которая имеет следующий вид:
$$\sup \bar{f}_i \ = \ (\sup f_{i1}, \ldots, \sup f_{in})$$
где $\forall\, i \geq 0 \quad \bar{f}_i = (f_{i1}, \ldots, f_{in})$.

Функционал, соответствующий ФП Σ – это отображение F_Σ вида

$$F_\Sigma : P_\Sigma \to P_\Sigma$$

которое сопоставляет каждому списку функций

$$\bar{f} = (f_1, \ldots, f_n) \in P_\Sigma$$

список функций $\bar{f}' = (f'_1, \ldots, f'_n) \in P_\Sigma$, определяемых следующим образом:

$$\forall\, i \in \{1, \ldots, n\} \quad f'_i \stackrel{\text{def}}{=} e_i[f_1/\varphi_1, \ldots, f_n/\varphi_n](\bar{x}_i)$$

2.4.3 Непрерывные функционалы на полных частично упорядоченных множествах

Пусть P – полное ЧУМ.

Функционал на P – это отображение F вида

$$F : P \to P$$

Функционал $F : P \to P$ называется

- **монотонным**, если для каждой пары p_1, p_2 элементов P верна импликация

$$p_1 \leq p_2 \quad \Rightarrow \quad F(p_1) \leq F(p_2)$$

- **непрерывным**, если
  - он является монотонным, и
  - для каждой цепи $p_0 \leq p_1 \leq p_2 \leq \ldots$ элементов P верно равенство

$$F(\sup p_i) = \sup F(p_i)$$

(отметим, что правая часть данного равенства имеет смысл, поскольку из свойства монотонности F следует, что последовательность $\{F(p_i) \mid i \geq 0\}$ является цепью)

Непрерывность функционала не является следствием его монотонности. Например, рассмотрим функционал F на D_{ft}, где $ft = (\mathtt{nat}) \to \mathtt{nat}$:

$$\forall f \in D_{ft} \quad F(f) \stackrel{\text{def}}{=} \begin{cases} f & \text{если } \forall k \in \mathbf{N} \quad f(k) = k \\ \textcircled{\omega} & \text{иначе} \end{cases}$$

Данный функционал является монотонным. Однако он не является непрерывным. Действительно, цепь

$$\{f_i \mid i \geq 0\}$$

функций из D_{ft}, где для каждого $i \geq 0$ функция f_i соответствует терму

$$(x < i)?\ x : \omega$$

обладает следующими свойствами:

- $\forall i \geq 0 \quad F(f_i) = \textcircled{\omega}$, поэтому

$$\sup F(f_i) = \textcircled{\omega}$$

- $\sup f_i = id = F(\sup f_i)$
  (где id – тождественная функция из D_{ft})

Мы будем использовать следующее обозначение: если F – функционал на P, то записи F^0, F^1, ... обозначают функционалы на P, определяемые следующим образом: $\forall\, p \in P$

- $F^0(p) \stackrel{\text{def}}{=} p$
- $\forall\, i \geq 0 \quad F^{i+1}(p) \stackrel{\text{def}}{=} F(F^i(p))$

2.4.4 Неподвижные точки функционалов на полных частично упорядоченных множествах

Пусть F – функционал на полном ЧУМ P.

Элемент $p \in P$ называется

- **неподвижной точкой (НТ)** функционала F, если верно равенство
$$F(p) = p$$
- **наименьшей НТ (ННТ)** функционала F, если
  - p – НТ функционала F, и
  - для каждой НТ p' функционала F верно неравенство
$$p \leq p'$$

Отметим, что если ННТ функционала F существует, то она единственна, т.к. если p' и p'' две ННТ функционала F, то, согласно определению ННТ, должны быть верны неравенства

$$p' \leq p'' \quad \text{и} \quad p'' \leq p'$$

откуда, в силу свойства антисимметричности отношения частичного порядка, получаем совпадение p' и p''.

Из определений в пунктах 2.4.1 и 2.4.2 следует, что для каждой ФП Σ и каждого списка функций $\bar{f} \in P_\Sigma$ следующие утверждения эквивалентны:

- Σ является описанием списка функций $\bar{f}$
- $\bar{f}$ является НТ функционала F_Σ.

Теорема 1

Для каждого непрерывного функционала F на полном ЧУМ P существует ННТ.

Доказательство.

Определим последовательность $\{p_i \mid i \geq 0\}$ элементов ЧУМ P следующим образом:

- $p_0 \stackrel{\text{def}}{=} \mathbf{0}$, где $\mathbf{0}$ – наименьший элемент P, и
- $\forall\, i \geq 0 \quad p_{i+1} \stackrel{\text{def}}{=} F(p_i)$.

Последовательность $\{p_i \mid i \geq 0\}$ является цепью, это можно доказать по индукции:

- $p_0 = \mathbf{0} \leq p_1$
- для каждого $i \geq 0$ из
  - неравенства $p_i \leq p_{i+1}$, и
  - монотонности функционала F

  следует соотношение

$$p_{i+1} = F(p_i) \leq F(p_{i+1}) = p_{i+2}$$

Обозначим символом p элемент $\sup p_i$. Имеем:

1. p является НТ функционала F, т.к., согласно свойству непрерывности F, верны соотношения

$$F(p) = F(\sup p_i) = \sup F(p_i) = \sup p_{i+1} = p$$

2. p является ННТ функционала F, т.к. если $F(p') = p'$, то

$$\forall\, i \geq 0 \quad p_i \leq p' \tag{2.20}$$

   (2.20) можно доказать по индукции:

   - $p_0 = \mathbf{0} \leq p'$

- если $p_i \leq p'$, то из монотонности F следует

$$p_{i+1} = F(p_i) \leq F(p') = p'$$

Согласно определению точной верхней грани, из (2.20) следует искомое неравенство

$$p = \sup p_i \leq p' \quad \blacksquare$$

2.4.5 Непрерывность функционалов, соответствующих функциональным программам

Теорема 2

Для каждой ФП Σ функционал F_Σ непрерывен.

Доказательство.

Мы рассмотрим лишь случай, когда ФП Σ состоит из одного уравнения (общий случай рассматривается аналогично), т.е. Σ имеет вид

$$\Sigma: \qquad \varphi(\bar{x}) = e \tag{2.21}$$

Каждый подтерм e' терма e обладает следующими свойствами:

- $FVar(e') \subseteq \{\varphi\}$, и
- каждая переменная, входящая в e', содержится в списке $\bar{x}$.

Поэтому для каждого подтерма e' терма e можно определить следующий функционал $F_{e'}$ на $D_{type(\varphi)}$:

$$\forall\, f \in D_{type(\varphi)} \quad F_{e'}(f) \stackrel{\text{def}}{=} e'[f/\varphi](\bar{x})$$

Докажем индукцией по структуре терма e', что функционал $F_{e'}$ непрерывен. Отметим, что из этого следует утверждение теоремы, поскольку $F_\Sigma = F_e$.

Если e' не содержит φ, то множество значений функционала $F_{e'}$ состоит из одного элемента. Очевидно, что такой функционал непрерывен.

Если e' содержит φ, то возможны два случая:

1. $e' = g(e_1, \ldots, e_m)$, где $g \in Fun$, и

2. $e' = \varphi(e_1, \ldots, e_m)$, где φ – функциональная переменная.

Мы разберём лишь первый случай (второй случай разбирается аналогично).

Предположим, что функционалы F_{e_1}, ..., F_{e_m} непрерывны. Докажем, что функционал $F_{e'}$, где

$$e' = g(e_1, \ldots, e_m), \quad g \in Fun$$

также будет непрерывен.

Согласно определению понятия функции, соответствующей терму (пункт 2.3.2), для

- каждой функции $f \in D_{type(\varphi)}$, и
- каждого $\bar{d} \in D_{\bar{x}}$

верны равенства

$$\begin{array}{l} F_{e'}(f)(\bar{d}) = \\ = e'[f/\varphi](\bar{d}) = \\ = g(e_1, \ldots, e_m)[f/\varphi](\bar{d}) = \\ = g(e_1[f/\varphi], \ldots, e_m[f/\varphi])(\bar{d}) = \\ = g(e_1[f/\varphi](\bar{d}), \ldots, e_m[f/\varphi](\bar{d})) = \\ = g(F_{e_1}(f)(\bar{d}), \ldots, F_{e_m}(f)(\bar{d})) \end{array} \tag{2.22}$$

Поэтому, если функции $f_1, f_2 \in D_{type(\varphi)}$ таковы, что $f_1 \leq f_2$, то, в силу монотонности функционалов

$$F_{e_1}, \ldots, F_{e_m}$$

для каждого $\bar{d} \in D_{\bar{x}}$ верны соотношения

$$\begin{array}{l} F_{e_1}(f_1)(\bar{d}) \leq F_{e_1}(f_2)(\bar{d}) \\ \ldots \\ F_{e_m}(f_1)(\bar{d}) \leq F_{e_m}(f_2)(\bar{d}) \end{array}$$

откуда, учитывая (2.22) и монотонность функции g, получаем соотношения

$$\begin{array}{l} F_{e'}(f_1)(\bar{d}) = \\ = g(F_{e_1}(f_1)(\bar{d}), \ldots, F_{e_m}(f_1)(\bar{d})) \leq \\ \leq g(F_{e_1}(f_2)(\bar{d}), \ldots, F_{e_m}(f_2)(\bar{d})) = \\ = F_{e'}(f_2)(\bar{d}) \end{array}$$

т.е. функционал $F_{e'}$ является монотонным.

Докажем, что для каждой цепи $\{f_i \mid i \geq 0\}$ функций из $D_{type(\varphi)}$ верно равенство

$$F_{e'}(\sup f_i) = \sup F_{e'}(f_i) \tag{2.23}$$

Для каждого $i \geq 0$ из неравенства

$$f_i \leq \sup f_i$$

и из монотонности $F_{e'}$ следует неравенство

$$F_{e'}(f_i) \leq F_{e'}(\sup f_i)$$

откуда, согласно определению точной верхней грани, следует неравенство

$$\sup\ F_{e'}(f_i) \leq F_{e'}(\sup f_i)$$

Для доказательства равенства (2.23) докажем, что верно обратное неравенство:

$$F_{e'}(\sup f_i) \leq \sup\ F_{e'}(f_i) \tag{2.24}$$

т.е. что для каждого $\bar{d} \in D_{\bar{x}}$ верно неравенство

$$F_{e'}(\sup f_i)(\bar{d}) \leq \sup\ F_{e'}(f_i)(\bar{d}) \tag{2.25}$$

Согласно (2.22), левую часть (2.25) можно переписать в виде

$$g(F_{e_1}(\sup f_i)(\bar{d}), \ldots, F_{e_m}(\sup f_i)(\bar{d})) \tag{2.26}$$

По индуктивному предположению, верны равенства

$$\begin{aligned} &F_{e_1}(\sup f_i) = \sup F_{e_1}(f_i) \\ &\ldots \\ &F_{e_m}(\sup f_i) = \sup F_{e_m}(f_i) \end{aligned}$$

Поэтому (2.26) можно переписать в виде

$$g\left(\sup\ F_{e_1}(f_i)(\bar{d}), \ldots, \sup\ F_{e_m}(f_i)(\bar{d})\right) \tag{2.27}$$

Из (2.12) следует, что существует номер j, такой, что

$$\begin{array}{l} \sup\ F_{e_1}(f_i)(\bar{d}) = F_{e_1}(f_j)(\bar{d}) \\ \ldots \\ \sup\ F_{e_m}(f_i)(\bar{d}) = F_{e_m}(f_j)(\bar{d}) \end{array}$$

Поэтому (2.27) можно переписать в виде

$$g\left(F_{e_1}(f_j)(\bar{d}), \ldots, F_{e_m}(f_j)(\bar{d})\right) \tag{2.28}$$

Согласно (2.22), значение выражения (2.28) равно

$$F_{e'}(f_j)(\bar{d})$$

Таким образом, доказываемое неравенство (2.25) можно переписать в виде

$$F_{e'}(f_j)(\bar{d}) \leq \sup\ F_{e'}(f_i)(\bar{d})$$

Очевидно, что последнее неравенство верно. ■

Таким образом, согласно теоремам 1 и 2, у функционала F_Σ существует ННТ. Мы будем обозначать эту ННТ символом σ.

Мы будем говорить, что список функций $\bar{f}$ является НТ (ННТ) ФП Σ, если $\bar{f}$ является НТ (ННТ) функционала F_Σ.

Мы будем использовать следующие обозначения.

1. Если ФП Σ имеет вид (2.18), то компоненты её ННТ σ, а также соответствующие им ФС, будут обозначаться записями

   $$\sigma_1, \ldots, \sigma_n$$

   (для каждого $i \in \{1, \ldots, n\}$ запись σ_i соответствует i–му уравнению в Σ).

2. Если ФП обозначается записью Σ^s, где s – некоторый символ, то её ННТ и компоненты этой ННТ (а также соответствующие им ФС) будут обозначаться записями

   $$\sigma^s,\ \sigma^s_1,\ \ldots$$

3. Если Σ состоит из одного уравнения, то список σ состоит из одной функции. В этом случае символ σ обозначает также

   - функцию, являющуюся единственной компонентой этого списка, и
   - ФС, которому соответствует эта функция.

2.4.6 Нахождение наименьших неподвижных точек функциональных программ

ННТ ФП Σ можно найти, например, путём

- вычисления списков термов $\bar{e}_i$, соответствуюших элементам $F^i_\Sigma(\mathbf{0}) \quad (i = 0, 1, \ldots)$, и
- нахождения по $\bar{e}_i$ списка термов $\bar{e}$, соответствущего элементу $\sup F^i_\Sigma(\mathbf{0})$ (который совпадает с σ по теореме 1).

Рассмотрим в качестве примера нахождение данным методом ННТ ФП

$$\Sigma: \quad \begin{cases} \varphi(x) \;=\; (x > 100) & ?\; x - 10 \\ & :\; \varphi\varphi(x + 11) \end{cases} \tag{2.29}$$

Нетрудно установить, что для $i = 1, \ldots, 11$

$$e_i \;=\; (x > 100) \quad ?\; x - 10 \\ \qquad\qquad\quad :\; (x > 101 - i) \quad ?\; 91 : \omega$$

а для $i \geq 11$

$$e_i \;=\; (x > 100) \quad ?\; x - 10 \\ \qquad\qquad\quad :\; (x > 90 - 11 \cdot (i - 11)) \quad ?\; 91 \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad :\; \omega \tag{2.30}$$

При фиксированном x и достаточно больших i выражение

$$x > 90 - 11 \cdot (i - 11)$$

в терме (2.30) будет истинным, поэтому функция σ соответствует терму

$$(x > 100)?\; x - 10 : 91 \quad \blacksquare$$

При вычислениях с термами можно пользоваться следующими теоремами (называемыми **перестановочными законами** для функции *if_ then_ else*).

Теорема 3
Пусть заданы функции

- $p : A \to \tilde{\mathbf{B}}$
- $h_1, h_2 : A \to \tilde{D}_1$
- $g : \tilde{D}_1 \to \tilde{D}$

Определим функции $f_1, f_2 : A \to \tilde{D}$:

$$f_1(a) = g(p(a)?\ h_1(a) : h_2(a))$$

$$f_2(a) = p(a)?\ gh_1(a) : gh_2(a)$$

Тогда $f_2 \leq f_1$, а если $g(\omega) = \omega$, то $f_2 = f_1$.

Доказательство.
Для каждого $a \in A$ имеет место один из следующих случаев:

- $p(a) = \top$
  в этом случае $f_2(a) = gh_1(a) = f_1(a)$
- $p(a) = \bot$
  в этом случае $f_2(a) = gh_2(a) = f_1(a)$
- $p(a) = \omega$
  в этом случае $f_2(a) = \omega \leq g(\omega) = f_1(a)$ ■

Теорема 4 (обобщение теоремы 3).
Пусть заданы функции

- $p : A \to \tilde{\mathbf{B}}$
- $\forall j \in \{1, \ldots, n\} \setminus \{i\} \quad h_j : A \to \tilde{D}_j$
- $h'_i, h''_i : A \to \tilde{D}_i$
- $g : \tilde{D}_1 \times \ldots \times \tilde{D}_n \to \tilde{D}$

Определим функции $f_1, f_2 : A \to \tilde{D}$:

$$f_1(a) = g\begin{pmatrix} h_1(a), \ldots, h_{i-1}(a), \\ p(a)?\ h'_i(a) : h''_i(a), \\ h_{i+1}(a), \ldots, h_n(a) \end{pmatrix}$$

$$f_2(a) = p(a) \quad ?\ g\begin{pmatrix} h_1(a), \ldots, h_{i-1}(a), \\ h'_i(a), \\ h_{i+1}(a), \ldots, h_n(a) \end{pmatrix} : g\begin{pmatrix} h_1(a), \ldots, h_{i-1}(a), \\ h''_i(a), \\ h_{i+1}(a), \ldots, h_n(a) \end{pmatrix}$$

Тогда

- $f_2 \leq f_1$, и
- если $g(\ldots, \omega, \ldots) = \omega$ (где ω в списке аргументов стоит на i-м месте), то $f_2 = f_1$. ■

2.4.7 Примеры неподвижных точек функциональных программ

1. ФП

$$\begin{cases} \varphi_1(x) = (x = 0)?\ 1 : \varphi_1(x-1) + \varphi_2(x-1) \\ \varphi_2(x) = (x = 0)?\ 0 : \varphi_2(x+1) \end{cases}$$

имеет следующие НТ:

$$\begin{pmatrix} (x = 0 \vee x = 1)?\ 1 : n \cdot (x-1) + 1 \\ (x = 0)?\ 0 : n \end{pmatrix}$$

где n – произвольный элемент $\tilde{\mathbf{N}}$.

ННТ этой ФП $= \begin{pmatrix} (x = 0 \vee x = 1)?\ 1 : \omega \\ (x = 0)?\ 0 : \omega \end{pmatrix}$

2. ФП

$$\begin{cases} \varphi_1(x) = (x > 100)?\ x - 10 : \varphi_1\varphi_2(x + 11) \\ \varphi_2(x) = (x > 100)?\ x - 10 : \varphi_2\varphi_1(x + 11) \end{cases}$$

имеет единственную НТ

$$\left(\begin{array}{l} (x > 100)?\ x - 10 : 91 \\ (x > 100)?\ x - 10 : 91 \end{array} \right)$$

3. ФП

$$\varphi(x, y) \ = \ (x = y) \ \ ?\ y + 1 \\ \qquad\qquad\qquad\quad : \varphi(x, \varphi(x - 1, y + 1))$$

имеет, например, следующие НТ:

   (a) $(x = y)?\ y + 1 : x + 1$

   (b) $(x \geq y)?\ x + 1 : y - 1$

   (c) $(x \geq y\ \wedge\ even(x - y))?\ x + 1 : \omega$
   (функция *even* принимает значение $\top$ на чётных числах, и $\bot$ на нечётных)

   Последняя функция – ННТ этой ФП.

4. У ФП

$$\varphi(x) \ = \ \varphi(x)$$

каждая функция является НТ, её ННТ = $\textcircled{\omega}$.

5. Каждая НТ ФП

$$\varphi(\bar{x}) \ = \ e_1?\ \varphi(\bar{x}) : e_2$$

имеет вид

$$e_1?\ e : e_2$$

где e – любой терм типа $type(e_2)$, такой, что $Var(e) \subseteq \bar{x}$.

ННТ этой ФП = $e_1?\ \omega : e_2$.

6. Каждая НТ ФП

$$\varphi(x) \ = \ (x = 0)?\ 1 : \varphi(x + 1)$$

имеет вид

$$(x = 0)?\ 1 : i$$

где i – произвольный элемент $\tilde{\mathbf{N}}$.

ННТ этой ФП = $(x = 0)?\ 1 : \omega$.

2.4.8 Немонотонные функциональные программы

Понятие **немонотонной ФП (НФП)** отличается от понятия ФП только в пункте (2d) определения в параграфе 2.4.1: НФП могут содержать ФС, которым соответствуют немонотонные функции.

У НФП могут отсутствовать НТ или ННТ, например:

1. НФП
$$\varphi(x) \;=\; (\varphi(x) \equiv 0)?\; 1 : 0$$
не имеет НТ.

   Действительно, если бы у этой НФП была НТ f, то

   - из $f(0) = 0$ следовало бы, что $f(0) = 1$, и
   - из $f(0) \neq 0$ следовало бы, что $f(0) = 0$.

2. НФП
$$\varphi(x) \;=\; (\varphi(x) \equiv 0)?\; 0 : 1$$
имеет две НТ (константы 0 и 1), но не имеет ННТ.

2.5 Алгоритмическая полнота функциональных программ

Свойство **алгоритмической полноты** множества ФП заключается в том, что каждая частичная функция на строках, вычислимая в интуитивном смысле, может быть описана некоторой ФП. Мы предполагаем, что для каждой частичной функции f на строках, вычислимой в интуитивном смысле, существует машина Тьюринга M, такая, что f совпадает с функцией, которую вычисляет M.

Теорема 5

Для каждой машины Тьюринга M существует ФП, описывающая ту функцию, которую вычисляет M.

Доказательство.

Пусть M – машина Тьюринга, компонентами которой являются

- множество состояний $Q = \{q_0, q_1, \ldots, q_n\}$, в котором выделены
  - начальное состояние q_0, и
  - заключительное состояние q_n
- алфавит символов $A = \{a_1, \ldots, a_m\}$, которые могут быть написаны в ячейках ленты, причём A содержит пробельный символ $\square$
- отображение переходов

$$\delta : Q \times A \to Q \times A \times \{\texttt{left}, \texttt{right}\}$$

Сопоставим каждому состоянию $q_i \in Q$ функциональную переменную φ_i, где

$$type(\varphi_i) = \quad (\texttt{string}, \texttt{string}) \to \texttt{string}$$

(тип `string` определён в параграфе 2.1.1).

ФП Σ_M, которая описывает функцию, вычисляемую машиной M, состоит из следующих уравнений:

- уравнение, соответствующее той функции, которую вычисляет машина M:

$$\varphi(x) = \varphi_0(\varepsilon, x)$$

- если q_i – незаключительное состояние, и

$$\begin{aligned} &\delta(q_i, a_1) = (q_j, a_k, \texttt{right}) \\ &\delta(q_i, a_2) = \ldots \end{aligned}$$

  то ФП Σ_M содержит уравнение

$$\begin{aligned} \varphi_i(x, y) = \quad (y = \varepsilon) \quad &? \ \varphi_i(x, \square) \\ &: (\hat{y} = a_1) \quad ? \ \varphi_j(a_k \cdot x, y') \\ &\qquad\qquad : (\hat{y} = a_2) \ ? \ldots \end{aligned}$$

  где

– значения, принимаемые переменной x, соответствуют записям на ленте машины M слева от головки, читаемым справа налево, и

– значения, принимаемые переменной y, соответствуют записям на ленте справа от головки (включая символ под головкой)

- если q_i – незаключительное состояние, и

$$\begin{aligned}&\delta(q_i, a_1) = (q_j, a_k, \texttt{left})\\ &\delta(q_i, a_2) = \ldots\end{aligned}$$

то ФП Σ_M содержит уравнение

$$\begin{aligned}&\varphi_i(x, y) =\\ &\quad = (y = \varepsilon) \;\; ?\; \varphi_i(x, \square)\\ &\quad : (x = \varepsilon) \;\; ?\; \varphi_i(\square, y)\\ &\qquad\qquad : (\hat{y} = a_1) \;\; ?\; \varphi_j(x', \hat{x} \cdot a_k \cdot y')\\ &\qquad\qquad\qquad\quad : (\hat{y} = a_2)\; ?\ldots\end{aligned}$$

- уравнение, соответствующее заключительному состоянию q_n:

$$\begin{aligned}&\varphi_n(x, y) = (y = \varepsilon)\; ?\; \varepsilon\\ &\quad : (last(y) = \square)\; ?\; \varphi_n(x, rem(y)) : y\end{aligned}$$

где

– функция $last$ возвращает последний символ своего аргумента, она описывается ФП

$$\varphi(x) = (x' = \varepsilon)\; ?\; \hat{x} : \varphi(x')$$

– функция rem возвращает строку, получаемую удалением последнего символа из своего аргумента, она описывается ФП

$$\varphi(x) = (x' = \varepsilon)\; ?\; \varepsilon : \hat{x} \cdot \varphi(x')$$

– функция, соответствующая функциональной переменной φ_n, возвращает строку, получаемую из значения второго аргумента удалением пробелов в его конце. ■

Глава 3

Вычисление значений наименьших неподвижных точек функциональных программ

В этой главе мы рассматриваем задачу вычисления значений ННТ ФП на заданных значениях аргументов.

В целях простоты изложения, мы рассматриваем в этой главе лишь ФП с одной функциональной переменной φ. Для каждого терма r, рассматриваемого в этой главе, выполнено условие

$$FVar(r) \subseteq \{\varphi\}.$$

3.1 Постановка задачи

Задача вычисления значений ННТ ФП заключается в построении алгоритма, который

- по заданной ФП Σ вида

$$\Sigma : \quad \varphi(\bar{x}) = e \tag{3.1}$$

- и заданному списку $\bar{d} \in D_{\bar{x}}$
  (обозначение $D_{\bar{x}}$ было введено в параграфе 2.3.2)

должен вычислить значение $\sigma(\bar{d})$.

Отметим, что для решения данной задачи не требуется нахождение терма, которому соответствует функция σ.

3.2 Метод решения

Один из возможных методов решения данной задачи заключается в том, что по заданным

- ФП Σ вида (3.1), и
- списку $\bar{d} \in D_{\bar{x}}$

строится последовательность термов

$$C^0_{\Sigma,\bar{d}} \quad C^1_{\Sigma,\bar{d}} \quad C^2_{\Sigma,\bar{d}} \quad \ldots \tag{3.2}$$

называемая **вычислительной последовательностью**. Каждый её член является, в некотором смысле, аппроксимацией искомого значения $\sigma(\bar{d})$. Если вычислительная последовательность (3.2) конечна, то её последний элемент должен быть константой, значение которой равно $\sigma(\bar{d})$.

Вычислительная последовательность (3.2) строится следующим образом.

1. Терм $C^0_{\Sigma,\bar{d}}$ имеет вид $\varphi(\bar{d})$.
2. Если терм $C^i_{\Sigma,\bar{d}}$ содержит функциональную переменную φ, то терм $C^{i+1}_{\Sigma,\bar{d}}$ получается из $C^i_{\Sigma,\bar{d}}$
   (a) заменой в нём некоторых подтермов вида
   $$\varphi(e_1, \ldots, e_n) \tag{3.3}$$
   на термы вида
   $$e(e_1/x_1, \ldots, e_n/x_n) \tag{3.4}$$
   где $(x_1, \ldots, x_n)$ – это список $\bar{x}$ в (3.1), и
   (b) упрощением получившегося терма.

3. Если терм $C^i_{\Sigma,\bar{d}}$ не содержит φ, то он является последним членом последовательности (3.2).

Ниже мы будем называть

- те подтермы вида (3.3) терма $C^i_{\Sigma,\bar{d}}$, которые выбираются для замены согласно пункту (2a), **раскрываемыми** подтермами, и
- замену (3.3) на (3.4) – **раскрытием** подтерма (3.3).

Если среди раскрываемых подтермов терма $C^i_{\Sigma,\bar{d}}$ одни подтермы содержатся в других, то раскрытие таких подтермов осуществляется по принципу "от меньших к большим", т.е. каждый раскрываемый подтерм раскрывается только после того, как будут раскрыты все содержащиеся в нём раскрываемые подтермы.

Преобразования термов, указанные в пунктах (2a) и (2b), объясняются более подробно ниже.

3.3 Вычислительные правила

Правило выбора раскрываемых подтермов терма $C^i_{\Sigma,\bar{d}}$ мы будем называть **вычислительным правилом**. Ниже будут рассматриваться следующие вычислительные правила.

1. **PO** (Parallel Outermost) – раскрываются все самые внешние подтермы вида (3.3) (т.е. не содержащиеся ни в каком подтерме вида (3.3)).
2. **LO** (Left Outermost) – раскрывается самый левый из самых внешних подтермов вида (3.3).
3. **PI** (Parallel Innermost) – раскрываются все самые внутренние подтермы вида (3.3) (т.е. не содержащие подтермов вида (3.3)).
4. **LI** (Left Innermost) – раскрывается самый левый из самых внутренних подтермов вида (3.3).

Мы будем считать, что символ C в (3.2) обозначает вычислительное правило, используемое при построении этой последовательности. Если это правило имеет специальное обозначение, то мы можем использовать это обозначение вместо символа C (т.е., например, если при построении термов вычислительной последовательности используется правило **PO**, то термы, входящие в данную последовательность, могут обозначаться записью $\mathbf{PO}^i_{\Sigma,\bar{d}}$).

3.4 Упрощение терма

Упрощение терма, о котором говорится в пункте (2b) описания построения вычислительной последовательности (3.2), представляет собой последовательность **упрощающих преобразований**.

Для определения понятия упрощающего преобразования введём следующие вспомогательные понятия.

1. Термы r и s называются **эквивалентными**, если

$$\forall f \in D_{type(\varphi)} \quad r[f/\varphi](\bar{x}) = s[f/\varphi](\bar{x}) \tag{3.5}$$

   где $\bar{x}$ состоит из переменных, входящих в r и s.

   Если r и s эквивалентны, то мы будем обозначать этот факт записью

$$r \equiv s.$$

2. Если терм v имеет вид

$$g(v_1, \ldots, v_n) \quad (g \in Fun) \tag{3.6}$$

   то запись $\hat{v}$ обозначает терм

$$g(h_1, \ldots, h_n) \tag{3.7}$$

   где $\forall\, i = 1, \ldots, n$

$$h_i \stackrel{\text{def}}{=} \begin{cases} v_i, & \text{если } v_i \text{ – константа} \\ x_i & \text{(переменная того же типа,} \\ & \text{что и } v_i\text{), иначе} \end{cases} \tag{3.8}$$

   причём все переменные, входящие в (3.7), различны.

Мы будем говорить, что терм s получен из терма r **упрощающим преобразованием**, если s является результатом замены в r подтерма v вида (3.6) на терм v', который равен

- константе d, если $\hat{v} \equiv d$, или
- терму v_i, если h_i – переменная, и $\hat{v} \equiv h_i$.

Нетрудно видеть, что при данной замене

- $v' \equiv v$, поэтому $s \equiv r$, и
- длина v' меньше длины v, поэтому длина s меньше длины r.

Терм r называется **неупрощаемым**, если не существует терма, который получается из r упрощающим преобразованием.

Терм s называется **упрощением** терма r, если

- s – неупрощаемый, и
- либо $s = r$, либо существует последовательность термов r_1, $\ldots$, r_n, такая, что
  - $r_1 = r$, $r_n = s$, и
  - $\forall\, i = 1, \ldots, n-1 \quad r_{i+1}$ получен из r_i упрощающим преобразованием.

Теорема 6

Для любого терма r существует терм s, являющийся его упрощением.

Доказательство.

Определим последовательность термов

$$r_1 \;\; r_2 \;\; \ldots \tag{3.9}$$

следующим образом: $r_1 \stackrel{\text{def}}{=} r$, и для каждого терма r_i в (3.9)

- если r_i неупрощаемый, то он – последний элемент в (3.9),

- иначе определяем r_{i+1} как результат какого-либо упрощающего преобразования терма r_i.

Последовательность (3.9) не может быть бесконечной, потому что длина каждого её терма r_i, где $i > 1$, меньше длины терма r_{i-1}.

Искомый терм s является последним элементом в (3.9). ■

Последовательность (3.9) с заданным первым элементом r может быть построена неоднозначно, т.к. при переходе от r_i к r_{i+1} возможно несколько вариантов выбора упрощающего преобразования.

Тем не менее, как устанавливает теорема 8, последний элемент последовательности (3.9) однозначно определяется её первым элементом.

Мы будем использовать следующие обозначения.

- Для любых термов r и s мы будем обозначать записью

$$r \to s \tag{3.10}$$

  тот факт, что $s = r$, или s получен из r упрощающим преобразованием.

- Если r – терм, и s – вхождение некоторого подтерма в r, то для каждого терма s', тип которого равен типу терма s, запись

$$r(s'/s)$$

  обозначает терм, получаемый из r заменой вхождения s на терм s'.

- Если заданы терм r и некоторые вхождения термов s_1 и s_2 в терм r, то

  – запись $s_1 \subset s_2$ означает, что вхождение терма s_1 содержится во вхождении терма s_2 и не совпадает с ним, и

  – запись $s_1 \subseteq s_2$ означает, что $s_1 \subset s_2$ или вхождение s_1 совпадает с вхождением s_2.

Теорема 7
Если

$$r \to s_1 \quad \text{и} \quad r \to s_2 \tag{3.11}$$

то

$$\exists\, s: \quad s_1 \to s \quad \text{и} \quad s_2 \to s \tag{3.12}$$

(в литературе по теории программирования данное свойство бинарного отношения иногда называют **свойством ромба**, или **свойством Чёрча-Россера**, или **конфлюентностью**).

Доказательство.
Если $s_1 = r$, то $s \stackrel{\text{def}}{=} s_2$, и если $s_2 = r$, то $s \stackrel{\text{def}}{=} s_1$.
Если $s_1 = r(u'_1/u_1)$ и $s_2 = r(u'_2/u_2)$ то возможны следующие случаи.

1. Вхождения u_1 и u_2 в терм r совпадают.

   В этом случае $u'_1 = u'_2$, т.к. нетрудно доказать, что для $i = 1, 2$ варианты

   - $\hat{u}_i \equiv d$, где d – константа, и
   - $\hat{u}_i \equiv h_j$, где u_i имеет вид (3.6), и h_j – переменная, определённая согласно (3.8)

   являются взаимоисключающими, и если имеет место второй вариант, то номер j определён однозначно. Доказательство этого факта опирается на то, что все пополненные домены состоят более чем из одного элемента.

2. $u_1 \subset u_2$.

   Пусть u_2 и $\hat{u}_2$ имеют вид

   $$g(v_1, \ldots, v_n) \quad \text{и} \quad g(h_1, \ldots, h_n) \tag{3.13}$$

   соответственно.

   Из $u_1 \subset u_2$ следует, что $u_1 \subseteq v_i$, где v_i – один из подтермов терма u_2, упомянутых в (3.13).

   Обозначим символом y подтерм $u_2(u'_1/u_1)$ терма s_1. Нетрудно видеть, что $s_1 = r(y/u_2)$.

Термы y и $\hat{y}$ отличаются от термов (3.13) не более чем в i-й компоненте списка, идущего сразу после ФС g. Мы обозначим эти i-е компоненты для y и $\hat{y}$ записями v_i^y и h_i^y соответственно.

Согласно определению правил упрощения, возможен один из следующих вариантов:

(a) $\hat{u}_2 \equiv d$, где d – константа.
В этом случае $\hat{y} \equiv d$, и $s \stackrel{\text{def}}{=} s_2 = s_1(d/y)$.

(b) $\hat{u}_2 \equiv h_j$, где h_j – переменная, и $j \neq i$.
В этом случае $\hat{y} \equiv h_j$ и $u_2' = y' = v_j$.
В качестве s можно взять s_2. Поскольку
- $s_1 = r(y/u_2)$, и
- $s = r(v_j/u_2)$

то $s = s_1(v_j/y)$.

(c) $\hat{u}_2 \equiv h_i$, где h_i – переменная.
В этом случае $u_2' = v_i$, и $s_2 = r(v_i/u_2)$.
Из $\hat{u}_2 \equiv h_i$ следует, что $\hat{y} \equiv h_i^y$.

i. Если h_i^y – переменная, то в качестве s можно взять терм $s_1(v_i^y/y)$.
$s_2 \to s$ следует из равенства $s = s_2(u_1'/u_1)$. По предположению, $u_1 \subseteq v_i$, и после замены u_1 на u_1' подтерм v_i терма s_2 преобразуется в терм v_i^y, в результате чего получается терм s.

ii. Если $h_i^y = d$ (константа), то

$$v_i^y = d \quad \text{и} \quad \hat{y} \equiv d$$

В данном случае $s \stackrel{\text{def}}{=} s_1(d/y)$.
$s_2 \to s$ следует из равенства $s = s_2(d/v_i)$, которое верно потому, что терм v_i^y
- равен константе d, и
- равен терму $v_i(u_1'/u_1)$

что может быть только в том случае, когда

$$v_i = u_1, \quad u_1' = d \quad \text{и} \quad \hat{v}_i \equiv d.$$

3. $u_2 \subset u_1$. Данный случай аналогичен предыдущему.

4. Вхождения термов u_1 и u_2 не пересекаются.

   В этом случае $u_2 \subseteq s_1,\ u_1 \subseteq s_2$, и

$$s \stackrel{\text{def}}{=} s_1(u_2'/u_2) = s_2(u_1'/u_1) \quad \blacksquare$$

Теорема 8
Если s_1 и s_2 – упрощения терма r, то $s_1 = s_2$.

Доказательство.
Если r неупрощаемый, то $r = s_1 = s_2$.

Если r упрощаемый, то существуют последовательности термов

$$\begin{array}{l} r = s_{11},\ \ s_{12},\ \ \ldots,\ \ s_{1n} = s_1 \\ r = s_{11},\ \ s_{21},\ \ \ldots,\ \ s_{m1} = s_2 \end{array}$$

такие, что имеют место соотношения

$$\begin{array}{ccccccc} s_{11} & \to & s_{12} & \to & \ldots & \to & s_{1n} \\ \downarrow & & & & & & \\ s_{21} & & & & & & \\ \downarrow & & & & & & \\ \ldots & & & & & & \\ \downarrow & & & & & & \\ s_{m1} & & & & & & \end{array} \tag{3.14}$$

(мы можем рисовать стрелки, изображающие отношение (3.10), не только горизонтально, но и вертикально).

Диаграмму (3.14) можно достроить до диаграммы

$$\begin{array}{ccccccc} s_{11} & \to & s_{12} & \to & \ldots & \to & s_{1n} \\ \downarrow & & \downarrow & & & & \downarrow \\ s_{21} & \to & s_{22} & \to & \ldots & \to & s_{2n} \\ \downarrow & & \downarrow & & & & \downarrow \\ \ldots & & \ldots & & & & \ldots \\ \downarrow & & \downarrow & & & & \downarrow \\ s_{m1} & \to & s_{m2} & \to & \ldots & \to & s_{mn} \end{array} \tag{3.15}$$

где термы s_{22}, ..., s_{mn} определяются индуктивно: если для некоторых $i \in \{1, \ldots, m-1\}$ и $j \in \{1, \ldots, n-1\}$ уже определены термы s_{ij}, $s_{i,j+1}$, $s_{i+1,j}$, удовлетворяющие условиям

$$\begin{array}{ccc} s_{ij} & \rightarrow & s_{i,j+1} \\ \downarrow & & \\ s_{i+1,j} & & \end{array}$$

то терм $s_{i+1,j+1}$ определяется как терм, удовлетворяющий условиям

$$\begin{array}{ccc} & & s_{i,j+1} \\ & & \downarrow \\ s_{i+1,j} & \rightarrow & s_{i+1,j+1} \end{array}$$

(существование такого терма гарантируется свойством ромба).

Поскольку термы $s_1 = s_{1n}$ и $s_2 = s_{m1}$ неупрощаемы, то все термы в последней строке и в последнем столбце диаграммы (3.15) совпадают. В частности,

$$s_1 = s_{1n} = s_{mn} = s_{m1} = s_2. \quad \blacksquare$$

Таким образом, для каждого терма r существует единственный терм, являющийся его упрощением.

3.5 Функция C_Σ

Пусть заданы вычислительное правило C и ФП Σ вида

$$\Sigma: \quad \varphi(\bar{x}) = e \tag{3.16}$$

Мы будем обозначать записью C_Σ функцию из $D_{type(\varphi)}$, сопоставляющую каждому $\bar{d} \in D_{\bar{x}}$ значение $C_\Sigma(\bar{d})$, которое

- называется **результатом вычисления** ФП Σ на списке $\bar{d}$, и
- определяется следующим образом:
  - по тройке $(C, \Sigma, \bar{d})$ строится вычислительная последовательность (3.2),

– если эта последовательность конечна, то $C_\Sigma(\bar{d})$ равно значению последнего терма в этой последовательности (который, как нетрудно видеть, является константой), и

– если эта последовательность бесконечна, то

$$C_\Sigma(\bar{d}) \stackrel{\text{def}}{=} \omega.$$

Теорема 9
Для любого $\bar{d} \in D_{\bar{x}}$ верно неравенство

$$C_\Sigma(\bar{d}) \leq \sigma(\bar{d}).$$

Доказательство.
Если последовательность (3.2) бесконечна, то

$$C_\Sigma(\bar{d}) = \omega \leq \sigma(\bar{d}).$$

Пусть последовательность (3.2) конечна.
Все термы в последовательности

$$C^0_{\Sigma,\bar{d}}[\sigma/\varphi] \quad C^1_{\Sigma,\bar{d}}[\sigma/\varphi] \quad \ldots \tag{3.17}$$

состоят только из констант и ФС, поэтому каждому из этих термов можно сопоставить некоторое значение. Докажем, что все эти значения совпадают.

Последовательность (3.2) является подпоследовательностью последовательности термов, в которой каждый терм, кроме первого, получается из предыдущего

- раскрытием некоторого подтерма, или
- упрощающим преобразованием.

Поэтому совпадение значений всех термов в (3.17) является следствием следующих двух утверждений.

1. Если термы s и r не содержат переменных, и s получен из r заменой подтерма

$$\varphi(e_1, \ldots, e_n) \tag{3.18}$$

на терм

$$e(e_1/x_1, \ldots, e_n/x_n) \tag{3.19}$$

то значения термов

$$s[\sigma/\varphi] \quad \text{и} \quad r[\sigma/\varphi] \tag{3.20}$$

совпадают.

2. Если терм s получен из терма r упрощающим преобразованием, то $s \equiv r$ (в частности, в том случае, когда s и r не содержат переменных, значения термов (3.20) совпадают).

Утверждение 2 верно потому, что при упрощающем преобразовании происходит замена подтерма на терм, эквивалентный заменяемому.

Обоснуем утверждение 1.

Если s получен из r заменой подтерма (3.18) на терм (3.19), то $s[\sigma/\varphi]$ получен из $r[\sigma/\varphi]$ заменой подтерма

$$\sigma(e_1[\sigma/\varphi], \ldots, e_n[\sigma/\varphi]) \tag{3.21}$$

на терм

$$e[\sigma/\varphi](e_1[\sigma/\varphi]/x_1, \ldots, e_n[\sigma/\varphi]/x_n) \tag{3.22}$$

Функция σ является НТ функционала, сопоставляющего каждой функции $f \in D_{type(\varphi)}$ функцию $e[f/\varphi](\bar{x})$, т.е. для каждого списка

$$(d_1, \ldots, d_n) \in D_{\bar{x}}$$

верно равенство

$$\sigma(d_1, \ldots, d_n) = e[\sigma/\varphi](d_1/x_1, \ldots, d_n/x_n). \tag{3.23}$$

Полагая в (3.23)

$$\forall i = 1, \ldots, n \quad d_i \stackrel{\text{def}}{=} e_i[\sigma/\varphi]$$

получаем, что значения термов (3.21) и (3.22) совпадают.

Следовательно, терм $s[\sigma/\varphi]$ получается из терма $r[\sigma/\varphi]$ заменой его подтерма на терм, значение которого равно значению заменяемого подтерма.

Таким образом, значения термов (3.20) совпадают, т.е. утверждение 1 обосновано.

Из утверждений 1 и 2 следует, что значения всех термов в (3.17) совпадают.

Поскольку

- значение первого терма в (3.17) равно $\sigma(\bar{d})$, и
- значение последнего терма в (3.17) равно $C_\Sigma(\bar{d})$

то мы заключаем, что если вычислительная последовательность (3.2) конечна, то верно равенство

$$C_\Sigma(\bar{d}) = \sigma(\bar{d}). \quad \blacksquare$$

3.6 Вспомогательные понятия

В этом параграфе мы определяем понятия и доказываем теоремы, которые будут использоваться в параграфе 3.7 при доказательстве теоремы 14 о свойстве безопасных вычислительных правил.

3.6.1 Полные раскрытия

Пусть задана ФП $\Sigma: \quad \varphi(\bar{x}) = e$.

Определим последовательность термов $e^{(0)}, e^{(1)}, \ldots$ (называемую **полным раскрытием** для ФП Σ) следующим образом:

- $e^{(0)} = \varphi(\bar{x})$, и
- $\forall\, i \geq 0$ терм $e^{(i+1)}$ получается из $e^{(i)}$ раскрытием каждого подтерма вида $\varphi(\ldots)$, причём эти раскрытия выполняются по принципу "от меньших подтермов к большим", т.е. каждый подтерм вида $\varphi(\ldots)$ раскрывается после того, как будут раскрыты все содержащиеся в нём подтермы вида $\varphi(\ldots)$.

  (эти раскрытия в $e^{(i)}$ можно выполнять в порядке "справа налево": сначала раскрывается самый правый подтерм вида

$\varphi(\ldots)$, затем - самый правый из подтермов вида $\varphi(\ldots)$, начало которого расположено левее начала первого раскрытого подтерма, и т.д.)

Теорема 10

Для любого $i \geq 0$ и любой функции $f \in D_{type(\varphi)}$

$$e^{(i)}[f/\varphi](\bar{x}) = F^i_\Sigma(f). \tag{3.24}$$

Доказательство.

Докажем (3.24) индукцией по i.

При $i = 0$ обе части (3.24) равны функции f.

Пусть (3.24) верно для некоторого $i \geq 0$ и произвольной функции f.

Обозначим новый ФС, которому соответствует функция f, той же буквой f.

Из определения терма $e^{(i+1)}$ следует, что терм

$$e^{(i+1)}[f/\varphi]$$

получается из $e^{(i)}[f/\varphi]$ заменой каждого подтерма вида

$$f(e_1, \ldots, e_n) \tag{3.25}$$

на терм

$$e[f/\varphi](e_1/x_1, \ldots, e_n/x_n) \tag{3.26}$$

причём данные замены выполняются по принципу “от меньших подтермов к большим”.

Обозначим записью $F_\Sigma(f)$ новый ФС, которому соответствует функция $F_\Sigma(f)$.

Из определения функционала F_Σ следует, что терм (3.26) эквивалентен терму

$$F_\Sigma(f)(e_1, \ldots, e_n) \tag{3.27}$$

поэтому терм $e^{(i+1)}[f/\varphi]$ эквивалентен терму, получаемому из терма $e^{(i)}[f/\varphi]$ заменой каждого подтерма вида (3.25) на терм (3.27), т.е. заменой каждого вхождения ФС f на ФС $F_\Sigma(f)$.

Таким образом, верно соотношение

$$e^{(i+1)}[f/\varphi] \equiv e^{(i)}[F_\Sigma(f)/\varphi]$$

откуда следует равенство функций

$$e^{(i+1)}[f/\varphi](\bar{x}) = e^{(i)}[F_\Sigma(f)/\varphi](\bar{x}) \tag{3.28}$$

Поскольку мы предполагаем, что (3.24) верно для каждой функции f, то, в частности, правая часть в (3.28) равна

$$F_\Sigma^i(F_\Sigma(f)) = F_\Sigma^{i+1}(f). \quad \blacksquare$$

3.6.2 Индексированные термы

Индексированным термом (ИТ) называется терм r, такой, что каждому вхождению функциональной переменной φ в r сопоставлено натуральное число, называемое **глубиной** этого вхождения, причем выполнено условие **монотонности**: если u – подтерм терма r, имеющий вид $\varphi(\ldots)$, то глубина первого вхождения φ в u меньше или равна глубины остальных вхождений φ в u.

Для каждого ИТ r и каждого $j \geq 0$

- $in(r, j)$ обозначает количество вхождений в r символа φ глубины j, и
- $in(r)$ обозначает последовательность $\{in(r, i) \mid i \geq 0\}$.

3.6.3 Σ–переходы

Пусть задана ФП $\Sigma: \quad \varphi(\bar{x}) = e$, где $\bar{x} = (x_1, \ldots)$.

Σ–**переходом** называется пара ИТ (r, s), удовлетворяющая одному из следующих условий.

1. $r \to s$, и глубина каждого вхождения φ в s равна глубине соответствующего вхождения φ в r.
   (поскольку в данном случае $s = r$ или s получается из r удалением некоторых символов или заменой их на константу, то каждому вхождению φ в s соответствует некоторое вхождение φ в r)
2. $s = r(u'/u)$, где $u = \varphi(e_1, \ldots), \quad u' = e(e_1/x_1, \ldots)$, причем для каждого вхождения φ в s, содержащегося в u',

- если это вхождение содержится в подтерме e_i, подставленном вместо переменной x_i в e, то глубина этого вхождения на 1 больше глубины соответствующего вхождения φ в u
- иначе глубина этого вхождения на 1 больше глубины первого вхождения φ в u

и для каждого вхождения φ в s, не содержащегося в u', его глубина равна глубине соответствующего вхождения φ в r.

Если пара (r, s) является Σ–переходом, то этот факт обозначается записью $r \xrightarrow{\Sigma} s$. Σ–переход (r, s) называется **упрощением** или **раскрытием**, если s получается из r в результате упрощающего преобразования или раскрытия соответственно.

Для любых ИТ r и s запись $r \xrightarrow{\Sigma*} s$ означает, что существует последовательность термов r_0, ..., r_n, обладающая свойствами

$$r = r_0 \xrightarrow{\Sigma} \ldots \xrightarrow{\Sigma} r_n = s \tag{3.29}$$

В излагаемых ниже теоремах используется следующее отношение порядка на множестве бесконечных последовательностей натуральных чисел: если $\bar{a} = \{a_i \mid i \geq 0\}$ и $\bar{b} = \{b_i \mid i \geq 0\}$ – две таких последовательности, то

$$\bar{a} \leq \bar{b} \;\Leftrightarrow\; \left\{ \begin{array}{ll} \bar{a} = \bar{b}, & \text{или} \\ \exists\, i \geq 0 & \left\{ \begin{array}{l} a_i < b_i \\ \forall\, j:\ 0 \leq j < i \quad a_j = b_j \end{array} \right. \end{array} \right.$$

(такой порядок называется **лексикографическим**).

Теорема 11
Если $r \xrightarrow{\Sigma*} s$, то $in(r) \geq in(s)$.

Доказательство.
Достаточно рассмотреть случай $r \xrightarrow{\Sigma} s$.

1. s получается из r упрощающим преобразованием, т.е. удалением некоторых символов (или заменой их на константу).

Поскольку глубина каждого вхождения φ в s совпадает с глубиной соответствующего вхождения φ в r, то

$$\forall\, j \geq 0 \quad in(r,j) \geq in(s,j)$$

откуда следует $in(r) \geq in(s)$.

2. $s = r(u'/u)$, где $u = \varphi(e_1, \ldots)$, $u' = e(e_1/x_1, \ldots)$.

   Пусть первое вхождение φ в u имеет глубину k.

   В этом случае

   - глубина вхождений φ в u больше или равна k,
   - для каждого $j \geq 0$ число вхождений φ глубины j в s, находящихся вне u', будет то же, что и число вхождений φ глубины j в r, находящихся вне u,
   - в подтерме u' терма s нет вхождений φ глубины k, и
   - для каждого $j < k$ в подтерме u терма r вхождения φ глубины j отсутствуют, и в подтерме u' терма s их тоже нет.

   Следовательно, верны соотношения

   - $\forall\, j : 0 \leq j < k \quad in(r,j) = in(s,j)$, и
   - $in(r,k) > in(s,k)$

   из которых следует $in(r) \geq in(s)$. ■

Пусть заданы ФП $\Sigma:\ \varphi(\bar{x}) = e$, вычислительное правило C, и список $\bar{d} \in D_{\bar{x}}$.

Термы вычислительной последовательности $\{C^i_{\Sigma,\bar{d}} \mid i \geq 0\}$ можно рассматривать как ИТ, такие, что каждая пара соседних термов удовлетворяет соотношению

$$C^i_{\Sigma,\bar{d}} \overset{\Sigma*}{\to} C^{i+1}_{\Sigma,\bar{d}} \tag{3.30}$$

Будем считать, что глубина вхождения φ в $C^0_{\Sigma,\bar{d}} = \varphi(\bar{d})$ равна 0.

Теорема 12

Если вычислительная последовательность $\{C^i_{\Sigma,\bar{d}} \mid i \geq 0\}$ бесконечна, то $\forall\, M \geq 0 \quad \exists\, N \geq 0$:

$$\forall\, j = 0, \ldots, M \qquad in(C^N_{\Sigma,\bar{d}}, j) = in(C^{N+1}_{\Sigma,\bar{d}}, j) = \ldots \tag{3.31}$$

Доказательство.
Индукция по M.

1. Если $M = 0$, то $N = 1$.

2. Пусть для некоторого $M \geq 0$ существует номер N, для которого верно (3.31). Тогда из теоремы 11 и из определения лексикографического порядка следует, что

   $$in(C^N_{\Sigma,\bar{d}}, M+1) \geq in(C^{N+1}_{\Sigma,\bar{d}}, M+1) \geq \ldots \tag{3.32}$$

   Цепочка (3.32) не может бесконечно убывать, т.е. $\exists\, N' \geq N$:

   $$in(C^{N'}_{\Sigma,\bar{d}}, M+1) = in(C^{N'+1}_{\Sigma,\bar{d}}, M+1) = \ldots \tag{3.33}$$

   Из (3.31) и из неравенства $N' \geq N$, следует, что для каждого $j = 0, \ldots, M$ цепочка (3.33), в которой второй аргумент функции in будет заменён на j, также будет верной.

   Таким образом, для $M+1$ в качестве искомого номера можно взять N'. ■

Теорема 13
Если вычислительная последовательность $\{C^i_{\Sigma,\bar{d}} \mid i \geq 0\}$ бесконечна, то $\forall\, M \geq 0 \quad \exists\, N$: в каждом раскрываемом подтерме терма $C^N_{\Sigma,\bar{d}}$ первое вхождение φ имеет глубину $> M$.

Доказательство.
Выберем N таким, чтобы было верно (3.31).
Если существует раскрываемый подтерм терма $C^N_{\Sigma,\bar{d}}$, в котором первое вхождение φ имеет глубину $j \leq M$, то

$$in(C^N_{\Sigma,\bar{d}}, j) > in(C^{N+1}_{\Sigma,\bar{d}}, j)$$

что противоречит (3.31). ■

3.7 Безопасные вычислительные правила

Пусть заданы ФП $\Sigma:\ \varphi(\bar{x}) = e$ и вычислительное правило C.

Правило C называется **безопасным** для Σ, если для любого $\bar{d} \in D_{\bar{x}}$, такого, что последовательность $\{C^i_{\Sigma,\bar{d}} \mid i \geq 0\}$ бесконечна, выполнено условие: $\forall\, i \geq 0$

$$C^{i,\omega}_{\Sigma,\bar{d}}[\sigma/\varphi] = \omega \tag{3.34}$$

где $C^{i,\omega}_{\Sigma,\bar{d}}$ получается из $C^i_{\Sigma,\bar{d}}$ заменой каждого самого внешнего раскрываемого подтерма (т.е. не содержащегося в другом раскрываемом подтерме) на константу ω.

Теорема 14
Если C безопасно для Σ, то $\forall\, \bar{d} \in D_{\bar{x}}$

$$C_\Sigma(\bar{d}) = \sigma(\bar{d}). \tag{3.35}$$

Доказательство.

Как было установлено в теореме 9, равенство (3.35) может нарушаться только в том случае, когда

$$\omega < \sigma(\bar{d}) \tag{3.36}$$

и вычислительная последовательность $\{C^i_{\Sigma,\bar{d}} \mid i \geq 0\}$ бесконечна.

Поскольку $\sigma = \sup F^i_\Sigma(\textcircled{\omega})$, то $\exists\, n$:

$$\sigma(\bar{d}) = F^n_\Sigma(\textcircled{\omega})(\bar{d}) = e^{(n)}[\textcircled{\omega}/\varphi](\bar{d}) \tag{3.37}$$

где $e^{(n)}$ – соответствующий терм из полного раскрытия для Σ (второе равенство верно согласно теореме 10).

Из определения полного раскрытия следует, что существуют ИТ $r_0, \ldots, r_k$, такие, что

$$\varphi(\bar{d}) = e^{(0)}(\bar{d}) = r_0 \xrightarrow{\Sigma} \ldots \xrightarrow{\Sigma} r_k = e^{(n)}(\bar{d}) \tag{3.38}$$

причём для каждого $i = 0, \ldots, k-1$ терм r_{i+1} получается из r_i раскрытием некоторого подтерма.

Из (3.36), (3.37), и определения r_k следует, что

$$\omega < r_k[\textcircled{\omega}/\varphi]. \tag{3.39}$$

Обозначим символом M максимальную глубину вхождения φ в r_k.

По теореме 13, $\exists\, N$: в каждом раскрываемом подтерме терма $C^N_{\Sigma,\bar{d}}$ первое вхождение φ имеет глубину $> M$.

Поскольку для каждого $i = 0, \ldots, N-1$ верно (3.30), то существует последовательность ИТ s_0, ..., s_l, такая, что

$$\varphi(\bar{d}) = s_0 \overset{\Sigma}{\to} \ldots \overset{\Sigma}{\to} s_l = C^N_{\Sigma,\bar{d}} \tag{3.40}$$

Если в (3.40) есть пара соседних Σ–переходов, первый из которых – упрощение, а следующий за ним – раскрытие, то мы преобразуем (3.40) в соответствии со следующим правилом. Пусть i – наименьший номер, такой, что (3.40) содержит пару

$$s_i \overset{\Sigma}{\to} s_{i+1} = s_i(u'/u) \overset{\Sigma}{\to} s_{i+2} = s_{i+1}(v'/v) \tag{3.41}$$

где $u = g(u_1, \ldots)$, $g \in Fun$, и $v = \varphi(v_1, \ldots)$.

1. Если вхождения u' и v в s_{i+1} не пересекаются, или $v \subseteq u'$, то можно считать, что $v \subseteq s_i$.

   Заменим в (3.40) пару (3.41) на пару Σ–переходов

   $$s_i \overset{\Sigma}{\to} s_i(v'/v) \overset{\Sigma}{\to} s_{i+2}$$

2. Если $u' \subset v = \varphi(v_1, \ldots)$, то $u' \subseteq v_j$ для некоторого j.

   В этом случае $\exists\, w = \varphi(w_1, \ldots) \subseteq s_i : u \subseteq w_j$.

   Заменим в (3.40) пару (3.41) на последовательность

   $$s_i \overset{\Sigma}{\to} s_{i1} \overset{\text{def}}{=} s_i(e(w_1/x_1, \ldots)/w) \overset{\Sigma}{\to} s_{i2} \overset{\Sigma}{\to} \ldots \overset{\Sigma}{\to} s_{ip} = s_{i+2}$$

   где $\forall\, q = 1, \ldots, p-1 \quad s_{i(q+1)}$ получается из s_{iq} заменой u на u' в одном из вхождений w_j.

Обозначим получившуюся последовательность тем же знакосочетанием (3.40), что и исходную последовательность. Если получившаяся последовательность будет содержать пару соседних Σ–переходов, первый из которых – упрощение, а следующий за ним – раскрытие, то опять преобразуем эту последовательность в соответствии с описанным выше правилом, и т.д.. Будем выполнять такие преобразования до тех пор, пока не получится такая последовательность (3.40), в которой сначала идут раскрытия, а затем - упрощения. Нетрудно видеть, что данный процесс завершается после конечного числа шагов.

Пусть s_m – тот член последовательности (3.40), на котором заканчиваются раскрытия и начинаются упрощения, т.е. все Σ–переходы в подпоследовательности

$$\varphi(\bar{d}) = s_0 \stackrel{\Sigma}{\to} \ldots \stackrel{\Sigma}{\to} s_m \tag{3.42}$$

являются раскрытиями, и все Σ–переходы в подпоследовательности

$$s_m \stackrel{\Sigma}{\to} \ldots \stackrel{\Sigma}{\to} s_l = C^N_{\Sigma,\bar{d}} \tag{3.43}$$

являются упрощениями.

Для каждого терма s и каждого $i \geq 0$ будем обозначать записью $\textcircled{\omega}_i(s)$ терм, получаемый из s заменой каждого вхождения φ глубины $> i$ на ФС $\textcircled{\omega}$.

Нетрудно видеть, что

$$\textcircled{\omega}_M(s_m)[\sigma/\varphi] = \omega \tag{3.44}$$

(это следует из того, что терм $C^{N,\omega}_{\Sigma,\bar{d}}[\sigma/\varphi]$, значение которого равно ω, можно получить из левой части (3.44) упрощениями и заменами некоторых вхождений ω на термы вида $\sigma(\ldots)$).

Если s_m содержит подтерм u вида $\varphi(u_1, \ldots)$, в котором первое вхождение φ имеет глубину $\leq M$, то добавим к (3.42) Σ–переход

$$s_m \stackrel{\Sigma}{\to} s_m(u'/u) \quad \text{где} \quad u' = e(u_1/x_1, \ldots) \tag{3.45}$$

Из (3.44) следует равенство

$$\textcircled{\omega}_M(s_m(u'/u))[\sigma/\varphi] = \omega$$

которое обосновывается соотношениями

$$\textcircled{\omega}_M(u')[\sigma/\varphi] \le \textcircled{\omega}_{M+1}(u')[\sigma/\varphi] = \textcircled{\omega}_M(u)[\sigma/\varphi]$$

Обозначим последовательность Σ–переходов, полученную добавлением (3.45) к (3.42), той же записью, что и исходную последовательность (3.42).

Будем выполнять описанную выше операцию добавления Σ–переходов к (3.42) до тех пор, пока последний терм s_m в получившейся последовательности не перестанет содержать вхождения φ глубины $\le M$. Из вышесказанного следует, что для этого терма верно равенство (3.44), из которого следует равенство

$$s_m[\textcircled{\omega}/\varphi] = \omega \tag{3.46}$$

Преобразуем последовательности (3.38) и (3.42) в такие последовательности Σ–переходов, в которых каждая пара соседних Σ–переходов удовлетворяет следующему условию: пусть данная пара имеет вид

$$a \overset{\Sigma}{\to} b = a(u'/u) \overset{\Sigma}{\to} c = b(v'/v) \tag{3.47}$$

тогда номер позиции в терме a, в которой расположен первый символ подтерма u, меньше номера позиции в терме b, в которой расположен первый символ подтерма v.

Предположим, что данное условие не выполнено для некоторой пары соседних переходов (3.47). Тогда возможен один из двух следующих случаев.

1. Вхождения u' и v в терм b не пересекаются.

   В этом случае заменим (3.47) на пару Σ–переходов

   $$a \overset{\Sigma}{\to} b' \overset{\text{def}}{=} a(v'/v) \overset{\Sigma}{\to} b'(u'/u) = c$$

2. $\exists\, w = \varphi(w_1, \ldots) \subseteq a :\ u \subseteq w_j$ для некоторого j, $v = w(u'/u)$.

   В этом случае заменим пару (3.47) на последовательность Σ–переходов

   $$a \overset{\Sigma}{\to} b_1 \overset{\text{def}}{=} a(e(w_1/x_1, \ldots)/w) \overset{\Sigma}{\to} b_2 \overset{\Sigma}{\to} \ldots \overset{\Sigma}{\to} b_p = c$$

   где $\forall\, q = 1, \ldots, p-1 \quad b_{q+1}$ получается из b_q заменой u на u' в одном из вхождений w_i.

Нетрудно убедиться, что после конечного числа преобразований данного типа последовательности (3.38) и (3.42) преобразуются в такие последовательности Σ–переходов, которые будут удовлетворять изложенному выше условию. Мы будем обозначать результаты данных преобразований теми же записями, что и исходные последовательности (3.38) и (3.42).

Таким образом, последовательности (3.38) и (3.42) обладают следующими свойствами:

- все Σ–переходы в них являются раскрытиями, причем в каждом из этих Σ–переходов (кроме последних) раскрываемый подтерм расположен левее раскрываемого подтерма в следующем за ним Σ–переходе
- (3.38) имеет вид $\varphi(\bar{d}) = r_0 \stackrel{\Sigma}{\to} \ldots \stackrel{\Sigma}{\to} r_k$, и все вхождения φ в r_k имеют глубину $\leq M$
- (3.42) имеет вид $\varphi(\bar{d}) = s_0 \stackrel{\Sigma}{\to} \ldots \stackrel{\Sigma}{\to} s_m$, и все вхождения φ в s_m имеют глубину $> M$.

Определим бинарное отношение R на множестве термов как множество пар вида (r_i, s_j), где $r_i \in$ (3.38), $s_j \in$ (3.42), обладающих следующими свойствами:

- r_i можно представить в виде конкатенации

$$a_1\ u_1\ \ldots\ a_n\ u_n\ a \tag{3.48}$$

  где $n \geq 0$ (т.е. компоненты a_1, …, u_n могут отсутствовать), и

  - $u_1, \ldots, u_n$ – вхождения подтермов, каждый из которых имеет вид $\varphi(\ldots)$
  - $a_1, \ldots, a_n, a$ – символьные строки
  - если $i < k$, то раскрываемый подтерм терма r_i содержится в подстроке a

- s_j можно представить в виде конкатенации

$$a_1\ v_1\ \ldots\ a_n\ v_n\ a \tag{3.49}$$

  где $n \geq 0$, и

– $v_1, \ldots, v_n$ – вхождения подтермов

– $a_1, \ldots, a_n, a$ – те же символьные строки, что и в (3.48)

– если $j < m$, то раскрываемый подтерм терма s_j содержится в подстроке a

Отметим, что

$$(s_0, r_0) \in R \tag{3.50}$$

(в данном случае $n = 0$ и $a = \varphi(\bar{d})$).

Если $(r_i, s_j) \in R$ и $i < k$, то $j < m$, т.к. если $j = m$, то терм s_m содержит подтерм вида $\varphi(\ldots)$ терма r_i (в подстроке a), и

- все вхождения φ в этот подтерм имеют глубину $\leq M$, но
- все вхождения φ в s_m имеют глубину $> M$.

Аналогично, если $(r_i, s_j) \in R$ и $j < m$, то $i < k$.

Докажем, что если $(r_i, s_j) \in R$ и $i < k$, то $\exists\, i' \geq i$ и $\exists\, j' \geq j$: $(i', j') \neq (i, j)$ и $(r_{i'}, s_{j'}) \in R$. Отметим, что на основании (3.50) отсюда будет следовать $(r_k, s_m) \in R$.

Обозначим символами α и β номера позиций в подстроке a, в которых расположены первые символы раскрываемых подтермов u и v термов r_i и s_j соответственно.

Рассмотрим три случая.

1. $\alpha = \beta$.

   В данном случае $u = v$, $i' = i + 1$, $j' = j + 1$.

2. $\alpha < \beta$.

   Этот случай невозможен, т.к. если

   - s_j в позиции α подстроки a содержит символ φ глубины $\leq M$, и
   - раскрываемый подтерм терма s_j расположен правее этого вхождения φ

   то поскольку в каждом из Σ–переходов в (3.42) (кроме первого) раскрываемый подтерм расположен правее раскрываемого подтерма в предыдущем Σ–переходе, то

- символ φ глубины $\leq M$ будет входить в s_{j+1}, ..., s_m,
- но все вхождения φ в s_m имеют глубину $> M$.

3. $\alpha > \beta$.

   Пусть r_i и s_j можно представить в виде конкатенаций (3.48) и (3.49) соответственно, компоненты которых обладают указанными выше свойствами.

   Представим подстроку a в виде конкатенации $b\ v\ c$, где v – раскрываемый подтерм в s_j.

   Нетрудно доказать, что $\exists\, i' \geq i$, $\exists\, j' \geq j$: $(i', j') \neq (i, j)$ и $r_{i'}$ и $s_{j'}$ можно представить в виде конкатенаций

$$\begin{aligned} r_{i'} &= a_1\ u_1\ \ldots\ a_n\ u_n\ b\ w\ c \\ s_{j'} &= a_1\ v_1\ \ldots\ a_n\ v_n\ b\ w'\ c \end{aligned}$$

   соответственно, где w и w' – термы, w имеет вид $\varphi(\ldots)$, и

   - либо $i' = k$ и $j' = m$,
   - либо раскрываемые термы в $r_{i'}$ и $s_{j'}$ содержатся в c.

Таким образом, во всех случаях существует пара $(r_{i'}, s_{j'}) \in R$ с требуемыми свойствами.

Как было отмечено выше, из доказанного следует соотношение

$$(r_k, s_m) \in R$$

т.е. r_k и s_m можно представить в виде конкатенаций (3.48) и (3.49) соответственно, компоненты которых обладают указанными выше свойствами.

Обозначим записями a_1^ω, u_1^ω, ..., v_n^ω, a^ω результаты замены в соответствующих компонентах конкатенаций (3.48) и (3.49) функциональной переменной φ на ФС $\textcircled{\omega}$. Поскольку термы u_1, ..., u_n имеют вид $\varphi(\ldots)$, то $u_1^\omega \equiv \omega$, ..., $u_n^\omega \equiv \omega$, откуда следуют соотношения

$$\begin{aligned} r_k[\textcircled{\omega}/\varphi] &= a_1^\omega\ u_1^\omega\ \ldots\ a_n^\omega\ u_n^\omega\ a^\omega \equiv \\ &\equiv a_1^\omega\ \omega\ \ldots\ a_n^\omega\ \omega\ a^\omega \leq \\ &\leq a_1^\omega\ v_1^\omega\ \ldots\ a_n^\omega\ v_n^\omega\ a^\omega = s_m[\textcircled{\omega}/\varphi] \end{aligned}$$

которые противоречат соотношениям (3.39) и (3.46). ■

3.8 Свойства правил PO, LO, PI, LI

3.8.1 Безопасность правила PO

Теорема 15

Вычислительное правило **PO** безопасно для любой ФП.

Доказательство.

Рассмотрим произвольный терм $C^i_{\Sigma,\bar{d}}$ в вычислительной последовательности, которая строится по правилу **PO**.

Если $C^i_{\Sigma,\bar{d}}$ имеет вид $\varphi(\ldots)$, то $C^{i,\omega}_{\Sigma,\bar{d}}[\sigma/\varphi] = \omega$, т.е. в этом случае условие (3.34) выполнено.

Рассмотрим случай, когда $C^i_{\Sigma,\bar{d}}$ имеет вид

$$g(\ldots\varphi(\bar{e}_1)\ldots\varphi(\bar{e}_k)\ldots) \tag{3.51}$$

где $g \in Fun$, и $\varphi(\bar{e}_1)$, …, $\varphi(\bar{e}_k)$ – раскрываемые подтермы.

Условие (3.34) для терма (3.51) имеет вид

$$g(\ldots\omega\ldots\omega\ldots) = \omega \tag{3.52}$$

где терм в левой части является результатом замены подтермов $\varphi(\bar{e}_i)$ терма (3.51) на ω.

Докажем соотношение (3.52). Предположим, что оно неверно, т.е. для некоторой константы $d \neq \omega$ верно соотношение

$$g(\ldots\omega\ldots\omega\ldots) = d$$

Тогда, в силу монотонности функции g, для каждого списка b_1, …, b_k констант соответствующих типов верно соотношение

$$g(\ldots b_1\ldots b_k\ldots) = d \tag{3.53}$$

откуда следует, что значение терма $C^i_{\Sigma,\bar{d}}$ равно d, поскольку

- $\varphi(\bar{e}_1)$, …, $\varphi(\bar{e}_k)$ – самые внешние подтермы терма $C^i_{\Sigma,\bar{d}}$, имеющие вид $\varphi(\ldots)$, и
- в терме $C^i_{\Sigma,\bar{d}}$ нет переменных.

Согласно нижеследующей лемме, из совпадения значения терма $C^i_{\Sigma,\bar{d}}$ с константой d следует совпадение самого терма $C^i_{\Sigma,\bar{d}}$ с константой d, которое противоречит предположению о том, что $C^i_{\Sigma,\bar{d}}$ имеет вид (3.51).

Лемма.

Для любого неупрощаемого терма v и любой константы $d \neq \omega$ верна импликация

$$v \equiv d \quad \Rightarrow \quad v = d \tag{3.54}$$

(где равенство понимается как совпадение термов, а не их значений).

Доказательство.

Индукция по длине терма v.

Соотношение $v \equiv d$ не может быть верным, когда

- $v = d'$, где d' – константа и $d' \neq d$
- $v = x$, где x – переменная, т.к. в этом случае

  $$v(d'/x) = d' \neq d$$

  если d' – константа и $d' \neq d$
- v имеет вид $\varphi(\ldots)$, т.к. в этом случае

  $$v[\textcircled{\omega}/\varphi] \equiv \omega \neq d$$

Предположим, что соотношение $v \equiv d$ верно, когда

$$v = g(v_1, \ldots, v_n), \quad \text{где} \quad g \in Fun$$

Докажем, что в этом случае будет верно совпадение термов

$$\hat{v} = d \tag{3.55}$$

которое противоречит предположению о неупрощаемости v.

(3.55) следует из доказываемого ниже соотношения

$$\begin{gathered} g(d_1, \ldots, d_n) \equiv d, \quad \text{где} \ \ \forall\, i = 1, \ldots, n \\ d_i \stackrel{\text{def}}{=} \begin{cases} v_i & \text{если } v_i \text{ – константа} \\ \omega & \text{иначе.} \end{cases} \end{gathered} \tag{3.56}$$

Для каждого терма u мы будем обозначать записью u^ω терм, получаемый из $u[\textcircled{\omega}/\varphi]$ заменой всех входящих в него переменных на константу ω.

Из $v \equiv d$ следует, что

$$d \equiv v^\omega = g(v_1^\omega, \ldots, v_n^\omega) \tag{3.57}$$

Докажем, что

$$\forall\, i = 1, \ldots, n \quad v_i^\omega \equiv d_i \tag{3.58}$$

- Если v_i – константа, то $d_i = v_i = v_i^\omega$.
- Если v_i – не константа, то $d_i = \omega$, и если бы v_i^ω не был эквивалентен ω, т.е.

  $$v_i^\omega \equiv d' \neq \omega$$

  то было бы верно соотношение

  $$v_i \equiv d' \neq \omega \tag{3.59}$$

  Поскольку терм v неупрощаемый, то терм v_i тоже неупрощаемый. По индуктивному предположению, верна импликация (3.54), с заменой v на v_i, т.е. из (3.59) следует равенство

  $$v_i = d'$$

  которое противоречит предположению о том, что v_i – не константа.

Соотношение (3.56) следует из (3.57) и (3.58).

Таким образом, соотношение $v \equiv d$ может быть верно только в том случае, когда v является константой d. ■

3.8.2 Безопасность правила LO

Теорема 16

Если в ФП Σ каждому входящему в неё ФС, за исключением *if_then_else*, сопоставлена функция, являющаяся естественным продолжением, то вычислительное правило **LO** является

безопасным для Σ.

Доказательство.

Обозначим символом r раскрываемый подтерм в терме $C^i_{\Sigma,\bar{d}}$ вычислительной последовательности, которая строится по правилу **LO**.

Если $r = C^i_{\Sigma,\bar{d}}$, то $C^{i,\omega}_{\Sigma,\bar{d}}[\sigma/\varphi] = \omega$, т.е. в этом случае условие (3.34) выполнено.

Пусть $r \neq C^i_{\Sigma,\bar{d}}$. Так как r – самый внешний подтерм вида $\varphi(\ldots)$ в терме $C^i_{\Sigma,\bar{d}}$, то существует последовательность r_1, ..., r_n подтермов терма $C^i_{\Sigma,\bar{d}}$, обладающая следующими свойствами:

- $r_1 = r$, $r_n = C^i_{\Sigma,\bar{d}}$
- $\forall\, j = 1, \ldots, n-1$ терм r_{j+1} имеет вид

$$g(v_1, \ldots, v_m) \quad (g \in Fun) \tag{3.60}$$

  где для некоторого $k \in \{1, \ldots, m\}$ $\quad v_k = r_j$.

Терм r является подтермом всех термов r_1, ..., r_n. Обозначим записью r^ω_j (где $j = 1, \ldots, n$) терм, получаемый из терма r_j заменой его подтерма r на константу ω.

Докажем индукцией по j, что $\forall\, j = 1, \ldots, n$

$$r^\omega_j = \omega. \tag{3.61}$$

Для $j = 1$ соотношение (3.61), очевидно, верно.

Пусть для некоторого $j \in \{1, \ldots, n-1\}$

- верно (3.61),
- r_{j+1} имеет вид (3.60), и
- $\exists\, k \in \{1, \ldots, m\} : v_k = r_j$

тогда

$$r^\omega_{j+1} = g(v_1, \ldots, v_{k-1}, r^\omega_j, v_{k+1}, \ldots, v_m) \tag{3.62}$$

Если $g \neq \mathit{if\_then\_else}$, то соотношение

$$r^\omega_{j+1} = \omega$$

следует из (3.61), (3.62), и предположения о том, что функция, соответствующая ФС g, является естественным продолжением.

Пусть $g = if\_then\_else$. Докажем, что в этом случае номер k, такой, что $v_k = r_j$, равен 1.

Если $k \neq 1$, то r входит в v_2 или v_3. По предположению, r является самым левым из самых внешних подтермов вида $\varphi(\ldots)$ в терме $C^i_{\Sigma,\bar{d}}$. Поэтому v_1 не содержит вхождений φ. Следовательно, v_1 состоит только из констант и ФС. Поскольку v_1 неупрощаемый, то, значит, он является константой.

Согласно определению функции if_then_else, если v_1 является константой, то терм

$$r_{j+1} = if\_then\_else(v_1, v_2, v_3)$$

является упрощаемым. Это противоречит тому, что терм r_{j+1} – неупрощаемый, поскольку он является подтермом неупрощаемого терма $C^i_{\Sigma,\bar{d}}$.

Итак, $k = 1$, и

$$r_{j+1} = if\_then\_else(r_j, v_2, v_3)$$

откуда следует, что

$$r^\omega_{j+1} = if\_then\_else(r^\omega_j, v_2, v_3) \tag{3.63}$$

Учитывая

- индуктивное предположение $r^\omega_j = \omega$, и
- определение функции if_then_else

заключаем, что (3.63) $= \omega$.

Таким образом, $\forall\, j = 1, \ldots, n$ верно (3.61).

В частности, $r^\omega_n = \omega$, поэтому

$$C^{i,\omega}_{\Sigma,\bar{d}}[\sigma/\varphi] = r^\omega_n[\sigma/\varphi] = \omega. \quad \blacksquare$$

3.8.3 Пример небезопасности правила LO

В излагаемом ниже примере используется ФС $\cdot$ типа

$$(\texttt{int}, \texttt{int}) \to \texttt{int}$$

которому соответствует следующая монотонная функция:

- на $\mathbf{Z} \times \mathbf{Z}$ она совпадает с обычным умножением
- $0 \cdot \omega = \omega \cdot 0 = 0$
- $\forall\, d \in \mathbf{Z} \setminus \{0\} \quad d \cdot \omega = \omega \cdot d = \omega$
- $\omega \cdot \omega = \omega$

Рассмотрим ФП

$$\Sigma: \quad \begin{array}{ll} \varphi(x) = (x = 0) & ? \; 0 \\ & : \varphi(x+1) \cdot \varphi(x-1) \end{array}$$

σ принимает на всех аргументах значение 0. Однако вычисление $\varphi(1)$ по правилу **LO** будет бесконечным.

3.8.4 Пример небезопасности правил PI и LI

Пусть ФП Σ имеет вид

$$\Sigma: \quad \begin{array}{ll} \varphi(x,y) = (x = 0) & ? \; 1 \\ & : \varphi(x-1, \varphi(x,y)) \end{array}$$

Тогда

- $\sigma = (x \geq 0)? \; 1 : \omega$, но
- $\mathbf{PI}_\Sigma = \mathbf{LI}_\Sigma = (x = 0)? \; 1 : \omega$.

Например, вычисление $\varphi(1,0)$ по правилу **LI** породит бесконечную последовательность

$$\begin{array}{l} \mathbf{LI}^0_\Sigma = \varphi(1,0) \\ \mathbf{LI}^1_\Sigma = \varphi(0, \varphi(1,0)) \\ \mathbf{LI}^2_\Sigma = \varphi(0, \varphi(0, \varphi(1,0))) \\ \ldots \end{array}$$

Глава 4

Верификация функциональных программ

4.1 Задача верификации функциональных программ

Задача **верификации** ФП заключается в доказательстве того, что ННТ анализируемых ФП обладают заданными свойствами. Наиболее часто эти свойства представляют собой утверждения вида

$$\forall\, \bar{x}\, e \tag{4.1}$$

где e – терм типа `bool`, и $\bar{x}$ – список переменных из $Var(e)$. Утверждение (4.1) считается верным, если для каждого $\bar{d} \in D_{\bar{x}}$ значение терма $e(\bar{d})$ равно $\top$.

К числу основных методов верификации ФП относятся излагаемые ниже методы вычислительной индукции и структурной индукции.

4.2 Метод вычислительной индукции

4.2.1 Описание метода

Многие задачи верификации ФП могут быть сведены к следующей задаче: пусть заданы

- полное ЧУМ P,
- непрерывный функционал $F : P \to P$, и
- подмножество $Q \subseteq P$, которое является **замкнутым**, т.е. удовлетворяет условию: для каждой цепи

$$p_0 \leq p_1 \leq \ldots$$

элементов P верна импликация

$$\Big(\forall\, i \geq 0 \quad p_i \in Q\Big) \;\Rightarrow\; \sup\, p_i \in Q$$

Требуется доказать, что ННТ функционала F (которую мы ниже будем обозначать записью $\mathbf{fix}_F$) удовлетворяет соотношению

$$\mathbf{fix}_F \in Q \tag{4.2}$$

Метод **вычислительной индукции (ВИ)** заключается в сведении задачи доказательства соотношения (4.2) к проверке следующих двух условий:

1. $\mathbf{0} \in Q$ (где $\mathbf{0}$ – наименьший элемент ЧУМ P)
2. $\forall\, p \in Q \quad F(p) \in Q$

Нетрудно видеть, что если эти условия выполнены, то для каждого $i \geq 0$ верно соотношение $F^i(\mathbf{0}) \in Q$, откуда на основании замкнутости Q следует, что $\mathbf{fix}_F = \sup F^i(\mathbf{0}) \in Q$.

Второе из этих условий можно заменить на следующее условие: для каждого $i > 0$ верна импликация

$$\Big(\forall\, j < i \quad F^j(\mathbf{0}) \in Q\Big) \;\Rightarrow\; F^i(\mathbf{0}) \in Q \tag{4.3}$$

P и F могут иметь, например, следующий вид:

- $P = P_\Sigma$, где Σ – ФП, и $F = F_\Sigma$, или
- $P = P_\Sigma \times P_{\Sigma'} \times \ldots$, где
  - Σ, Σ', $\ldots$ – ФП,

– отношение порядка на P определяется покомпонентно:

$$(p_1, p_1', \ldots) \leq (p_2, p_2', \ldots) \quad \Leftrightarrow \quad p_1 \leq p_2, \;\; p_1' \leq p_2', \;\; \ldots$$

– наименьшим элементом P является список наименьших элементов ЧУМ P_Σ, $P_{\Sigma'}$, ...

и $\forall\, \bar{f} = (f, f', \ldots) \in P \quad F(\bar{f}) \stackrel{\text{def}}{=} (F_\Sigma(f), F_{\Sigma'}(f'), \ldots)$

При верификации ФП методом ВИ можно использовать следующие утверждения.

- Если Q_1 и Q_2 – замкнутые подмножества полного ЧУМ P, то подмножество $Q_1 \cap Q_2$ тоже замкнуто.
- Для любых непрерывных функционалов F_1 и F_2 на P подмножества

  $$\{p \in P \mid F_1(p) \leq F_2(p)\} \quad \text{и} \quad \{p \in P \mid F_1(p) = F_2(p)\}$$

  являются замкнутыми.

4.2.2 Примеры верификации функциональных программ методом вычислительной индукции

В этом параграфе излагаются некоторые примеры верификации ФП методом ВИ. Во всех этих примерах проверка первого условия ($\mathbf{0} \in Q$) опущена по причине ее тривиальности. Мы будем предполагать, что функции, используемые в ФП в этом пункте $(p, h, k, \ldots)$, принимают значение ω на аргументе ω.

Пример 1

Требуется доказать, что $\forall\, x \quad \sigma\sigma(x) = \sigma(x)$, где

$$\Sigma: \quad \varphi(x) = p(x)\,?\,x : \varphi\varphi h(x)$$

Определим $Q \stackrel{\text{def}}{=} \{f \in P_\Sigma \mid \sigma f = f\}$.

Пусть $f \in Q$. Докажем, что $F_\Sigma(f) \in Q$, т.е.

$$\sigma F_\Sigma(f) = F_\Sigma(f) \tag{4.4}$$

$$\begin{aligned}
&\text{Лев.Ч.}(4.4) \ = \sigma(p(x)?x : ffh(x)) = \\
&= p(x)?\ \sigma(x) : \sigma ffh(x) = \\
&= p(x)?\ (p(x)?\ x : \sigma\sigma h(x)) : \sigma ffh(x) = \\
&= p(x)?\ x : \sigma ffh(x) = \\
&= p(x)?\ x : ffh(x) = \ \text{Прав.Ч.}(4.4)
\end{aligned}$$

Отметим, что если определить $Q \stackrel{\text{def}}{=} \{f \in P_\Sigma \mid ff = f\}$, то тогда доказать требуемое свойство не получится.

Пример 2

Требуется доказать, что $\forall\, x, y \quad h\,\sigma(x, y) = \sigma(x, h(y))$, где

$$\Sigma: \quad \varphi(x, y) = p(x)?\ y : h\varphi(k(x), y)$$

Определим $Q \stackrel{\text{def}}{=} \{f \in P_\Sigma \mid \forall\, x, y \quad hf(x, y) = f(x, h(y))\}$.
Пусть $f \in Q$. Докажем, что $F_\Sigma(f) \in Q$, т.е. $\forall\, x, y$

$$h(F_\Sigma(f)(x, y)) = F_\Sigma(f)(x, h(y)) \tag{4.5}$$

$$\begin{aligned}
&\text{Лев.Ч.}(4.5) = h(p(x)?\ y : hf(k(x), y)) = \\
&= p(x)?\ h(y) : hhf(k(x), y) = \\
&= p(x)?\ h(y) : hf(k(x), h(y)) = \text{Прав.Ч.}(4.5)
\end{aligned}$$

Пример 3

Требуется доказать, что $\forall\, x \quad \sigma_1(x) = \sigma_3(x)$, где

$$\Sigma: \quad \begin{cases}
\varphi_1(x) = p(x)?\ \varphi_1\varphi_2 h(x) : \varphi_2 g(x) \\
\varphi_2(x) = q(x)?\ k\varphi_2\varphi_1(x) : kh(x) \\
\varphi_3(x) = p(x)?\ \varphi_3 k\varphi_4 h(x) : k\varphi_4 g(x) \\
\varphi_4(x) = q(x)?\ k\varphi_4\varphi_3(x) : h(x)
\end{cases}$$

Определим $Q \stackrel{\text{def}}{=} \{\bar{f} \in P_\Sigma \mid (f_1 = f_3) \wedge (f_2 = kf_4)\}$.
Пусть $\bar{f} \in Q$. Докажем, что $F_\Sigma(\bar{f}) \in Q$, т.е.

$$(F_\Sigma(\bar{f}))_1 = (F_\Sigma(\bar{f}))_3 \tag{4.6}$$

$$(F_\Sigma(\bar{f}))_2 = k(F_\Sigma(\bar{f}))_4 \tag{4.7}$$

Лев.Ч.(4.6) $= p(x)?\ f_1 f_2 h(x) : f_2 g(x) =$
$= p(x)?\ f_3 k f_4 h(x) : k f_4 g(x) =$ Прав.Ч.(4.6)
Лев.Ч.(4.7) $= q(x)?\ k f_2 f_1(x) : k h(x) =$
$= k(q(x)?\ f_2 f_1(x) : h(x)) =$
$= k(q(x)?\ k f_4 f_3(x) : h(x)) =$ Прав.Ч.(4.7)

Пример 4

Требуется доказать, что $\forall\, x \quad \sigma(x, 0, x) = \sigma'(x, 0)$, где

$$\begin{aligned} \Sigma: &\quad \varphi(x, y, z) = (x = 0)?\ y : \varphi(x - 1, y + z, z) \\ \Sigma': &\quad \varphi(x, y) = (x = 0)?\ y : \varphi(x - 1, y + 2x - 1) \end{aligned}$$

Мы докажем более общее соотношение:

$$\forall\, x, y \quad (\sigma(y, x(x - y), x) = \sigma'(y, x^2 - y^2)) \tag{4.8}$$

Искомое соотношение является частным случаем (4.8) (при $y = x$).

Определим

- $P \stackrel{\text{def}}{=} P_\Sigma \times P_{\Sigma'}$
- $F : P \to P, \quad F(f, f') \stackrel{\text{def}}{=} (F_\Sigma(f), F_{\Sigma'}(f'))$, и
- $Q \stackrel{\text{def}}{=} \left\{ (f, f') \in P \;\middle|\; \begin{array}{ll} \forall\, x, y & f(y, x(x - y), x) = \\ & = f'(y, x^2 - y^2) \end{array} \right\}$

Пусть $(f, f') \in Q$. Докажем, что $(F_\Sigma(f), F_{\Sigma'}(f')) \in Q$, т.е. $\forall\, x, y$

$$F_\Sigma(f)(y, x(x - y), x) = F_{\Sigma'}(f')(y, x^2 - y^2) \tag{4.9}$$

Лев.Ч.(4.9) $=$
$= (y = 0)?\ x(x - y) : f(y - 1, x(x - y) + x, x) =$
$= (y = 0)?\ x^2 : f(y - 1, x(x - (y - 1)), x) =$
$= (y = 0)?\ x^2 : f'(y - 1, x^2 - (y - 1)^2) =$
$= (y = 0)?\ x^2 - y^2 : f'(y - 1, (x^2 - y^2) + 2y - 1) =$
$=$ Прав.Ч.(4.9)

Пример 5

Требуется доказать, что $\forall\, x \quad \sigma(x) = \sigma'(x)$, где

$$\begin{array}{ll} \Sigma: & \varphi(x,y) = p(x)?\ y : h\varphi(k(x),y) \\ \Sigma': & \varphi(x,y) = p(x)?\ y : \varphi(k(x),h(y)) \end{array}$$

Мы докажем более сильное соотношение:

$$\forall\, x,y \quad \left\{ \begin{array}{l} \sigma(x,y) = \sigma'(x,y) \\ \sigma'(x,h(y)) = h\sigma'(x,y) \end{array} \right\}$$

(где запись вида $\left\{ \begin{array}{l} e_1 \\ e_2 \end{array} \right\}$ является другой формой записи $e_1 \wedge e_2$).

Определим

- P и F так же, как в предыдущем примере, и
- $Q \stackrel{\text{def}}{=} \{(f,f') \in P \mid \forall\, x,y \quad \left\{ \begin{array}{l} f(x,y) = f'(x,y) \\ f'(x,h(y)) = hf'(x,y) \end{array} \right\}\}$

Пусть $(f,f') \in Q$. Докажем, что $(F_\Sigma(f), F_{\Sigma'}(f')) \in Q$, т.е. $\forall\, x,y$

$$F_\Sigma(f)(x,y) = F_{\Sigma'}(f')(x,y) \tag{4.10}$$

$$F_{\Sigma'}(f')(x,h(y)) = hF_{\Sigma'}(f')(x,y) \tag{4.11}$$

$$\begin{array}{l} \text{Лев.Ч.}(4.10) = p(x)?\ y : hf(k(x),y) = \\ = p(x)?\ y : hf'(k(x),y) = \\ = p(x)?\ y : f'(k(x),h(y)) = \text{Прав.Ч.}(4.10) \\ \text{Лев.Ч.}(4.11) = p(x)?\ h(y) : f'(k(x),hh(y)) = \\ = p(x)?\ h(y) : hf'(k(x),h(y)) = \\ = h(p(x)?\ y : f'(k(x),h(y))) = \text{Прав.Ч.}(4.11) \end{array}$$

Другой способ доказательства соотношения $\sigma = \sigma'$ заключается в том, что вместо условия 2 в определении метода ВИ, т.е. вместо импликации

$$(f,f') \in Q \quad \Rightarrow \quad F(f,f') \in Q$$

доказывается следующее утверждение: $\forall\, i \geq 0$ верна импликация (4.3), в которой

$$Q \stackrel{\text{def}}{=} \{(f,f') \in P \mid f = f'\}$$

Для каждого $i \geq 0$ элемент $F^i(\mathbf{0})$ является парой функций, которую мы будем обозначать записью $(f^i, (f')^i)$.

(4.3) доказывается следующим образом.

1. Если $i = 0$, то (4.3) представляет собой утверждение

$$f^0 = (f')^0$$

т.е. $\textcircled{\omega} = \textcircled{\omega}$, что, очевидно, верно.

2. Если $i = 1$, то (4.3) представляет собой утверждение

$$(f^0 = (f')^0) \;\Rightarrow\; (f^1 = (f')^1)$$

т.е. $F_\Sigma(\textcircled{\omega}) = F_{\Sigma'}(\textcircled{\omega})$. Данное равенство обосновывается следующим образом:
$F_\Sigma(\textcircled{\omega})(x, y) = p(x)?\ y : h\textcircled{\omega}(k(x), y) =$
$= p(x)?\ y : h(\omega) = p(x)?\ y : \omega =$
$= p(x)?\ y : \textcircled{\omega}(k(x), h(y)) = F_{\Sigma'}(\textcircled{\omega})(x, y)$

3. Пусть $i > 1$. Для доказательства равенства

$$f^i = (f')^i$$

мы будем использовать предположение об истинности равенств

$$f^{i-1} = (f')^{i-1} \quad \text{и} \quad f^{i-2} = (f')^{i-2}$$

$f^i(x, y) = p(x)?\ y : hf^{i-1}(k(x), y) =$
$= p(x)?\ y : h(f')^{i-1}(k(x), y) =$
$= p(x)?\ y : h(pk(x)?\ y : (f')^{i-2}(kk(x), h(y))) =$
$= p(x)?\ y : h(pk(x)?\ y : f^{i-2}(kk(x), h(y))) =$
$= p(x)?\ y : (pk(x)?\ h(y) : hf^{i-2}(kk(x), h(y))) =$
$= p(x)?\ y : f^{i-1}(k(x), h(y)) =$
$= p(x)?\ y : (f')^{i-1}(k(x), h(y)) = (f')^i(x, y)$

4.3 Метод структурной индукции

4.3.1 Описание метода

Метод **структурной индукции (СИ)** для верификации ФП может использоваться в том случае, когда

- доказываемое утверждение о ФП имеет вид (4.1), и
- существует фундированное ЧУМ P, такое, что каждому элементу $\bar{d} \in D_{\bar{x}}$ может быть сопоставлен некоторый элемент $c(\bar{d}) \in P$.

Напомним, что ЧУМ называется **фундированным (ФЧУМ)**, если в нём нет бесконечно убывающих цепей, т.е. подмножеств вида $\{p_0, p_1, \ldots\}$, где $p_0 > p_1 > \ldots$.

Приведем два примера ФЧУМ.

1. Множество $\mathbf{N}$ натуральных чисел с обычным отношением порядка.
2. Множество P^n списков длины n, состоящих из элементов ФЧУМ P, с **лексикографическим** отношением порядка: неравенство
$$(p_1, \ldots, p_n) < (p'_1, \ldots, p'_n)$$
верно, если $p_1 < p'_1$, или существует $i \in \{2, \ldots, n\}$, такой, что $p_1 = p'_1$, …, $p_{i-1} = p'_{i-1}$, $p_i < p'_i$.

Метод СИ для обоснования утверждения
$$\forall\, \bar{d} \in D_{\bar{x}} \quad e(\bar{d}) = \top$$
в случае, когда определена функция c вида
$$c : D_{\bar{x}} \to P \tag{4.12}$$
где P – ФЧУМ, заключается в доказательстве того, что для произвольного $\bar{d} \in D_{\bar{x}}$ верно утверждение
$$(\forall\, \bar{d}' \in D_{\bar{x}} \quad c(\bar{d}') < c(\bar{d}) \;\Rightarrow\; e(\bar{d}') = \top) \quad \Rightarrow \quad e(\bar{d}) = \top. \tag{4.13}$$
Данный метод обосновывается следующим образом: если для некоторого $\bar{d} \in D_{\bar{x}}$ значение $e(\bar{d})$ было бы равно $\bot$, то $\exists\, \bar{d}_0 \in D_{\bar{x}}$:

- $e(\bar{d}_0) = \bot$, и
- $\forall\, \bar{d}' \in D_{\bar{x}} \quad c(\bar{d}') < c(\bar{d}_0) \;\Rightarrow\; e(\bar{d}') = \top$.

(нетрудно доказать, что если такого $\bar{d}_0$ не существует, то в P можно построить бесконечно убывающую цепь). Эта ситуация противоречит утверждению (4.13) для $\bar{d} = \bar{d}_0$.

4.3.2 Примеры верификации функциональных программ методом структурной индукции

В излагаемых ниже примерах 1, 2 и 3 ФЧУМ P совпадает с $D_{\bar{x}}$, и (4.12) является тождественной функцией. В примерах 4 и 5 $D_{\bar{x}} = \mathbf{S}$, $P = \mathbf{N}$, и функция $c : \mathbf{S} \to \mathbf{N}$ сопоставляет каждой строке её длину (т.е. количество символов в ней). В этих примерах для любых $x, y \in \mathbf{S}$ запись $x < y$ означает, что длина x меньше длины y.

Пример 1 (функция Аккермана)

Требуется доказать, что $\forall\, x, y \in \mathbf{N} \quad \sigma(x, y) \neq \omega$, где

$$\Sigma : \begin{cases} \varphi(x,y) = (x = 0) & ?\ \ y + 1 \\ & :(y = 0) \quad ?\ \ \varphi(x-1, 1) \\ & \qquad\qquad :\varphi(x-1, \varphi(x, y-1)) \end{cases}$$

(переменные x, y имеют тип `nat`).

Поскольку $D_{(x,y)} = \mathbf{N}^2$ – ФЧУМ относительно лексикографического порядка, то требуемое свойство можно доказать методом СИ.

- Если $x = 0$, то $\sigma(x, y) = y + 1 \neq \omega$.
- Пусть $x > 0$, и $\forall\, (x', y') < (x, y) \;\; \sigma(x', y') \neq \omega$. Докажем, что $\sigma(x, y) \neq \omega$.
  - Если $y = 0$, то $\sigma(x, y) = \sigma(x - 1, 1)$.
    Т.к. $(x - 1, 1) < (x, y)$, то, по индуктивному предположению, $\sigma(x - 1, 1) \neq \omega$.
  - Если $y \neq 0$, то, т.к. $(x, y - 1) < (x, y)$, то, по индуктивному предположению,

    $$\sigma(x, y - 1) \neq \omega$$

    откуда, используя неравенство

    $$(x - 1, \sigma(x, y - 1)) < (x, y)$$

    получаем требуемое соотношение $\sigma(x, y) \neq \omega$.

Пример 2 (вычисление факториала)

Требуется доказать, что $\forall\, x \in \mathbf{N}$

$$\sigma'(x, 0) = \sigma(x) \tag{4.14}$$

где

$$\begin{array}{ll} \Sigma: & \varphi(x) = (x = 0)?\ 1 : x \cdot \varphi(x-1) \\ \Sigma': & \varphi(x, y) = (x = y)?\ 1 : \varphi(x, y+1) \cdot (y+1) \end{array}$$

(переменные x, y имеют тип `nat`).

Мы докажем более общее утверждение: $\forall\, x \in \mathbf{N}$

$$\forall\, y \in \mathbf{N} \quad \Big(\sigma'(x+y, y) \cdot \sigma(y) = \sigma(x+y)\Big) \tag{4.15}$$

(4.14) является частным случаем (4.15) (при $y = 0$).

- Если $x = 0$, то (4.15) имеет вид

  $$\forall\, y \in \mathbf{N} \quad (\sigma'(y, y) \cdot \sigma(y) = \sigma(y))$$

  что верно, т.к. $\forall\, y \in \mathbf{N} \quad \sigma'(y, y) = 1$.

- Пусть $x > 0$, и для каждого натурального $z < x$ верно утверждение

  $$\forall\, y \in \mathbf{N} \quad \Big(\sigma'(z+y, y) \cdot \sigma(y) = \sigma(z+y)\Big)$$

  Докажем, что тогда будет верно и (4.15).

  Для каждого натурального y

  $$\begin{array}{l} \sigma'(x+y, y) \cdot \sigma(y) = \\ = \sigma'(x+y, y+1) \cdot (y+1) \cdot \sigma(y) = \\ = \sigma'(x+y, y+1) \cdot \sigma(y+1) = \\ = \sigma'((x-1)+(y+1), y+1) \cdot \sigma(y+1) = \\ = \sigma((x-1)+(y+1)) = \sigma(x+y) \end{array}$$

Пример 3 (функция Фибоначчи)

Требуется доказать, что $\forall\, x \in \mathbf{N}$

$$\sigma'(x, 1, 1) = \sigma(x) \tag{4.16}$$

где

$$\begin{array}{ll} \Sigma: & \varphi(x) = (x \in \{0, 1\})?\ 1 : \varphi(x-1) + \varphi(x-2) \\ \Sigma': & \varphi(x, a, b) = (x = 0)?\ a : \varphi(x-1, b, a+b) \end{array}$$

(переменные x, a, b имеют тип `nat`).

Мы докажем более общее утверждение: $\forall\, x \in \mathbf{N}$

$$\forall\, y \in \mathbf{N} \quad \sigma(x + y) = \sigma'(x, \sigma(y), \sigma(y + 1)) \tag{4.17}$$

((4.16) является частным случаем (4.17) (при $y = 0$).

- Если $x = 0$, то (4.17) имеет вид

  $$\forall\, y \in \mathbf{N} \quad \sigma(y) = \sigma'(0, \sigma(y), \sigma(y + 1))$$

  что верно согласно определению ФП Σ'.

- Пусть $x > 0$, и для каждого натурального $z < x$ верно утверждение

  $$\forall\, y \in \mathbf{N} \quad \sigma(z + y) = \sigma'(z, \sigma(y), \sigma(y + 1))$$

  Докажем, что тогда будет верно и (4.17).

  Для каждого натурального y

  $$\begin{array}{l} \sigma(x + y) = \\ = \sigma((x - 1) + (y + 1)) = \\ = \sigma'(x - 1, \sigma(y + 1), \sigma(y + 2)) = \\ = \sigma'(x - 1, \sigma(y + 1), \sigma(y + 1) + \sigma(y)) = \\ = \sigma'(x, \sigma(y), \sigma(y + 1)) \end{array}$$

Пример 4 (функция инвертирования строк)

Требуется доказать, что $\forall\, x \in \mathbf{S}$

$$r(x) \neq \omega \quad \text{и} \quad rr(x) = x \tag{4.18}$$

где r – функция инвертирования строк, определяемая следующим образом: $r(x) \stackrel{\text{def}}{=} \sigma(x, \varepsilon)$, где

$$\Sigma: \quad \varphi(x, y) = (x = \varepsilon)?\ y : \varphi(x', \hat{x} \cdot y)$$

(переменные x, y имеют тип `string`).

Мы докажем более общее утверждение: $\forall\, x \in \mathbf{S}$

$$\forall\, y \in \mathbf{S} \quad \sigma(x, y) \neq \omega \quad \text{и} \quad r\sigma(x, y) = \sigma(y, x) \tag{4.19}$$

(4.18) является частным случаем (4.19) (при $y = \varepsilon$).

- Если $x = \varepsilon$, то (4.19) имеет вид

  $$\forall\, y \in \mathbf{S} \quad \sigma(\varepsilon, y) \neq \omega \quad \text{и} \quad r\sigma(\varepsilon, y) = \sigma(y, \varepsilon) \tag{4.20}$$

  Согласно определению ФП Σ и функции r, (4.20) можно переписать в виде

  $$\forall\, y \in \mathbf{S} \quad y \neq \omega \quad \text{и} \quad r(y) = r(y)$$

  что, очевидно, верно.

- Пусть $x \neq \varepsilon$, и для каждой строки $z < x$ верно утверждение

  $$\forall\, y \in \mathbf{S} \quad \sigma(z, y) \neq \omega \quad \text{и} \quad r\sigma(z, y) = \sigma(y, z) \tag{4.21}$$

  Докажем, что тогда будет верно и (4.19).

  Используя (4.21) для $z \stackrel{\text{def}}{=} x'$ (что возможно, т.к. $x' < x$) и определение ФП Σ, получаем: для каждой строки y

  - $\sigma(x, y) = \sigma(\hat{x} \cdot x', y) = \sigma(x', \hat{x} \cdot y) \neq \omega$
  - $r\sigma(x, y) = r\sigma(\hat{x} \cdot x', y) = r\sigma(x', \hat{x} \cdot y) = \sigma(\hat{x} \cdot y, x') =$
    $= \sigma(y, \hat{x} \cdot x') = \sigma(y, x)$.

Пример 5 (сортировка)

Требуется доказать, что $\forall\, x \in \mathbf{S}$

$$\mathbf{ord}(\mathbf{sort}(x)) = 1 \tag{4.22}$$

где

- **sort** – функция сортировки строк, определяемая ФП

$$\begin{cases} \mathbf{sort}(x) = (x = \varepsilon)?\ \varepsilon : \mathbf{insert}(\hat{x}, \mathbf{sort}(x')) \\ \mathbf{insert}(a, y) = (y = \varepsilon) \quad ?\ a \cdot \varepsilon \\ \qquad\qquad\qquad\qquad\quad : (a \leq \hat{y}) \quad ?\ a \cdot y \\ \qquad\qquad\qquad\qquad\qquad\quad : \hat{y} \cdot \mathbf{insert}(a, y') \end{cases}$$

- **ord** – функция проверки упорядоченности строки, определяемая ФП

$$\begin{cases} \mathbf{ord}(x) = (x = \varepsilon) \quad ?\ 1 \\ \qquad\qquad\qquad\quad : (x' = \varepsilon) \quad ?\ 1 \\ \qquad\qquad\qquad\qquad\quad : (\hat{x} \leq (x')^\wedge) \quad ?\ \mathbf{ord}(x') \\ \qquad\qquad\qquad\qquad\qquad\quad : 0 \end{cases}$$

Ниже мы будем обозначать терм вида $\mathbf{insert}(a, y)$ сокращённой записью $a \to y$.

Если $x = \varepsilon$, то $\mathbf{sort}(x) = \varepsilon$, и

$$\mathbf{ord}(\mathbf{sort}(x)) = \mathbf{ord}(\varepsilon) = 1$$

Если $x \neq \varepsilon$, то доказываемое равенство (4.22) можно переписать в виде

$$\mathbf{ord}(\hat{x} \to \mathbf{sort}(x')) = 1 \tag{4.23}$$

По индуктивному предположению, верно равенство

$$\mathbf{ord}(\mathbf{sort}(x')) = 1$$

из которого следует (4.23) по нижеследующей лемме.

Лемма.

Имеет место импликация

$$\mathbf{ord}(y) = 1 \quad \Rightarrow \quad \mathbf{ord}(a \to y) = 1 \tag{4.24}$$

Доказательство.
Доказываем лемму индукцией по y.
Если $y = \varepsilon$, то правая часть в (4.24) имеет вид

$$\mathbf{ord}(a \cdot \varepsilon) = 1$$

что верно по определению **ord**.

Пусть $y \neq \varepsilon$, и для каждого $z < y$ верна импликация

$$\mathbf{ord}(z) = 1 \quad \Rightarrow \quad \mathbf{ord}(a \to z) = 1 \tag{4.25}$$

Обозначим $c \stackrel{\text{def}}{=} \hat{y}$, $d \stackrel{\text{def}}{=} y'$.
(4.24) имеет вид

$$\mathbf{ord}(c \cdot d) = 1 \quad \Rightarrow \quad \mathbf{ord}(a \to c \cdot d) = 1 \tag{4.26}$$

Для доказательства импликации (4.26) нужно доказать, что при условии $\mathbf{ord}(c \cdot d) = 1$ верны импликации

(a) $a \leq c \quad \Rightarrow \quad \mathbf{ord}(a \cdot (c \cdot d)) = 1$,

(b) $c < a \quad \Rightarrow \quad \mathbf{ord}(c \cdot (a \to d)) = 1$

(a) верно потому, что из $a \leq c$ следует

$$\mathbf{ord}(a \cdot (c \cdot d)) = \mathbf{ord}(c \cdot d) = 1$$

Докажем (b).

- $d = \varepsilon$. В этом случае правая часть в (b) имеет вид

$$\mathbf{ord}(c \cdot (a \cdot \varepsilon)) = 1 \tag{4.27}$$

  (4.27) следует из $c < a$.

- $d \neq \varepsilon$. Обозначим $p \stackrel{\text{def}}{=} \hat{d}$, $q \stackrel{\text{def}}{=} d'$.

  В этом случае надо доказать, что при $c < a$

$$\mathbf{ord}(c \cdot (a \to p \cdot q)) = 1 \tag{4.28}$$

1. Если $a \leq p$, то (4.28) имеет вид

$$\mathbf{ord}(c \cdot (a \cdot (p \cdot q))) = 1 \tag{4.29}$$

Т.к. $c < a \leq p$, то (4.29) следует из равенств

$$\mathbf{ord}(c \cdot (a \cdot (p \cdot q))) = \mathbf{ord}(a \cdot (p \cdot q)) = \mathbf{ord}(p \cdot q) =$$
$$= \mathbf{ord}(c \cdot (p \cdot q)) = \mathbf{ord}(c \cdot d) = 1$$

2. Если $p < a$, то (4.28) имеет вид

$$\mathbf{ord}(c \cdot (p \cdot (a \to q))) = 1 \tag{4.30}$$

Поскольку по предположению

$$\mathbf{ord}(c \cdot d) = \mathbf{ord}(c \cdot (p \cdot q)) = 1$$

то $c \leq p$, и поэтому (4.30) можно переписать в виде

$$\mathbf{ord}(p \cdot (a \to q)) = 1 \tag{4.31}$$

При $p < a$

$$a \to d = a \to p \cdot q = p \cdot (a \to q)$$

поэтому (4.31) можно переписать в виде

$$\mathbf{ord}(a \to d) = 1 \tag{4.32}$$

(4.32) следует по индуктивному предположению для леммы (т.е. из импликации (4.25), в которой $z \stackrel{\text{def}}{=} d$) из равенства

$$\mathbf{ord}(d) = 1$$

которое обосновывается цепочкой равенств

$$1 = \mathbf{ord}(c \cdot d) = \mathbf{ord}(c \cdot (p \cdot q)) = \quad (\text{т.к. } c \leq p)$$
$$= \mathbf{ord}(p \cdot q) = \mathbf{ord}(d).$$

4.4 Другие методы верификации функциональных программ

4.4.1 Оценка наименьшей неподвижной точки функциональной программы сверху

Если проверяемое свойство ФП Σ имеет вид $\sigma \leq f$, где

- f – заданная монотонная функция, и
- отношение порядка на функциях понимается в смысле определения из пункта 2.1.5

то для доказательства этого свойства достаточно доказать неравенство

$$F_\Sigma(f) \leq f$$

Пример

Требуется доказать, что $\sigma \leq f$, где

$$\begin{array}{ll} \Sigma: & \varphi(x) = (x > 100)?\ x - 10 : \varphi\varphi(x + 11) \\ f & \stackrel{\text{def}}{=} (x > 100)?\ x - 10 : 91 \end{array}$$

Докажем, что $F_\Sigma(f) \leq f$, т.е.

$$(x > 100)?\ (x - 10) : ff(x + 11) \leq f$$

Поскольку

$$\begin{array}{l} f(x + 11) = \\ = (x + 11 > 100)?\ x + 11 - 10 : 91 = \\ = (x \geq 90)?\ x + 1 : 91 \end{array}$$

то

$$\begin{array}{l} ff(x + 11) = \\ = f((x \geq 90)?\ x + 1 : 91) = \\ = (x \geq 90)?\ f(x + 1) : f(91) = \\ = (x \geq 90)?\ f(x + 1) : 91 \end{array}$$

Таким образом,

$$F_\Sigma(f) = (x > 100) \quad ? \ (x - 10) \\ \qquad\qquad\quad : (x \geq 90) \quad ? \ f(x + 1) \\ \qquad\qquad\qquad\quad\ : 91 \tag{4.33}$$

Подвыражение $f(x + 1)$ в (4.33) вычисляется только для $x \in \{90, \ldots, 100\}$, и для всех таких значений x значение $f(x+1)$ равно 91. Отсюда следует равенство $(4.33) = f$.

4.4.2 Эквивалентные преобразования функциональных программ

Иногда доказательство свойств ННТ ФП можно существенно упростить, если вместо анализируемой ФП Σ рассматривать ФП Σ', получаемую из Σ

- раскрытием некоторых подтермов, или
- заменами вида $[\sigma/\varphi]$.

Можно доказать, что Σ и Σ' имеют одинаковые ННТ. Ниже мы доказываем это для случая, когда Σ' получается из Σ раскрытием одного подтерма или одной заменой вида $[\sigma/\varphi]$.

Теорема 17
Пусть задана ФП Σ вида

$$\Sigma: \quad \varphi(\bar{x}) = e$$

и терм e' получается из e раскрытием одного подтерма.

Тогда $\sigma = \sigma'$, где $\Sigma': \ \varphi(\bar{x}) = e'$.

Доказательство.
Для каждого $\bar{d} \in D_{\bar{x}}$ терм $e'(\bar{d})$ получается из терма $e(\bar{d})$ заменой подтерма вида $\varphi(e_1, \ldots, e_n)$ на терм $e(e_1/x_1, \ldots, e_n/x_n)$. Согласно утверждению 1 в доказательстве теоремы 9, значения термов $e(\bar{d})[\sigma/\varphi]$ и $e'(\bar{d})[\sigma/\varphi]$ совпадают. Поскольку значение первого из этих термов равно $\sigma(\bar{d})$, то

$$\sigma(\bar{d}) = e'[\sigma/\varphi](\bar{d})$$

т.е. σ – НТ Σ', откуда следует неравенство $\sigma' \leq \sigma$.

Докажем методом ВИ обратное неравенство: $\sigma \leq \sigma'$.

Определим

- $P \stackrel{\text{def}}{=} P_\Sigma \times P_{\Sigma'}$
- $F : P \to P$, $F(f, f') \stackrel{\text{def}}{=} (F_\Sigma(f), F_{\Sigma'}(f')) = (e[f/\varphi], e'[f'/\varphi])$
- $Q \stackrel{\text{def}}{=} \{(f, f') \in P \mid \left\{ \begin{array}{l} f \leq f' \\ f \leq e[f/\varphi] \end{array} \right\}\}$

Докажем, что верна импликация $p \in Q \;\Rightarrow\; F(p) \in Q$, т.е.

$$\left\{ \begin{array}{l} f \leq f' \\ f \leq e[f/\varphi] \end{array} \right\} \quad \Rightarrow \quad \left\{ \begin{array}{l} e[f/\varphi] \leq e'[f'/\varphi] \\ e[f/\varphi] \leq e[e[f/\varphi]/\varphi] \end{array} \right\} \tag{4.34}$$

Обозначим конъюнктивные члены в правой части (4.34) через A и B, и докажем, что каждый из них следует из левой части (4.34).

A: Из $f \leq f'$ и монотонности $F_{\Sigma'}$ следует неравенство

$$e'[f/\varphi] \leq e'[f'/\varphi] \tag{4.35}$$

Искомое неравенство следует из (4.35) и из неравенства

$$e[f/\varphi] \leq e'[f/\varphi] \tag{4.36}$$

(4.36) верно потому, что

- $e'[f/\varphi]$ получается из $e[f/\varphi]$ заменой подтерма вида $f(e_1, \ldots, e_n)$ на $e[f/\varphi](e_1/x_1, \ldots, e_n/x_n)$, и
- из $f \leq e[f/\varphi]$ следует неравенство

$$f(e_1, \ldots, e_n) \leq e[f/\varphi](e_1/x_1, \ldots, e_n/x_n)$$

B: Из $f \leq e[f/\varphi]$ и монотонности F_Σ следует неравенство

$$e[f/\varphi] \leq e[e[f/\varphi]/\varphi].$$

Таким образом, $\mathbf{fix}_F = (\sigma, \sigma') \in Q$, откуда следует $\sigma \leq \sigma'$. ■

Теорема 18
Пусть задана ФП Σ вида

$$\Sigma: \quad \varphi(\bar{x}) = e$$

и терм e' получается из e заменой одного из вхождений функциональной переменной φ на ФС σ.
Тогда $\sigma = \sigma'$, где $\Sigma': \ \varphi(\bar{x}) = e'$.

Доказательство.
Т.к. $e'[\sigma/\varphi] = e[\sigma/\varphi] = \sigma$, то σ – НТ Σ', поэтому $\sigma' \leq \sigma$.
Докажем методом ВИ, что $\sigma \leq \sigma'$.
Определим

- $P \stackrel{\text{def}}{=} P_\Sigma \times P_{\Sigma'}$
- $F : P \to P$, $F(f, f') \stackrel{\text{def}}{=} (F_\Sigma(f), F_{\Sigma'}(f'))$
- $Q \stackrel{\text{def}}{=} \{(f, f') \in P \mid \left\{ \begin{array}{l} f \leq f' \\ f \leq \sigma \end{array} \right\}\}$

Докажем, что верна импликация $p \in Q \ \Rightarrow \ F(p) \in Q$, т.е.

$$\left\{ \begin{array}{l} f \leq f' \\ f \leq \sigma \end{array} \right\} \Rightarrow \left\{ \begin{array}{l} e[f/\varphi] \leq e'[f'/\varphi] \\ e[f/\varphi] \leq \sigma \end{array} \right\} \tag{4.37}$$

Обозначим конъюнктивные члены в правой части (4.37) через A и B, и докажем, что каждый из них следует из левой части (4.37).

A: Из $f \leq f'$ и монотонности $F_{\Sigma'}$ следует неравенство

$$e'[f/\varphi] \leq e'[f'/\varphi] \tag{4.38}$$

Искомое неравенство следует из (4.38) и из неравенства

$$e[f/\varphi] \leq e'[f/\varphi] \tag{4.39}$$

(4.39) верно потому, что

– $e[f/\varphi]$ получается из e заменой всех вхождений φ на f,
– $e'[f/\varphi]$ можно получить из e заменой
  * одного из вхождений φ на σ, и
  * остальных вхождений φ – на f
– $f \leq \sigma$.

В: Из $f \leq \sigma$ и монотонности F_Σ следует соотношение

$$e[f/\varphi] \leq e[\sigma/\varphi] = \sigma.$$

Таким образом, $\mathbf{fix}_F = (\sigma, \sigma') \in Q$, откуда следует $\sigma \leq \sigma'$. ■

Приведем два примера верификации ФП с использованием эквивалентных преобразований ФП.

Пример 1

Требуется доказать, что $\sigma = \sigma'$, где

$$\begin{array}{ll} \Sigma: & \varphi(x) = (x > 10)?\ x - 10 : \varphi\varphi(x + 13) \\ \Sigma': & \varphi(x) = (x > 10)?\ x - 10 : \varphi(x + 3) \end{array}$$

где x – типа `nat`.

Обозначим через Σ'' ФП, получаемую из Σ раскрытием подтерма, начинающегося со второго φ:

$$\Sigma'': \quad \varphi(x) = (x > 10)?\ x - 10 : A$$

где $A = \varphi \begin{pmatrix} (x + 13 > 10) & ?\ x + 13 - 10 \\ & : \varphi\varphi(x + 13 + 13) \end{pmatrix}$

Согласно теореме 17, $\sigma = \sigma''$.

Т.к. $x \geq 0$, то $x + 13 > 10$, поэтому можно упростить A до $\varphi(x + 3)$, в результате чего получим Σ'.

Пример 2

Требуется доказать, что $\sigma_1 = \sigma_3$, где

$$\Sigma: \quad \left\{ \begin{array}{l} \varphi_1(x) = p(x)?\ \varphi_3\varphi_2\varphi_2 f(x) : g(x) \\ \varphi_2(x) = q(x)?\ \varphi_3 h(x) : k(x) \\ \varphi_3(x) = p(x)?\ \varphi_1\varphi_4 f(x) : g(x) \\ \varphi_4(x) = q(x)?\ \varphi_2\varphi_3 h(x) : \varphi_2 k(x) \end{array} \right.$$

Заменим в Σ первое вхождение φ_2 в e_1 и оба вхождения φ_2 в e_4 на ФС σ_2. ФП, получившуюся в результате этой замены, обозначим тем же символом Σ. Согласно вышесказанному, ННТ ФП

$$\Sigma: \quad \left\{ \begin{array}{l} \varphi_1(x) = p(x)?\ \varphi_3\sigma_2\varphi_2 f(x) : g(x) \\ \varphi_2(x) = q(x)?\ \varphi_3 h(x) : k(x) \\ \varphi_3(x) = p(x)?\ \varphi_1\varphi_4 f(x) : g(x) \\ \varphi_4(x) = q(x)?\ \sigma_2\varphi_3 h(x) : \sigma_2 k(x) \end{array} \right.$$

совпадает с ННТ исходной ФП.

Определим замкнутое подмножество $Q \subseteq P_\Sigma$:

$$Q \stackrel{\text{def}}{=} \{\bar{f} \in P_\Sigma \mid \left\{ \begin{array}{l} f_1 = f_3 \\ \sigma_2 f_2 = f_4 \end{array} \right\}\}$$

Нетрудно доказать, что верна импликация

$$p \in Q \ \Rightarrow\ F_\Sigma(p) \in Q$$

т.е. из $\left\{ \begin{array}{l} f_1 = f_3 \\ \sigma_2 f_2 = f_4 \end{array} \right\}$ следует, что

$$\left\{ \begin{array}{l} p(x)?\ f_3\sigma_2 f_2 f(x) : g(x) = p(x)?\ f_1 f_4 f(x) : g(x) \\ \sigma_2(q(x)?\ f_3 h(x) : k(x)) = q(x)?\ \sigma_2 f_3 h(x) : \sigma_2 k(x) \end{array} \right\}$$

Из замкнутости Q следует, что $\mathbf{fix}_{F_\Sigma} \in Q$, т.е.

$$\left\{ \begin{array}{l} \sigma_1 = \sigma_3 \\ \sigma_2\sigma_2 = \sigma_4 \end{array} \right\}$$

откуда следует искомое равенство $\sigma_1 = \sigma_3$.

Отметим, что в данном примере доказать требуемое свойство без использования эквивалентных преобразований довольно сложно.

Глава 5

Задачи

5.1 Нахождение наименьших неподвижных точек функциональных программ

1. Найти ННТ ФП

$$\Sigma: \ \varphi(x,y) = \ (x=0)?\ 0 : 1 + \varphi(x-1, \varphi(y-2, x))$$

где x, y – типа `int`.

2. Найти ННТ ФП

$$\Sigma: \ \varphi(x,y) = \ (x=0)?\ 1 : \varphi(x-1, \varphi(x-y, y))$$

где x, y – типа `int`.

5.2 Доказательство того, что наименьшая неподвижная точка функциональной программы имеет заданный вид

1. Доказать, что $\sigma = (xy=0)?\ x+y : 1$, где x, y – типа `nat`, и $\Sigma: \begin{cases} \varphi(x,y) = (xy=0) & ?\ x+y \\ & : \varphi(x-1, \varphi(x, y-1)) \end{cases}$

2. Доказать, что $\sigma = x + y + 1$, где x, y – типа `nat`, и
$$\Sigma : \begin{cases} \varphi(x,y) = (xy = 0) & ? \; x + y + 1 \\ & : \varphi(\varphi(x-1,1), y-1) \end{cases}$$

3. Доказать, что $\sigma = (x = 0)? \; y : 1$, где x, y – типа `nat`, и
$$\Sigma : \begin{cases} \varphi(x,y) = (x = 0) & ? \; y \\ & : (y = 0) \quad ? \; \varphi(x-1,1) \\ & \qquad\quad : \varphi(x-1, \varphi(x, y-1)) \end{cases}$$

4. Доказать, что
$$\sigma = \begin{cases} (x > a) & ? \; x - b \\ & : a - b + c - rem(a - x, c) \end{cases}$$
где x – типа `int`, и
   - $\Sigma : \; \varphi(x) = (x > a)? \; x - b : \varphi\varphi(x + b + c)$
   - a, b, $c \in \mathbf{Z}, \quad b, c > 0$
   - rem – функция взятия остатка от деления нацело.

5. Доказать, что
$$\sigma = (x > 10)? \; x - 10 : rem(x + 1, 3) + 1$$
где x – типа `nat`, и
   - $\Sigma : \; \varphi(x) = (x > 10)? \; x - 10 : \varphi\varphi(x + 13)$
   - rem – функция взятия остатка от деления нацело.

6. Доказать, что $\sigma = (x < 0)? \; x + 1 : 0$, где x – типа `int`, и
$$\Sigma : \; \varphi(x) = (x < 0)? \; x + 1 : \varphi\varphi(x - 2)$$

7. Доказать, что $\sigma = (x > 100)? \; x - 10 : 91$, где x – типа `int`, и
$$\Sigma : \quad \begin{cases} \varphi(x) = (x > 100) & ? \; x - 10 \\ & : \underbrace{\varphi \ldots \varphi}_{k}(x + 10(k-1) + 1) \end{cases}$$
где k – произвольное целое число, такое, что $k \geq 2$.

8. Доказать, что $\sigma_1 = (x \leq 2)?\ x : 4$, где x – типа `nat`, и

$$\Sigma : \left\{ \begin{array}{l} \varphi_1(x) = \varphi_2(x, 0, 0) \\ \varphi_2(x, y, z) = \varphi_3(even(x)?\ \frac{x}{2} : 3x + 1, x, y, z) \\ \varphi_3(u, x, y, z) = (u = z)?\ u : \varphi_2(u, x, y) \end{array} \right.$$

где x, y, z, u – типа `nat`, и $even$ – функция проверки чётности.

5.3 Совпадение функций, определяемых различными функциональными программами

1. Верно ли, что $\sigma = \sigma'$, где

$$\Sigma : \left\{ \begin{array}{lll} \varphi(x, y) = & (x = 0) & ?\ y \\ & & : (x > 0) \quad ?\ \varphi(x - 1, y + 1) \\ & & \quad\ : \varphi(x + 1, y - 1) \end{array} \right.$$

$$\Sigma' : \left\{ \begin{array}{lll} \varphi(x, y) = & (x = 0) & ?\ y \\ & & : (x > 0) \quad ?\ \varphi(x - 2, y + 2) \\ & & \quad\ : \varphi(x + 2, y - 2) \end{array} \right.$$

2. Доказать, что $\sigma = \sigma'_1$, где

$$\begin{array}{l} \Sigma :\ \varphi(x) = (x = 0)?\ 0 : \varphi h(x) \\ \Sigma' : \left\{ \begin{array}{l} \varphi_1(x) = (x = 0)?\ 0 : \varphi_1\varphi_2(x) \\ \varphi_2(x) = (x = 0)?\ 0 : \varphi_2\varphi_2 h(x) \end{array} \right. \end{array}$$

(x – типа `nat`)

3. Доказать, что $\sigma = \sigma'_1$, где

$$\begin{array}{l} \Sigma :\ \varphi(x) = (x = 0)?\ 0 : (p(x)?\ \varphi g(x) : \varphi h(x)) \\ \Sigma' : \left\{ \begin{array}{l} \varphi_1(x) = (x = 0)?\ 0 : \varphi_1\varphi_2(x) \\ \varphi_2(x) = (x = 0)?\ 0 : (p(x)?\ g(x) : \varphi_2\varphi_2 h(x)) \end{array} \right. \end{array}$$

(x – типа `nat`)

4. Доказать, что $\sigma = \sigma'$, где

$$\begin{array}{l} \Sigma: \ \varphi(x) = p(x)?\ \varphi\varphi f(x) : x \\ \Sigma': \varphi(x) = p(x)?\ \varphi f(x) : x \end{array}$$

5. Доказать, что $\sigma_1 = \sigma'$, где

$$\begin{array}{l} \Sigma: \left\{ \begin{array}{l} \varphi_1(x) = p(x)?\ \varphi_2\varphi_1 f(x) : x \\ \varphi_2(x) = p(x)?\ x : \varphi_1 g(x) \end{array} \right. \\ \Sigma': \ \varphi(x) = p(x)?\ \varphi g\varphi f(x) : x \end{array}$$

6. Доказать, что $\sigma_1 = \sigma'_1$, где

$$\begin{array}{l} \Sigma: \left\{ \begin{array}{l} \varphi_1(x) = p(x)?\ \varphi_1\varphi_2 f(x) : g(x) \\ \varphi_2(x) = p(x)?\ \varphi_2 h(x) : x \end{array} \right. \\ \Sigma': \left\{ \begin{array}{l} \varphi_1(x) = p(x)?\ \varphi_2 f(x) : g(x) \\ \varphi_2(x) = p(x)?\ \varphi_2 h(x) : g(x) \end{array} \right. \end{array}$$

7. Доказать, что $\sigma_1 = \sigma'_1$, где

$$\begin{array}{l} \Sigma: \left\{ \begin{array}{l} \varphi_1(x) = p(x)?\ \varphi_1\varphi_2 h(x) : \varphi_3(x) \\ \varphi_2(x) = q(x)?\ f\varphi_2\varphi_1(x) : fh(x) \\ \varphi_3(x) = qg(x)?\ \varphi_3(x) : lgg(x) \end{array} \right. \\ \Sigma': \left\{ \begin{array}{l} \varphi_1(x) = p(x)?\ \varphi_1 f\varphi_2 h(x) : l\varphi_3 g(x) \\ \varphi_2(x) = q(x)?\ f\varphi_2\varphi_1(x) : h(x) \\ \varphi_3(x) = q(x)?\ \varphi_3(x) : g(x) \end{array} \right. \end{array}$$

8. Доказать, что $\sigma_1 = \sigma'_1$, где

$$\begin{array}{l} \Sigma: \left\{ \begin{array}{l} \varphi_1(x) = p(x)?\ \varphi_2\varphi_3 f(x) : \varphi_2 f\varphi_4 g(x) \\ \varphi_2(x) = q(x)?\ \varphi_2 f(x) : h(x) \\ \varphi_3(x) = r(x)?\ f\varphi_4 g\varphi_5 f(x) : \varphi_5 gk(x) \\ \varphi_4(x) = s(x)?\ \varphi_4 k(x) : g(x) \\ \varphi_5(x) = s(x)?\ \varphi_5 k(x) : fg(x) \end{array} \right. \\ \Sigma': \left\{ \begin{array}{l} \varphi_1(x) = p(x)?\ \varphi_2 g\varphi_3 f(x) : \varphi_2 g(x) \\ \varphi_2(x) = s(x)?\ \varphi_2 k(x) : \varphi_4 fg(x) \\ \varphi_3(x) = r(x)?\ \varphi_5 f(x) : k(x) \\ \varphi_4(x) = q(x)?\ \varphi_4 f(x) : h(x) \\ \varphi_5(x) = s(x)?\ \varphi_5 k(x) : fg(x) \end{array} \right. \end{array}$$

9. Доказать, что $\sigma_1 = \sigma'_1$, где

$$\Sigma: \begin{cases} \varphi_1(x) = \begin{array}{c} p(x) \\ \\ \end{array} \begin{array}{c} ? \\ \\ : h(x) \end{array} \begin{pmatrix} q(x) & ? \varphi_3\varphi_2 g(x) \\ & : \varphi_2\varphi_3 g(x) \end{pmatrix} \\ \varphi_2(x) = q(x)?\ \varphi_1 f(x) : k(x) \\ \varphi_3(x) = q(x)?\ \varphi_4 f(x) : k(x) \\ \varphi_4(x) = p(x)?\ \varphi_5 g(x) : h(x) \\ \varphi_5(x) = q(x)?\ \varphi_2\varphi_4 f(x) : \varphi_3 k(x) \end{cases}$$

$$\Sigma': \begin{cases} \varphi_1(x) = p(x)?\ \varphi_2\varphi_2 g(x) : h(x) \\ \varphi_2(x) = q(x)?\ \varphi_1 f(x) : k(x) \end{cases}$$

10. Доказать, что $\sigma = \sigma'$, где

$$\Sigma:\ \varphi(x) = (x = 0)?\ 1 : 2 \cdot \varphi(x-1)$$

$$\Sigma':\ \varphi(x) = (x = 0)?\ 1 : \begin{pmatrix} odd(x) & ? 2 \cdot \varphi(\frac{x-1}{2})^2 \\ & : \varphi(\frac{x}{2})^2 \end{pmatrix}$$

11. Доказать, что $\forall x \quad \sigma(x, 0, 0) = \sigma'(x, 0)$, где

$$\begin{aligned} \Sigma:&\ \varphi(x, y, z) = \\ &= (y = x)?\ z : \varphi(x, y+1, z + 3y(y+1) + 1) \\ \Sigma':&\ \varphi(x, y) = (x = 0)?\ y : \varphi(x-1, y + 3x(x-1) + 1) \end{aligned}$$

12. Доказать, что $\forall x \quad \sigma(x) = \sigma'(x, 0) = \sigma''(x, 1, 0)$, где

$$\begin{aligned} \Sigma:&\ \varphi(x) = (x = 0)?\ 0 : x + \varphi(x-1) \\ \Sigma':&\ \varphi(x, y) = (x = 0)?\ y : \varphi(x-1, x+y) \\ \Sigma'':&\ \varphi(x, y, z) = (y = x+1)?\ z : \varphi(x, y+1, y+z) \end{aligned}$$

13. Доказать, что $\forall x, y > 0 \quad \sigma(x, y) = \sigma'(x, y)$, где

$$\Sigma: \begin{cases} \varphi(x, y) = \begin{array}{c} (x = 0)?\ 0 \\ \\ \end{array} \begin{array}{c} \\ : (\varphi(x-1, y) + 1 = y)?\ 0 \\ \end{array} \begin{array}{c} \\ \\ : \varphi(x-1, y) + 1 \end{array} \end{cases}$$

$$\Sigma':\ \varphi(x, y) = (x < y)?\ x : \varphi(x-y, y)$$

(обе ФП вычисляют остаток от деления x на y).

14. Доказать, что $\forall x \quad \sigma(x) = \sigma'(x,1) = \sigma''(x,0) = \sigma'''(x,0,1)$, где

$$\Sigma:\ \varphi(x) = (x = 0)?\ 1 : x \cdot \varphi(x-1)$$
$$\Sigma:\ \varphi(x,y) = (x = 0)?\ y : \varphi(x-1, xy)$$
$$\Sigma'':\ \varphi(x,y) = (y = x)?\ 1 : (y+1) \cdot \varphi(x, y+1)$$
$$\Sigma''':\ \varphi(x,y,z) = (y = x)?\ z : \varphi(x, y+1, (y+1) \cdot z)$$

15. Доказать, что $\sigma_1 = r$, где

$$\Sigma:\ \begin{cases} \varphi_1(x) = (x = \varepsilon)?\ x : \varphi_2(\hat{x}, \varphi_1(x')) \\ \varphi_2(x,y) = (y = \varepsilon)?\ x : \hat{y} \cdot \varphi_2(x, y') \end{cases}$$

и r – функция инвертирования строк, определённая в примере 4 пункта 4.3.2.

16. Доказать, что $\sigma = r$, где

$$\Sigma:\ \begin{cases} \varphi(x) = (x = \varepsilon)?\ \varepsilon \\ \qquad\quad : (x' = \varepsilon)?\ x : (\varphi(x'))^{\wedge} \cdot \varphi(\hat{x} \cdot \varphi(\varphi(x')')) \end{cases}$$

и r – функция инвертирования строк, определённая в примере 4 пункта 4.3.2.

17. Доказать, что $\forall x \quad \sigma(x) = \sigma'(x, NIL)$, где

$$\Sigma:\ \begin{cases} \varphi(x) = atom(x) \quad ?\ cons(x, NIL) \\ \qquad\qquad\qquad\quad : append(\varphi(car(x)), \varphi(cdr(x))) \end{cases}$$
$$\Sigma':\ \begin{cases} \varphi(x,y) = atom(x) \quad ?\ cons(x,y) \\ \qquad\qquad\qquad\quad : \varphi(car(x), \varphi(cdr(x), y)) \end{cases}$$

и $atom$, $cons$, car, cdr – функции языка LISP, NIL – константа, которой соответствует пустой список.

(обе ФП определяют LISP-функции сплющивания дерева, например $(a(bc)(de))$ преобразуется в $(abcde)$)

5.4 Свойства наименьших неподвижных точек функциональных программ

1. Доказать, что верна импликация

$$x \geq y \quad \Rightarrow \quad \sigma(x) \geq \sigma'(y) \geq 1$$

где x, y – типа `nat`, и

$$\Sigma: \ \varphi(x) = (x = 0) \vee (x = 1)?\ 1 : \varphi(x-1) + \varphi(x-2)$$
$$\Sigma': \ \varphi(x) = (x = 0) \vee (x = 1)?\ 1 : 2 \cdot \varphi(x-2)$$

2. Доказать, что верна импликация

$$x > y \quad \Rightarrow \quad \sigma(x,x) > \sigma(y,y)$$

где x, y – типа `nat`, и

$$\Sigma: \begin{cases} \varphi(x,y) = (x = 0) & ?\ y+1 \\ & :(y=0) \quad ?\ \varphi(x-1,1) \\ & \qquad\quad :\varphi(x-1,\varphi(x,y-1)) \end{cases}$$

3. Доказать, что верна импликация

$$x > 1 \quad \Rightarrow \quad \sigma(x) = \omega \quad \text{или} \quad \sigma(x) \leq \frac{x}{2}$$

где x – типа `int`, и

- $\Sigma: \begin{cases} \varphi(x) = (x \leq 1) & ?\ 1 \\ & :even(x) \quad ?\ div(x,2) \\ & \qquad\quad :\varphi\varphi(3 \cdot div(x,2) + 2) \end{cases}$
- $even$ – функция проверки чётности
- div – функция целочисленного деления

4. Доказать, что

   (a) $\forall x \quad \sigma(x) = \sigma'(x,0,1)$

   (b) $\forall x,y \quad \sigma''(x,y) \leq \Big(\sigma(x) = \sigma'(x,y,\sigma(y))\Big)$

где x, y – типа `nat`, и

$$\begin{array}{l}\Sigma:\ \varphi(x) = (x=0)?\ 1 : x\cdot\varphi(x-1)\\ \Sigma':\varphi(x,y,z) = (x=y)?\ z : \varphi(x,y+1,(y+1)z)\\ \Sigma'':\varphi(x,y) = (x=y)?\ \top : \varphi(x,y+1)\end{array}$$

5. Доказать, что $\forall x \quad \sigma(x) \neq \omega$, где x – типа `nat`, и

$$\Sigma:\ \varphi(x) = even(x)?\ \frac{x}{2} : \varphi\varphi(3x+1)$$

(использовать бинарное представление натуральных чисел)

6. Верно ли, что $\forall x > 0 \quad \sigma(x) \neq \omega$, где
$\Sigma:\ \varphi(x) = (x=1)?\ 0 : (even(x)?\ \varphi(\frac{x}{2}) : \varphi(3x+1))$

7. Доказать, что $\forall x > 0 \quad \sigma(x) = \sigma_1'(x)$, где x – типа `nat`, и
$$\Sigma:\ \left\{\begin{array}{ll}\varphi(x) = (x=1) & ?\ 0\\ & :(even(x)?\ \varphi(\frac{x}{2}) : \varphi(3x+1))\end{array}\right.$$
$$\Sigma':\ \left\{\begin{array}{l}\varphi_1(x) = (x=1)?\ 0 : \varphi_1\varphi_2(x)\\ \varphi_2(x) = (x=1)?\ 1 : (even(x)?\ \frac{x}{2} : \varphi_2\varphi_2(\frac{3x+1}{2}))\end{array}\right.$$

8. Доказать, что $\sigma(x,h(y)) = h\sigma(x,y)$, где
$\Sigma:\ \varphi(x,y) = p(x)?\ y : \varphi(k(x),h(y))$

9. Доказать, что $\sigma_1(x,x) = \sigma_2(x)$, где
$$\Sigma:\ \left\{\begin{array}{l}\varphi_1(x,y) = p(x)?\ \varphi_2(y) : \varphi_1(h(x),y)\\ \varphi_2(x) = p(x)?\ x : \varphi_2 h(x)\end{array}\right.$$

10. Доказать, что $\sigma_1\sigma_2 = \sigma_3\sigma_4$, где
$$\Sigma:\ \left\{\begin{array}{l}\varphi_1(x) = q(x)?\ g(x) : \varphi_3\varphi_4 f(x)\\ \varphi_2(x) = p(x)?\ x : \varphi_2 f(x)\\ \varphi_3(x) = p(x)?\ g(x) : \varphi_1\varphi_2 f(x)\\ \varphi_4(x) = q(x)?\ x : \varphi_4 f(x)\end{array}\right.$$

11. Доказать, что $\sigma_1 = \sigma_4$, где
$$\Sigma:\ \left\{\begin{array}{l}\varphi_1(x) = p(x)?\ \left(\begin{array}{ll}q(x) & ?\ \varphi_3\varphi_2 g(x)\\ & :\varphi_2\varphi_3 g(x)\end{array}\right) : h(x)\\ \varphi_2(x) = q(x)?\ \varphi_1 f(x) : k(x)\\ \varphi_3(x) = q(x)?\ \varphi_4 f(x) : k(x)\\ \varphi_4(x) = p(x)?\ \varphi_5 g(x) : h(x)\\ \varphi_5(x) = q(x)?\ \varphi_2\varphi_4 f(x) : \varphi_3 k(x)\end{array}\right.$$

Для этого необходимо сначала доказать, что

$$\sigma_5 = \sigma_3\sigma_2 = \sigma_2\sigma_3 \quad \text{и} \quad \sigma_3 = \sigma_2$$

12. Доказать, что $\forall\, x, y \in \mathbf{N}$

    (a) $succ(\sigma(x, y)) = \sigma(x, succ(y))$

    (b) $\sigma(\sigma(x, y), z) = \sigma(\sigma(x, z), y)$

    (c) $\sigma(x, y) = \sigma(y, x)$

    где

    - $succ$ – функция "следующее число"
    - $pred$ – функция "предыдущее число"
    - $\Sigma : \varphi(x, y) = (x = 0)?\ y : \varphi(pred(x), succ(y))$
      (σ является сложением натуральных чисел)

13. Доказать, что

$$\forall\, x \neq \varepsilon \quad \sigma(x, \sigma'(x)) = 1$$

    где x – типа `string`, и

    - $\Sigma :\ \varphi(x, y) = (x = \varepsilon)?\ 0 : ((\hat{x} = y)?\ 1 : \varphi(x', y))$
      где y – типа `char`
    - $\Sigma' : \varphi(x) = (x = \varepsilon)?\ \omega : ((x' = \varepsilon)?\ \hat{x} : \varphi(x'))$

    ($\sigma(x, y) =$ результат проверки вхождения символа y в строку x, и $\sigma'(x) =$ последний символ в строке x)

14. Доказать следующие свойства функции **sort** (определение которой см. в примере 5 пункта 4.3.2):

    (a) $\forall\, x \in \mathbf{S} \quad \mathbf{sort}(\mathbf{sort}(x)) = \mathbf{sort}(x)$

    (b) $\forall\, x, y \in \mathbf{S} \quad \mathbf{sort}(x * y) = \mathbf{sort}(y * x)$
    где $*$ обозначает функцию конкатенации строк, описываемую ФП (1.1)

(c) $\forall\, x \in \mathbf{S} \quad \mathbf{sort}(x * x) = \mathbf{dup}(\mathbf{sort}(x))$
где функция $\mathbf{dup} : \mathbf{S} \to \mathbf{S}$ является ННТ ФП

$$\Sigma :\ \varphi(x) = (x = \varepsilon)?\ \varepsilon : \hat{x} \cdot (\hat{x} \cdot \varphi(x'))$$

(d) $\forall\, x \in \mathbf{S} \quad \mathbf{sort}(r(x)) = \mathbf{sort}(x)$
где $r : \mathbf{S} \to \mathbf{S}$ – функция инвертирования строк, определённая в примере 4 пункта 4.3.2.

15. Определить ФП, одна из компонент ННТ которой представляет собой функцию

$$\mathbf{equal} :\ \mathbf{S}^2 \to \{0, 1\}$$

принимающую на паре строк $(x, y) \in \mathbf{S}^2$ значение

- 1, если каждый символ входит в строку x столько же раз, сколько раз он входит в строку y, и
- 0, иначе,

и доказать следующие свойства функции **equal**:

(a) $\forall\, x \in \mathbf{S} \quad \mathbf{equal}(x, \mathbf{sort}(x)) = 1$

(b) $\forall\, x \in \mathbf{S} \quad \mathbf{equal}(x, x) = 1$

(c) $\forall\, x, y \in \mathbf{S} \quad \mathbf{equal}(x, y) = \mathbf{equal}(y, x)$

(d) $\forall\, x, y, z \in \mathbf{S} \quad \mathbf{equal}(x, y) \cdot \mathbf{equal}(y, z) \leq \mathbf{equal}(x, z).$

Заключение

В книге была изложена одна из математических моделей функциональных программ, основанная на понятии наименьшей неподвижной точки непрерывного функционала на полном частично упорядоченном множестве. Было введено понятие вычислительного правила, и описано достаточное условие (Теорема 14), при выполнении которого функция, определяемая вычислительным правилом и функциональной программой, совпадает с наименьшей неподвижной точкой этой функциональной программы. Также были изложены основные методы верификации функциональных программ – вычислительная индукция и структурная индукция. В следующей части будут изложены другие математические модели функциональных программ, основанные на λ–исчислении и на теории типов.

Перечислим некоторые актуальные проблемы, связанные с изложенным материалом.

1. Нахождение необходимого и достаточного условия, которому должны удовлетворять вычислительное правило C и функциональная программа Σ, при выполнении которого функции C_Σ и σ совпадают.

2. Построение методов автоматизированной верификации функциональных программ. Нахождение достаточно широкого класса функциональных программ (включающего все рассмотренные в этой книге функциональные программы на символьных строках), такого, что проблема распознавания совпадения наименьших неподвижных точек двух различных функциональных программ из этого класса алгоритмически разрешима.

3. Построение автоматизированных методов синтеза функциональных программ, где проблема синтеза понимается следующим образом: по заданной системе Sys функциональных уравнений, неизвестными в которой являются функциональные переменные $\varphi^1, \ldots, \varphi^n$, соответствующие ННТ некоторых ФП, определить, разрешима ли эта система, и если да, то построить это решение, т.е. найти такой список $\Sigma^1, \ldots, \Sigma^n$ функциональных программ, что подстановка их ННТ вместо соответствующих функциональных переменных $\varphi^1, \ldots, \varphi^n$ в систему Sys будет приводить к истинным равенствам.

4. Нахождение методов оптимизации функциональных программ и вычислительных правил, т.е. таких методов, которые позволяют по заданной ФП Σ и заданному вычислительному правилу C построить (если это возможно) такие ФП Σ' и вычислительное правило C', что что $C_\Sigma = C'_{\Sigma'}$ и для каждого аргумента $\bar{d}$ функции C_Σ вычисление значения $C'_{\Sigma'}(\bar{d})$ требует меньшего числа шагов чем вычисление значения $C_\Sigma(\bar{d})$.

   Отметим, что формулировка данной задачи может уточняться и модифицироваться различными способами. Одно из таких уточнений заключается в том, чтобы рассматривать вычислительные правила, связанные с параллельным выполнением операций над термами.

Литература

1. **Lawrence C. Paulson.** ML for the working programmers, 2nd edition. *Cambridge University Press, 2000.*

2. Биография Джона Маккарти.
<http://ru.wikipedia.org/wiki/Маккарти,_Джон>

3. Описание языка Lisp.
<http://ru.wikipedia.org/wiki/Лисп>

4. **John McCarthy.** Recursive Functions of Symbolic Expressions and Their Computation by Machine. *Communications of the ACM, April 1960.*
<http://www-formal.stanford.edu/jmc/recursive.pdf>

5. Семантика языков программирования.
<http://ru.wikipedia.org/wiki
/Семантика_(программирование)>

6. Основные понятия лямбда-исчисления.
<http://ru.wikipedia.org/wiki/Лямбда-исчисление>

7. **Барендрегт Х.** Ламбда-исчисление. Его синтаксис и семантика. *Мир, 1985.*

8. Описание языка Common Lisp.
<http://ru.wikipedia.org/wiki/Common_Lisp>

9. Типизация в языках программирования.
<http://ru.wikipedia.org/wiki/Типизация>

10. Абстракция данных в программировании.
<http://ru.wikipedia.org/wiki/Абстракция_данных>

11. Полиморфизм в языках программирования.
<http://ru.wikipedia.org/wiki
/Полиморфизм_в_языках_программирования>

12. Описание языка ML.
<http://ru.wikipedia.org/wiki/ML>

13. Описание языка Scheme.
<http://ru.wikipedia.org/wiki/Scheme>

14. Описание языка Hope.
<http://ru.wikipedia.org/wiki/Hope>

15. Описание языка Miranda.
<http://ru.wikipedia.org/wiki
/Миранда_(язык_программирования)>

16. Описание языка Haskell.
<http://ru.wikipedia.org/wiki/Haskell>

17. Биография Хаскелла Карри.
<http://ru.wikipedia.org/wiki/Карри,_Хаскелл>

18. Описание языка OCaml.
<http://ru.wikipedia.org/wiki/OCaml>

19. Описание языка F#.
<http://ru.wikipedia.org/wiki/F_Sharp>

20. Описание языка Standard ML.
<http://ru.wikipedia.org/wiki/Standard_ML>

21. Описание языка XQuery.
<http://ru.wikipedia.org/wiki/XQuery>

22. Описание языка Рефал.
<http://refal.ru/>

23. Программирование на языке Рефал и суперкомпиляция.
<http://pat.keldysh.ru/~roman/doc/Turchin
/index_Turchin.html>

24. Биография В.Ф. Турчина.
<http://ru.wikipedia.org/wiki
/Турчин,_Валентин_Фёдорович>

25. Императивное программирование.
<http://ru.wikipedia.org/wiki
/Императивное_программирование>

26. Функциональное программирование на языке Python.
<http://ru.wikipedia.org/wiki
/Функциональное_программирование_на_Python>

27. Описание программной системы Coq.
<http://ru.wikipedia.org/wiki/Coq>

28. Описание программной системы Agda.
<http://ru.wikipedia.org/wiki/Agda>

29. **William Sonnex**. Zeno: a tool for the fully automated verification of functional program properties. *Final Report, Department of Computing, Imperial College London, 2010.*
<http://www.doc.ic.ac.uk/teaching
/distinguished-projects/2010/w.sonnex.pdf>

30. **C.-H. Luke Ong, Steven J. Ramsay**. Verifying Higher-Order Functional Programs with Pattern-Matching Algebraic Data Types *POPL'11, January 26–28, 2011, Austin, Texas, USA.*
<https://mjolnir.comlab.ox.ac.uk/papers/pmrs.pdf>

31. **N. Kobayashi.** Types and higher-order recursion schemes for verification of higher-order programs. *In Proceedings of POPL 2009, pages 416–428. ACM Press, 2009.*

32. **N. D. Jones and N. Andersen.** Flow analysis of lazy higher-order functional programs. *Theoretical Computer Science, 375:120–136, 2007.*

33. **N. Kobayashi.** Types and higher-order recursion schemes for verification of higher-order programs. *In Proceedings of POPL 2009, pages 416–428. ACM Press, 2009.*

34. **N. Kobayashi.** Model-checking higher-order functions. *In PPDP, pages 25–36, 2009.*

35. **N. Kobayashi and C.-H. L. Ong.** A type theory equivalent to the modal mu-calculus model checking of higher-order recursion schemes. *In Proceedings of LICS 2009. IEEE Computer Society, 2009.*

36. **N. Kobayashi, N. Tabuchi, and H. Unno.** Higher-order multiparameter tree transducers and recursion schemes for program verification. *In POPL, pages 495–508, 2010.*

37. **J. Kochems.** Approximating reachable terms of functional programs. *University of Oxford MMathsCompSc thesis, 2010.*

38. **J. Midtgaard.** Control-flow analysis of functional programs. *Technical Report BRICS RS-07-18, DAIMI, Department of Computer Science, University of Aarhus, Aarhus, Denmark, Dec 2007.*
<http://www.brics.dk/RS/07/18/BRICS-RS-07-18.pdf>

39. **C.-H. L. Ong.** On model-checking trees generated by higherorder recursion schemes. *In Proceedings 21st Annual IEEE Symposium on Logic in Computer Science, Seattle, pages 81–90. Computer Society Press, 2006.*
<http://users.comlab.ox.ac.uk/luke.ong/>

40. **R.M. Burstall and J. Darlington.** A transformation system for developing recursive programs. *Journal of the Association for Computing Machinery 24(1):44–67 (1977).*
<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.4684>

41. **Манна 3.** Теория неподвижной точки программ. *Кибернетический сборник (новая серия), вып. 15, 1978, стр. 38-100.*

42. **Zohar Manna**. Mathematical Theory of Computation. *(Reprint), Dover, 2003.*

43. **А.Филд, П.Харрисон.** Функциональное программирование. *М., Мир, 1993.*

44. **Хендерсон П.** Функциональное программирование. Применение и реализация. *М.: Мир, 1983.*

45. **Андерсон Р.** Доказательство правильности программ. *М., Мир, 1982.*

46. **J. Loeckx, K. Sieber.** The Foundations of Program Verification. *Teubner, second edition, 1987.*

47. **Simon Thompson**. Type Theory and Functional Programming. *Addison-Wesley. 1991.*

48. **Andrew D. Gordon.** Functional Programming and Input/Output. *Distinguished Dissertations in Computer Science. Cambridge University Press, 1994.*
<http://research.microsoft.com/en-us/um/people/adg/publications/fpio.pdf>

49. **Zhang Yu.** Theory and Practice of Functional Programming.
<http://lcs.ios.ac.cn/~yzhang/fopl/>

50. **Городняя Л. В.** Основы функционального программирования. Курс лекций. *М.: Интернет-университет информационных технологий, 2004.*
<http://www.intuit.ru/studies/courses/29/29/lecture/463>

51. **Душкин Р. В.** Функциональное программирование на языке Haskell. *М.: ДМК Пресс, 2006.*

52. **Душкин Р. В.** Основы функционального программирования. *Электронный учебный курс.*
<http://roman-dushkin.narod.ru/fp.html>

53. **Н. А. Роганова.** Функциональное программирование: Учебное пособие для студентов высших учебных заведений. *М.: ГИНФО, 2002.*

54. **John Harrison.** Introduction to Functional Programming.
<http://www.cl.cam.ac.uk/teaching/Lectures/funprog-jrh-1996/index.html>

55. **Р. Берд.** Жемчужины проектирования алгоритмов. Функциональный подход. *ДМК Пресс, 2013.*
<http://www.ozon.ru/context/detail/id/19435989/>

56. **А. Отт.** Обзор литературы по функциональному программированию.
<http://alexott.net/ru/fp/books/>

Abstract

Andrew M. Mironov
A theory of functional programs

The book covers mathematical models and methods for the analysis of functional programs. First part of the book is related to a theory of functions which are computed by functional programs (these functions are called least fixed points of functional programs). Basic methods of verification of functional programs are described in this part, namely: a method of computational induction and a method of structural induction. The book contains a large number of problems related to properties of functions computed by functional programs. In second part of the book a model of functional programs based on lambda-calculus and theory of types will be presented.

The book is intended for university students studying in the field "theoretical foundations of computer science"and "information security". It is also of interest for specialists in these areas.

Научное издание

Теория функциональных программ
Часть 1

Андрей Михайлович Миронов

Оригинал-макет подготовлен в ИПИ РАН автором

Подписано в печать 08.07.2013
Тираж 70 экз.
Заказ 13-03

Издательство ИПИ РАН
119333, Москва, ул. Вавилова, д.44, корп.2