Московский инженерно-физический институт
(государственный университет)

В.Э. Вольфенгаген,
Л.В. Гольцева,
Л.Ю. Исмаилова

АППЛИКАТИВНЫЕ ВЫЧИСЛЕНИЯ

на основе комбинаторов и λ-исчисления

Издание 3-е, стереотипное

Проект: Аппликативные Вычислительные Системы

Руководитель проекта, кандидат технических наук
Л.Ю.Исмаилова

МИФИ
Москва
2007

Факультет кибернетики

Проект: Аппликативные Вычислительные Системы
Руководитель проекта, кандидат технических наук
Исмаилова Л.Ю.

Д.т.н., профессор **Вольфенгаген В.Э.**, к.т.н., **Гольцева Л.В.**, к.т.н., **Исмаилова Л. Ю.**

Аппликативные вычисления на основе комбинаторов и λ-исчисления.— 3-е изд.— М.: МИФИ, 2007.—72 с.

Основы аппликативных вычислительных систем изложены элементарными средствами, что обеспечивает студентов и аспирантов кратким и исчерпывающим руководством, которое может использоваться ‘для первого чтения’. Настоящее руководство в различных вариантах течение ряда лет использовалось для проведения практический занятий и лабораторных работ по соответствующим разделам курса компьютерных наук. Охватываются вопросы использования комбинаторов и λ-исчисления при реализации аппликативных вычислений. Приводится необходимый теоретический минимум, основное внимание уделено выполнению упражнений, иллюстрирующих применения основных вычислительных идей, понятий и определений. Для облегчения овладения предметом руководство снабжено простой обучающей программой, которая может использоваться в качестве вводного лабораторного практикума. При практической работе с обучающей программой следует иметь ввиду, что решение задач предполагает проведение дополнительных преобразований с целью оптимизации выражений, устранения в них переменных, упрощения целевого исполняемого выражения. Аппликативные вычисления излагаются как набор таких методов и средств.

Для студентов и аспирантов всех специальностей. Может быть использована для первоначального самостоятельного изучения предмета.

ⓒВ.Э. Вольфенгаген, 1992–2007
ⓒИнститут Актуального Образования “ЮрИнфоР-МГУ”, 2001
ⓒМосковский инженерно-физический институт, 1992

Московский инженерно-физический институт
Лаборатория ИИТ и ВС, комната 119а
Москва, 115409 РФ
Fax: +7 (495) 323-94-37
E-mail: vew@jmsuice.msk.ru

Оглавление

Предисловие к 1-му изданию

Настоящее учебное пособие подготовлено для проведения семинарских занятий и лабораторных работ и охватывает вопросы использования комбинаторов и λ-исчисления при реализации аппликативных вычислений. Наряду с теоретическими сведениями основное внимание уделено выполнению упражнений, закрепляющих навык владения комбинаторной логикой. Материал ориентирован на развитие на развитие у студентов умения соединять теоретическое рассмотрение задачи с ее практической реализацией на компьютере. Решение задач предполагает проведение дополнительных преобразований с целью оптимизации выражений, устранения в них переменных, упрощения целевого исполняемого выражения. Аппликативные вычисления излагаются как набор таких методов и средств.

Москва, 1992
МИФИ

Вольфенгаген В.Э.
Гольцева Л.В.

Предисловие ко 2-му изданию

Настоящее руководство написано как краткий и элементарный курс, который надо практически изучить, чтобы помочь в овладении специальной литературой в области компьютерных наук. В обычных университетских курсах по математике или информатике, как правило, остаются без должного внимания вопросы исчисления объектов. В мировой практике за последние два десятилетия сложился относительно новый язык изложения результатов научных исследований в области программирования или информационных технологий, который основан на систематическом и явном использовании метаоператоров аппликации и функциональной абстракции. Книга принесет наибольшую пользу студенту, аспиранту или специалисту, не имеющего должной подготовки в области теоретических компьютерных наук, но имеющему необходимость войти в круг идеи аппликативных вычислений.

Общая идея разработки практикума, который бы позволил начинающему без излишних трудозатрат освоить методы записи аппликативных форм возникла в 1986 г. при проведении в МИФИ практических занятий со студентами по соответствующим разделам дисциплин по компьютерным наукам. Постановка задачи принадлежит к.т.н. Л.Ю. Исмаиловой, под чьим общим руководством аспиранткой Л.В. Гольцевой было реализовано программное обеспечение.

Москва, 2001
Институт Актуального Образования "ЮрИнфоР-МГУ"

Вольфенгаген В.Э.

1. Введение

λ-исчисление – это бестиповая теория, рассматривающая функции как правила, а не как графики. Понятие ‘функции как закона или правила’ подразумевает переход от аргумента к значению, процесс, закодированный некоторым определением. λ-исчисление рассматривает функции с точки зрения их вычисления.

Как теория комбинаторов, так и λ-исчисление имеют общую цель – описать некоторые базисные и общие свойства операторов и комбинаций операторов. Обе теории можно рассматривать как абстрактные языки программирования, так как они содержат в себе концепцию вычисления в ее наиболее полном и общем виде, но с минимумом синтаксических деталей и сложностей. λ-исчисление с успехом используется в качестве метатеории для различных формальных систем, основанных на определении функции. Такие системы строятся для описания основных способов комбинирования операторов и функций для формирования других операторов. На практике каждая λ-система обладает собственной синтаксической структурой, определяемой способами ее применения. В некоторых системах есть дополнительные константы, большинство из них имеет встроенные синтаксические ограничения, например, типовые ограничения. Будем рассматривать наиболее простую из синтаксических систем.

Часть I

Элементарная техника аппликативных вычислений

2. λ-исчисление, основные определения

2.1 λ-обозначения

Для пояснения λ-обозначений рассмотрим математическое выражение '$x - y$'. Его можно рассматривать как определение функции f от x или функции g от y:

$$f : x \mapsto x - y, \quad g : y \mapsto x - y$$

Для их различения требуется такое определение, которое дает функциям f и g различные имена некоторым систематическим способом. Нотация Черча – это систематический способ построения для выражения, включающего переменную 'x', обозначения для соответствующей функции от x (аналогичным способом для y):

$$f = \lambda x.x - y, \quad g = \lambda y.x - y$$

Например, рассмотрим равенства

$$f(0) = 0 - y, \quad f(1) = 1 - y$$

В λ-обозначениях они принимают вид:

$$(\lambda x.x - y)(0) = 0 - y, \quad (\lambda y.x - y)(1) = 1 - y$$

λ-обозначения могут быть распространены на случай функций от более, чем одной переменной. Например, выражение '$x - y$' определяет две функции h и k от двух переменных как

$$h(x, y) = x - y, \quad k(y, x) = x - y$$

Эти определения можно переписать как

$$h(x, y) = \lambda xy.x - y, \quad k(y, x) = \lambda yx.x - y$$

Однако можно избежать использования специальных обозначений для функций с различными переменными, используя функции, значения которых являются не числами, а представляют собой другие функции. Например, вместо 2-местной функции h, определенной выше, рассмотрим 1-местную функцию h^*, определенную посредством

$$h^* = \lambda x.(\lambda y.x - y)$$

Для каждого числа 'a' получаем:

$$h^*(a) = \lambda y.a - y$$

Следовательно, для каждой пары чисел a и b:

$$\begin{aligned}(h^*(a))b &= (\lambda y.a - y)(b)\\ &= a - b\\ &= h(a,b)\end{aligned}$$

2.2 Формальная система λ-исчисления

Построим формальную систему λ-исчисления.

Положим, что дана бесконечная последовательность различных символов, называемых *переменными*, и конечная, бесконечная или пустая последовательность различных символов, называемых *константами* (когда последовательность констант пуста, система будет называться *чистой*, в противном случае – *прикладной*).

Определение 2.1 (λ-терм). *Множество выражений, называемых λ-термами, определяется следующим способом:*

i) все переменные и константы, являющиеся λ-термами, называются атомами;

ii) если M и N – произвольные λ-термы, то (MN) – λ-терм, называемый аппликацией;

iii) если M – произвольный λ-терм, а x - произвольная переменная, то $(\lambda x.M)$ – λ-терм, называемый абстракцией.

Пример 2.1. Примеры λ-термов:

$$(\lambda x.(xy)), \quad ((\lambda y.y)(\lambda x.(xy))), \quad (x(\lambda x(\lambda x.x))), \quad (\lambda x.(yz))$$

Прописными буквами будем обозначать произвольные λ-термы. Строчными буквами 'x', 'y', 'z', 'u', 'v', 'w' будем обозначать переменные. Скобки опускаются таким образом, что, например, '$MNPQ$' обозначает $(((MN)P)Q)$. Это преобразование называется сокращением (восстановлением) скобок *по ассоциации влево*. В дальнейшем будут использоваться также сокращения:

$$\begin{aligned} \lambda x.PQ &= (\lambda x.(PQ)) \\ \lambda x_1, \ldots, x_n.M &= (\lambda x_1.(\lambda x_2.(\ldots(\lambda x_n.M)\ldots))) \end{aligned}$$

Графическое равенство будет обозначаться знаком '$\doteq$'. Например, запись '$M \doteq N$' будет обозначать, что M в точности такой же терм, что и N. В дальнейшем будет предполагаться, что если $MN = PQ$, то $M = P$ и $N = Q$ и если $\lambda x.M = \lambda y.P$, то $x = y$. Также будет предполагаться, что 4-е класса термов – *переменные*, *константы*, *аппликации*, *абстракции*, – не имеют общих элементов. Это принимается всегда при определении множеств выражений. Такое определение терма позволяет сформировать больше термов, чем это необходимо для приложений системы. Следовательно, часть термов остается неинтерпретированной.

Если терм M был проинтерпретирован как функция или оператор, то (MN) интерпретируется как результат приложения (аппликирования) M к аргументу N с обеспечением того, что этот результат определен. Терм $(\lambda x.M)$ представляет оператор, значение которого на аргументе N вычисляется подстановкой N вместо x всюду, где x имеет свободные вхождения в M.

Примеры

Пример 2.2. λ x.x(xy) представляет операцию применения функции дважды к объекту, обозначаемому y, и равенство (λ x.x(xy))N = N(Ny) выполняется для всех термов N в том смысле, что обе стороны имеют одинаковую интерпретацию.

Пример 2.3. λ x.y представляет константную функцию, которая принимает значение y для всех аргументов, и равенство

$$(\lambda x.y)N = y$$

несет тот же смысл, что и предыдущее.

Упражнения

Упражнение 2.1. *Представить следующие выражения в формальном λ-обозначении, используя D для обозначения оператора дифференцирования:*

$$\begin{aligned} D(x^2) &= 2x \\ D(x^2)3 &= 6 \\ g(x) &= f(x^2) \end{aligned}$$

Упражнение 2.2. *Расставить недостающие скобки:*

$$ux(yz)(\lambda v.vy)$$
$$(\lambda xyz.xz(yz))uvw$$
$$w(\lambda xyz.xz(yz))uv$$

3. Структура терма и подстановка

3.1 Структура терма

Определение 3.1 (длина терма). *Длина терма M (обозначается $lgh(M)$) – это количество вхождений атомов в M, которое более точно определим следующим образом:*

a) $lgh(a) = 1$;
b) $lgh(MN) = lgh(M) = lgh(N)$;
c) $lgh(\lambda x.M) = 1 + lgh(M)$.

Определение 3.2 (подтерм). *Отношение 'P содержится в Q' (также читается как 'P является подтермом Q', или 'Q содержит P') определяется индукцией по Q следующим образом:*

i) *P содержится в P;*
ii) *если P содержится в M или N, то P содержится в (MN);*
iii) *если P содержится в M или $P = x$, то P содержится в $(\lambda x.M)$.*

Определение 3.3 (связанные и свободные переменные). *Вхождение переменной x в терм P связано, если и только если она содержится в подтерме P вида $(\lambda x.M)$; в противном случае она является свободной. Если x имеет, по крайней мере, одно свободное вхождение в P, она называется свободной переменной P.*

Множество всех свободных переменных P обозначается посредством $FV(P)$. Замкнутый терм – это терм без свободных переменных.

Пример 3.1. В терме $xv(\lambda y.yv)w$ обе y связаны, обе v свободны, а x и w – свободны.

3.2 Подстановка

Для любых M, N, x определим $[N/x]M$ как результат подстановки N для любых свободных вхождений x в M и заменим связанные переменные для избежания конфликта:

i) $[N/x]x = N$;
ii) $[N/x]a = a$ для всех атомов $a \neq x$;
iii) $[N/x]PQ = ([N/x]P)([N/x]Q)$;
iv) $[N/x](\lambda x.P) = \lambda x.P$;
v) $[N/x](\lambda y.P) = \lambda y.[N/x]P$, если $y \neq x$ и $y \notin FV(N)$ и $x \notin FV(P)$;
vi) $[N/x](\lambda y.P) = \lambda z.[N/x][z/y]P$, если $y \neq x$ и $y \notin FV(N)$ и $x \in FV(P)$.

Условие (vi) вводится для того, чтобы предотвратить зависимость интуитивного значения $[N/x](\lambda y.P)$ от связанной переменной y. Например, для трех различных переменных w, x, y рассмотрим

$$[w/x](\lambda y.x)$$

Терм $(\lambda y.x)$ представляет *функцию-константу*, значение которой всегда x, таким образом, интуитивно можем ожидать, что $[w/x](\lambda y.x)$ представляет функцию-константу, значение которой всегда w. По правилу (v) получаем $[w/x](\lambda y.x) = (\lambda y.w)$. Теперь предположим, что мы заменяем y на w: так как $w \neq x$, терм $\lambda w.x$ все равно представляет функцию-константу, значение которой x, как и $\lambda y.x$. Итак, можно ожидать, что $[w/x](\lambda w.x)$ представляет ту же функцию-константу, что и $[w/x](\lambda y.x)$. Но если бы $[w/x](\lambda w.x)$ было вычислено по (v), то получилось бы $[w/x](\lambda w.x) = \lambda w.w$, что является *функцией-тождественностью*, а не функцией-константой. Но по условию (vi) с $N = y = w$ имеем вычисление

$$\begin{aligned} [w/x](\lambda w.x) &= \lambda z.[w/x][z/w]x \\ &= \lambda z.[w/x]x \\ &= \lambda z.w, \end{aligned}$$

которое представляет ту же константную функцию, что и $\lambda y.w$.

Упражнения

Упражнение 3.1. *Входит ли терм $x(yz)$ в $ux(yz)$?*

Упражнение 3.2. *В какой из термов упражнений предыдущего раздела входит терм $(\lambda xyz.xz(yz))$?*

Упражнение 3.3. *Вычислить:*

$$[(\lambda y.xy)/x](\lambda y.x(\lambda x.x))$$
$$[(\lambda y.vy)/x](y(\lambda v.xv))$$

4. Графическое представление λ-термов и комбинаторных термов

В предыдущих разделах были рассмотрены λ-термы, записанные в текстовом виде. Для лучшего понимания процесса редукции терма можно обратиться к внутреннему представлению λ-терма. К настоящему времени применяется два различных способа представления: в виде стека и в виде графа [8].

При реализации вычислительной среды наиболее универсальным можно считать графовый способ представления. Рассмотрим все возможные типы выражений и покажем, как каждое из них представляется ориентированным графом. Каждый из рассматриваемых графов будет направленным, ациклическим. Под ацикличностью понимается отсутствие циклов в графе, то есть невозможно вернуться в тот узел, в котором уже побывали. Будем рассматривать следующие виды конструкций.

1. Константы представляются просто в виде деревьев, состоящих из одного узла, содержащего значение константы, то есть из листьев.
2. Функциональная аппликация MN представляется специальным типом узла, узлом-аппликацией, который помечен символом @. На рис. 4.1 a) приведен пример графового представления аппликации.

   Примитивные функции от более, чем одного аргумента записываются в каррированной форме, то есть + 1 3 интерпретируется как (+ 1) 3, где результат применения функции + к 1 является другой функцией, эквивалентной функции прибавления 1 (‘следования за’) на целых числах. На рис. 4.1 b) приведен пример графового представления функции + 1 3.
3. λ-абстракции вида $\lambda x.B$ представлены вторым типом узла, λ-узлом со связанной переменной, ассоциирующейся с листом по левой стрелке и подграфом справа, представляющим те-

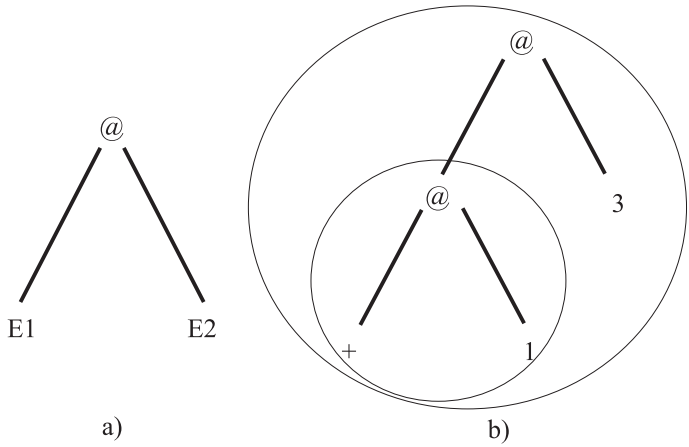

Рис. 4.1. Пример графового представления аппликации.

ло λ-выражения. Тело выражения в общем случае содержит вхождения связанных переменных и возможно также вхождения связанных переменных других (внешних) λ-абстракций как свободных переменных. Все такие ссылки на переменные представлены листьевыми вершинами. На рис. 4.2 приведен пример графового представления функции $\lambda x.\lambda y.+x(*yy)$.

Упражнения

Упражнение 4.1. *Нарисовать графическое представление следующих λ-термов:*

$(\lambda x.\lambda y.y(\lambda h.hx))7(\lambda x.x)$
$(\lambda f.(\lambda y.fy))(\lambda x.+2x)3$
$(\lambda T.TT)(\lambda f.\lambda x.f(fx))S\ 0$, *где* $S = \lambda x.+\ 1\ x$.

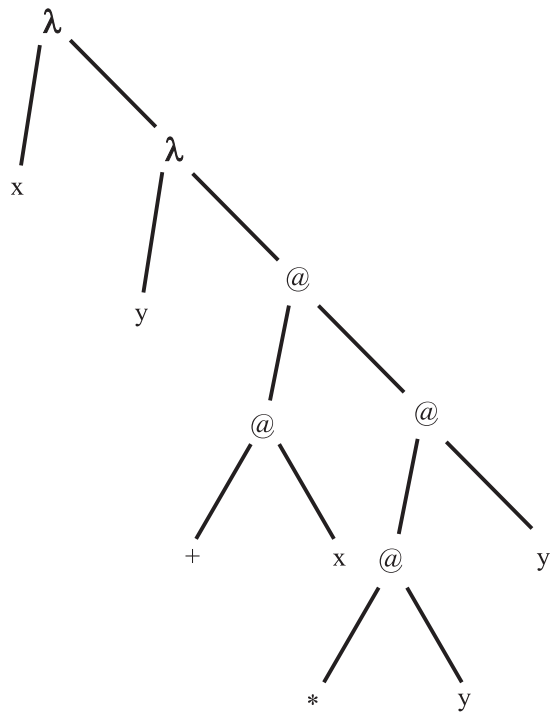

Рис. 4.2. Пример графового представления функции $\lambda x.\lambda y. + x(*yy)$.

5. Свертывание

Терм вида $(\lambda x.M)N$ обозначает оператор $\lambda x.M$, приложенный к аргументу N. В неформальной интерпретации $\lambda x.M$ его значение в 'точке' N вычисляется заменой N на x в M, таким образом, терм $(\lambda x.M)N$ может быть упрощен до $[N/x]M$.

Определение 5.1 (β-свертывание, β-редукция). *Любой терм вида $(\lambda x.M)N$ называется β-редексом, а соответствующая ему конструкция терма $[N/x]M$ называется его контрактом. Если терм P содержит вхождение $(\lambda x.M)N$, то, заменяя его на $[N/x]M$ и получая в результате P', говорим, что выполнили свертывание вхождения редекса в P, или что P β-свертывается к P', или $P \Rightarrow P'$.*

Конструкция терма $[N/x]M$ понимается как подстановка N вместо переменной x в M всюду, где x имеет свободные вхождения в M.

Говорим, что P β-редуцируется к Q, если и только если Q получается из P путем конечной, возможно, пустой последовательности β-свертываний и изменений связанных переменных.

Пример 5.1. Найти результат редукции термов:

$$
\begin{array}{lrcl}
a) & (\lambda x.x(xy))N & \to & N(Ny); \\
b) & (\lambda x.y)N & \to & y; \\
c) & (\lambda x.(\lambda y.yx)z)v & \to & [v/x]((\lambda y.yx)z) \to (\lambda y.yv)z \to zv; \\
d) & (\lambda x.xxy)(\lambda x.xxy) & \to & (\lambda x.xxy)(\lambda x.xxy)y \\
 & & \to & (\lambda x.xxy)(\lambda x.xxy)yy \to \ldots
\end{array}
$$

Определение 5.2 (нормальная форма). *Терм Q, который не содержит β-редексов, называется β-нормальной формой, или термом в β-нормальной форме. Класс всех β-нормальных форм называется β-nf или $\lambda\beta$-nf. Если терм β-редуцируется к Q в β-nf, то Q называется β-нормальной формой.*

Пример 5.2. Пусть $L = (\lambda x.xxy)(\lambda x.xxy)$.

a) $L = Ly = Lyy$;

b) Пусть $P = (\lambda u.v)L$ для вышеуказанного терма. Тогда P может редуцироваться двумя различными путями:

$$\begin{array}{rlcl} (i) & P = (\lambda u.v)L & = & [L/u]v = v \\ (ii) & P = (\lambda u.v)L & = & (\lambda u.v)(Ly) \\ & & = & (\lambda u.v)(Lyy) \\ & & = & \ldots \end{array}$$

Было показано, что некоторые термы могут быть редуцированы более, чем одним способом. Пусть $P = (\lambda x.(\lambda y.yx)z)v$. Он имеет две редукции:

$P = (\lambda y.yv)z = zv$ со сверткой $(\lambda y.yv)z$;
$P = (\lambda x.zx)v = zv$ со сверткой $(\lambda y.yx)z$;

В этом случае обе редукции достигают одной нормальной формы. Всегда ли это правильно? Для любой вычислительной системы конечный результат должен быть независимым от пути вычисления. Если это свойство невыполнимо для β-редукции, то любая попытка реализации λ-исчисления в языке программирования потерпит неудачу с самого начала. Нормальная форма действительно единственна, и речь здесь может идти только о конгруэнтных термах.

Определение 5.3. *Терм P β-эквивалентен, или β-конвертируем к Q (обозначается $P = Q$), если и только если Q получается из P путем конечной, возможно, пустой последоательности β-свертываний и обратных им β-развертываний и изменений связанных переменных. Это значит, что $P = Q$ если и только если существует последовательность $P_0, \ldots, P_n$ ($n \geq 0$) такая, что $P_0 = P$ и $P_n = Q$.*

6. Комбинаторная логика

Системы комбинаторов были построены для решения тех же задач, что и системы λ-исчисления, но без использования связанных переменных. Работая с комбинаторами, можно избежать технических сложностей, вызванных подстановками и конгруэнтностью. Однако ради этого технического преимущества приходится жертвовать интуитивной ясностью λ-обозначений.

6.1 Комбинаторы

Для пояснения комбинаторов рассмотрим закон коммутативности сложения в арифметике. Он может быть записан как

$$(\forall x, y)x + y = y + x$$

Но этот закон может быть записан без использования связанных переменных x и y. Для этого сначала определим $A(x, y) = x + y$ (для всех x, y), а затем оператор C как $(Cf)(x, y) = f(y, x)$ (для всех f, x, y). Тогда закон коммутативности запишется как

$$A = CA$$

Оператор C может быть назван *комбинатором*. Приведем примеры комбинаторов:

B – представляет композицию двух функций $(B(f, g))x = f(gx)$;
I – оператор тождества $If = f$;
K – формирует константные функции $(Ka)x = a$;
S – оператор сильной композиции $(S(f, g))x = f(x, gx)$;
W – оператор дублирования $(Wf)x = f(xx)$.

Вместо того, чтобы дать представление о комбинаторе в этом неформальном контексте, построим формальную систему термов, в которой будет построен комбинатор с требуемыми свойствами.

6.2 Формальная система комбинаторов

Как и в предыдущих разделах, изучаемая система будет наипростейшей, без синтаксических ограничений или усложнений, но с предупреждением, что системы, используемые на практике, выглядят сложнее. Однако те идеи, которые здесь представлены, являются общими для всех систем.

Положим, что дана бесконечная последовательность символов, называемых переменными, и конечная или бесконечная последовательность символов, называемых константами, включающая два названных базисных комбинатора K, S. (Если K и S – единственные константы, то система будет называться *чистой*, в противном случае она будет называться *прикладной*).

Определение 6.1 (CL-термы). *Множество выражений, называемых CL-термами, или термами комбинаторной логики, определяется индуктивно следующим образом:*

i) все переменные и константы, включающие K и S, являются CL-термами;
ii) если X и Y – CL-термы, то (XY) – тоже CL-терм.

Атом является переменной или константой. ‘Нередексный атом’ – это атом, который отличается от K и S. ‘Нередексная константа’ – константа, отличная от K и S. *Замкнутый* терм – это такой терм, который не содержит переменных. *Комбинатор* – это терм, атомами которого являются только K и S. (В чистой системе это то же, что и замкнутый терм.)

Примеры CL-термов:

$((S(KS))K)$,
$((S(Kx))((SK)K))$

Прописными латинскими буквами будем обозначать CL-термы, слово ‘терм’ будет использоваться как синоним для ‘CL-терм’. Строчными буквами ‘x’, ‘y’, ‘z’, ‘u’, ‘v’ будем обозначать переменные.

Определение 6.2 (длина CL-терма). *Длина X, или $(lgh(X))$, определяется как количество вхождений атомов в X:*

i) $lgh(a) = 1$ для атома a;

ii) $lgh(uv) = lgh(u) + lgh(v)$.

Определение 6.3 (вхождение). *Отношение 'X входит в Y' определяется следующим образом:*

i) X *входит в* X;
ii) если X *входит в* u *или в* v, *то* X *входит в* (uv).

Определение 6.4 (подстановка). $[U/x]Y$ *определяется как результат подстановки для каждого вхождения* x *в* Y *индукцией по построению* Y:

i) $[U/x]x = U$;
ii) $[U/x]a = a$ *для атомов* $a \neq x$;
iii) $[U/x](VW) = ([U/x]V)([U/x]W)$.

Для любых $U_1, \ldots, U_n$ и любых $x_1, \ldots, x_n$ (различных) определим $[U/x_1, \ldots, U/x_n]Y$ как результат одновременной подстановки U_1 вместо x_1, U_2 вместо x_2, $\ldots$, U_n вместо x_n в Y.

7. Слабая редукция

Терм вида Kxy или $Sxyz$ называется (слабым) редексом. Свертывание вхождения редекса в терм означает замещение одного вхождения Kxy на x, а $Sxyz$ на $xz(yz)$. Если эта процедура изменяет U на U', то говорим, что U (слабо) свертывается к U'.

Определение 7.1 (слабая редукция). *Говорим, что U (слабо) редуцируется к V, если и только если V получается из U путем конечной, возможно, пустой последовательностью слабых свертываний.*

Определение 7.2 (слабая н.ф.). *Слабая нормальная форма (или терм в слабой нормальной форме) – это терм, не содержащий слабых редексов. Если U слабо редуцируется к слабой нормальной форме X, то X называется слабой нормальной формой (слабой н.ф.) U.*

Примеры

Пример 7.1. Определим $B = S(KS)K$, тогда $Bxyz \to x(yz)$:

$$\begin{array}{rcl} Bxyz & \to & S(KS)Kxyz \\ & \to & KSx(Kx)yz \\ & \to & S(Kx)yz \\ & \to & Kxz(yz) \\ & \to & x(yz) \end{array}$$

Пример 7.2. Определим $B = S(BBS)(KK)$, тогда $Cxyz \to xzy$:

$$\begin{aligned} Cxyz &\to S(BBS)(KK)xyz \\ &\to BBSx(KKx)yz \\ &\to B(Sx)Kyz \\ &\to Sx(Ky)z \\ &\to xz(Kyz) \\ &\to xzy \end{aligned}$$

Определение 7.3 (абстракция). *Для каждого CL-терма M и для всякой x CL-терм, называемый $\lambda^* x.M$, определяется индукцией по M следующим образом:*

i) $\lambda^* x.M = KM$, *если* $x \notin FV(M)$;
ii) $\lambda^* x.x = I$, *где* $I = SKK$;
iii) $\lambda^* x.Ux = U$, *если* $x \notin FV(U)$;
iv) $\lambda^* x.UV = S(\lambda^* x.U)(\lambda^* x.V)$.

Примеры

$$\begin{aligned} \lambda^* x.xy &= S(\lambda^* x.x)(\lambda^* x.y) \\ &= SI(Ky) \\ \lambda^* xy.x &= \lambda^* x.(\lambda^* y.x) \\ &= \lambda^* x.(Kx) \\ &= K \\ \lambda^* xyz.xz(yz) &= \lambda^* x.(\lambda^* y.(\lambda^* z.xz(yz))) \\ &= \lambda^* x.(\lambda^* y.(S(\lambda^* z.xz)(\lambda^* z.yz))) \\ &= \lambda^* x.(\lambda^* y.(Sxy)) \\ &= \lambda^* x.Sx \\ &= S \end{aligned}$$

Определение 7.4 (слабая конвертируемость). *Будем говорить, что X слабо эквивалентируется, или слабо конвертируется к Y, или $X =_\omega Y$, если и только если Y может быть получен из X в результате некоторой последовательности (возможно, пустой) свертываний и обратных свертываний.*

Сравнивая основные понятия, на которых строится комбинаторная логика и в λ-исчисление, можно показать, что слабая редукция и эквивалентность в комбинаторной логике параллельны

λ-редукции и эквивалентности. Однако, эти системы не являются одинаковыми. Основное отличие в том, что λ-эквивалентность обладает свойством (ξ):

$$(\xi) \qquad X = Y \Rightarrow \lambda x.X = \lambda x.Y$$

Это свойство выполняется, поскольку все свертки и замены связанных переменных в X отражаются в Y. В комбинаторной логике это свойство будет выглядеть как:

$$X = Y \Rightarrow \lambda^* x.X = \lambda^* x.Y,$$

но для CL-термов λ^* не является частью синтаксиса, и свойство (ξ) не выполняется. Для примера возьмем

$$X = Sxyz, \quad Y = xz(yz)$$

Действительно, $X = Y$, но

$$\begin{array}{rcl} \lambda^* x.S & = & S(SS(Ky))(Kz) \\ \lambda^* x.Y & = & S(SI(Kz))(K(yz)) \end{array}$$

Во многих приложениях отсутствие свойства (ξ) не вызывает проблем, и простота слабой эквивалентности дает ей преимущества над λ.

Упражнения

Упражнение 7.1. *Докажите, что*

$$\begin{array}{rcl} S(Kx)(Ky) & = & K(xy) \\ S(Kx)I & = & x \\ S(Kx)y & = & xy \end{array}$$

Упражнение 7.2. *Постройте комбинатор пары и его проекции, то есть постройте D, D_1, D_2 такие, что $D_1(Dxy) = x$, $D_2(Dxy) = y$.*

Упражнение 7.3. *Покажите, что нет комбинатора, различающего атомы и сложные термы, то есть покажите, что не существует A такого, что $AX = S$, если X – атом, $AX = K$, если $X = UV$ для некоторых U, V.*

Упражнение 7.4. *Представить следующие λ-термы в базисе S, K, I:*

$$\lambda x.+\ x\ 1,$$
$$\lambda x.x(\lambda x.x)I,$$
$$\lambda x.cond(=\ x\ 0)1(-\ x\ 1)$$

8. Теорема о неподвижной точке

В мире чистых комбинаторов и λ-исчисления каждый оператор имеет неподвижную точку. То есть, для каждого терма X существует терм P такой, что $XP = P$. Более того, существует комбинатор Y, который находит эту неподвижную точку, такой, что YX является неподвижной точкой X для всех X.

Теорема 8.1 (о неподвижной точке). *И в λ-исчислении, и в комбинаторной логике имеется комбинатор Y такой, что:*

a) $YX =_\beta X(YX)$.

Фактически, существует комбинатор с более сильным свойством:

b) $YX \Rightarrow X(YX)$.

Y не является единственным, можно привести несколько определений Y:

1) $Y = \lambda x.VV,\ V = \lambda y.x(yy)$;
2) $Y = ZZ,\ Z = \lambda x.(\lambda y.y(xxy))$

Пусть $Y = ZZ$, докажем, что он удовлетворяет теореме о неподвижной точке:

$$\begin{array}{rcl} Yx & = & \lambda z.(\lambda y.y(zzy))Zx \text{ по определению } Z, \\ & \to & ([Z/z](\lambda y.y(zzy)))x \\ & \to & x(ZZx) \\ & \to & x(Yx) \end{array}$$

Упражнения

Упражнение 8.1. *Доказать, что первое определение комбинатора Y удовлетворяет теореме о неподвижной точке.*

Упражнение 8.2. $X = (\lambda x.(\lambda y.y(xxy)))(\lambda x.(\lambda y.y(xxy)))$. *Доказать, что X – комбинатор неподвижной точки.*

9. Формальные теории $\lambda\beta$ и $\mathbf{CL}_\omega$

Формальная теория $\mathcal{T}$ содержит три множества: *формулы*, *аксиомы* и *правила вывода*. Правила записываются с посылками, расположенными над горизонтальной чертой, и с заключениями, расположенными под горизонтальной чертой. Дедуктивный вывод в $\mathcal{T}$ формулы F из множества формул B – это дерево формул. Формулы, расположенные на вершине веток, являются аксиомами или элементами множества B, а формулы, полученные путем вывода по правилам непосредственно из исходных формул, расположены ниже, в самом же низу располагается F. Аксиомы наверху веток называются посылками. Если и только если такой вывод существует, то говорим, что

$$\mathcal{T},\ B\ \vdash\ F \text{ или } B \overset{\mathcal{T}}{\vdash}\ F.$$

Если и только если B – пустое, то называем F выводимой формулой или теоремой $\mathcal{T}$, а дедуктивный вывод называем доказательством и записываем:

$$\mathcal{T}\ \vdash\ F \text{ или } \overset{\mathcal{T}}{\vdash}\ F.$$

Если $\mathcal{T}$ содержит правило бесконечных посылок, то дедуктивный вывод является деревом, в котором бесконечная совокупность веток может с одной стороны возрастать, но каждая линейная часть, направленная вверх, имеет только конечную длину. Схема аксиом – это множество аксиом, удовлетворяющих данной модели.

Определение 9.1 ($\lambda\beta$-теории β-эквивалентности). *Формулами $\lambda\beta$ являются просто равенства $M = N$ для всех λ-термов M, N.*

Аксиомами считаются частные случаи трех схем аксиом, приведенных далее для всех λ-термов M, N и всех переменных x и y.

Схемы аксиом:

$$(\alpha) \qquad \lambda x.M = \lambda y.[y/x]M,\ y \notin FV(M),$$

$$(\beta) \qquad (\lambda x.M)N = [N/x]M,$$

$$(\rho) \qquad M = M$$

Правила вывода:

$$(\mu) \quad \frac{M = M'}{NM = NM'} \qquad (\nu) \quad \frac{M = M'}{MN = M'N}$$

$$(\sigma) \quad \frac{M = N}{N = M} \qquad (\tau) \quad \frac{M = N, \quad N = P}{M = P}$$

$$(\xi) \quad \frac{M = M'}{\lambda x.M = \lambda x.M'}$$

Если и только если равенство $M = N$ доказуемо в $\lambda\beta$, тогда записываем:

$$\lambda\beta \vdash M = N.$$

Определение 9.2 (фомальная $\lambda\beta$-теория β-редукции).
Эта теория называется $\lambda\beta$, как и рассмотренная выше. Ее формулами являются выражения $M \Rightarrow N$ для всех λ-термов M и N.

Ее схемы аксиом и правила те же, что и в предыдущем определении, только знак '=' заменяется на '$\Rightarrow$' и правило (τ) опускается. Если и только если выражение $M \Rightarrow N$ доказуемо в $\lambda\beta$, то записываем:

$$\lambda\beta \vdash M \Rightarrow N$$

Определение 9.3 (CL теория слабой эквивалентности).
В формальной CL теории слабой эквивалентности формулами являются равенства $X = Y$ для всех CL-термов X, Y. Аксиомы – частные случаи трех схем аксиом, представленных ниже для всех CL-термов X, Y, Z.

Схемы аксиом:

$$\begin{aligned} (K) \qquad KXY &= X, \\ (S) \qquad SXYZ &= XZ(YZ), \\ (\rho) \qquad X &= X \end{aligned}$$

Правила вывода:

$$(\mu)\ \frac{X = X'}{ZX = ZX'} \qquad (\nu)\ \frac{X = X'}{XZ = X'Z}$$

$$(\sigma)\ \frac{X = Y}{Y = X} \qquad (\tau)\ \frac{X = Y, \quad Y = Z}{X = Z}$$

Если и только если $X = Y$ доказуемо в CL, то записываем:

$$CL \vdash\ X = Y$$

Определение 9.4 (формальная CL-теория слабой редукции). *Формулами CL являются выражения $X \Rightarrow Y$ для всех CL-термов X, Y. Схемы аксиом – те же, что и в предыдущем определении, только '=' заменено на '$\Rightarrow$' и опущено правило (ξ).*

Если в CL доказуемо то, что $X \Rightarrow Y$, то записываем

$$CL \vdash\ X = Y.$$

Теория первого порядка является частным случаем формальной теории. Ее формулы построены из термов с использованием предикатных символов и обычных связок и кванторов. Ее аксиомы поделены на два класса: логические аксиомы, общие для всех теорий первого порядка, и собственные аксиомы, соответствующие теории $\mathcal{T}$. Ее правила – обычные правила классической логики первого порядка и одинаковые для всех теорий первого порядка. Под теорией первого порядка будем понимать теорию первого порядка с эквивалентностью, то есть с символом '=' и обычными аксиомами для него. Модель в теории первого порядка всегда будет обозначать модель, в которой '=' интерпретируется как отношение тождества.

Теорема 9.1. *Ни одна из теорий $\lambda\beta$, CL не является теорией первого порядка.*

Для $\lambda\beta$ это справедливо потому, что в ней нет связок и кванторов и потому, что ее теоремы содержат λ; в теориях первого порядка не разрешены операторы образования термов, связывающие переменные. В CL также нет связок, поэтому она не является теорией первого порядка. Но этот недостаток тривиальный, и CL может быть преобразована в теорию первого порядка CL+.

Определение 9.5 (теория первого порядка CL+).
Термами CL+ являются CL-термы. Формулы построены из равенств $X = Y$, используя связки и равенство обычным способом.

Правила вывода и логические аксиомы обычные для классической логики первого порядка с эквивалентностью.
Собственные аксиомы:

$$(a) \qquad (\forall x, y)(Kxy = x),$$

$$(b) \qquad (\forall x, y, z)(Sxyz = xz(yz)),$$

$$(c) \qquad (S \neq K)$$

Положим, что имеется формальная теория $\mathcal{T}$ и рассматривается расширение $\mathcal{T}$ добавлением нового правила Ω. Возникает вопрос, строится ли Ω в $\mathcal{T}$.

Поясним смысл высказывания ‘новое правило Ω’. Пусть F – множество всех формул $\mathcal{T}$ и пусть $n \geq 1$.

Определение 9.6. *Каждая частичная функция f из F^n в F определяет правило $\Omega(f)$ следующим образом: каждая n-ка $(A_1, \ldots, A_n)$ в области определения f может быть названа последовательностью посылок, и если $f(A_1, \ldots, A_n) = B$, то соответствующее заключение и выражение*

$$\frac{(A_1, \ldots, A_n)}{B}$$

называется расширением (подстановкой) $\Omega(f)$.

Аналогично, если F^n – множество всех бесконечных последовательностей формул, то каждая частичная функция $f : F^\omega \to F$

определяет правило бесконечной посылки $\Omega(f)$. Правило Ω, определенной этой частичной функцией, называется образованным в $\mathcal{T}$, если и только если для каждого расширения Ω существует дедуктивный вывод в $\mathcal{T}$ от посылки до заключения:

$$\mathcal{T}, A_1, A_2, \ldots \vdash B$$

С другой стороны Ω называют приемлемым в $\mathcal{T}$, если и только если для каждого расширения Ω выполняется: если все предпосылки доказуемы в $\mathcal{T}$, то же верно для заключения, то есть если и только если

$$(\mathcal{T} \vdash A_1), (\mathcal{T} \vdash A_2), \ldots \Rightarrow (\mathcal{T} \vdash B).$$

Наонец, формула C называется и приемлемой, и образованной в $\mathcal{T}$, если и только если $\mathcal{T} \vdash C$.

10. Модели $\mathbf{CL}_\omega$

В данном разделе под термом будем понимать CL-терм. Тождество будем обозначать как '='. '$Vars$' будет обозначать класс переменных, обозначенных через 'x', 'y', 'z', 'u', 'v'. Напротив, 'a', 'b', 'c', 'd', 'e' будут использоваться для обозначения элементов данного множества D. Если точка является отображением из D^2 в D, то выражение вида $(((a \cdot b) \cdot c) \cdot d)$ будет сокращено до $a \cdot b \cdot c$ (соглашение о восстановлении опущенных скобок 'по ассоциации влево').

Определение 10.1 (оценка). *Если D – множество, то любое отображение $\rho : Vars \to D$ будет называться оценкой.*

Для $d \in D$ и $x \in Vars$

$$[d/x]\rho$$

будет обозначать оценку ρ', которая совпадает с ρ всюду, за исключением того, что $\rho'(x) = d$. (Если $\rho(x) = d$, то $[d/x]\rho = \rho$). Модель называется нормальной, когда знак '=' интерпретируется как тождество. В данном разделе под моделью всегда будет пониматься нормальная модель.

Определение 10.2 (аппликативная структура). *Аппликативная структура – это пара $< D, \cdot >$, где D – множество, называемое доменом структуры, содержащее, самое меньшее, два элемента, а точка является некоторым отображением из D^2 в D.*

Моделью CL будет аппликативная структура с дополнительными свойствами, например, 'точка' обладает свойствами функции-аппликации. Условие наличия двух элементов во множестве введено для того, чтобы избежать тривиальности модели.

Определение 10.3 (представимость). *Пусть $< D, \cdot >$ – аппликативная структура и пусть $n \geq 1$. Функция $\theta : D^n \to D$ представима в D, если и только если D имеет элемент a такой, что*

$$(\forall d_1, \ldots, d_n \in D)\ \ a \cdot d_1 \cdot \ldots \cdot d_n = \theta(d_1, \ldots, d_n)$$

Каждый такой a называется представлением θ. Множество всех представимых функций из D^n в D называется

$$(D^n \to D)rep$$

Определение модели CL_ω такое же, как и определение модели в логике первого порядка.

Определение 10.4 (комбинаторная алгебра). *Комбинаторная алгебра – это пара $\mathcal{A} = < D, \cdot >$, где D – множество, содержащее, по крайней мере, два элемента, '$\cdot$' отображает D^2 в D, в D входят элементы k и s такие, что*

$$(1) \qquad (\forall a, b \in D)\ \ k \cdot a \cdot b \ = \ a,$$

$$(2) \qquad (\forall a, b, c \in D)\ \ s \cdot a \cdot b \cdot c \ = \ a \cdot c \cdot (b \cdot c)$$

Моделью CLω является четверка $< D, \cdot, k, s >$ такая, что $< D, \cdot >$ – комбинаторная алгебра, а k, s удовлетворяют условиям (1) и (2).

Определение 10.5 (комбинация). *Комбинацией $x_1, \ldots, x_n$ является любой CL-терм X, единственными атомами которого являются $x_1, \ldots, x_n$. (Терм X не обязательно должен содержать все $x_1, \ldots, x_n$, но он не должен содержать K или S.)*

Любой такой X может быть проинтерпретирован как произвольная аппликативная структура $< D, \cdot >$ естественным образом.

Определение 10.6 (комбинаторная полнота). *Аппликативная структура $< D, \cdot >$ комбинаторно полна, если и только если для каждой последовательности u, $x_1, \ldots, x_n$ и каждой комбинации X из $x_1, \ldots, x_n$ формула*

$$(\exists u)(\forall x_1, \ldots, x_n)(ux_1 \ldots x_n = X)$$

истинна в D.

Это выполняется, если и только если существует $a \in D$ такой, что

$$(\forall d_1, \ldots, d_n \in D)\ a \cdot d1 \cdot \ldots \cdot d_n = [X][d_1/x_1] \ldots [d_n/x_n]\rho,$$

где ρ произвольна.

Часть II

Лабораторный практикум

11. Структура практикума

Курс охватывает все основные понятия и обозначения, используемые в аппликативных вычислительных системах, и позволяет проводить их изучение в полном объеме: от языка и соглашений об обозначениях до метатеорем о соответствующих формальных системах. Курс дает базовые знания об использовании аппликативных вычислительных систем (АВС) в качестве основы функциональных языков, систем функционального программирования и чисто объектных языков.

Центральным используемым понятием является терм, рассматриваемый как *представление* объекта. Терм синтаксически соответствует программе на функциональном языке.

На этом основании *первый* раздел практикума посвящен технике выполнения правильных синтаксических преобразований – расстановке скобок,– для термов различных видов.

Второй раздел предназначен для изучения редукции в АВС. Результатом выполнения функциональной программы является значение, полученное по окончании процесса вычисления. В АВС результату выполнения редукции соответствует нормальная форма терма. В соответствии с теоремой Черча-Россера нормальная форма не зависит от порядка выполнения шагов редукции, что дает возможность построения различных стратегий вычисления в системах функционального программирования. Построение λ-абстракции в комбинаторной логике можно рассматривать в качестве примера для интерпретации логических систем средствами комбинаторной логики.

Так пара комбинаторов K и S образует базис для произвольных λ-термов, в то время как комбинаторы I, B, C, S являются базисом только для термов, в которых отсутствуют свободные переменные. Использование этих двух базисов для разложения термов из двух разделов практикума обеспечивает достаточную полноту рассмот-

рения материала, поскольку будут представлены произвольные λ-термы и комбинаторные термы, которые являются λ-термами без свободных переменных.

Последний раздел практикума является наиболее творческим, так как предполагает конструирование собственных комбинаторных систем. Показано, как путем добавления к базисным комбинаторам дополнительных комбинаторов увеличить выразительные возможности реализованной среды.

В целом, каждый из разделов лабораторного практикума вырабатывает у обучаемого практические навыки по определенному кругу вопросов и может быть использован как отдельная лабораторная работа. Если практикум используется как компьютерное пособие, то разделы могут быть скомпонованы в соответствии с подготовкой обучаемого либо как это нужно преподавателю. Схема экрана при выборе раздела для выполнения представлена на рис. 11.1

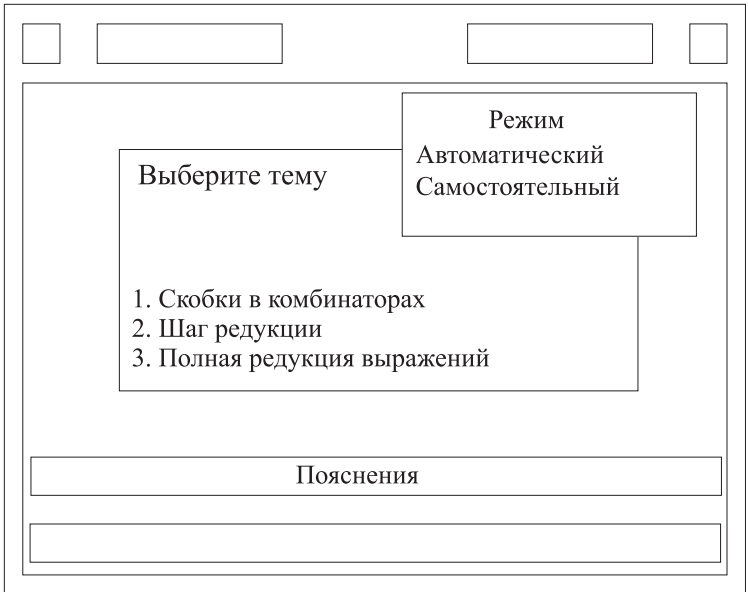

Рис. 11.1. Схема экрана при выборе раздела.

Предложенный порядок следования разделов может быть взят за основу.

12. Механизмы построения практикума

Программное обеспечение поддерживает два режима: ознакомительный и рабочий. В *ознакомительном* режиме обучаемый считывает информацию с экрана, а затем переходит к следующему кадру. В *рабочем* режиме обучаемый либо выбирает альтернативу из меню, либо вводит сообщение в указанное поле. Практически можно выделить три режима решения задач.

1. Просмотр результатов выполнения задания системой. В этом режиме качество знаний обучаемого не оценивается.
2. Динамический просмотр результатов выполнения задания с акцентированием наиболее трудных мест. Как альтернатива может использоваться режим, ориентированный на проверку знания обучаемого и его уверенности в своих знаниях – корректировка промежуточных результатов.
3. Самостоятельное пошаговое решение.

Система осуществляет контроль каждого промежуточного ответа. Качество знаний обучаемого оценивается.

По умолчанию установлен режим динамического просмотра результатов. Однако практически предусмотрено переключение режима работы в любой удобный для обучаемого момент:

1) при входе в систему (см. рис. 11.1),
2) между отдельными заданиями,
3) в процессе выполнения задания.

Для перехода от одной стратегии решения к другой используется функциональная клавиатура, выведенная в нижней строке экрана.

В режиме самостоятельного решения обучаемому предлагается только две попытки для ввода правильного ответа. Если обе попытки были неудачными, то система решает задание сама и выдает результат в поле ответа.

В системе поддерживается режим синтаксического контроля, предполагающий отслеживание неправильно записанных термов, и режим семантического контроля, отслеживающий неправильно примененные правила преобразований. В случае синтаксической ошибки в окно сообщений выводится диагностика ошибки и курсором указывается возможное место ошибки. В случае неправильного применения правила выводится информация о том, какое правило следует применять и как оно будет применено для конкретных аргументов. Синтаксические преобразования выполняются с точностью до правильно расставленных скобок в λ-термах.

Различаются стратегии подбора заданий в различных разделах. В качестве альтернативы обучаемый выполняет только текущее задание, постепенно продвигаясь к следующим без возможности вернуться назад и повторить неправильно сделанное задание. В другом случае он имеет возможность выбрать произвольное задание из списка задач.

Осуществляется контроль, исключающий возможность правильного выполнения каждого задания более двух раз. Раздел считается выполненным, когда исчерпаны все задания, либо по прерыванию пользователя. После каждого раздела обучаемому выводится полученная сумма баллов и информационное сообщение, оценивающее результаты его работы по данному разделу. Результаты лучших игроков заносятся в протокол системы.

13. Восстановление скобок в λ-термах

Цель: отрабатывание практических навыков правильной синтаксической записи комбинаторных термов и λ-термов со скобками.

Схема экрана в лабораторной работе приведена на рис. 13.1.

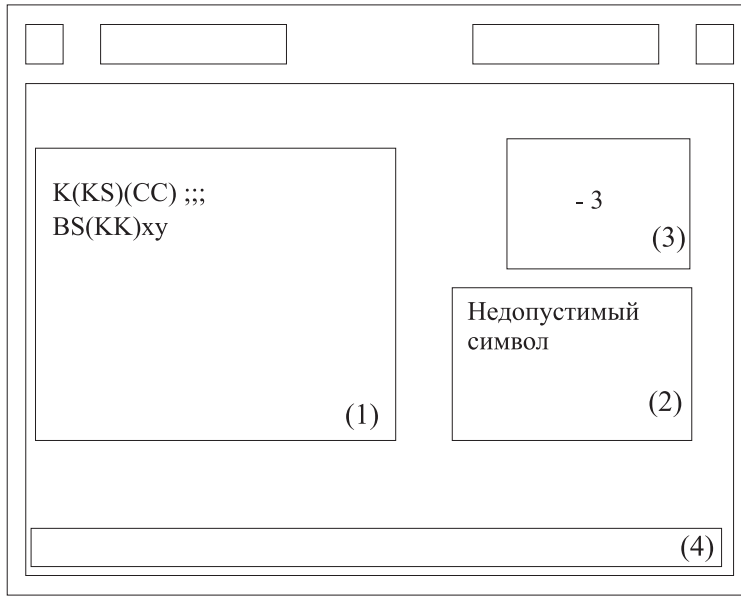

Рис. 13.1. Схема экрана при проверке скобочной записи.

В окне заданий (1) приведены восемь правильно построенных термов в аппликативной записи. Предлагается восстановить возможную расстановку скобок, не нарушая синтаксической правильности терма. В окне (2) высвечивается диагностика ошибки или правило редукции выбранного терма. В случае ошибки набранный результат задания стирается и обучаемому предлагается ввести от-

вет повторно. В окне (4) представлена функциональная клавиатура, которая доступна при выполнении раздела. В окне (3) в режиме самостоятельного решения высвечивается сумма баллов, набранных на текущий момент времени за выполнение заданий данного раздела. В работе используется три режима решения: просмотр результата, корректирование промежуточного результата и самостоятельный ввод.

В процессе выполнения обучаемый видит все задания, но выполнять может только одно. Продвижение к следующему заданию происходит строго последовательно. Возврат к предыдущим заданиям невозможен. Результат вводится в поле результата в окне (1). При успешном выполнении *первых восьми заданий* обучаемому предлагаются более сложные задания.

14. Шаг редукции

Цель: редуцирование комбинаторных выражений.

Схема экрана для выполнения лабораторной работы приведена на рис. 14.1.

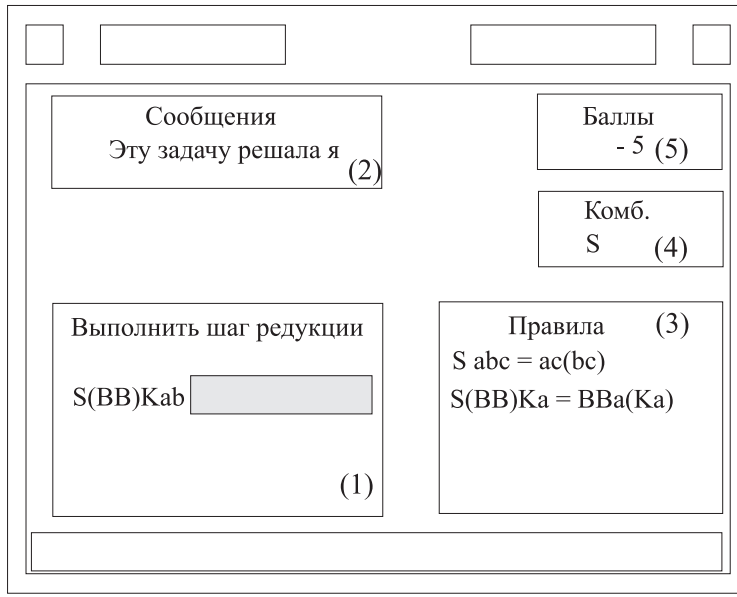

Рис. 14.1. Схема экрана при редукцировании комбинаторных выражений.

В данной работе обучаемому предлагаются различные комбинаторные выражения. Ему следует либо выполнить один шаг редукции, либо выполнить преобразования, обратные редукции, – экспансию, – то есть восстановить исходный терм. Задания выбираются произвольным образом из системного задачника и высве-

чиваются в окне (1). При этом обучаемый видит только текущее задание. По мере выполнения задания система вынуждает его продвигаться на шаг. Стратегия решения только одна – самостоятельный ввод результата.

При правильном решении в окне (2) выводится сообщение, подтверждающее его правильность. В случае неправильного решения система выдает правильный ответ текущего задания, следующего задания и информационное сообщение в окне (2). Сумма набранных на текущий момент времени баллов выводится в окне (5). Правильное синтаксическое преобразование выполняется с точностью до расстановки скобок. Решение задания сопровождается показом имени применяемого правила в окне (4).

При ошибочном вводе обучаемому демонстрируется общее применение нужного правила и применение этого же правила к конкретным аргументам из задания в окне (3). Введенный результат при этом стирается.

15. Полная редукция

Цель: закрепить у обучаемого понятие нормальной формы комбинаторного терма, дать практические навыки по выполнению редукции комбинаторных выражений.

Схема экрана при выполнении лабораторной работы приведена на рис. 15.1.

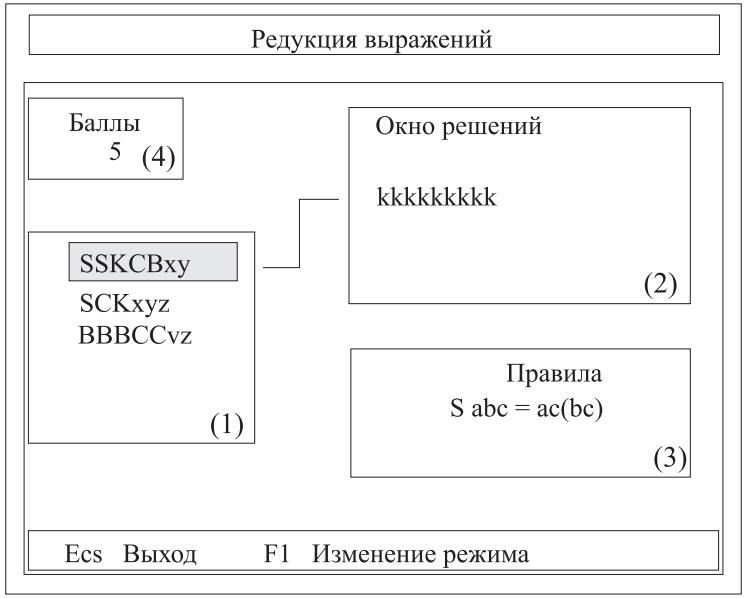

Рис. 15.1. Схема экрана при выполнении редукции.

В данной работе обучаемому предлагается набор комбинаторных выражений. Все эти выражения нужно привести к нормальной форме. Обучаемый видит на экране все задания, относящиеся

к данному разделу и может выполнять любое из них. В случае невыполнения какого-либо из заданий к нему можно вернуться.

Решение задания происходит в трех стандартных режимах: просмотр результата, пошаговый просмотр результата и самостоятельный пошаговый ввод. Системный задачник находится в окне (1). Результаты решения каждого задания вводятся в поля ответов окна (2). Набранная на текущий момент сумма баллов в режиме самостоятельного ввода высвечивается в окне (4).

В случае ошибки в окне (3) выводится общий вид применяемых в данном случае правил либо диагностика синтаксической ошибки. Введенный неправильный результат стирается.

16. Разложение термов в базисах I, K, S и I, B, C, S

Цель: произвести разложение λ-терма в различных базисах. Продемонстрировать несколько возможных решений одного задания.

Схема экрана при выполнении лабораторной работы приведена на рис. 16.1.

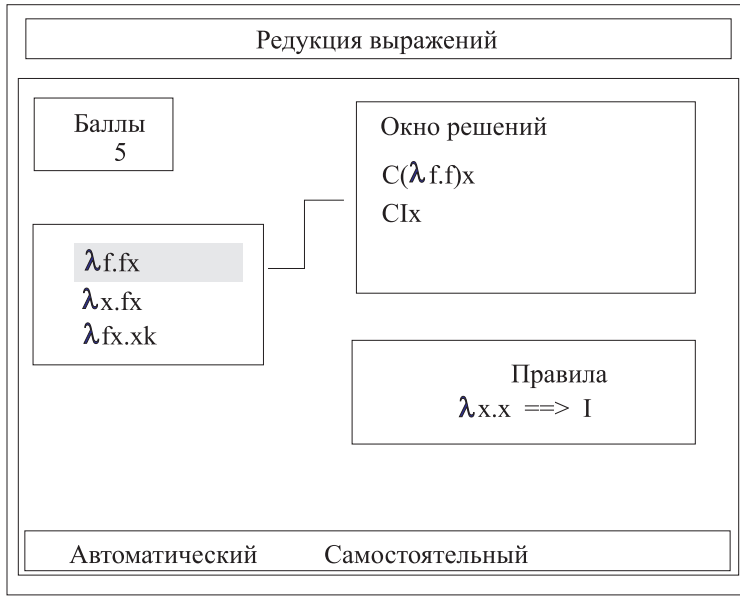

Рис. 16.1. Схема экрана при разложении в базисе.

Обучаемому предлагается набор λ-термов, которые нужно разложить либо в базисе I, K, S, либо в базисе I, B, C, S. Часть заданий в этих разделах повторяется, то есть λ-термы, не имеющие свободных переменных раскладываются в обоих базисах. В свою очередь ряд λ-термов, предложенных для разложения в бази-

се I, K, S, не может быть разложен в базисе I, B, C, S, а попытка разложения приводит к зацикливанию.

Схема выполнения работ аналогична схеме выполнения работы “Полная редукция” из раздела 15.

17. Расширение базисной системы комбинаторов

Цель: добавлением в базисную систему комбинаторов двух-трех дополнительных комбинаторов получить среду, достаточную для описания некоторого потенциального приложения. Проверить правильность введенных комбинаторов, выполняя полную редукцию придуманного обучаемым выражения.

В ряде приложений для представления задачи неудобно пользоваться базисной системой комбинаторов, поскольку результирующие термы получаются громоздкими, а операции по их конвертированию требуют значительного времени и усилий. В качестве примера рассмотрим представление логики. Возможно введение комбинаторов, представляющих истинностные значения:

$T\ x\ y\ =\ x$ – обозначает истину,
$F\ x\ y\ =\ y$ – обозначает ложь.

На основе введенных комбинаторов легко представить комбинатор, вычисляющий условное выражение:

$$IF\ x\ y\ z\ =\ x\ y\ z,$$

в котором терм 'x' может принимать истинностные значения T или F, а IF рассматривается как конструкция $if\ \ldots\ then\ \ldots\ else\ \ldots$. С использованием этого условного выражения возможно выразить логические связки, то есть логические функции, пользуясь комбинаторами:

$$\begin{aligned} NOT\ x &= IF\ x\ F\ T \\ AND\ x\ y &= IF\ x\ y\ F \\ OR\ x\ y &= IF\ x\ T\ y \end{aligned}$$

Предлагается самостоятельно убедиться в корректности приведенных равенств.

В другом случае, при введении арифметики в рамках комбинаторной логики требуется комбинаторное представление числа 0

и функции “следования за”, то есть функции прибавления 1. Скажем, при введении оператора каррирования требуется ввести специальный комбинатор пары, точно представляющий интуитивное понятие об упорядоченной паре.

Схема экрана для выполнения лабораторной работы приведена на рис. 17.1.

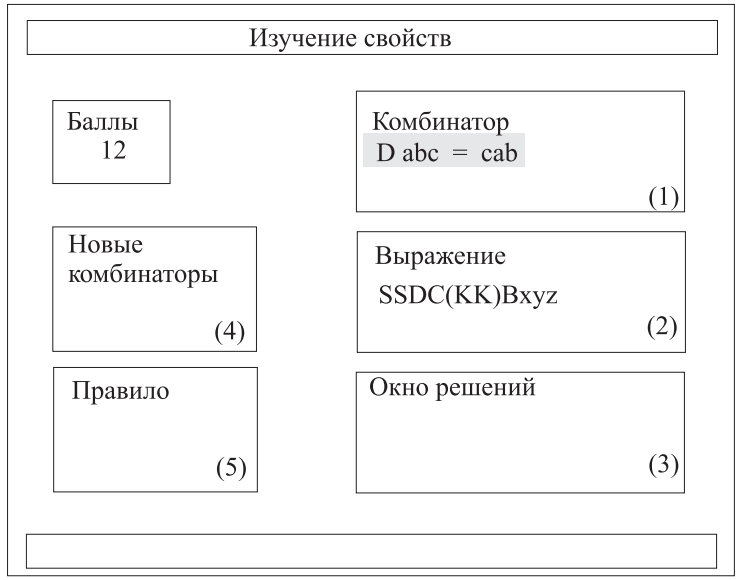

Рис. 17.1. Схема экрана при при добавлении к среде нового комбинатора.

Обучаемому предлагается в базисную систему комбинаторов добавить комбинатор в виде его комбинаторной характеристики:

имя комбинатора	список аргументов	правило преобразования
D	$a\ b\ c$	$c\ a\ b$,

где имя комбинатора образует один символ, в списке аргументов перечисляются имена переменных (не должны повторяться), а правило преобразования заключается в добавлении или удалении одного или нескольких аргументов, изменении порядка следования

аргументов, появлении скобочных структур. Затем предлагается ввести комбинаторное выражение для выполнения редукции.

Выражение должно состоять из определенных в системе на текущий момент комбинаторов. Далее выполняется редукция выражения. В процессе работы оценивается умение обучаемого выполнять редукцию. Выполнение работы начинается с введения нового комбинатора в окно (1).

При введении каждого нового комбинатора происходит контроль логической непротиворечивости списка комбинаторов.

В случае, если комбинатор с таким же названием или с таким же содержанием уже определен в системе, то выдается сообщение и комбинатор не добавляется к списку. Если же комбинатор введен синтаксически верно, то он добавляется к уже зарегистрированным в системе, в противном случае выдается сообщение о характере ошибки. До тех пор, пока комбинатор не будет правильно введен, не возможен переход к следующему этапу задания.

Добавленный в систему комбинатор высвечивается в окне (4). В результате правильного ввода комбинаторного выражения в окне (2), в окне (3) выполняется пошаговая редукция этого выражения.

В случае ошибочного ввода в окне (5) высвечивается диагностика ошибок или правило редукции на текущем шаге. Работа считается выполненной, если обучаемый добавил в систему хотя бы три комбинатора.

18. Техника анализа введенных результатов

Выполнение пошагового контроля вводимых результатов обеспечивается внутренним представлением λ-термов и комбинаторных термов в виде бинарных деревьев. Построение терма осуществляется с самого нижнего левого листа. Таким образом, каждый анализируемый элемент терма становится правой ветвью надстраиваемого дерева. Такое построение поддерживает правило восстановления опущенных скобок ‘по ассоциации влево’. Синтаксический анализ терма производится слева направо. Если очередной анализируемый символ является атомарным объектом, то строится лист из текущего узла, если анализируется список, то из текущего узла строится дерево. Внутреннее представление терма $SK(KK)Bcxyz$ приведено на рис. 18.1 a).

Представление λ-термов в памяти несколько отличается от представления комбинаторных термов. В этом случае при построении дерева учитывается то, что в λ-выражениях скобки восстанавливаются ‘по ассоциации вправо’. Построение дерева такого терма до самого правого символа точки в терме организовано следующим образом: построение начинается с верхнего узла, левым листом текущего узла является переменная, связанная с ассоциированным символом λ, а правым – узел для построения дерева для следующей переменной. Если очередной анализируемый символ – список, то вместо листа строится дерево разбора этого списка.

На рис. 18.1 b) приведено построение внутреннего представления для λ-терма $\lambda xyz.x(yz)(BB)z$.

Комбинатор задается списком своих аргументов и правилом преобразования, то есть формулой из этих аргументов и скобок. Дерево комбинатора аналогично дереву комбинаторного терма. Однако при этом на месте имен переменных стоят номера аргументных мест этих переменных.

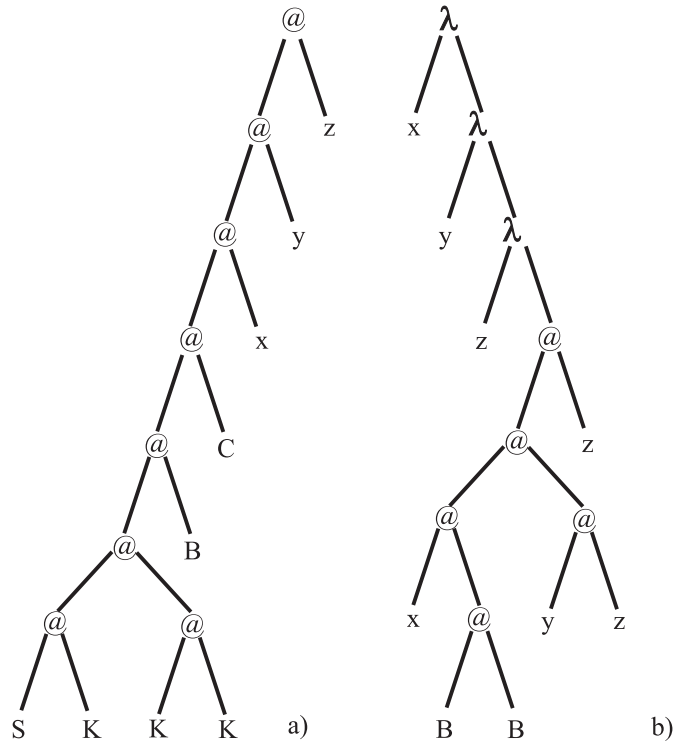

Рис. 18.1. Внутреннее представление термов: a) $SK(KK)Bcxyz$, b) $\lambda xyz.x(yz)(BB)z$.

Внутреннее представление комбинатора $Sxyz = xz(yz)$ приведено на рис. 18.2.

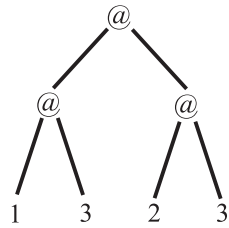

Рис. 18.2. Внутреннее представление комбинатора $Sxyz = xz(yz)$.

Таким образом, в представлении комбинаторов заложены правила их применения. При выполнении редукции терма происходит выделение части терма, необходимой для совершения редукции. Механизм применения правила (принципа) свертывания заключается в том, что создается новое дерево, в котором копируется структура комбинатора, только на место номеров переменных (аргументных мест) восстанавливаются реальные переменные, ассоциированные этим номерам. Далее происходит замещение выделенного элемента на построенный.

19. Начисление баллов

В режиме самостоятельного ввода, если правильный ответ был введен с первого раза, за него начисляется 5 баллов. Если правильный ответ был введен со второго раза, за него начисляется 3 балла. В случае невыполнения задания из общей суммы баллов вычитается 3 балла. В режиме корректировки промежуточного результата за правильный ответ, введенный с первого раза, начисляется 3 балла, а за правильный ответ, введенный со второго раза, начисляется 1 балл. В случае невыполнения задания из общей суммы баллов вычитается 3 балла.

Ответы к упражнениям

Ответ к упражнению 2.1.

$$\begin{aligned} D(\lambda x.x^2) &= \lambda x.2x \\ (D(\lambda x.x^2))3 &= 6 \end{aligned}$$

Ответ к упражнению 2.2.

$$\begin{array}{l} (((ux)(yz))(\lambda v.(vy))); \\ ((((\lambda x.(\lambda y.(\lambda z.((xz)(yz)))))u)v)w); \\ (((w(\lambda x.(\lambda y.(\lambda z.((xz)(yz))))))u)v) \end{array}$$

Ответ к упражнению 3.1.
Не входит. При расстановке скобок терм $ux(yz)$ превращается в терм $((ux)(yz))$.
Ответ к упражнению 3.2.
Этот терм принадлежит терму (ii), но не (iii).
Ответ к упражнению 3.3.

$$\begin{array}{l} \lambda y.(\lambda y.xy)(\lambda x.x) \\ y(\lambda z.(\lambda y.vy)z) \end{array}$$

Ответ к упражнению 4.1.
На рис. 19.1 и рис. 19.2 приведены древовидные диаграммы только для первых двух примеров.
Ответ к упражнению 7.1.
Доказательство:

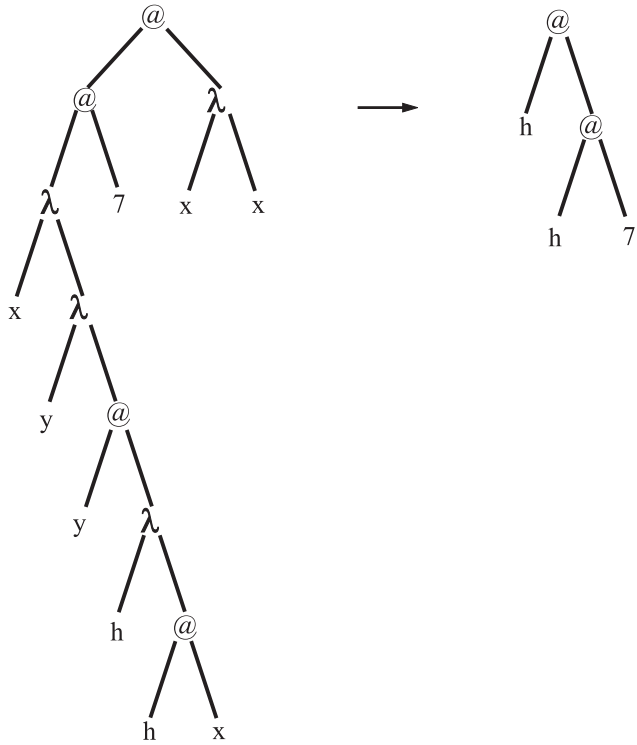

Рис. 19.1. Диаграмма для первого примера.

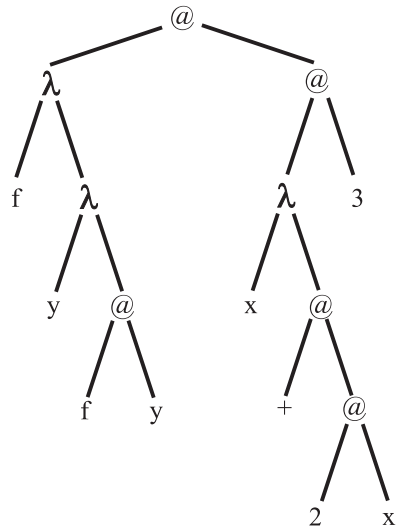

Рис. 19.2. Диаграмма для второго примера.

$$\begin{aligned} S(Kx)(Ky)a &= Kxa(Kya) \\ &= xy \\ K(xy)a &= xy \\ S(Kx)Ia &= Kxa(Ia) \\ &= x(Ia) \\ &= xa \\ S(Kx)ya &= Kxa(ya) \\ &= x(ya) \\ Bxya &= x(ya) \end{aligned}$$

Ответ к упражнению 7.4.

$S(S(K+)1)(K1)$
$S(SI(KI))(KI)$
$S(S(S(Kcond)(S(S(K=)1)(K0)))(K1))(S(S(K-)+)(K1))$

Ответ к упражнению 8.2.
Доказательство.
$X = ZZ$, где $Z = \lambda x.\lambda y.y(xxy)$. Тогда для λ-выражения E имеем:

$$XE = (ZZ)E = (\lambda y.y(ZZy))E = E(ZZE) = E(XE).$$

Литература

[1] Вольфенгаген В.Э. *Комбинаторная логика в программировании. Вычисления с объектами в примерах и задачах.* М.: МИФИ, 1994; М.: ИАО “ЮрИнфоР-МГУ”, 2000.

[2] Воскресенская О.В. *Методы разработки реляционной системы управления базой данных.* Диссертация на соискание ученой степени кандидата технических нвук (специальность 05.13.06 – Автоматизированные системы управления), М.: МИФИ, 1985.

[3] Гаврилов А.В. *Настраиваемая система программирования для категориальных вычислений.* Диссертация на соискание ученой степени кандидата технических нвук (специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов, систем и сетей), М.: МИФИ, 1995.

[4] Гольцева Л.В. *Аппликативная вычислительная система с интенсиональными отношениями.* Диссертация на соискание ученой степени кандидата технических нвук (специальность 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов, систем и сетей), М.: МИФИ, 1995.

[5] Джордж Ф. *Основы кибернетики.* Пер. с англ../Под ред. А.Л. Горелика. М.: Радио и связь, 1984. 272 с.

[6] Кузин Л.Т. *Основы кибернетики.* 2-е изд., перераб. и доп. М.: Энергоатаомиздат, 1994. 576 с.

[7] Amadio R. and Curien P.-L. *Domains and lambda-calculi.* Cambridge University Press, 1998.

[8] Field R., Harrison J. *Functional programming.* Academic Press, New York, 1990. 4

[9] Frege G. et al. *From Frege to Gödel, A Source Book in Mathematical Logic 1897-1931.* Cambridge. Mass., 1967

[10] Hindley R. and Seldin J. *Introduction to combinators and lambda-calculus.* Cambridge University Press, 1986.

[11] Curry H.B., Hindley R., Seldin J.P. *Combinatory Logic.* Vol. II, North-Holland Co., 1972.

[12] Hindley R., Lercher B., Seldin J.P. *Introduction to Combinatory Logic*. Cambridge Univ. Press, 1972.

[13] Hindley R., Seldin J.P. (Eds.) *To H. B. Curry*. Academic Press, 1980.

[14] Hindley R., Seldin J.P. *Introduction to Combinators and Lambda-calculus*. Cambridge Univ. Press, 1986.

[15] Hindley R. *Basic Simple Type Theory*. Cambridge Univ. Press, 1995.

[16] Hindley R. *The principal type-scheme of an object in combinatory logic*. Trans. American Math. Soc. 146 (1969). – pp.29-60.

[17] Hindley R. *Standard and normal reductions*. Trans. American Math. Soc.241 (1978). – pp. 253-271.
[18] Hindley R., Longo G. *Lambda-calculus models and extensionality*. Zeit. Math. Logik 26 (1980). – pp. 289-310.
[19] Hindley R. *The completeness theorem for typing lambda terms*. Theoretical Computer Sci. 22 (1983). – pp. 1-17.
[20] Hindley R., Meredith D. *Principal type-schemes and condensed detachment*. J. Symbolic Logic 55 (1990). – pp. 90-105.
[21] Hindley R., Dezani M. *Intersection types for combinatory logic*. Theoretical Computer Sci. 100 (1992). – pp. 303-324.
[22] Hindley R. *Types with intersection: an introduction*. Formal Aspects of Computing 4 (1992). – pp. 470-486.

Предметный указатель

Вячеслав Эрнстович **Вольфенгаген**
Людмила Владимировна **Гольцева**
Лариса Юсифовна **Исмаилова**

Аппликативные вычисления на основе комбинаторов и λ-исчисления.

Корректор:
Макет: Автор

ЗАО “ЮрИнфоР”
125009, Москва, Брюсов пер., 8-10, стр. 2, тел. (495) 743-73-96,
http://www.jurinfor.ru, эл. почта: info@jurinfor.ru

Все права принадлежат авторам