УДК 378.146:004

И. А. Кудрявцева

ЛОГИКО-СЕМИОТИЧЕСКИЙ АНАЛИЗ СОДЕРЖАНИЯ ОБУЧЕНИЯ МОНОМОРФНОЙ СИСТЕМЕ ТИПОВ λ→

Автор предлагает направление развития содержания обучения теоретическому программированию, которое понимается как наука о математических моделях понятия «программа» и включающая следующие разделы: λ-исчисление, теория комбинаторов, теория типов, теория категорий. Проведен логико-семиотический анализ содержания одного из разделов обучения теоретическому программированию «мономорфная система типов $\lambda_{\rightarrow}$» (демонстрирующая зависимость термов от термов). Основной акцент сделан на методах решения основных типов задач системы с использованием интерпретатора GHC, содержащего инструментарий по установлению существования вывода типизированного λ-терма и вывода типа λ-терма. В результате сформулированы компетенции, приобретаемые обучаемыми по итогам изучения мономорфной системы типов.

Ключевые слова: *теоретическое программирование, λ-куб, мономорфная система типов, логико-семиотический анализ содержания, изоморфизм Карри-Говарда, минимальная пропозициональная логика (Prop).*

Теоретическое программирование представляет собой раздел математической науки, предназначенный для рассмотрения моделей программ с целью выделения и разработки новых свойств языков программирования. При этом математические модели программ рассматриваются с помощью исчислений таких теорий, как λ-исчисление, теории комбинаторов, теории типов, теории категорий, семантической теории языков программирования.

Теория типов изучает одно из свойств современных языков программирования – типизируемость, т. е. приписывание типов программным объектам. В настоящее время теория типов превратилась в одну из основных разновидностей логических исчислений высшего порядка, основной принцип которой состоит в том, что логические понятия располагаются в определенную иерархию «типов»: функция имеет в качестве аргументов лишь понятия, предшествующие ей в этой иерархии.

Системой типов [1, Введение, с. 1] называется гибко управляемый синтаксический метод доказательства отсутствия в программе определенных видов поведения при помощи классификации выражений языка по разновидностям вычисляемых ими значений.

Необходимость типов отмечается следующими положениями:

(а) типы позволяют получить частичную спецификацию;

(б) правильно типизированные программы не могут «сломаться»;

(в) проверка типов позволяет обнаружить простые ошибки.

Возможные зависимости между типами и λ-термами (как одного из средств описания моделей программ) отображены в системах, которые Х. Барендрегт (1991) представил λ-*кубом* или *кубом Барендрегта*.

Единообразное описание восьми различных систем типизированного λ-исчисления с явным приписыванием типов (систем, типизированных по Чёрчу), составляющих вершины λ-куба, предоставляет инструмент для построения языков с системой вывода типов и доказательства предполагаемого поведения программы до ее запуска (соответствия некоторой спецификации).

Простейшая система типов λ-куба – просто типизированное λ-исчисление $\lambda_{\rightarrow}$ (STT, англ. *Simple Type Theory*), в которой термы зависят от термов.

Дефицит русскоязычной литературы по основам теории типов языков программирования [1–4], а также малое число переводов (приводящее к терминологическим трудностям) сказались на отсутствии в университетских курсах должного внимания к рассматриваемым ниже вопросам. Поэтому данное содержание представляет собой информационную базу, являющуюся неотъемлемой частью системы содействия профессиональному саморазвитию преподавателя вуза [5] и способствующую формированию информационной компетентности будущих специалистов в области теоретического программирования [6].

Приступим к логико-семиотическому анализу мономорфной системы типов $\lambda_{\rightarrow}$ в обучении теоретическому программированию.

Логико-семиотический анализ позволит рассмотреть содержание обучения со следующих позиций (по [7, с. 223]):

(1) семиотики (с помощью явного выделения метаязыка, предметного языка, синтаксиса, семантики);

(2) формальной логики (с помощью явного выделения формальных систем, формальных языков, формальных и содержательных аксиоматических теорий);

(3) алгоритмов (с помощью явного выделения вычислительных моделей, алгоритмов и их реализаций с помощью вычислительных систем).

I. *Семиотический компонент содержания*.

Предметный язык: язык типизации λ-термов.

Метаязыки: язык канторовской теории множеств, язык минимальной пропозициональной логики (*Prop*), язык бестипового λ-исчисления.

II. *Формально-логический компонент содержания*.

Формальные системы: просто типизированное λ-исчисление в стиле А. Чёрча и в стиле Х. Карри.

Формальная система включает описание двух компонентов (по [8, с. 324–326]): язык в алфавите (класс исходных символов, правильные слова в алфавите, формулы) и исчисление (аксиомы, правила вывода).

1. ***Язык в алфавите***:

(а) *класс исходных символов* (*алфавит*): V={x, y, z,…} – *множество предметных переменных*; **0** – *основной* (*элементарный*) *тип*, *тип индивида*; → – *синтаксический конструктор типа*; = – *двухместный предикатный символ*; λ – *логический символ*; **(**, **)**,. – *вспомогательные символы*;

(б) *класс правильных слов в алфавите*.

Множество типов (Typ) – это множество слов в алфавите {**0**,**(**,**)**}, называемых *типами*, которое индуктивно определяется следующим образом (по [9, с. 548]): (1) **0** ∈ Typ (*элементарный тип)*; (2) если α ∈ Typ и β ∈ Typ, то **(**αβ**)** ∈ Typ; (3) других типов, кроме построенных по пп. (1) и (2), нет.

Скобки в типе восстанавливаются *по ассоциативности вправо*: $(\alpha_1(\alpha_2(...\alpha_n)))$.

Далее по тексту множество типов будем обозначать Т.

Предтермы (*псевдотермы*) имеют вид:

в стиле Чёрча (ΛТ):	*в стиле Карри (Λ):*
$\begin{cases} x\in V & \Rightarrow x\in\Lambda_T; \\ M,N\in\Lambda_T & \Rightarrow (MN)\in\Lambda_T; \\ M\in\Lambda_T,\ x\in V,\ \alpha\in T & \Rightarrow (\lambda x{:}\alpha.M)\in\Lambda_T \end{cases}$	$\begin{cases} x\in V & \Rightarrow x\in\Lambda; \\ M,N\in\Lambda & \Rightarrow (MN)\in\Lambda; \\ M\in\Lambda,\ x\in V & \Rightarrow (\lambda x.M)\in\Lambda. \end{cases}$

Приписывание типа α терму x обозначим x: α или x^{α}.

Типизированный λ-терм имеет вид [10, с. 132]:

(1) каждая типизированная переменная x^{α} (x ∈ V, α ∈ T) есть *типизированный λ-терм*;

(2) если $M^{\alpha\beta}$ и N^{α} – типизированные λ-термы (α,β ∈ T), то $(M^{\alpha\beta}N^{\alpha})^{\beta}$ или $(M^{\alpha\to\beta}N^{\alpha})^{\beta}$ является *типизированным λ-термом*;

(3) если x^{α} – типизированная переменная (α ∈ T), а M^{β} – типизированный λ-терм (β ∈ T), то $(\lambda x^{\alpha}.\ M^{\beta})^{\alpha\beta}$ или $(\lambda x^{\alpha}.M^{\beta})^{\ \alpha\to\beta}$ является *типизированным λ-термом*.

Утверждение о типизации в $\lambda_{\to}$ имеет вид

в стиле Чёрча:	*в стиле Карри:*
$M{:}\tau,\ M\in\Lambda_T,\ \tau\in T$	$M{:}\tau,\ M\in\Lambda,\ \tau\in T.$

Тип τ называют *предикатом*, а λ-терм – *субъектом утверждения*. Разница между стилем Чёрча и Карри очевидна: абстракция в стиле Чёрча определяется однозначно, а в стиле Карри – нет:

в стиле Чёрча:	*в стиле Карри:*
$(\lambda x{:}\alpha.x){:}\alpha\to\alpha,$	$\lambda x.x{:}\alpha\to\alpha,$
$(\lambda x{:}\alpha\to\beta.x){:}(\alpha\to\beta)\to\alpha\to\beta$	$\lambda x.x{:}(\alpha\to\beta)\to\alpha\to\beta$

(в) *класс формул*.

Формулами типизированного λ-исчисления называются слова вида

$M = N,\ M,N \in \Lambda_{\alpha},\ \alpha \in T;$

(г) *свойства* λ-*термов*.

Понятия «*свободная переменная*», «*связанная переменная*» и «*замкнутый λ-терм*» для системы $\lambda_{\to}$ вводятся аналогично соответствующим понятиям для бестипового λ-исчисления; понятие «*нормальная форма*» аналогично определению в бестиповом λ-исчислении;

(д) *операция над* λ-*термами*.

Операция «*подстановка*» $(M)^{x}_{N}$ определяется для типизированных λ-термов точно так же, как и для бестиповых λ-термов, за исключением того, что x и N должны иметь один и тот же тип; в том случае, если x и N имеют разные типы, то подстановка $(M)^{x}_{N}$ не определена;

(е) *основные теоремы и леммы*.

Из многочисленных теорем выделим следующую [9, с. 550]: любой типизированный λ-терм имеет нормальную форму.

2. ***Исчисления***.

Типизированные исчисления в стиле Чёрча и в стиле Карри отличаются с точки зрения синтаксиса записью правила обработки абстракции, в остальном они схожи и представимы как в терминах гильбертовского исчисления, так и в терминах генценовского исчисления.

(а) Приведем *аксиомы* и *правила вывода* типизированных λ-термов, предварительно введя понятие «*контекст*».

Контекст (*окружение*) – это частичная функция из множества *различных* переменных V в множество типов Т:

$$\Gamma \overset{\text{fin}}{=} \{x_1{:}\sigma_1, x_2{:}\sigma_2, \ldots, x_n{:}\sigma_n\}.$$

Утверждением, выводимым в контексте **Γ**, называется утверждение M:τ, вывод которого *выполняется по правилам типизации системы* λ→:

в гильбертовском исчислении	*в генценовском исчислении*
$\Gamma \vdash M{:}\tau$	$M{:}\tau\in\Gamma$

(а) *схемы аксиом*:

в стиле Карри

(0) $\Gamma\cup\{M{:}\beta, M\equiv_{\alpha}N\}\vdash N{:}\beta;$	(0) $\dfrac{M{:}\beta \quad M\equiv_{\alpha}N}{N{:}\beta}$ (*общелогический принцип замены равного на равное*);

в стиле Карри и Чёрча

(1) $\Gamma\cup\{x{:}\sigma\}\vdash x{:}\sigma,\ x\notin\Gamma;$	(1) $\dfrac{x{:}\sigma\in\Gamma}{x{:}\sigma};$

(б) *правила вывода*:

в стиле Карри и Чёрча

в гильбертовском исчислении	*в генценовском исчислении*
$\Gamma \vdash M:\tau$	$M:\tau \in \Gamma$
(2) $\dfrac{\Gamma \vdash M:\sigma\to\tau \quad \Gamma \vdash N:\sigma}{\Gamma \vdash MN:\tau}$;	(2) $\dfrac{M:\sigma\to\tau \quad N:\sigma}{MN:\tau}$ (*удаление* →);
в стиле Чёрча	
$(3_1)\ \dfrac{\Gamma\cup\{x:\sigma\} \vdash M:\tau}{\Gamma \vdash \lambda x:\sigma.M:\sigma\to\tau}$;	$(3_1)\ \dfrac{\begin{matrix}[x:\sigma]\\ M:\tau\end{matrix}}{\lambda x:\sigma.M:\sigma\to\tau}$ (*введение* →);
в стиле Карри	
$(3_2)\ \dfrac{\Gamma\cup\{x:\sigma\} \vdash M:\tau}{\Gamma \vdash \lambda x.M:\sigma\to\tau}$.	$(3_2)\ \dfrac{\begin{matrix}[x:\sigma]\\ M:\tau\end{matrix}}{\lambda x.M:\sigma\to\tau}$ (*введение* →).

Если существуют Γ и τ такие, что $\Gamma \vdash M:\tau$, то предтерм M называют *допустимым λ-термом* (или просто *λ-термом*).

(б) *определение понятия* «*вывод*».

Понятие «*древесный вывод утверждения* $M:\tau$ *в контексте* Γ» определяется подобно Г. Генцену [11, с. 15–16]: «...вывод называется *древовидным*, если все его утверждения являются нижними утверждениями не более чем одной фигуры заключения. *Исходными утверждениями вывода* называются *утверждения*, которые не являются нижними утверждениями никаких фигур заключения... Исходные утверждения некоторого вывода называются *допущениями*... *Конечным утверждением вывода* называется утверждение, которое не является верхним утверждением никакой фигуры заключения... Вывод с конечным утверждением $M:\tau$ называется *выводом утверждения* $M:\tau$».

3. *Интерпретация*.

При *программистском подходе* (*в практическом программировании*) λ-термы интерпретируются как программы, а типы – как их частичные *спецификации*.

При *логическом подходе* (*в теории типов*) типы интерпретируются как формулы некоторого логического языка, а λ-термы – как доказательства этих формул.

По *изоморфизму Карри-Говарда:* формулы в логической системе *Prop* – это типы в системе $\lambda_{\to}$. Поэтому если в представленных правилах оставить только типы, то получатся правила системы *Prop*; доказательства в логической системе *Prop* – это термы в системе $\lambda_{\to}$.

III. *Алгоритмический компонент содержания*.

Важнейшими задачами в системе $\lambda_{\to}$ являются:

(1) *задача проверки типа* (*TCP*, англ. «*Type Checking Problem*») – проверка соответствия заданному λ-терму заданного типа;

(2) *задача синтеза* (*реконструкции*) *типа* (*TSP*, англ. «*Type Synthesis Problem*») – восстановление типа по заданному λ-терму;

(3) *задача обитаемости типа* (*TIP*, англ. «*Type Inhabitation Problem*») – восстановление λ-терма по заданному типу.

Используя правила типизации как в стиле гильбертовского и генценовского исчисления, рассмотрим методы решения представленных выше задач.

Предварительно отметим, что запись в стиле генценовского исчисления выглядит более лаконично по сравнению с записью в стиле гильбертовского исчисления, поскольку в этом случае используется исчисление формул со списком допущений (фактически являющегося стеком); в случае гильбертовского исчисления используется исчисление выводов. Далее приведем примеры соответствующих методов решения задач в стиле генценовского исчисления (за исключением задачи TCP, где продемонстрируем также и стиль гильбертовского исчисления).

1. *Решение задачи TCP*.

1.1. *В стиле Чёрча* проверку соответствия λ-терму некоторого типа можно осуществить одним из двух способов:

(1) обратного вывода (*снизу вверх*), в котором список допущений (контекст) пополняется типизированными переменными с заданными типами;

(2) прямого вывода (*сверху вниз*), в котором в список допущений (контекст) предварительно добавляются типизированные переменные с заданными типами.

Приведем пример решения задачи TCP для λ-терма с контекстом Γ:

$$\underbrace{\lambda x:\alpha.\lambda y:(\beta\to\alpha)\to\gamma.y(\lambda z:\beta.x)}:\alpha\to((\beta\to\alpha)\to\gamma)\to\gamma,$$

$$\Gamma \stackrel{\text{def}}{=} \{x:\alpha, y:(\beta\to\alpha)\to\gamma, z:\beta\}$$

(а) в стиле гильбертовского исчисления (рис. 1):

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash y:(\beta\to\alpha)\to\gamma \quad \dfrac{\Gamma \vdash x:\alpha}{\Gamma\setminus\{z:\beta\} \vdash \lambda z:\beta.x:\beta\to\alpha}}{\Gamma\setminus\{z:\beta\} \vdash y(\lambda z:\beta.x):\gamma}}{\Gamma\setminus\{y:(\beta\to\alpha)\to\gamma, z:\beta\} \vdash \lambda y:(\beta\to\alpha)\to\gamma.y(\lambda z:\beta.x):((\beta\to\alpha)\to\gamma)\to\gamma}}{\Gamma\setminus\{x:\alpha, y:(\beta\to\alpha)\to\gamma, z:\beta\} \vdash \lambda x:\alpha.\lambda y:(\beta\to\alpha)\to\gamma.y(\lambda z:\beta.x):((\beta\to\alpha)\to\gamma)\to\gamma}$$

Рис. 1. Решение задачи TCP в оформлении гильбертовского исчисления

(б) в стиле генценовского исчисления (рис. 2):

$$\dfrac{\dfrac{\dfrac{\begin{matrix}2\\ \downarrow\\ y:(\beta\to\alpha)\to\gamma\end{matrix} \quad \dfrac{\begin{matrix}1\\ \downarrow\\ x:\alpha\end{matrix}}{\lambda z.x:\beta\to\alpha}}{y(\lambda z:\beta.x):\gamma}}{\lambda y:(\beta\to\alpha)\to\gamma.y(\lambda z:\beta.x):((\beta\to\alpha)\to\gamma)\to\gamma}}{\lambda x:\alpha.\lambda y:(\beta\to\alpha)\to\gamma.y(\lambda z:\beta.x):\alpha\to((\beta\to\alpha)\to\gamma)\to\gamma}$$

Список допущений:
1. $x:\alpha$
2. $y:(\beta\to\alpha)\to\gamma$
3. $z:\beta$

Рис. 2. Решение задачи TCP в оформлении генценовского исчисления

1.2. *В стиле Карри* проверка соответствия λ-терму некоторого типа осуществляется путем построения:

(1) обратного вывода (*снизу вверх*), как было представлено выше в стиле Чёрча, исключая из записи типы типизированных переменных при абстракции;

(2) прямого вывода терма (*сверху вниз*), вводя обозначения типов для типизированных переменных и предварительно добавляя их в список допущений (контекст).

Если действовать по правилам типизации, то возможно конструирование системы равенств типов, которая затем решается для получения главного типа λ-терма. Затем полученный результат унифицируется с заданным типом λ-терма; если унификация удастся, то λ-терму соответствует заданный тип, в противном случае – нет.

Приведем решение задачи TCP (в стиле Карри, используя прямой вывод в направлении сверху вниз (рис. 3)) для λ-терма с заданным типом:

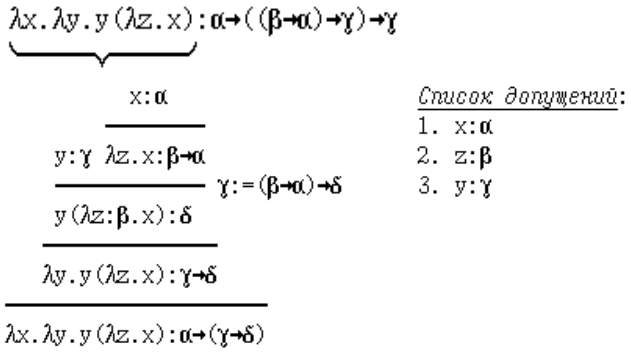

Рис. 3. Дерево вывода терма (в стиле Карри) с типом

Получим главный тип λ-терма:

$$[\gamma:=(\beta\to\alpha)\to\delta]\ (\alpha\to(\gamma\to\delta))=\alpha\to((\beta\to\alpha)\to\delta)\to\delta.$$

Унифицируем полученный главный тип λ-терма с его заданным типом, используя алгоритм Робинсона, который фактически представляет обход бинарных деревьев, содержащих типы.

Отобразим унифицируемые типы в виде деревьев; тем самым неформально получим результат унификации (рис. 4):

$$\alpha\to((\beta\to\alpha)\to\delta)\to\delta,\quad \alpha\to((\beta\to\alpha)\to\gamma)\to\gamma$$

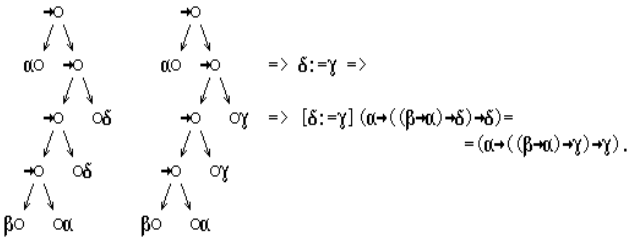

Рис. 4. Унифицируемые типы, представленные в виде деревьев

Унифицирующая подстановка [δ:=γ] найдена, т. е. TCP решена.

2. *Решение задачи TSP*.

2.1. *В стиле Чёрча* восстановление типа по заданному λ-терму осуществляется путем построения:

(1) прямого вывода (*сверху вниз*), при котором заданный λ-терм выстраивается из типизированных переменных (по аналогии с действиями решения задачи TCP);

(2) обратного вывода (*снизу вверх*), который начинается с ввода обозначения типа λ-терма некоторой буквой. В зависимости от вида λ-терма возможна конкретизация типа (например, известно, что терм, начинающийся с абстракции, должен иметь тип «со стрелкой»), учитывая то, что типизированные переменные представлены с типами (рис. 5):

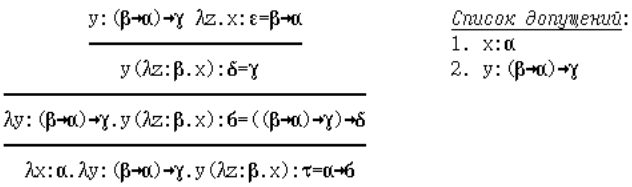

Рис. 5. Дерево вывода терма (в стиле Чёрча) с конструированием типа

Далее строить вывод не имеет смысла, поскольку все введенные буквы типа конкретизированы типами типизированных переменных заданного λ-терма.

В результате получится простая система равенств типов, решение которой по алгоритму Робинсона представит тип λ-терма (рис. 6):

$$\begin{cases}\delta=\gamma\\ б=((\beta\to\alpha)\to\gamma)\to\delta\\ \tau=\alpha\to б\end{cases} \Rightarrow \begin{cases} б=((\beta\to\alpha)\to\gamma)\to\gamma\\ \tau=\alpha\to б\end{cases} \Rightarrow \tau=\alpha\to((\beta\to\alpha)\to\gamma)\to\gamma.$$

Рис. 6. Система равенств типов

Полученный тип τ проверим с помощью интерпретатора GHC (версия 6.10.3), указав соответствующую теорему в файле th1.hs:

```
-- λx:α.λy:(β→α)→γ.y(λz:β.x):τ
-----------------------------------------------

{-# LANGUAGE RankNTypes,ScopedTypeVariables #-}
v = \(x::forall a. a)
     (y::forall a b g. (b -> a) -> g)
   -> y (\(z::forall b. b) -> x) :: a -> ((b -> a) -> g) -> g
```

Интерпретатор установит существование вывода:

```
> ghci.hs th1.hs
*Main> :t v

v :: (forall a1. a1) -> (forall a1 b1 g1. (b1 -> a1) -> g1)
                     -> a -> ((b -> a) -> g) -> g
```

2.2. *В стиле Карри* восстановление типа по заданному λ-терму осуществляется путем построения:

(1) обратного вывода (*снизу вверх*) по аналогии с тем, как это делалось при рассмотрении данной задачи в стиле Чёрча (рис. 7). Однако в силу отсутствия в явном виде типов типизированных переменных λ-терма получаемая система уравнений типов может быть более сложной:

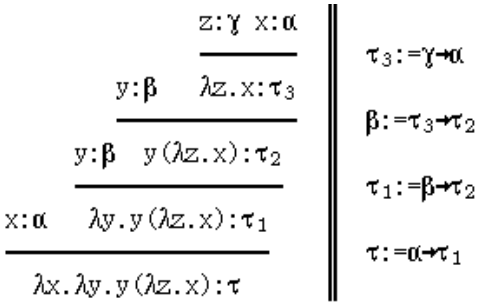

Рис. 7. Дерево вывода терма (в стиле Карри) с конструированием типа

Приступим к *формальной унификации* системы уравнений вида с помощью алгоритма Робинсона (рис. 8):

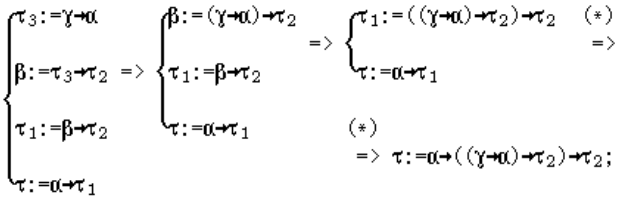

Рис. 8. Решение системы уравнений типов

(2) прямого вывода (*сверху вниз*), исходя из предположения о типах типизированных переменных (пусть x:α, y:β, z:γ, тогда дерево вывода будет таким же, как при построении обратного вывода).

Для того чтобы убедиться в правильности «ручного» вывода типа, унифицируем его с типом, выводимым GHC:

```
> ghci.exe
Prelude> :t (\x y -> y (\z -> x))
(\x y -> y (\z -> x)) :: t -> ((t1 -> t) -> t2) -> t2
```

В результате применения алгоритма унификации Робинсона к типам

$\alpha\to((\gamma\to\alpha)\to\tau_2)\to\tau_2,\ t\to((t1\to t)\to t2)\to t2$

получится унифицирующая подстановка

$[\tau_2:=t2]\circ[\alpha:=t]\circ[\gamma:=t1]$.

Осуществим проверку результатов унификации:

$[\tau_2:=t2]\circ[\alpha:=t]\circ[\gamma:=t1]\,(\alpha\to((\gamma\to\alpha)\to\tau_2)\to\tau_2)=t\to((t1\to t)\to t2)\to t2$

3. *Решение задачи TIP*.

Сформулируем правила вывода для решения задачи TIP *в стиле фигур натурального вывода*. При этом нумерация правил соответствует правилам типизации для $\lambda_{\to}$ *в стиле Чёрча* и *в стиле Карри*.

А. Правило, используемое при построении вывода в направлении *снизу вверх*:

в стиле Чёрча	*в стиле Карри*
$(3_1')\ \dfrac{\begin{matrix}[x:\sigma]\\ M_1:\tau\end{matrix}}{M:\sigma\to\tau},\ M=\lambda x:\sigma.M_1.$	$(3_2')\ \dfrac{\begin{matrix}[x:\sigma]\\ M_1:\tau\end{matrix}}{M:\sigma\to\tau},\ M=\lambda x.M_1.$

Б. Правило, используемое при построении вывода в направлении сверху вниз (в *стиле Чёрча* и в стиле Карри):

$$(2')\ \frac{M:\sigma\to\tau;\ N:\sigma}{MN:\tau}.$$

Восстановление λ-терма по заданному типу осуществляется путем построения вначале обратного вывода в направлении *снизу вверх* (при котором выделяются подтермы данного терма), затем – прямого вывода в направлении *сверху вниз* (при котором подтермы собираются в единый λ-терм с помощью операции «*подстановка*») (рис. 9).

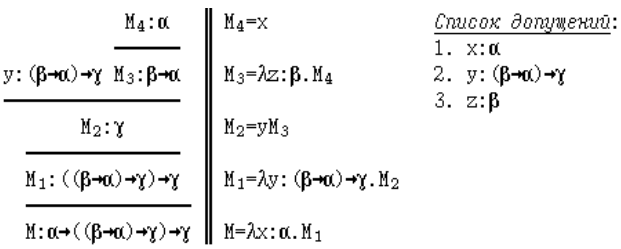

Рис. 9. Дерево вывода типа с конструированием терма (в стиле Чёрча)

Осталось выполнить подстановку термов, двигаясь в направлении сверху вниз по дереву вывода

$M=\lambda x:\alpha.\lambda y:(\beta\to\alpha)\to\gamma.y(\lambda z:\beta.x)$ (*запись в стиле Чёрча*);

$M=\lambda x.\lambda y.y(\lambda z.x)$ (*запись в стиле Карри*).

Проверка результатов осуществляется с помощью интерпретатора GHC аналогично тому, как было указано в разделе решения задачи TSP.

Напомним, что по изоморфизму Карри-Говарда тип λ-терма является доказываемой формулой в системе *Prop*, а λ-терм – доказательством. Легко видеть, что если удастся свести заданный тип к общезначимой формуле с помощью эквивалентных преобразований, то указанный тип будет обитаемым.

Описанные методы решения задач TCP, TSP и TIP легко превратить в соответствующие алгоритмы (известно [2], что все указанные задачи *разрешимы* в системе $\lambda_{\to}$).

Например, приведем алгоритмы решения задачи TSP:

(1) *алгоритм построения ограничений* (системы равенства типов) при выводе λ-терма в направлении снизу вверх [2, с. 71] (рис. 10):

$$\begin{cases} E(\Gamma, x, \delta) \stackrel{\text{def}}{=} \{x{:}\Gamma(x)\}; \\ E(\Gamma, MN, \delta) \stackrel{\text{def}}{=} E(\Gamma, M, \alpha{\to}\delta) \cup E(\Gamma, N, \alpha); \\ E(\Gamma, \lambda x.M, \delta) \stackrel{\text{def}}{=} E(\Gamma \cup \{x{:}\alpha\}, M, \beta) \cup \{\delta{:=}\alpha{\to}\beta\}; \end{cases}$$

Рис. 10. Формальная запись алгоритма построения ограничений

(2) *алгоритм Хиндли–Милнера* (двухэтапный) [3, с. 98–99]: первый этап состоит в построении системы уравнений на типах, а второй – в решении системы уравнений на типах с помощью алгоритма унификации;

(3) *алгоритм Хиндли–Милнера с немедленным разрешением* (по [3, с. 99–100]): уравнения на типах решаются по мере добавления в систему (S – главный унификатор типов, T, T' – подстановки, $\mathfrak{U}$ – функция унификации типов):

$$(1)\ \Gamma \cup \{x{:}\delta\} \vdash x{:}\delta, \varnothing;$$

$$(2)\ \frac{T(\Gamma) \vdash M{:}\varphi, T' \quad \Gamma \vdash N{:}\delta, T}{\Gamma \vdash MN{:}S(\tau), S \circ T' \circ T}, \text{ если } S = \mathfrak{U}(\varphi, T'(\delta) \to \tau);$$

$$(3)\ \frac{\Gamma \cup \{x{:}\delta\} \vdash M{:}\tau, T}{\Gamma \vdash \lambda x.M{:}T(\delta) \to \tau, T}.$$

В заключение выделим основные понятия предшествующего учебного материала, задействованные в изучаемом разделе.

Семиотика. *Знать* понятия: формальный язык, формальная система, интерпретация, синтаксис и семантика формальных языков.

Дискретная математика:

(1) *слова в алфавите*. *Знать* понятия: алфавит, слово в алфавите, операция и результат приписывания буквы (слова) к слову, подслово, вхождение (буквы) подслова в слово, операция и результат замены подслова исходного слова на слово, операция и результат подстановки слова вместо буквы в исходном слове. *Уметь*: осуществлять операции замены и подстановки;

(2) *формальные грамматики Хомского*. *Знать* понятия: формальная, порождающая грамматика; отношение непосредственной выводимости; вывод цепочки; выводимая цепочка; терминальная цепочка, порождаемая грамматикой; язык, порождаемый формальной грамматикой; *Уметь*: выводить терминальную цепочку, порождаемую грамматикой.

Комбинаторная логика: *язык в алфавите бестипового λ-исчисления*.

Знать: алфавит бестипового λ-исчисления; понятия: λ-терм, λ-формула бестипового λ-исчисления, λ-подтерм. *Уметь*: определять, является ли слово λ-термом, λ-формулой бестипового λ-исчисления; опускать и восстанавливать скобки в записях λ-термов.

Математическая логика:

(1) *система минимальной логики нулевого порядка* (Prop). *Знать*: подмножества формул языка нулевого порядка, содержащие только логическую связку "импликация"; подмножества правил натурального вывода (*modus ponens*, *дедукция*); подмножества множества аксиом и непрямых правил вывода гильбертовского исчисления нулевого порядка; понятия: линейное и древесное доказательство формулы, линейный и древесный вывод формулы. *Уметь*: устанавливать существование доказательств и выводов формул с помощью непрямых допустимых правил вывода в исчислении *Prop*;

(2) *генценовское исчисление натурального вывода*. *Знать*: аксиомы, фигуры заключения, правила заключения, правила вывода. *Уметь*: устанавливать существование доказательств (выводов) формул с помощью фигур заключения.

Логическое программирование: *алгоритмы унификации термов*.

Знать понятия: конкретизация, унификация; подстановка, композиция подстановок; унифицируемые термы, наиболее общий унификатор термов; рекурсивный и итеративный алгоритм Робинсона. *Уметь*: находить наиболее общий унификатор термов; применять композицию подстановок к терму.

Содержание предлагаемой нами дисциплины «Теоретическое программирование» может быть поставлено в соответствие с образовательной программой ФГОС ВО по направлению подготовки 230400 "Информационные системы и технологии" (квалификация (степень) «бакалавр»), так как оно ориентировано на формирование следующих компетенций:

(1) понимание социальной значимости своей будущей профессии, обладание высокой мотивацией к выполнению профессиональной деятельности (ОК-3).

Оценивается в процессе проведения занятий в интерактивных формах;

(2) владение широкой общей подготовкой (базовыми знаниями) для решения практических задач в области информационных систем и технологий (ОК-6);

(3) способность разрабатывать средства реализации информационных технологий (методические, информационные, математические, алгоритмические, технические и программные) (ПК-12);

(4) способность к инсталляции, отладке программных настройке технических средств для ввода информационных систем в опытную эксплуатацию (ПК-29).

Приведенные выше компетенции (2) – (4) оцениваются через решение задач, сопровождающих тексты лабораторных работ и упражнений по курсу "Теоретическое программирование" и направлен-

ных на выработку умений. Также имеет место оценка вариативной составляющей самостоятельной работы, которая подразумевает написание программ, реализующих алгоритмы, рассматриваемые в курсе.

В настоящее время на третьем курсе бакалавриата по направлению «Информационные системы и технологии» (институт компьютерных наук и технологического образования) проводится двухсеместровый курс «Прикладное программирование на языках высокого уровня». Первая часть курса посвящена бестиповому лямбда-исчислению, вторая часть курса посвящена использованию типизированного лямбда-исчисления для изучения систем типизации: λ->, λ2, λω, λP, λω, λPω, λP2, λPω (λ-куб Барендрегта).

Статья носит теоретический характер и посвящена структуризации содержания обучения некоторым аспектам теоретического программирования, поэтому не упоминаются методы и формы обучения и, следовательно, не исследуется вопрос эффективности предложенной методики.

Проведен логико-семиотический анализ содержания раздела «Мономорфная система типов $\lambda_{\to}$» в учебном курсе «Теоретическое программирование», в результате которого определены компетенции, которые представим в виде следующих типов задач.

Под компетенцией понимается способность специалиста воспользоваться для решения задач трансформированием декларативных знаний в «алгоритмические» умения. Такой взгляд сформировало следующее определение: профессиональная компетентность (по [12, с. 8]) – это интегральная характеристика, определяющая способность специалиста решать профессиональные проблемы и типичные профессиональные задачи, возникающие в реальных ситуациях профессиональной деятельности, с использованием знаний, профессионального и жизненного опыта, ценностей и наклонностей.

1. *Задачи системы* $\lambda_{\to}$: проверка типа λ-терма в стиле Чёрча и в стиле Карри (TCP); синтез типа по λ-терму в стиле Чёрча и в стиле Карри (TSP); установка обитаемости типа λ-терма в стиле Чёрча и в стиле Карри (TIP).

2. *Задачи на программирование*:

(1) работа с программной моделью вывода типов (в системе $\lambda_{\to}$), реализованной на языке программирования Haskell: вывод типовой схемы заданного λ-терма и проверка полученного результата с помощью интерпретатора языка Haskell; восстановление λ-терма по заданному дереву вывода типа (результат работы программы с конкретными λ-термом); восстановление типов некоторых подтермов дерева вывода типа λ-терма; определение главного типа λ-терма по заданному списку типовых схем подтермов λ-терма (получаемого с помощью программы), предварительно решив систему равенств вручную;

(2) работа с интерпретатором HUGS или GHCi: вывод типа заданного λ-терма; работа с интерпретатором GHCi-6.10.3, в который встроен механизм проверки типа для заданного λ-терма;

(3) реализация алгоритмов: стирание типа λ-терма; построение ограничений при выводе λ-терма в направлении снизу вверх; Хиндли–Милнера (двухэтапного); Хиндли–Милнера (с немедленным разрешением).

Список литературы

1. Пирс Б. Типы в языках программирования: пер. с англ. М.: Лямбда пресс: Добросвет, 2012. 656 с.
2. Barendregt H. Lambda calculi with types // Handbook of logic in computer science. Oxford University Press, vol. 2, 1993.
3. Довек Ж., Леви Ж.-Ж. Введение в теорию языков программирования. М.: ДМК Пресс, 2013. 134 с.
4. Митчелл Дж. Основания языков программирования: пер. с англ. М.; Ижевск: НИЦ «Регулярная и хаотическая динамика», 2010. 720 с.
5. Вековцева Т. А. Система содействия профессиональному саморазвитию преподавателя вуза // Научно-педагогическое обозрение (Pedagogical Review). 2014. Вып. 2 (4). С. 68–72.
6. Артеменко Н. А., Белогуров С. В. Об особенностях организации процесса формирования информационной компетентности будущего специалиста // Научно-педагогическое обозрение (Pedagogical Review). 2014. Вып. 1 (3). С. 13–18.
7. Лаптев В. В., Рыжова Н. И., Швецкий М. В. Методическая теория обучения информатике. Аспекты фундаментальной подготовки. СПб.: Изд-во СПб. ун-та, 2003. 352 с.
8. Френкель А., Бар-Хиллел И. Основания теории множеств: пер. с англ. М.: Мир, 1966. 555 с.
9. Барендрегт Х. Ламбда-исчисление. Его синтаксис и семантика: пер. с англ. М.: Мир, 1985. 606 с.
10. Хиндли Дж. Р. Комбинаторы и лямбда-исчисление. Краткий обзор // Математическая логика в программировании: сб. ст. 1980–1988 гг.: пер. с англ. М.: Мир, 1991. С. 119–140.
11. Генцен Г. Исследования логических выводов // Математическая теория логического вывода: пер. с нем. М.: Наука, 1967. С. 9–76.
12. Компетентностный подход в педагогическом образовании / под ред. В. А. Козырева, Н. Ф. Радионовой, А. П. Тряпицыной. СПб.: РГПУ им. А. И. Герцена, 2005. 392 с.

Кудрявцева И. А., кандидат педагогических наук, доцент.
Институт компьютерных наук и технологического образования РГПУ им. А. И. Герцена.
Наб. р. Мойки, 48, корп. 2, Санкт-Петербург, Россия, 191186.
E-mail: Yarina22@gmail.com.

Материал поступил в редакцию 21.03.2016.

I. A. Kudryavtseva

LOGICAL AND SEMIOTIC ANALYSIS OF THE CONTENT OF STUDYING THE MONOMORPHIC TYPE SYSTEM

The author offers the direction of development of content of training in theoretical programming, which is the science of mathematical models of the concept of "program" and includes the following topics: lambda calculus, type theory, combinators theory, theory of categories. The author carried out a logical and semiotic analysis of the one of the sections of studying theoretical programming "The monomorphic type system λ→" (showing dependence of terms on terms). The main emphasis is placed on the methods of solution of the main types of tasks of system with use of the GHC interpreter, containing tools on establishment of existence of an output of the typified λ-term and type inference of a λ-term. The competences acquired by trainees following the results of study of monomorphic type system are as a result formulated.

Key words: *theoretical programming, λ-cube, monomorphic type system, logical and semiotic analysis of the content, Curry-Howard isomorphism, minimum propositional logic (Prop).*

References

1. Pierce B. C. *Types and programming languages.* The MIT Press, Cambridge, Massachusetts, London, England, 2002. 645 p. (Russ. ed.: Pirs B. *Tipy v yazykakh programmirovaniya: per. s angl.* Moscow, Lyambda press: Dobrosvet Publ., 2012. 656 p.).
2. Barendregt H. Lambda calculi with types. *Handbook of logic in computer science*. Oxford University Press, vol. 2, 1993.
3. Dowek G., Levy J.-J. *Introduction to the Theory of Programming Languages.* Springer-Verlag London Limited, 2011. 108 p. (Russ. ed.: Dovek Zh., Levi Zh.-Zh. *Vvedeniye v teoriyu yazykov programmirovaniya: per. s angl.* Moscow, DMK Press, 2013. 134 p.).
4. Mitchell John C. *Foundations for Programming Languages*. The MIT Press, Cambridge, Massachusetts, London, England, 1996, 854 p. (Russ. ed.: Mitchell Dzh. *Osnovaniya yazykov programmirovaniya.* Moscow, Izhevsk, NIC "Regulyarnaya i khaoticheskaya dinamika" Publ., 2010. 720 p.).
5. Vekovtseva T. A. Sistema sodeystviya professoinal'nomu samorazvitiyu prepodavatelya vuza [The scheme of assistance in professional self-development of university teachers]. *Nauchno-pedagogicheskoye obozreniye – Pedagogical Review*, 2014, vol. 2 (4), pp. 68–72 (in Russian).
6. Artemenko N. A., Belogurov S. V. Ob osobennostyakh organizatsii protsessa formirovaniya informatsionnoy kompetentnosti budushchego spetsialista [Some features of the organization process of formation of information competence of the future expert]. *Nauchno-pedagogicheskoye obozreniye – Pedagogical Review,* 2014, vol. 1 (3), pp. 13–18 (in Russian).
7. Laptev V. V., Ryzhova N. I., Shvetskiy M. V. *Metodicheskaya teoriya obucheniya informatike. Aspekty fundamental'noy podgotovki* [Methodical theory of learning to computer science. Aspects of fundamental preparation]. St. Petersburg, S.-Peterb. un-t Publ., 2003. 352 p. (in Russian).
8. Fraenkel A. A., Bar-Hillel Y. *Foundations of set theoty.* North-holland publishing company, Amsterdam, 1958. 412 p. (Russ. ed.: Frenkel' A., Bar-Hillel I. *Osnovaniya teorii mnozhestv: per. s angl*. Moscow, Mir Publ., 1966. 555 p.)
9. Barendregt H. *The lambda calculus, its syntax and semantics (studies in logic and the foundations of mathematics)*. North-Holland, 1984. 654 p. (Russ. ed.: Barendregt H. *Lambda-ischisleniye. Ego sintaksis i semantika: per. s angl.* Moscow, Mir Publ., 1985. 606 p.).
10. Hindley J. R. Combinators and lambda-calculus, a short outline. *Combinators and functional programming languages, Thirteenth spring school of the LITP Val d'Ajol,* France, May 6–10, 1985 Proceedings, 1986. Pp. 104–122. DOI: 10.1007/3-540-17184-3_42. (Rus. ed.: Khindli Dzh.R. Kombinatory i lyambda-ischisleniye. Kratkiy obzor. *Matematicheskaya logika v programmirovanii: sb. statey 1980–1988 gg.: per. s angl.* Moscow, Mir Publ., 1991. Pp. 119–140).
11. Gentzen G. Untersuchungen uber das logische Schliessen. *Mathematische Zeitschriff*. 1935. (Rus. ed.: Gentsen G. Issledovaniya logicheskikh vyvodov. *Matematicheskaya teoriya logicheskogo vyvoda: per. s nem.* Moscow, Nauka Publ., 1967. Pp. 9–76).
12. *Kompetentnostnyy podkhod v pedagogicheskom obrazovanii* [Competence-based approach in pedagogical education: Collective monograph]. Under the editorship of V. A. Kozyrev, N. F. Radionova, A. P. Tryapichyna. St. Petersburg, Herzen State Pedagogical University Publ., 2005. 392 p. (in Russian).

Kudryavtseva I. A.
Herzen State Pedagogical University of Russia.
Nab. r. Moyka, 48, korp. 2, St. Petersburg, Russia, 191186.
E-mail: Yarina22@gmail.com.