И. А. Кудрявцева

ПОЛИМОРФНАЯ СИСТЕМА ТИПОВ Λ2 В СОДЕРЖАНИИ ОБУЧЕНИЯ ТЕОРЕТИЧЕСКОМУ ПРОГРАММИРОВАНИЮ

Автором проведён логико-семиотический анализ содержания одного из разделов обучения теоретическому программированию «полиморфная система типов λ2» (демонстрирующая зависимость термов от типов). Построена сводная таблица о разрешимости (наличия алгоритма решения) классических задач системы λ2 в слабом и сильном полиморфизме произвольного ранга, а также в слабом полиморфизме с let-правилом в стиле Карри и в стиле Чёрча. В результате сформулированы компетенции, приобретаемые обучаемыми по итогам изучения полиморфной системы типов λ2.

Ключевые слова: теоретическое программирование, λ-куб, полиморфная система типов, изоморфизм Карри-Говарда, интуиционистская пропозициональная логика второго порядка (Prop2).

I. Kudryavtseva

POLYMORPHIC TYPE SYSTEM Λ2 IN THE CONTENT OF STUDYING THEORETICAL PROGRAMMING

The author carried out a logical and semiotic analysis of the of one of the sections of studying theoretical programming «The polymorphic type system λ2» (showing dependence of terms from types). Built a summary table on the solvability of (solution algorithm) classical tasks of the λ2 system in strong and weak arbitrary rank polymorphism, as well as in weak polymorphism with let-rule-style in style of the Curry and in Church style. The competences acquired by trainees following the results of study of polymorphic type system of λ2 are as a result formulated.

Keywords: theoretical programming, λ-cube, polymorphic type system, Curry-Howard isomorphism, intuitionistic propositional logic of the second order (Prop 2).

Раскрытие содержания полиморфной системы типов λ2 в обучении теоретическому программированию осуществим через логико-семиотический анализ.

Теоретическое программирование, как раздел математической науки, имеет в обозрении модели программ, представляющие концепции для разработки языков программирования. Отметим теории, в которых обнаруживаются такие модели программ и которые послужили основой для развития современных функциональных языков программирования: λ-исчисление, теория комбинаторов, теория типов, теория категорий, семантическая теория языков программирования.

Введение понятия типа в λ-исчисление и языки программирования основано (по [11]):

1) с точки зрения логики на преодолении парадокса Рассела, препятствующего попыткам построить непротиворечивое расширение лямбда-исчисления теорией множеств. Связано это с циклической системой приёма вычисления — применения функции к самой себе;

2) с точки зрения программирования на применении типизации в языках программирования, в том числе отличных от функциональных. Связано это с эффективностью порождаемого компилятором кода и рациональностью распределения памяти

при наличии информации о допустимых способах использования переменной;

3) с точки зрения теоретического программирования на выявлении инструмента типизации как:

− ограниченной статической проверки корректности программ (выявление ошибок, концептуальных просчётов на этапе компиляции программы);

− документации текста программы;

− улучшения модульности программ и скрытия информации при помощи определения различных структур данных, которые явно разделены на интерфейс и подробности реализации.

Несмотря на то, что типизация накладывает определённые ограничения на стиль программирования (что выражается более детальным описанием вычислительного процесса), развитие типизированных языков программирования прогрессирует.

Современные функциональные языки программирования имеют в основе разнообразные системы типов, полученные из теории типизированных λ-исчислений. Возможные зависимости между типами и выражениями формального языка λ-термов отображены в системах, которые Х. Барендрегт (1991) классифицировал и представил структурой λ-*куба*. Рассмотрим одну из систем λ-куба — типизированное полиморфное λ-исчисление (λ2, система F), на которой основаны такие языки программирования, как Haskell ML.

Структура логико-семиотического анализа позволит отобразить в содержании (по [3, с. 223]) следующие компоненты:

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

2) *формально-логический* (с помощью явного выделения формальных систем, формальных языков, формальных и содержательных аксиоматических теорий);

3) *алгоритмический* (с помощью явного выделения вычислительных моделей, алгоритмов и их реализации с помощью вычислительных систем).

I. *Семиотический компонент содержания*.

Предметный язык: язык типизации λ-термов.

Метаязык: язык канторовской (наивной) теории множеств, язык интуиционистской пропозициональной логики второго порядка (*Prop*2) и язык типизированного λ-исчисления $\lambda_{\rightarrow}$.

II. *Формально-логический компонент содержания*.

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

Формальная система включает описание двух компонентов (по [6, 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*):

1) V={x,y,z,... } — *множество предметных* (*термовых*) *переменных*;

2) V_{type} = {α,β,γ,...,χ,ψ,ω} — *множество типовых переменных* (*неинтерпретируемые базовые типы*);

3) →, ∀ — *синтаксические конструкторы типа*;

4) = — *двухместный предикатный символ*;

5) λ и Λ — *логические символы*;

6) (,),. — *вспомогательные символы*;

б) *класс правильных слов в алфавите*.

Множество типов определяется следующей формальной грамматикой (по [10]):

- *в сильном полиморфизме*

$\mathbf{T} ::= \forall_{type} \mid \mathbf{T} \rightarrow \mathbf{T} \mid \forall \forall_{type}.\mathbf{T}$

- *в слабом полиморфизме* (*в стиле ML*)

$\mathbf{T}_w ::= \mathbf{T}_{->} \mid \forall \forall_{type}.\mathbf{T}_w,$

$\mathbf{T}_{->} ::= \forall \mid \mathbf{T}_{->} \rightarrow \mathbf{T}_{->}$ (тип для мономорфной системы $\lambda_{->}$)

Область действия знака «.» начинается с места его расположения, и простирается вправо от этого знака до окончания записи

типа. Например, записи следующего вида являются синонимами:

$\forall\alpha.\alpha\to\alpha$, $(\forall\alpha.\alpha\to\alpha)$, $\forall\alpha.(\alpha\to\alpha)$.

Предтермы (*псевдотермы*) имеют вид (по [10]):

- *в стиле Чёрча* (Λ_T):

$\Lambda_T ::= V \mid \Lambda_T \Lambda_T \mid \lambda V^T.\Lambda_T \mid \Lambda_T T \mid \Lambda V_{type}.\Lambda_T$,

- *в стиле Карри* (Λ):

$\Lambda ::= V \mid \Lambda\Lambda \mid \lambda V.\Lambda$

V^T — множество термовых переменных, каждый элемент которого аннотирован типом.

В *системе А. Чёрча* тип предметной переменной указывается в её аннотации; при этом разрешено изменять тип, указанный в аннотации, что осуществляется с помощью указания символа «Λ» (читается: «*большая лямбда*») в записи λ-терма и использования следующего *расширения* β-*редукции* (M — λ-терм, α, β — типовые переменные):

$(\Lambda\alpha.M)\beta \to_{\beta} M[\alpha:=\beta]$.

Например: $(\Lambda\alpha.\lambda x{:}\alpha.x)\beta \to_{\beta} \lambda x{:}\beta.x$

Самоприменением называется λ-терм вида λf. ff.

let-терм (let-конструкция) — это запись вида $\mathbf{let}\ x=M\ \mathbf{in}\ N ::= (\lambda x.N)M$.

Утверждение о типизации в λ2 имеет вид

- *в стиле Чёрча*:

(а) $M:\tau,\ M\in\Lambda_T,\ \tau\in T$;

(б) $M:\tau,\ M\in\Lambda_T,\ \tau\in T_w$.

- *в стиле Карри:*

(а) $M:\tau,\ M\in\Lambda,\ \tau\in T$;

(б) $M:\tau,\ M\in\Lambda,\ \tau\in T_w$.

Тип τ называют *предикатом*, а λ-терм — *субъектом утверждения*;

в) *класс формул*.

Формулами полиморфного типизированного λ-*исчисления* называются слова вида M = N, где M, N — λ-термы одного типа;

г) *свойства типизированных* λ-*термов*.

Понятия «*свободная переменная*» и «*связанная переменная*» и «*замкнутый* λ-*терм*» для системы λ2 вводятся аналогично соответствующим понятиям для бестипового λ-исчисления; понятия «*свободная типовая переменная*» и «*связанная типовая переменная*» для системы λ2 вводятся аналогично понятиям для языка второго порядка; понятие «*нормальная форма*» аналогично определению в бестиповом λ-исчислении.

Длина типизированного λ-*терма* (по [4, с. 548]) — это количество (λ+***let***)-подтермов.

let-*высота* (*глубина*) (λ+***let***)-*терма M* (по [4, с. 548]) — это количество его λ-подтермов наибольшей длины, не содержащих ***let***-выражений.

λ-*терм N является* ***let***-*нормальной формой* (λ+***let***)-*терма M* (по [4, с. 548]), если N является λ-термом (не содержащим ***let***), полученным из M неоднократной ***let***-редукцией;

д) *операции 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, подстановка $(M)^x_N$ не определена.

Редукция на λ-*термах* M и N (в стиле Чёрча) определяется так:

$$\left(\lambda x^{\alpha}.M\right)N \to_{\beta} M[x:=N] = \left(M\right)^x_N.$$

Для типизированных λ-термов в стиле Чёрча определяется операция «*подстановка*» типов для возможного изменения типа, указанного в аннотации предметной переменной λ-терма.

(λ+***let***)-*терм M' let-редуцируем к* (λ+***let***)-*терму N'*, если можно получить N' из M' неоднократным применением ***let***-правила к подтермам и переименованием связанных переменных (α-конверсией).

let-правилом называется следующее правило:

$$\frac{\Gamma \vdash M:\delta \quad \Gamma \vdash N[x:=M]:\tau}{\Gamma \vdash (\mathbf{let}\ x=M\ \mathbf{in}\ N):\tau},\ \delta\in T_w,\ \tau\in T_{->}\ \text{для}\ \lambda 2(T_w).$$

Переменная x имеет полиморфный тип σ, то есть «скрыто» выполняется подстановка полиморфного типа.

Применение ***let***-правила — это способ фактически использовать полиморфные по σ редексы:

$$(\lambda x{:}\mathfrak{G}.N)M{:}\tau \to_{\beta} \left(N\right)^{x:\mathfrak{G}}_{M:\mathfrak{G}}$$

Стирающее отображение $|\circ|{:}\Lambda_T \to \Lambda$ (по [10]):

(1) $|x| \equiv x$;
(2) $|MN| \equiv |M||N|$;
(3) $|M\mathfrak{G}| \equiv |M|$;
(4) $|\lambda x^{\alpha}.M| \equiv \lambda x.|M|$;
(5) $|\Lambda\alpha.M| \equiv |M|$;

е) *основные теоремы и леммы*.

Из многочисленных лемм и теорем приведём основные.

Лемма (*о контекстах*) (по [8]):

1) (*Утончение*, англ. *thinning*) Расширение контекста не влияет на выводимость утверждения о типизации;

2) свободные переменные типизированного λ-терма должны присутствовать в контексте;

3) *сужение* контекста до множества свободных переменных λ-терма не влияет на выводимость утверждения о типизации.

Теорема (*о типизируемости подтерма*).

Если λ-терм имеет тип, то есть $\Gamma \vdash M{:}\mathfrak{G}$, то каждый его подтерм также имеет тип.

Теорема (*о единственности типов*) *не имеет места* для λ2 в стиле *Карри*.

Леммы (*о сохранении типов при подстановке термов*) [10]; (*о подстановке типа)* [8; 10].

Теорема (*о редукции субъекта*) [8].

Теорема (*о сильной нормализации*) [7].

Система λ2 является *SN-системой*.

Если $\Gamma \vdash M{:}\mathfrak{G}$, то M является сильно нормализуемым, то есть любая последовательность β-редукций приводит λ-терм к нормальной форме за конечное число шагов.

Теорема (*о типизации самоприменения*) [10]:

Для λ-терма λf. ff верны следующие утверждения:

$T_1 \equiv \forall\alpha.\alpha\to\alpha \equiv \forall\alpha(\alpha\to\alpha)$
$T_2 \equiv \forall\alpha.\alpha$ - пустой тип
$f{:}T_1 \vdash ff{:}T_1$ в $\lambda 2(\mathbf{T}_w)$;
$f{:}T_2 \vdash ff{:}T_2$ в $\lambda 2(\mathbf{T}_w)$;
$\vdash \lambda f.ff{:}T_1\to T_1$ в $\lambda 2(\mathbf{T})$;
$\vdash \lambda f.ff{:}T_2\to T_2$ в $\lambda 2(\mathbf{T})$.

Теорема (по [4, с. 548]).

Пусть M — произвольный бестиповый (λ+***let***)-терм.

Существует λ-терм N (свободный от let), в котором каждая максимальная последовательность ***let***-редукций из M завершается в N.

Таким образом, можно утверждать, что не существует бесконечной последовательности ***let***-редукций. ***let***-нормальная форма существует и единственна.

2. *Исчисления*.

Типизированные исчисления в стиле Чёрча и в стиле Карри отличаются лишь записью правила обработки абстракции, в остальном они схожи и представимы в терминах моносукцедентного секвенциального исчисления.

Приведём *аксиомы* и *правила вывода* типизированных λ-термов, предварительно введя понятия «*контекст*», «*тип, выводимый из контекста*» и правила построения допустимого контекста.

Контекстом называется следующий кортеж объявлений типов:

$$\Gamma \equiv \left(\underbrace{\alpha_1{:}*,\alpha_2{:}*,\ldots,\alpha_p{:}*}, \underbrace{x_1{:}t_1,x_2{:}t_2,\ldots,x_r{:}t_r}\right),$$

где:

1) α_1, α_2,..., α_p — свободные типовые переменные (с типом $*$);

2) $t_1,t_2,\ldots,t_r$ — типы, которые должны содержать вхождения хотя бы одного из типов $\alpha_1, \alpha_2,\ldots, \alpha_p$;

3) $x_1{:}t_1$, $x_2{:}t_2,\ldots,x_r{:}t_r$ — типизированные переменные.

Областью определения контекста Γ называется множество

$$dom(\Gamma) \equiv \left\{\alpha_1,\alpha_2,\ldots,\alpha_p,x_1,x_2,\ldots,x_r\right\}.$$

Тип σ *выводим из контекста* Γ (по [10]): $\Gamma \vdash \sigma : *$,

если все свободные переменные σ принадлежат Γ.

Правила построения допустимого контекста (по [10]):

1) *пустой контекст* (контекст, не содержащий компонентов) является допустимым;

2) если контекст Γ является допустимым, то допустим контекст $\Gamma \cup \{\alpha : *\}$, где $\alpha \notin dom(\Gamma)$;

3) если тип σ выводим из Γ (то есть все свободные переменные σ входят в Γ), то при $x \notin dom(\Gamma)$ допустим контекст $\Gamma \cup \{x : \sigma\}$.

Для постулирования некоторых правил вывода типизированных λ-термов необходимо устанавливать выводы типов из контекста. В этом процессе помогут правила конструирования типов, выводимых из контекста.

Правила конструирования типов таковы (по [10]):

1) если тип $\alpha : * \in \Gamma$, то тип $\alpha : *$ выводим из контекста Γ;

2) если типы $\sigma : *$ и $\tau : *$ выводимы из контекста Γ, то тип $\sigma \to \tau : *$ выводим из Γ;

3) если тип $\sigma : *$ выводим из контекста Γ, $\alpha : *$, то тип $\forall \alpha.\ \sigma : *$ выводим из Γ.

Утверждением, выводимым в контексте Γ, называется утверждение $M : \tau$, вывод которого выполняется по *правилам типизации системы* λ2.

По *изоморфизму Карри-Говарда:*

1) формулы в логической системе *Prop*2 — это типы в системе λ2;

2) доказательства в логической системе *Prop*2 — это термы в системе λ2.

Таблица 1

Правила типизации системы λ2

	Система λ2 *(гильбертовский вариант)*		*Система Prop2 (гильбертовский вариант)*
	схема аксиом:		
(1)	$\Gamma, x : \sigma \vdash x : \sigma$;		$\Gamma \cup \{\sigma\} \vdash \sigma$;
	правила вывода:		
(2)	$\dfrac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$;		$\dfrac{\Gamma \vdash \sigma \to \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau}$;
	в стиле Карри	*в стиле Чёрча*	
(3)	$\dfrac{\Gamma \cup \{x : \sigma\} \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}$,	$\dfrac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma.M : \sigma \to \tau}$,	$\dfrac{\Gamma \cup \{\sigma\} \vdash \tau}{\Gamma \vdash \sigma \to \tau}$;
	$\sigma \in T \to$ для $\lambda 2(T_w)$, $\sigma \in T$ для $\lambda 2(T)$;		
(4)	$\dfrac{\Gamma, \alpha : * \vdash M : \sigma}{\Gamma \vdash M : \forall \alpha.\sigma}$;	$\dfrac{\Gamma, \alpha : * \vdash M : \sigma}{\Gamma \vdash \Lambda \alpha.M : \forall \alpha.\sigma}$;	$\dfrac{\Gamma \vdash \sigma}{\Gamma \vdash \forall \alpha.\sigma}$, $\alpha \notin FV(\Gamma)$;
(5)	$\dfrac{\Gamma \vdash \tau : * \quad \Gamma \vdash M : \forall \alpha.\sigma}{\Gamma \vdash M : \sigma[\alpha := \tau]}$,	$\dfrac{\Gamma \vdash \tau : * \quad \Gamma \vdash M : \forall \alpha.\sigma}{\Gamma \vdash M\tau : \sigma[\alpha := \tau]}$,	$\dfrac{\Gamma \vdash \forall \alpha.\sigma}{\Gamma \vdash \left(\sigma\right)_\tau^\alpha}$.
	$\tau \in T \to$ для $\lambda 2(T_w)$, $\tau \in T$ для $\lambda 2(T)$.		

ж) *определение понятия* «*вывод*».

Понятие «*древесный вывод утверждения* $M{:}\tau$ *в контексте* Γ» определим подобно [2, с. 31]: *древесным выводом секвенции* $\Gamma \vdash M{:}\tau$ (*выводом секвенции в виде дерева* $\mathcal{D}$) называется *дерево секвенций* $\mathcal{D}$, всякая начальная секвенция которого является *доказуемой секвенцией*, заключительной секвенцией является секвенция $\Gamma \vdash M{:}\tau$, а переходы являются применениями правил типизации системы λ2.

Дерево секвенций определим индуктивно [2, с. 28]:

1) *деревом секвенций* является всякая секвенция;

2) если $\mathcal{D}_1,\ldots,\mathcal{D}_n$ — деревья секвенций, а Σ — секвенция, то

$$\frac{\mathcal{D}_1 \quad \mathcal{D}_2 \quad \ldots \quad \mathcal{D}_n}{\Sigma}$$

также является *деревом секвенций*.

Доказуемая секвенция (по: 2, с. 27–28]) — это секвенция, для которой существует последовательность секвенций, где каждая секвенция либо является аксиомой, либо получается из предшествующих секвенций при помощи правила типизации системы λ2.

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 (интуиционистской пропозициональной логики второго порядка), а λ-термы — как доказательства этих формул.

III. *Алгоритмический компонент содержания*.

Важнейшими задачами в системе λ2 являются:

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

Решение задачи TCP осуществляется путём построения дерева вывода заданного λ-терма с помощью правил типизации системы λ2;

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

Решение задачи TSP осуществляется путём построения дерева вывода заданного λ-терма снизу вверх (с изначальным обозначением типа какой-либо типовой переменной) с помощью правил типизации системы λ2. По мере выстраивания дерева вывода будет формироваться система равенств типов, которую затем можно решить с помощью алгоритма Робинсона [1, с. 98–99] и получить конкретизацию введённой типовой переменной;

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

Для решения задачи TIP сформулируем правила вывода *в стиле фигур натурального вывода*, опираясь на правила типизации системы λ2.

При этом нумерация правил соответствует правилам типизации для $\lambda2(T_w)$ и $\lambda2(T)$ *в стиле Карри* ($\sigma,\tau \in T_{\to}$).

A. Правила, используемые при построении вывода в направлении снизу вверх:

- *в стиле Чёрча*

$$(3')\ \frac{[x:\sigma]\quad M_1:\tau}{M:\sigma\to\tau},\ M=\lambda x{:}\sigma.M_1;$$

$$(4')\ \frac{[\alpha:*]\quad M_1:\sigma}{M:\forall\alpha.\sigma},\ M=\Lambda\alpha.M_1.$$

- *в стиле Карри*

$$(3')\ \frac{[x:\sigma]\quad M_1:\tau}{M:\sigma\to\tau},\ M=\lambda x.M_1;$$

$$(4')\ \frac{[\alpha:*]\quad M:\sigma}{M:\forall\alpha.\sigma}.$$

Б. Правила, используемые при построении вывода в направлении сверху вниз:

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

- *в стиле Чёрча*

$$(5')\ \frac{[\tau:*]\quad M:\forall\alpha.\sigma}{M\tau:\sigma[\alpha:=\tau]}.$$

- *в стиле Карри*

$$(5')\ \frac{[\tau:*]\quad M:\forall\alpha.\sigma}{M:\sigma[\alpha:=\tau]}.$$

Восстановление λ-терма по заданному типу осуществляется путём 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*»).

Для систем $\lambda2(T_w)$, $\lambda2(T(1))$ (*система типов ранга* 1), $\lambda2(T)$, $\lambda2(T(2))$ (*система типов ранга* 2) установлены следующие результаты о разрешимости приведённых выше задач — наличия алгоритмов решения (по [5; 8; 10; 12]) (табл. 2):

Заметим (по [13, с. 9]), что

1) неразрешимость TCP и TSP для λ2 в стиле Карри доказана в [14];

2) разрешимость TSP для $\lambda2(T_w)$ с ***let***-правилом в стиле Карри доказана в [9].

Для решения задачи TSP с ***let***-правилом в стиле Карри существует алгоритм Дамаса-Милнера [1, с. 104–105], в котором уравнения на типах (получаемых при построении дерева вывода λ-терма) решаются по мере добавления в систему.

Также отметим, что для некоторого класса нетипизируемых λ-термов — λ-термов *в слабой заголовочной нормальной форме* (начинающиеся с абстракции) с рекурсивным типом — нами определён метод решения задачи TSP в системе $\lambda2(T(k))$ (k=1,2,...) в стиле Карри, который будет представлен в соответствующей отдельной статье. Метод основывается на двухэтапном алгоритме Дамаса-Милнера (Хиндли-Милнера).

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.

Таблица 2

Разрешимость классических задач системы λ2

Задача / ***Система типов***	***TCP*** (M:σ?)	***TSP*** (M:?)	***TIP*** (?:σ)
$\lambda2(T_w)$ в стиле Карри	разрешима	разрешима	разрешима
$\lambda2(T_w)$ в стиле Чёрча	разрешима	разрешима	разрешима
$\lambda2(T(1))$	разрешима	разрешима	разрешима
$\lambda2(T_w)$ с **let**-правилом в стиле Карри	неразрешима	разрешима	неразрешима
$\lambda2(T_w)$ с **let**-правилом в стиле Чёрча	неразрешима	разрешима	неразрешима
$\lambda2(T)$ в стиле Карри	неразрешима	неразрешима	неразрешима
$\lambda2(T)$ в стиле Чёрча	разрешима	разрешима	неразрешима
$\lambda2(T(2))$		разрешима	
$\lambda2(T(k))$, k>2		неразрешима	

Дискретная математика:

1) *слова в алфавите*. *Знать* понятия: алфавит, буква, 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 операцию замены подслова исходного слова на другое слово; осуществлять операцию подстановки вместо буквы исходного слова на другое слово;

2) *формальные грамматики Хомского*. *Знать* понятия: формальная грамматика, 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:

1) *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; доказывать, что слово является λ-термом, λ-формулой бестипового λ-исчисления; опускать и восстанавливать скобки в записях λ-термов.

2) *язык типизированного λ-исчисления* $\lambda_{\rightarrow}$. *Знать*: алфавит типизированного λ-исчисления; понятия: тип, типизированный λ-терм, утверждение о типизации в $\lambda_{\rightarrow}$ в стиле Чёрча и в стиле Карри. *Уметь*: определять, является ли слово типизированным λ-термом; опускать и восстанавливать скобки в записях λ-термов и типов.

Математическая логика:

1) *синтаксис языка первого порядка*. *Знать*: язык в алфавите языка первого порядка; понятия «*терм*», «*формула*»; язык формальной системы первого порядка. *Уметь*: определять связанное и свободное вхождение переменной в формулу;

2) *подсистема гильбертовского исчисления второго порядка*, содержащая только символы «→» и «∀» (*Prop*2) — интуиционистская пропозициональная логика второго порядка. *Знать*: подмножества формул языка второго порядка, содержащие только логическую связку «→» (импликация) и квантор всеобщности «∀»; подмножества правил гильбертовского исчисления (удаление →, дедукция, правило универсального обобщения, правило универсальной конкретизации); подмножества правил натурального вывода (дедукция, modus ponens, введение ∀, удаление ∀). *Уметь*: доказывать формулы в исчислении *Prop*2;

3) *исчисление дедуктивных эквивалентностей*. *Знать* понятия: дедуктивно эквивалентные формулы, универсальная формула (∀-формула); доказуемые формулы гильбертовского исчисления, содержащие логическую связку «↔» (эквиваленция) и квантор всеобщности «∀». *Уметь*: приводить формулу к универсальной формуле;

Логическое программирование: *алгоритмы унификации термов*.

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

Выводы

Проведён логико-семиотический анализ содержания раздела «Полиморфная система типов λ2» в учебном курсе «Теоретическое программирование», в результате которого можно определить компетенции, которые представим в виде следующих типов задач (под *компетенциями* понимаются способы преобразования знаний в умения).

0. *Вспомогательные задачи*:

1) определение выводимости типа из контекста;

2) построение типа в заданном контексте;

3) доказательство выводимости типа из контекста;

4) приведение с помощью дедуктивных эквивалентностей формулы к универсальной формуле.

1. *Классические задачи системы* λ2:

Среди λ-термов в стиле Чёрча рассматриваются в том числе те, семантика которых определяет: логические связки и кванторы, булевы типы и логические операции, упорядоченные пары, нумералы, списки, примитивную рекурсию на нумералах.

2. *Задачи на программирование*:

1) Работа с программой M. Grabmueller (2006) [7], написанной на языке Haskell, в которой реализован алгоритм W, предложенный Р. Милнером (1978) (по некоторым источникам и Л. Дамасом), по выводу полиморфного типа в языке Haskell: запись λ-терма с помощью let-конструкции и определение его типа при запуске программы.

2) Работа с программной моделью метода решения системы равенства типов с рекурсивным типом, получаемых при построении дерева вывода λ-терма в процессе решения задачи TSP в системе λ2(T(k)) (k=1,2,...) в стиле Карри, реализованной нами на языке программирования Haskell:

а) получение типа заданного λ-терма в стиле Карри, предварительно вводя в программу систему равенства типов, составленную при «ручном» построении дерева вывода заданного λ-терма;

б) преобразование системы равенств типов, содержащих неоднозначное определение одного и того же типа (в том числе и рекурсивного) к виду, пригодному для решения с помощью алгоритма Робинсона [1, с. 98–99];

в) осуществление унификации двух типов с обработкой равенства типов, содержащего рекурсию;

3) Работа с интерпретатором GHCi-6.10.3, в который встроен механизм проверки типа для заданного λ-терма: запись в файл типизированного утверждения, например:

```
{-# LANGUAGE RankNTypes #-}
test :: (g -> b2) -> (forall b1.
((b1 -> d2) -> d2) -> g) -> b2
test = (\z x -> z (x (\y -> y x)))
```

и решение задачи TCP для терма, указанного в файле:

```
> ghci.exe Exampls.hs
*Main>
```

Загрузка главного модуля Main будет являться подтверждением того, что тип λ-терма соответствует.

4) Реализация алгоритма, определяющего длину типизированного λ-терма.

5) Реализация алгоритма, определяющего let-высота (глубина) (λ+let)-терма.

6) Реализация алгоритма стирания типа λ-терма.

7) Реализация двухэтапного алгоритма Дамаса-Милнера (Хиндли-Милнера) для решения задачи TSP в стиле Карри в системе λ2(T).

8) Реализация алгоритма Дамаса — Милнера с немедленным разрешением.

9) Реализация алгоритма W (по [5, с. 360]), который приспособлен для «чистой реконструкции типов», то есть присвоения типов совершенно нетипизированным термам в стиле ML (let-полиморфизм).

10) Компьютерная реализация экспоненциальной регрессии для длины λ-терма и, соответственно, его типа с возрастанием вложенных let-выражений при выполнении let-редукции. Проблема вывода «патологического» типа термов с вложенными let-выражениями была продемонстрирована независимо H. G. Mairson (1990) и группой исследователей A. J. Kfoury, J. Assaf, J. Tiuryn, P. Urzyczyn (1990).

СПИСОК ЛИТЕРАТУРЫ

1. *Довек Ж., Леви Ж.-Ж.* Введение в теорию языков программирования. М.: ДМК Пресс, 2013. 134 с.
2. *Ершов Ю. Л., Палютин Е. А.* Математическая логика. М.: Наука, 1979. 320 с.; 1987. 336 с.
3. *Лаптев В. В., Рыжова Н. И., Швецкий М. В.* Методическая теория обучения информатике. Аспекты фундаментальной подготовки. СПб.: Изд-во С.-Петерб. ун-та, 2003. 352 с.
4. *Митчелл Дж.* Основания языков программирования. М.; Ижевск: НИЦ «Регулярная и хаотическая динамика», 2010. 720 с.
5. *Пирс Б.* Типы в языках программирования. М.: Лямбда пресс; Добросвет, 2012. 656 с.
6. *Френкель А., Бар-Хиллел И.* Основания теории множеств. М.: Мир, 1966. 555 с.
7. *Algorithm W Step by Step*. [Электронный ресурс] // Home Comics Posts Publications Talks Misc About RSS. Электрон. дан. Режим доступа: http://catamorph.de/publications/2007-09-01-algorithm-w-step-by-step.html (дата обращения: 13.07.2015).
8. *Barendregt H.* Lambda calculi with types / Handbook of logic in computer science. Vol. 2. Oxford University Press, 1993.
9. *Damas L., Milner R.* Principal Type-schemes for Functional Programs // Principles of Programming Languages. ACM. 1982. P. 207–212.
10. *Geuvers H.* Introduction to Type Theory / Alfa Lernet summer school, Uruguay, 2008.
11. *Harrison J.* Introduction to Functional Programming. Cambridge University, 1996/1997.
12. *Kfoury A. J., Wells J. B.* Principality and decidable type inference for finite-rank intersection types // ACM Symposium on Principles of Programming Languages (POPL), 1999. P. 161–174.
13. *Severi P. G.* Normalization in Lambda Calculus and its relation to Type Inference // Thesis Technische Universiteit Eindhoven, 1996.
14. *Wells J. B.* Typeability and Type Checking in the Second Order λ-calculas are Equivalent Science // IEEE, 1994. P. 176–185.

REFERENCES

1. *Dovek Zh., Levi Zh.-Zh.* Vvedenie v teoriyu yazykov programmirovaniya. M.: DMK Press, 2013. 134 s.
2. *Ershov YU. L., Palyutin E. A.* Matematicheskaya logika. M.: Nauka, 1979. 320 s.; 1987. 336 s.
3. *Laptev V. V., Ryzhova N. I., SHveckij M. V.* Metodicheskaya teoriya obucheniya informatike. Aspekty fundamental'noj podgotovki. SPb.: Izd-vo S.-Peterb. un-ta, 2003. 352 s.
4. *Mitchell Dzh.* Osnovaniya yazykov programmirovaniya. M.; Izhevsk: NIC «Regulyarnaya i haoticheskaya dinamika», 2010. 720 s.
5. *Pirs B.* Tipy v yazykah programmirovaniya. M.: Lyambda press; Dobrosvet, 2012. 656 s.
6. *Frenkel' A., Bar-Hillel I.* Osnovaniya teorii mnozhestv. M.: Mir, 1966. 555 s.
7. *Algorithm W Step by Step.* [Ehlektronnyj resurs] // Home Comics Posts Publications Talks Misc About RSS. Ehlektron. dan. Rezhim dostupa: http://catamorph.de/publications/2007-09-01-algorithm-w-step-by-step.html (data obrashcheniya: 13.07.2015).
8. *Barendregt H.* Lambda calculi with types / Handbook of logic in computer science. Vol. 2. Oxford University Press, 1993.
9. *Damas L., Milner R.* Principal Type-schemes for Functional Programs // Principles of Programming Languages, ACM, 1982. P. 207–212.
10. *Geuvers H.* Introduction to Type Theory / Alfa Lernet summer school, Uruguay, 2008.
11. *Harrison J.* Introduction to Functional Programming. Cambridge University, 1996/1997.
12. *Kfoury A. J., Wells J. B.* Principality and decidable type inference for finite-rank intersection types // ACM Symposium on Principles of Programming Languages (POPL), 1999. P. 161–174.
13. *Severi P. G.* Normalization in Lambda Calculus and its relation to Type Inference // Thesis Technische Universiteit Eindhoven, 1996.
14. *Wells J. B.* Typeability and Type Checking in the Second Order λ-calculas are Equivalent Science // IEEE, 1994. P. 176–185.