УДК 510.51:515.122.5;515.126;510.57

НЕПРЕРЫВНОСТЬ КАК ВЫЧИСЛИМОСТЬ[#]

Н. Н. Непейвода

Институт программных систем им. А. К. Айламазяна РАН, г. Переславль-Залесский

✉ nnn@nnn.botik.ru

Аннотация. Исследуются соотношения между вычислимостью и непрерывностью. Рассматривается вычислимость над произвольным исходным базисом типов данных и функций (основой) при помощи рекурсивных схем Маккарти и строго типизированных функционалов конечных типов. Доказано, что в этом случае вычислимые функционалы сильно непрерывны в бэровском смысле: для функций-параметров при любом значении остальных аргументов можно найти конечную совокупность их значений, однозначно определяющую результат. На базе относительной вычислимости развивается подход к конструктивной топологии, в котором эквиваленты точечный подход (элемент – фундаментальная последовательность окрестностей) и аппроксимационный подход абстрактной топологии (функция над топологическими пространствами является окрестностным отношением). Формулируется понятие **Б**-пространств, позволяющее конструктивизировать сепарабельные пространства со счётной базой окрестностей. Доказывается непрерывность точечных конструктивных функций **Б**-пространств и возможность их преобразования в окрестностные отношения. Устанавливается эквивалентность понятий вычислимости относительно некоторой основы и непрерывности. Формулируется понятие относительно конструктивной функции, перерабатывающей каждый элемент в элемент, конструктивный относительно аргумента и фиксированной основы. Устанавливается его эквивалентность понятию счётно непрерывной функции, образующейся объединением счётного семейства непрерывных на подпространствах функций. Поскольку любое сепарабельное пространство со счётной базой может быть описано как **Б**-пространство, этот результат не содержит конструктивных ограничений. Обсуждается взаимосвязь предлагаемого подхода с другими подходами к конструктивной топологии.

Ключевые слова: относительная вычислимость, сепарабельность, счётная база, точечные пространства, абстрактная топология, аппроксимационный подход, бэровская непрерывность, непрерывность на подпространствах.

ВВЕДЕНИЕ

В работах, посвящённых вычислимости в топологических пространствах, с необходимостью присутствует её связь с непрерывностью (их обсуждению посвящён § 3). Вычислимость рассматривается на базе единой стандартной модели: машин Тьюринга. Возникает вопрос о взаимосвязи абстрактного общего понятия вычислимости с непрерывностью. Ответу на него посвящена данная статья.

[#] Работа выполнена в рамках проекта 125021302067-9 Института программных систем им. А. К. Айламазяна РАН.

1. АЛГОРИТМИЧЕСКИЙ ФОРМАЛИЗМ

1.1. Рекурсивные схемы

Практика программирования показала, что функциональные программы для дискретных и непрерывных задач пишутся в разном стиле. Стандартное в информатике определение вычислимости через машины Тьюринга либо λ-исчисление [1] прагматически корректно прежде всего для дискретных задач. Хотя практически все рассматриваемые в теории варианты вычислимости формально равносильны, области их практической применимости могут различаться. Поскольку

здесь анализируется вычислимость для непрерывных задач, нужно проявить некоторую аккуратность, выбрав вариант определения вычислимости, подходящий для используемого стиля программирования. В данной работе основным понятием вычислимости является относительная вычислимость рекурсивных схем по Маккарти [2] над *основой*: базисом исходных функций и предикатов. Она обобщается на строго типизированное исчисление функционалов конечных типов. Формально это является ограничением определения λ-функционалов конечных типов с комбинатором неподвижной точки, прямыми произведениями и условными операторами, детально исследованного в информатике [1, 3, 4].

Определение 1. Башня типов данных $\mathbb{T}$.

1. Конечная совокупность типов данных $\mathbb{T}_0$, обязательно включающая тип **bool**.

2. Все типы из $\mathbb{T}_0$ принадлежат $\mathbb{T}$.

3. Тип $(\tau_1,\ldots,\tau_n)$ называется *записью*. Если $\tau_1,\ldots,\tau_n \in \mathbb{T}$ и не являются записями, то $(\tau_1,\ldots,\tau_n) \in \mathbb{T}$. Если все члены записи – типы из $\mathbb{T}_0$, то запись принадлежит подклассу $\mathbb{T}_1$.

4. Если $\tau_1,\ldots,\tau_n \in \mathbb{T}$, то $(\tau_1,\ldots,\tau_n \to \tau) \in \mathbb{T}$. Такие типы называются *функциональными*.

5. Каждому типу сопоставляется неограниченная совокупность переменных этого типа.

Определение 2. Основа.

1. Каждому типу данных из $\mathbb{T}_0$ сопоставлено непустое множество данных, обязательно включающее элемент «мягкая ошибка» $\perp$. Типу **bool** сопоставляется множество $\{T, F, \perp\}$.

2. Для каждых типов вида $(\tau_1,\ldots,\tau_n)$ имеются функции

$$join:(\tau_1,\ldots,\tau_n \to (\tau_1,\ldots,\tau_n));$$
$$pr_i:((\tau_1,\ldots,\tau_n) \to \tau_i),$$

удовлетворяющие условию

$$pr_i(join(a_1,\ldots,a_n)) = a_i \text{ при } 1 \leqslant i \leqslant n. \qquad (1)$$

3. Конечная совокупность исходных типизированных функций с аргументами и значениями из $\mathbb{T}_1$ и констант с типами из $\mathbb{T}_0$ (логические константы **true** и **false** имеются в любом случае). Все функции устойчивы относительно $\perp$:

если $f(x_1,\ldots,x_n,\perp,y_1,\ldots,y_k) = z$, $z \neq \perp$,

то для всех x $f(x_1,\ldots,x_n,x,y_1,\ldots,y_k) = z$

(такие функции в информатике называют *генерическими*). Константы из основы называются *исходными*. Структуры, все элементы которых – константы, также считаются константами.

Функции с логическим значением результата называются *предикатами*. ♦

В дальнейшем последовательность выражений $a_1,\ldots,a_n$ часто будет обозначаться $\vec{a}$. Соответственно, $\vec{\perp}$ – последовательность $\perp$. Условие (1) означает, что тип записи изоморфен прямому произведению множеств значений составляющих его типов. Но ситуация несколько тоньше, поскольку множества значений функциональных типов не определяются и, соответственно, констант этих типов до появления явных определений нет, за исключением исходных функций. Поэтому условие (1) означает, что если какие-то значения соответствующих типов имеются, то имеется и значение, соответствующее их *n*-ке.

Далее, никаких ограничений на появление $\perp$ в качестве элементов записи нет. Легко видеть, что функции *join* и pr_i устойчивы относительно $\perp$. Это позволяет обойти возражения Кларка и рассматривать лишь функции с одним аргументом или же с одним предметным аргументом и одним функциональным.

В дальнейшем, если исключены двусмысленности, функция *join* опускается.

Определение 3. Термы.

1. Константы и переменные типа τ являются термами типа τ.

2. Если f – терм типа $(\tau_1,\ldots,\tau_n \to \tau)$, а $t_1,\ldots,t_n$ – термы типов $\tau_1,\ldots,\tau_n$ соответственно (такая последовательность термов $\vec{t}$ называется *подходящей* для *f*), то $f(t_1,\ldots,t_n)$ – терм типа τ.

3. Если b – терм типа **bool**, r и u – термы типа τ, то

if b **then** r **else** u **fi**

– терм типа τ. Такие термы называются *условными*. ♦

Обозначение $t\{f_1,\ldots,f_n,x_1,\ldots,x_k\}$, или сокращённо $t\{\vec{f},\vec{x}\}$, означает, что в терме используются лишь перечисленные константы и переменные, кроме исходных. Заметим, что некоторые из переменных могут быть функционального типа. *Параметрами* терма $f(t_1,\ldots,t_n)$ называются все t_i, не имеющие вида $(r_1,\ldots,r_k)$, и все r_i для таких записей. Ввиду ограничений на типы это определение не рекурсивно. Метаобозначение $f<t_1,\ldots,t_n>$ означает, что $t_1,\ldots,t_n$ – список всех параметров терма, переставленных в удобном порядке. В одном и том же предложении порядок в выражениях $f<r_1,\ldots,r_n>$ не меняется.

Определение 4. Рекурсивная схема.

Пусть $f_1,\ldots, f_n$ – новые константы функциональных типов, называемые рекурсивными функциями. Тогда совокупность определений вида

$$f_i(x_1,\ldots, x_k) \leftarrow t[f_1,\ldots, f_n, x_1,\ldots, x_k],$$

где $\vec{x}$ – последовательность переменных, подходящая для f_i, называется *рекурсивной схемой*. ♦

Последовательность определений внутри рекурсивной схемы не имеет значения.

Определение 5. *Вычислительная обстановка*, определяемая рекурсивной схемой, состоит из основы и функций, определённых схемой (рекурсивных функций).

1.2. Абстрактная вычислимость

Рассмотрим программы, работающие на основе рекурсивных схем, строго определим их исполнение и вычислительные возможности.

Терм называется *замкнутым*, если он не содержит переменных.

Процесс вычисления определим в духе правил переписывания термов, чтобы вычислительная система оказалась подсистемой рекурсивных функционалов конечных типов из работы [1], и воспользуемся установленными там результатами.

По принципиальным соображениям (функционалы должны работать не только с определимыми с их помощью аргументами) в вычислениях разрешается использовать константы для всех элементов типов из $\mathbb{T}_0$.

Определение 6. *Нормальная форма* – терм, к которому не может быть применено ни одно правило преобразования. Процесс вычисления замкнутого терма определяется как последовательность замен подтермов, пока не получится нормальная форма (если это когда-то произойдёт). Если процесс вычисления не завершается, то значением считается $\perp$. Терм *вычисляется*, когда к нему применяется одно из правил переписывания. Терм *исчезает* в случаях, оговорённых в правилах переписывания, или когда он является подтермом исчезающего терма.

1. Если терм имеет нормальную форму, то процесс завершён и эта нормальная форма является его значением.

2. Если терм имеет вид $f(\vec{t})$, где f – базовая функция, и все параметры – константы, то он заменяется на результат вычисления f.

3. Если терм имеет вид $f<\vec{c}, \vec{t}>$, где f – базовая функция, $\vec{c}$ – константы, и $f<\vec{c}, \vec{\perp}> \neq \perp$, то он заменяется на результат вычисления $f<\vec{c}, \vec{\perp}>$, а все элеиенты $\vec{t}$ исчезают.

4. Терм $pr_i(join(t_1,\ldots, t_n))$ заменяется на t_i, все остальные t_j исчезают.

5. **if true then** r **else** u **fi** заменяется на r, u исчезает.

6. **if false then** u **else** r **fi** заменяется на r, u исчезает.

7. **if** $\perp$ **then** r **else** u **fi** заменяется на $\perp$, r и u исчезают.

8. Если терм имеет вид $f(\vec{t})$, где f – рекурсивная функция, то он заменяется на определение f, в котором вместо переменных подставлены термы из $\vec{t}$.

Результат преобразований *промежуточный*, если к нему может быть применено лишь правило 8.

Замечание 1. Условие генеричности содержательно означает, что данный аргумент $\perp$ не используется в вычислениях. Правило 3 выражает условие, чтобы это выполнялось для всех исходных функций.

Рассмотрим структуру промежуточных результатов.

Определим дисциплину вычислений, позволяющую с гарантией получить результат, если он возможен.

Определение 7. Вычисление *регулярно*, если оно удовлетворяет следующим ограничениям.

1. Правило 8 применяется лишь в промежуточном терме.

2. В терме **if** $f(t_1,\ldots, t_n)$ **then** r **else** u **fi** термы r и u и их подтермы не вычисляются.

3. В промежуточном терме в подтерме $f(t_1,\ldots, t_n)$, где f не является исходной функцией, t_i не вычисляются.

4. Любой подтерм $f(t_1,\ldots, t_n)$, появившийся в промежуточном терме, в дальнейшем вычислении либо исчезает, либо вычисляется. ♦

Определение 8. Рекурсивная функция *тотальна*, если любое её регулярное вычисление заканчивается. Функция *генеральна*[1], если она тотальна и любое её вычисление при аргументах, не равных $\perp$, не даёт $\perp$. Рекурсивный функционал *тотален* (*генерален*), если при любой подстановке в него вместо функциональных параметров тотальных (генеральных) функций получающаяся функция – тотальная (генеральная).

[1] Название дано по аналогии с общерекурсивной.

Функция f типа $\vec{\tau}\to\pi$, где все τ_i, $\pi\in\mathbb{T}_1$, *вычислима*, если есть рекурсивная функция, результатом вычисления которой для каждого значения $\vec{c}$ аргументов f является $f(\vec{c})$. Функционал $\varphi<\vec{f},\vec{x}>$, где $\vec{x}$ и результат типов из $\mathbb{T}_1$, а $\vec{f}$ типов $\vec{\tau}\to\pi$, где все $\tau_i,\pi\in\mathbb{T}_1$, *вычислим*, если имеется рекурсивный функционал Φ такой, что при расширении основы любыми подходящими по типам $\vec{f}$ и подстановке $\vec{c}$ вместо $\vec{x}$ результат вычисления $\Phi(\vec{f},\vec{c})$ равен $\varphi(\vec{f},\vec{c})$. ♦

Замечание 2. В тотальных функциях ошибка появляется как вычисленное значение и может быть обработана. Если же функция зацикливается, вычисляясь бесконечно, то обработать такую ситуацию в общем случае невозможно.

Определения для функционалов не ограничиваются функциями, определимыми в основе. Они должны работать над любыми функциями. Также отметим, что любая внешняя функция тотальна.

Для функционалов более высоких типов определить вычислимость не пытаемся, поскольку множество функций таких типов остаётся неопределённым.

1.3. Свойства вычислимости

Теорема 1 (теорема корректности вычислений).

1. *Все завершающиеся вычисления дают один и тот же результат.*

2. Если какое-то вычисление завершается, то завершаются все регулярные вычисления.

Доказательство. Первый пункт – следствие теоремы о корректности правил переписывания для типизированного λ-исчисления с рекурсией [1, *гл. 5*].

Второй пункт – следствие теоремы о головной рекурсии [1, *гл. 3.7*].

Лемма 1. *Никакой вычислимый функционал не может выдать в качестве компонента результата новую функцию.*

Доказательство. Согласно синтаксическим ограничениям, все нормальные формы являются константами.

Поскольку вычислить новую функцию невозможно, определим понятие преобразования функции при помощи вычислимого функционала.

Определение 9. Функционал $\Psi<f,x>$ преобразует функцию f в g, если после добавления функции f к основе для всех c $\Psi<f,c>=g(c)$. ♦

Это определение естественно обобщается на переработку структуры функций в структуру функций.

Лемма 2. *Любой вычислимый функционал* Ψ *преобразует вычислимые функции в вычислимые.*

Доказательство. Добавляем определения функций-аргументов $\vec{f}$ в определение функционала Ψ и без расширения основы получаем определение функции-результата

$$\psi(\vec{x})\leftarrow\Psi(\vec{f},\vec{x}).$$

Определение 10. Конечная совокупность X значений аргументов $f_i(\vec{b_i})=d_i$ функционала $\Psi<f,x>$ *гарантирует* на c значение y, если при подстановке любых функций, удовлетворяющих $f(\vec{b_i})=d_i$, значение $\Psi<f,c>=y$. Если любое её подмножество не гарантирует значения, она его *в точности гарантирует*. Совокупность X будем задавать как множество пар (c_i,d_i) (называемое *фрагментом* функции). Функция имеет фрагмент X, если она принимает на всех c_i значения d_i. ♦

Практический критерий проверки, что фрагмент в точности гарантирует значение: запустить функционал на тестирование со следующими функциями: F, равной $\perp$ повсюду, кроме данного множества, а на c_i выдающей d_i, и её модификации, дополнительно выдающие ошибку на одном из c_i. Но это работает не всегда. Если рекурсивная схема зацикливается, то не получаем ответа. Такое может случиться и когда был «незаконный» вызов функции-параметра, в результате которого вычисление пошло по другому пути.

Введённые ограничения (отсутствие исходных функционалов) позволяют установить нарушающуюся в более общем случае бэровскую непрерывность вычислимых функций.

Теорема 2 (теорема непрерывности 1). *Для любого замкнутого терма* t*, в рекурсивной схеме определения которого присутствуют лишь функциональные переменные* $f_1,\ldots,f_n$ *и регулярное вычисление которого конечно, имеются фрагменты* X_i *таких, что при подстановке в него других функций, имеющих те же фрагменты, значение терма не меняется.*

Доказательство. В вычислении присутствует лишь конечное число вызовов функций-параметров. Если другие функции имеют те же значения на данных параметрах, то вычисление будет повторено с тем же результатом вычислений.

Пример 1. Существенность ограничений сложности типов в основе показывает следующий пример. Теорема непрерывности нарушается, если добавить исходный функционал второго типа $FZ:((\mathbf{nat}\to\mathbf{nat})\to\mathbf{bool})$ с определением

$$FZ(f)\leftarrow\begin{cases}T, \text{ если нет } i \text{ такого, что } f(i)=\perp,\\ \perp \text{ иначе.}\end{cases}\ \blacklozenge$$

Таким образом, определена концепция вычислимости, приспособленная к релятивизации на многие исходные пространства и любые исходные функции, дающая сильную непрерывность вычислимых функционалов второго типа.

Пример 2. Минимальной основой, порождающей стандартные вычислимые функции натуральных чисел, является система типов $\{\mathbf{nat, bool}\}$, где **nat** интерпретируется как тип натуральных чисел, константы $0\in\mathbf{nat}$, предиката $Z\in(\mathbf{nat}\to\mathbf{bool})$, функций S, Pd типа $(\mathbf{nat}\to\mathbf{nat})$, интерпретируемых как

$$Z(x)\equiv(x=0),\quad S(x)=x+1,$$

$$Pd(x)=\begin{cases}\perp, \text{ если } x=0,\\ x-1, \text{ если } x>0.\end{cases}$$

Напомним, что истина и ложь добавляются всегда.

Сложение и умножение определяются рекурсивной схемой

$$\begin{cases}A(x,\,y)\leftarrow \textbf{if } Z(y) \textbf{ then } x \textbf{ else } A(S(x),\,Pd(y)) \textbf{ fi},\\ M(x,\,y)\leftarrow \textbf{if } Z(y) \textbf{ then } 0 \textbf{ else } A(M(x,\,Pd(y)),\,x) \textbf{ fi}.\end{cases}$$

Функция второго порядка I итерирует применение аргумента f x раз, Err формально всегда даёт ошибку, а фактически её выполнение никогда не заканчивается и она ничего не выдаёт:

$$\begin{cases}I(x,\,y,\,f)\leftarrow \textbf{if } Z(x) \textbf{ then } y\\ \qquad\qquad\qquad \textbf{else } I(Pd(x),\,f(y),\,f) \textbf{ fi},\\ Err(x)\leftarrow Err(x).\end{cases}$$

Связки lisp-логики [5] определяются схемами, поэтому далее используем их свободно:

$$\begin{cases}A\wedge B\leftarrow \textbf{if } A \textbf{ then } B \textbf{ else } F \textbf{ fi},\\ A\vee B\leftarrow \textbf{if } A \textbf{ then } T \textbf{ else } B \textbf{ fi},\\ \neg A\leftarrow \textbf{if } A \textbf{ then } F \textbf{ else} T \textbf{ fi}.\end{cases}$$

Если в обстановке есть или моделируются натуральные числа, то пользуемся стандартными обозначениями для операций и отождествляем натуральные числа с их кодом.

1.4. Свойства обобщённого графика

Как было замечено выше, можно без ограничения общности рассматривать функционалы с двумя аргументами: исходного типа и функциональным. Следующее соглашение обеспечивает возможность пользоваться результатами стандартной теории алгоритмов.

Соглашение о перечислимости и списках. *Для каждого исходного типа добавлен предикат равенства и функция перечисления* en *такая, что последовательность* $en^n(a_0)$ *перечисляет без повторений все элементы данного типа, где* a_0 *– константа. Хотя бы один из исходных типов бесконечен.*

Тогда этот бесконечный тип может быть использован как изоморф натуральных чисел. Будем обозначать i-й элемент просто как i. Конечно, если среди исходных типов есть натуральные числа, то можно ими воспользоваться прямо. Как в стандартной теории вычислимости, определим примитивно рекурсивные кодирования n-к, списков и конечных множеств числами, операции объединения и пересечения закодированных множеств и предикат принадлежности числа множеству. Для удобства несколько конкретизируем кодирование множества. Это список, в котором нет повторяющихся элементов.

Определение 11. *Обобщённый график* – перечислимое множество троек (X, c, d), где значение c не содержит[2] константы $\perp$, и для каждых троек (X_i, c, d_i) с одним и тем же c ни одно из X_i не вложено в другое X_j. Обобщённый график является *графиком функционала* $F(f, x)$, если во всех тройках X – фрагмент, гарантирующий d на c. График *полон*, если для любых φ, c, для которых вычисление $F(\varphi, c)$ заканчивается, найдётся (X, c, d), где X – фрагмент φ. ♦

Зададим в виде спецификации, реализуемой стандартными методами программирования, оператор вычисления функционала $F(f, x)$ по полному графику G:

$$\begin{cases}\text{если в перечислении } G \text{ есть тройка } (X, x, d),\\ \qquad \text{где } X-\text{фрагмент } f, \text{ то } d;\\ \text{если такой нет, работаем бесконечно.}\end{cases}$$

Таким образом, полный обобщённый график – сильное и адекватное определение функционала. Было бы отлично, если бы рекурсивную схему для него удалось построить синтаксическими преобразованиями схемы функционала. Рассмотрим возникающие при этом сложности.

Лемма 3 (прослеживание вызовов). *Пусть рекурсивная схема определяет функционал* Φ *с функциональными параметрами* f_i *и предметным*

[2] Напомним, что элемент $\perp$ не превращает в ошибку всю запись.

(т. е. из $\mathbb{T}_1$) параметром x. Тогда выполнено следующее.

1. При вычислении других функций этой схемы f_i может быть применено, лишь если данная функция также представляет собой функционал и в качестве некоторых его функциональных параметров подставлено f_i.

2. Все места возможного применения f_i в терме t могут быть статически найдены при анализе рекурсивной схемы и они имеют вид $pr_j(F)(t)$, где F – функциональный параметр некоторой функции.

3. Применение f_i произойдёт тогда и только тогда, когда место возможного применения f_i окажется на пути вычисления со значением f_i параметра $pr_j(F)$.

Доказательство. Дадим эскиз доказательства. Поскольку f_i не является ни исходной, ни определяемой в схеме функцией, она может быть лишь значением некоторого параметра. Это доказывает пункт 1.

Построим список функциональных параметров схемы для Ф, компоненты которых могут быть заменены на f. Само f_i принадлежит этому списку. Если внутри определения какой-либо функции вместо параметра g подставляется функция из Ф, то g добавляется к Ф. Применения параметров из Ф являются возможными применениями f_i. Пункт 2 доказан. Пункт 3 следует из пункта 2. ♦

Лемма о прослеживании работает и в случае, когда в схеме есть функционалы, вычисляющие функции, поскольку они не могут вычислить новые функции. Последующие построения в принципе можно проделать для случая, когда такие функционалы разрешены. Но поскольку это непринципиально для основной цели работы (операторы над функциями топологического пространства здесь не рассматриваются), ограничимся схемами второго порядка, не определяющими функции, выдающие функции.

Задача 1. Модифицировать определение функционала так, чтобы в случае завершения работы на данных параметрах он выдавал бы в точности гарантирующий результат фрагмент функции-параметра.

Задача 2. Модифицировать определение терма с одной функциональной переменной так, чтобы он вместо функции принимал в качестве параметра фрагмент и по возможности давал бы не только результат, но и диагностику, достаточен ли фрагмент для корректного вычисления.

Функцию-параметр обозначим f, а вычисляемый терм $T[f]$. Заметим, что преобразования схемы проводятся для данного конкретного терма, поскольку необходимо предварительное прослеживание. Базируясь на прослеживании, можно в задаче 1 разметить подтермы в исходной рекурсивной схеме, а в задаче 2 – подтермы преобразуемого терма и опять-таки определений в исходной рекурсивной схеме.

Определение 12. Подтерм *красный*, если он содержит вызов f. Подтерм *жёлтый*, если он такого не содержит, но является непосредственной составляющей красного терма (аргумент красной функции, компонента красной записи, альтернатива условного выражения с красным условием). Он *белый* в остальных случаях. Если $T[f]$ белый, то его помечаем как жёлтый. ♦

С помощью леммы о прослеживании можно раскрасить подтермы любого терма.

Несколько удобнее, чтобы индикатор ошибки обрабатывался первым, поэтому он всегда снабжается флагом **true**. В дальнейшем через $\varnothing$ обозначается пустое множество, через $\varnothing$ n-ка пустых множеств, через $\{x\}$ одноэлементное множество. Без ограничения общности можно предполагать, что все определяемые в схеме функции выдают результатом запись с n компонентами. Если S – запись, то (x, S) – результат добавления к ней компонента x. Это не приводит к двусмысленностям, поскольку в башне типов записи не могут быть компонентами записей. Заметим, что стандартное сворачивание n-ки в код списка не всегда корректно в связи с замечаниями Кларка: код списка, один из элементов которого ошибка, всегда ошибка. Чтобы не тратить излишние усилия с конкретными номерами, определим для каждого встречающегося в схеме типа n-к функцию $tail((x_1,\ldots, x_n)) = (x_2,\ldots, x_n)$.

Стандартными методами строим следующие вспомогательные функции.

1. Функцию $Ev(X, a)$, выдающую $(\mathbf{false}, b)$, если при некотором b пара $(a, b) \in X$, и $(\mathbf{true}, A)$ иначе.

2. Функцию $\cup$, строящую объединение двух множеств.

Через A обозначим константу, которой будет заменяться ненужное далее значение.

Модификация 1. $t^p[X, t]$ – вычисление при гарантированной успешности. Оно получается добавлением ко всем определённым в схеме функционалам, содержащим потенциальные применения f, предметного параметра X, и рекурсивной заменой всех вызовов $f(r)$ функции-

параметра, прослеженных по лемме 5, на $tail(Ev(X, r^p))$. Эта модификация даёт простейшее частичное решение задачи 2 без диагностики ошибок, связанных с параметром, отсутствующим в фрагменте. Заметим, что даже это частичное решение индивидуально для $T[f]$, поскольку раскраска зависит от T.

Модификация 2. Отслеживание и сбор применений функции-параметра: переработка T в T^v.

Для каждой функции, имеющей красное вхождение, определяем её красный вариант с дополнительным параметром X_1 типа фрагмент.

1. Красный вариант f:

$$f^v(x) \leftarrow \textbf{if } pr_1(x) \textbf{ then } (\textbf{true}, pr_2(x), A)$$
$$\textbf{else } (pr_1(Ev(X, tail(tail(x))), X_1 \cup \{pr_2(x)\}, \quad (2)$$
$$pr_2(Ev(X, tail(tail(x)))) \textbf{ fi}.$$

2. Красный вариант остальных исходных функций:

$$\varphi^v(X_1, x) \leftarrow \textbf{if } pr_1(x) \textbf{then}(\textbf{true}, pr_2(x), A)$$
$$\textbf{else } (\textbf{false}, pr_2(x), \varphi(tail(tail(x)))) \textbf{ fi}.$$

3. Красный вариант функции, определяемой как $\varphi(\alpha, x) \leftarrow t$, есть

$$\varphi^v(X_1, \alpha, x) \leftarrow t^v,$$

где t^v вычисляется рекурсивно по указанным ниже правилам.

Рекурсивно производим следующие замены.

1. Все белые подтермы остаются без изменений. Все жёлтые t заменяем на

$$t^v = (\textbf{true}, \varnothing, t).$$

2. Запись $(t_1, \ldots, t_n)^v(t_1, \ldots, t_n)^v$:

$$(\textbf{if } pr_1(t_1^v) \textbf{ then } (\textbf{true}, pr_2(t_1^v) \cup \cdots \cup pr_2(t_n^v), A) \textbf{ elif } \ldots$$
$$\textbf{elif } pr_1(t_n^v) \textbf{ then } (\textbf{true}, pr_2(t_1^v) \cup \cdots \cup pr_2(t_n^v), A)$$
$$\textbf{else } (\textbf{false}, pr_2(t_1^v) \cup \cdots \cup pr_2(t_n^v),$$
$$tail(tail(t_1^v)), \ldots, tail(tail(t_n^v))) \textbf{ fi}.$$

3. Условный терм **if** b **then** r **else** u **fi**^{v}:

$$\textbf{if } pr_1(b^v) \textbf{ then } (\textbf{true}, pr_2(b^v), A)$$
$$\textbf{elif } pr_2(b^v) \textbf{then } (pr_1(r^v),$$
$$pr_2(b^v) \cup pr_2(r^v), tail(tail(r^v))),$$
$$\textbf{else}(pr_1(r^v), pr_2(b^v) \cup pr_2(u^v), tail(tail(u^v))) \textbf{ fi}.$$

4. Красная функция $\varphi(s)$:

$$\textbf{if } pr_1(s^v) \textbf{ then } (\textbf{true}, pr_2(s^v), A)$$
$$\textbf{else}(pr_1(r^v), pr_2(b^v) \cup pr_2(u^v), tail(tail(u^v))) \textbf{ fi}.$$

Эта сложная перестройка программы соответствует понятию *завершителя* (*continuation*) в функциональных программах [1, 6]. В отличие от указанных работ здесь не понадобилось добавлять оператор третьего уровня, удалось перестроить функцию с понижением уровня.

Теорема 3 (обобщённый график). *По схеме генерального функционала* Φ *можно построить определение функции, перечисляющей его обобщённый график.*

Доказательство. Внесём дополнительную модификацию в схему T^v для терма $T = \Phi(f_0, x_0)$, где $\Phi(f_0, x_0)$ – константы[3]. Введём новую переменную для фрагмента X и добавим её в качестве параметра ко всем определениям и применениям функций, не используя её нигде внутри определений, кроме обязательного параметра при вызове любой функции. Единственное исключение – (2), в котором она используется в Ev. В связи с этим переопределим исходный функционал Φ как

$$\Phi^g(X, x) \leftarrow \Phi^v[X, x].$$

Стандартными методами построим некоторое перечисление $Enum(i)$ пар «фрагмент и предметный параметр» (X, x), начинающееся с нуля.

Определим функцию построения начального отрезка обобщённого графика. Для сокращения и обеспечения понятности записи обозначим неоднократно встречающееся выражение $\Phi^g(pr_1(Enum(i)), pr_2(Enum(i)))$ просто $[\Phi^g]$:

$$Gr(i, G, n) \leftarrow \textbf{if } i = n+1 \textbf{ then } G$$
$$\textbf{elif } pr_1([\Phi^g]) \textbf{ then } Gr(i+1, G, n)$$
$$\textbf{else } Gr(i+1, G \cup \{(pr_2([\Phi^g]),$$
$$pr_3([\Phi^g]), pr_2(Enum(i)))\}, n) \textbf{ fi}.$$

Она шаг за шагом добавляет найденные фрагменты, верифицированно гарантирующие $pr_2(Enum(i))$ на $pr_1(Enum(i))$. В пределе получается полный график функционала.

Замечание 3. Этот алгоритм даёт перечисление с повторениями. От этого можно избавиться стандартными методами.

Полное исследование соотношений между функционалами и обобщёнными графиками требует отдельного рассмотрения. Ограничимся упоминанием, что синтаксическое преобразование произвольного функционала в график в общем случае невозможно: его существование означало бы, в частности, разрешимость проблемы зацикливания. Поэтому важны частные случаи.

[3] Здесь f_0 может быть как новой исходной функцией, так и определённой в той же схеме; это влияет на процесс построения T^v, но не на новые модификации.

2. ПРИЛОЖЕНИЕ К ТОПОЛОГИИ

2.1. Б-пространства

Ю. Л. Ершов определил конструктивизацию топологических пространств со счётным базисом окрестностей (А-пространства) [7, 8]. Дадим обобщённое для случая подпространств произвольных сепарабельных пространств со счётной базой окрестностей определение, воспользовавшись идеями П. Мартин-Лёфа [9] и А. Лакомба [10]. При этом избавимся от исходной привязки к натуральным числам, имеющейся у всех перечисленных авторов и стандартной для всех работ по конструктивной топологии, перечисленных в фундаментальной монографии [11]. В случае обращения к натуральным числам это явно отмечается.

Определение 13. $\mathbf{Б}_0$-пространство – сепарабельное полное пространство, в котором выделен счётный базис окрестностей. Этот базис с добавленным пустым множеством обозначим $\mathfrak{A}$. Определим на $\mathfrak{A}$ основу.

1. Константы $\varnothing$ и $\mathfrak{U}$ (пустое множество и всё пространство).

2. Генеральная функция счётности $e:(\mathfrak{A}\to\mathfrak{A})$ такая, что последовательность $\lambda n.e^n(\varnothing)$ пробегает без повторений[4] всё $\mathfrak{A}$.

3. Генеральная операция пересечения окрестностей $A\cap B$.

4. Генеральный предикат $A\le B$ (*A тоньше B*), означающий, что либо замыкание A вложено в B, либо $A=B$ и B – одноэлементное открыто-замкнутое множество.

5. Генеральный предикат $A\#B$ (A *отделено* от B), означающий, что замыкания окрестностей не пересекаются.

6. (Необязательный) Некоторые дополнительные функции и предикаты на $\mathfrak{A}$. ♦

Пункт 2 задаёт биморфизм $\mathfrak{A}$ в **nat**, который не является изоморфизмом, поскольку в основе нет функции Pd и предиката равенства (даже $a=\varnothing$). Он необходим, чтобы гарантировать конструктивизацию счётности $\mathfrak{A}$. Конструктивными объектами, согласно определению 2, являются также n-ки окрестностей.

Предложение 1. *Если в* $\mathbf{Б}_0$*-пространстве вычислим предикат равенства окрестностей* $a=b$*, то окрестности образуют модель натуральных чисел и вычислимы все частично-рекурсивные функции на них.*

[4] В данном случае λ является общепринятым квантором функциональности и принадлежит метаязыку.

Д о к а з а т е л ь с т в о. Для доказательства достаточно построить функцию $Pd(a)$, поскольку $Z(a)\equiv a=\varnothing$.

$$\begin{cases} Pda(x,a) & \leftarrow \textbf{if } a=\varnothing \textbf{ then } \perp \\ & \quad \textbf{elif } e(x)=a \textbf{ then } x \textbf{ else } Pda(e(x),a) \textbf{ fi} \\ Pd(a) & \leftarrow Pda(\varnothing, a). \end{cases}$$

Если есть равенство, то свободно используются известные функции натуральных чисел, в частности, кортежи и операции над ними. Кортежи в данном случае являются также кортежами окрестностей. ♦

Определение 14. Метрическое $\mathbf{Б}_0$-пространство.

$\mathbf{Б}_0$-пространство *метрическое*, если имеется вычислимая генеральная функция меры μ, сопоставляющая каждой окрестности рациональное число $\mu(a)\geqslant 0$, и

1. $\mu(A)=0$ тогда и только тогда, когда A содержит не более чем одну точку.

2. Если $A\le B$ и $A\ne B$, то $\mu(A)<\mu(B)$.

3. Если $A\#B$, $A\le C$, $B\le C$, то $\mu(A)+\mu(B)<\mu(C)$.

4. У каждой точки $x\in\mathbf{Б}_0$ есть базовая окрестность сколь угодно малой меры.

Пример 3. Если базовые окрестности в некотором пространстве имеют по вложению структуру дерева, то $A\le B$ означает, что, когда B не лист, A находится на пути из B. Но лист – конечная точка пути, и тогда $A=B$.

Рассмотрим хаусдорфовы $\mathbf{Б}_0$-пространства. Вычислимым элементом $\mathbf{Б}_0$-пространства естественно считать класс эквивалентности вычислимых сходящихся последовательностей вложенных окрестностей. Н. А. Шанин [12] показал, что этого недостаточно: нужно иметь также явно заданный вычислимый регулятор сходимости данной последовательности. Поскольку не предполагается мера либо равномерная структуры окрестностей, необходимо соблюсти аккуратность.

Определение 15. Вычислимый элемент. *Шанинской точкой* называется пара всюду определённых функций f, r, где $\forall x(f(e(x)))\le f(x))$, а функция $r(x,y)$ (*регулятор*) такова, что для любой пары окрестностей $x\le y$ выполнено $f(r(x,y))\le y$ либо $f(r(x,y))\#x$. Если f, r вычислимы, то элемент $\mathbf{Б}_0$-пространства *конструктивен* относительно данной основы.

Шанинские точки (f,r) и (f_1, r_1) *эквивалентны* $(f, r)\equiv(f_1, r_1)$, если для любых n $f(r(f_1(e(n)), f_1(n)))\le f(n)$ и $f_1(r_1(f(e(n)), f(n)))\le$ $\le f_1(n)$.

Замечание 4. Содержательный смысл регулятора состоит в том, что он позволяет попасть либо

в меньшую окрестность, либо в зазор между меньшей и большей, тем самым обходя неприятный с конструктивной точки зрения случай, когда граница испытуемой окрестности попадает в точности на предел f, и в результате она не может отделиться от членов $f(n)$, а $f(n)$ не могут попасть внутрь.

Определение 16. **Б**-пространство – непустое подмножество **Б**$_0$-пространства X_{ϖ} (материнского пространства) $X \subseteq X_{\varpi}$ с базой обобщённых окрестностей, каждая из которых есть пересечение базисной окрестности из $\mathfrak{A}$ с множеством X. ♦

Таким образом, все перечисленные выше функции и предикаты наследуются **Б**-пространством от материнского **Б**$_0$-пространства.

Лемма 4. *Свойства шанинских точек.*

1. *В случае, если пара функций – шанинская точка, пересечение всех* $f(a)$ *содержит не более одного элемента.*

2. *В полном пространстве это пересечение непусто.*

3. *В метрическом пространстве шанинская последовательность* f *имеет меру членов, стремящуюся к нулю.*

Доказательство. Пусть даны две различные точки x, y. Тогда по сепарабельности есть базовые окрестности $x \in A$, $y \in B$, $A \# B$. Тогда возьмём $A_1 \le A$, $x \in A_1$. Поскольку $f(r(A_1, A)) \le A_1$, то $f(r(A_1, A)) \# B$, и тем самым y не принадлежит пересечению всех $f(n)$. Первый пункт доказан. Второй пункт выполнен по определению полноты. Третий пункт выполнен, поскольку для любой окрестности, где $\bigcap_a f(a) \in A$, имеем $f(r(A)) \le A$. ♦

В случае метрических пространств (как у П. Мартин-Лёфа и в других работах, перечисленных в книге [13]) регуляторы не нужны, поскольку достаточно потребовать быстрой сходимости последовательности (например, что мера каждого $f(n)$ не превосходит 2^{-n}). В общем случае при отсутствии регуляторов справедливы контрпримеры из основополагающей работы [12], разрушающие конструктивность построений.

Предложение 2. *Любое сепарабельное пространство* Ξ *со счётной базой окрестностей может быть представлено как* **Б**-*пространство.*

Доказательство. **Б**$_0$-пространством Ξ_0 служит пополнение Ξ. Оно тоже имеет счётную базу окрестностей. На ней определяем основу, и получаем **Б**-пространство. Окрестностями будут пересечения окрестностей из базы Ξ_0 с Ξ.

Заметим, что конкретизация основы для Ξ не оговаривается. Все функционалы работают над Ξ_0 и принадлежность Ξ оговаривается внешним образом, никак не используясь в вычислениях.

Определение 17. *Вычислимая функция* над элементами **Б**-пространств $f : \Xi \to \Upsilon$ – вычислимый функционал

$$\Phi : (((\mathfrak{A}, \mathfrak{A}) \to (\mathfrak{A}, \mathfrak{A})), \mathbf{nat}, \mathbf{nat} \to (\mathfrak{A}, \mathfrak{A})),$$

преобразующий любую шанинскую точку (f, r); $\bigcap_a f(a) \in \Xi$ в функцию $\lambda n, m.\Phi((s, r), n, m)$, являющуюся шанинской точкой (g, q); $\bigcap_a g(a) \in \Upsilon$, причём

$$(s, r) \equiv (s_1, r_1) \supset \Phi((s, r)) \equiv \Phi((s_1, r_1)). \qquad (3)$$

Здесь (s, r) понимается в смысле определённого выше преобразования функций.

Замечание 5. Таким образом, определение 17 можно перевести на более привычный язык следующим образом: по шанинской паре (s, r), представляющей x, Φ выдаёт шанинскую точку (g, q), представляющую $f(x)$, следующим образом:

$$\Phi(s, r, i, j) = (g(i), q(i, j)).$$

Аргументы функций пробегают всё пространство Ξ_0, ограничиваются лишь результаты и требование корректности. Тотальность и генеральность также рассматриваются на всём пространстве Ξ_0.

Пример 4. Приведём пример, иллюстрирующий важную роль регуляторов непрерывности. Возьмём обычное пространство действительных чисел с базисом окрестностей: интервалы рациональных точек $\left(a - \frac{1}{2^n}, a + \frac{1}{2^n}\right)$. Рассмотрим функционал, перерабатывающий каждую окрестность $\left(a - \frac{1}{2^n}, a + \frac{1}{2^n}\right)$ в $\left(1 - \frac{1}{2^n}, 1 + \frac{1}{2^n}\right)$, если её нижняя граница больше нуля, $\left(-1 - \frac{1}{2^n}, -1 + \frac{1}{2^n}\right)$, если верхняя граница меньше нуля, и $\left(-\frac{1}{2^n}, +\frac{1}{2^n}\right)$, если 0 находится внутри интервала. Он перерабатывает каждую вычислимую сходящуюся последовательность в вычислимую сходящуюся последовательность, но получить регулятор невозможно, поскольку он должен выдавать окрестности нуля по последовательностям, все члены которых включают нуль. Но тогда он должен выдать $\left(-\frac{1}{2}, \frac{1}{2}\right)$ по конечному числу её членов, и подменив её на последовательность, отделяющуюся от нуля на следующем шаге, получаем неверный результат (метод обмана).

Замечание 6. Функции не предполагаются всюду определёнными как операторы на элементах

пространства. Более того, возникает ещё один случай, когда функция не определена: если нарушено условие корректности (3) для шанинских пар, представляющих данный элемент. Корректная работа должна обеспечиваться лишь на элементах множества Ξ. Эквивалентности регуляторов результата для различных шанинских представлений аргумента не требуется.

2.2. Основная теорема

Теорема 4. *Функция непрерывна на* **Б**-*пространстве тогда и только тогда, когда она вычислима относительно некоторой основы.*

Доказательство теоремы требует нескольких дополнительных построений.

Лемма 5. *Вычислимая функция преобразует конструктивные точки* Ξ *в конструктивные точки* Υ.

Следствие леммы 2. Заметим, что аргументы вычислимой функции не предполагаются вычислимыми. Она должна корректно работать над любыми. В этом принципиальное отличие от стандартных концепций конструктивности [11], даже если их релятивизовать.

Лемма 6. *Все вычислимые функции непрерывны на своей области определения.*

Доказательство.

Следствие теоремы непрерывности 1. Пусть дана окрестность Y результата g,q. Тогда его регулятор даёт $q(Y)$ такое, что $g(q(Y)) \le Y$. По теореме непрерывности используется лишь конечное число значений $[f(\varnothing),\ldots, f(e^n(\varnothing))]$ функции-аргумента f для того, чтобы найти $g(q(Y))$. Возьмём теперь произвольную шанинскую точку f_1, g_1 из $f(e^n(\varnothing))$. Подменив в ней $[f_1(\varnothing),\ldots, f_1(e^n(\varnothing))]$ на $[f(\varnothing),\ldots, f(e^n(\varnothing))]$ и все значения $g_1(a)$ такие, что $f(e^n(\varnothing)) \le g(g_1(a))$, на $e^n(\varnothing)$, получаем эквивалентную шанинскую точку. Она принадлежит $g(q(Y))$ и, тем самым, Y. Итак, по любой окрестности результата можно найти окрестность аргумента, отображающуюся в неё.

Основная теорема в одну сторону доказана. Остаётся для каждой непрерывной на **Б**-пространстве функции найти основу, относительно которой она конструктивна. ♦

Функции в данном контексте не являются полноправными значениями. Совокупности вычислимых функций нет. Они образуют множество, внешнее по отношению к вычислительной модели. Тем более это относится к функционалам. Но имеется известное топологическое преобразование: перейти от непрерывной функции к отношению окрестностей результата и аргумента. Его конструктивная форма предложена П. Мартин-Лёфом [9]: здесь она обобщена на пространства, не являющиеся метрическими.

Определение 18. Понижение типа.

Аппроксимация – перечисляющая окрестности функция $\mathbb{A}$ такая, что для любых a и b есть c, $\mathbb{A}(a) \cap \mathbb{A}(b) = \mathbb{A}(c)$. Аппроксимация *максимальна*, если для любых $a \le b$ есть такое c, что $a\#\mathbb{A}(c)$ либо $\mathbb{A}(c) \le b$.

Открытое множество – функция $\mathbb{O}$, значением которой являются окрестности и для любого a, если $b \le \mathbb{O}(a)$, то имеется c, такое, что $\mathbb{O}(c) = b$.

Окрестностное отношение – функция R из $\mathfrak{A}$ в пары окрестностей (X, Y) точек $x \in \Xi$, $y \in \Upsilon$ такая, что $R<X>$ – аппроксимация в Υ, $R^{-1}<Y>$ – открытое множество в Ξ. ♦

Для обеспечения конструктивности построим через R перечисляющие функции аппроксимации и открытого множества.

Лемма 7. *Функция как отношение. Для любой непрерывной функции* $f : \Xi \to \Upsilon$ **Б**-*пространств можно найти окрестностное отношение* R *такое, что для любой окрестности* a *аргумента* x $R<a>$ *содержит* $f(x)$, *а для окрестности* b *результата* $f(x)$ *объединение окрестностей* $R^{-1}<b>$ *содержит*[5] $f^{-1}(f(x))$, *и для любого* $z\#f(x)$ *есть окрестность* a *такая, что некоторая окрестность из* $R<a>$ *отделена от* z.

Доказательство. Возьмём произвольную непрерывную функцию $f : \Xi \to \Upsilon$. Поскольку f непрерывна, то для любой окрестности Y результата $f(x)$ имеются окрестности X аргумента x такие, что для любого $x \in X$ $f(x) \in Y$. Они образуют искомое отношение R. ♦

При этом для $x \in X$ $R<X>$ максимально. Построим вычислимую относительно материнского пространства и R функцию поиска $sf(a,b)$:

$$
\begin{aligned}
sf\,0(a, b, x) \leftarrow\ & \textbf{if } a \le b \textbf{ then} \\
& \quad \textbf{if } R(a, x) \textbf{ then} \\
& \quad\ \textbf{if } x \le a \textbf{ then } x \textbf{ elif } x\#a \textbf{ then } x \textbf{ fi} \\
& \quad\ \textbf{else } sf\,0(a, b, x+1) \textbf{ fi} \\
& \textbf{else } \perp \textbf{ fi} \\
sf(a, b) \leftarrow\ & sf\,0(a, b, 0).
\end{aligned}
\qquad (4)
$$

[5] f^{-1} рассматривается как отношение, а не как функция: $R^{-1} = \{(x, y) \mid (y, x) \in R\}$.

Эта функция находит в аппроксимации окрестность, вложенную либо отделённую от меньшей для любой пары вложенных окрестностей из аппроксимации[6].

Лемма 8 (повышение типа). *Если окрестностное отношение R представляет функцию f, то f вычислима относительно материнских* **Б**$_0$*-пространств, предиката $=$ и R.*

Доказательство. Поскольку имеется предикат равенства, можно отождествить окрестности с патуральными числами и пользоваться стандартными рекурсивными функциями.

Построим последовательность-результат $\lambda n.g(n)$ по функции f из шанинской точки. Инвариантом создаваемой функции является

Для любого n $f(n), g(n) \in R$ и $g(n+1) \le g(n)$.

Определим вспомогательную функцию $sr(a, b)$:

$$\begin{cases} sr0(a, b, n) \leftarrow \textbf{if } pr_1(R(n)) = a \wedge pr_2(R(n)) \le b \\ \qquad\qquad \textbf{then } pr_2(R(n)) \textbf{ else } sr0(a, b, S(n)) \textbf{ fi} \\ sr(a, b) \leftarrow sr0(a, b, 0). \end{cases}$$

При условии, что вычисление sr для данных a,b конечно, она удовлетворяет следующему инварианту:

$$(a, sr(a, b)) \in R \wedge sr(a, b) \le b.$$

Функция g определяется схемой

$$g(n) \leftarrow \textbf{if } Z(n) \textbf{ then } f_2(0) \textbf{ else } sr(f_1(n), g(Pd(n))) \textbf{ fi}.$$

Конечность вычисления g гарантируется в случае, если f принадлежит области определения функции, послужившей основой для построения R.

Регулятор по окрестностному отношению уже построен выше (см. формулу (4)), нужно лишь заменить найденную окрестность на её номер в результате искомой функции.

Тем самым основная теорема доказана. ♦

Следствием этого является

Лемма 9. *Результат f(x) непрерывной функции конструктивен относительно R и x.*

Доказательство. Функция sf даёт результат, а sr – его регулятор.

2.3. Вычислимость окрестностного отношения

Возникает вопрос о вычислимости окрестностного отношения относительно основы, для которой вычислима данная непрерывная функция. Здесь приходится использовать многоступенчатые нетривиальные трансформации определения функции и нетривиальную технику программирования. Поскольку для логических результатов работы эта техника вторична, в данном пункте она описывается в виде идей преобразований и формулировок результатов, а исследование получившейся концепции вычислимости, обладающей некоторыми нетривиальными свойствами и при этом достаточной силой, – предмет других работ.

Для вычисления окрестностного отношения хотелось бы запустить функцию хотя бы для всех конструктивных чисел. Это невозможно ввиду отсутствия средств изменения значений функциональных переменных. Но можно обойти эту трудность для генеральных функций, пользуясь основной теоремой. Поэтому доказательство теоремы требует нескольких вспомогательных построений. Они представляют самостоятельный интерес и поэтому проведены для общего случая, но достаточно громоздки технически. Идея же, на которой они базируются, проста и прозрачна.

Теорема 5 (вычисление отношения). *По определению функционала* Ф*, вычисляющего генеральную функцию f, можно построить определение окрестностного отношения для этой функции в той же основе, пополненной предикатом равенства.*

Доказательство. Построим, проанализировав определение функции, перечисление обобщённого графика функционала (основная техническая часть работы) и заменим в нём все пары $((F, y), z)$ на $(x, pr_1(z))$, где x – наименьшая из окрестностей в $pr_1(F)$. Первый элемент шанинской пары монотонно убывает, значит, если оказалось достаточно других объемлющих его элементов, то тем более в окрестности x результат функции попадёт в окрестность $pr_1(z)$, а построения регулятора на значение функции не влияют и они опускаются.

2.4. Относительная конструктивность

Возникает задача: является ли относительная конструктивность характеристикой какого-то класса функций?

Определение 19. Относительная конструктивность. Функция ψ на **Б**-пространстве *конструктивна относительно* $f : (\mathbb{N} \to \mathbb{N})$, если для каждого числа x есть конструктивная функция φ над основой S, Pd, Z, f такая, что $\psi(x) = \varphi(x)$. ♦

Прежде всего, тривиален ответ: этот класс шире вычислимых (непрерывных) функций. Результат функции Дирихле 0 либо 1 и конструктивен. Но сама эта функция не конструктивна относительно никакого базиса.

[6] С конструктивной точки зрения доказательство корректности данной функции требует применения бар-индукции Брауэра [16], что ещё раз показывает существенность требования счётности базиса окрестностей.

Определение 20. Функция *счётно непрерывна*, если **Б**-пространство можно разбить на счётную совокупность **Б**-пространств, на каждом из них она непрерывна.

Теорема 6 (эквивалентность понятий). *Функция относительно конструктивна тогда и только тогда, когда она счётно непрерывна.*

Доказательство.

Только тогда. Пусть функция относительно конструктивна. Поскольку каждый её результат порождается функционалом, определимым над основой S, Pd, Z, F и являющимся конструктивной функцией, сопоставим всем числам вычисляющие их функционалы. Поскольку совокупность вычислимых функционалов счётна, и каждая конструктивная функция непрерывна, получили требуемое разбиение.

Тогда. Пусть функция счётно непрерывна. Возьмём окрестностные функционалы Φ_i, определяющие каждый из непрерывных фрагментов, и объединим их в функцию $\Psi(i, n) = \Phi_i(n)$. Тогда каждый $f(x)$ вычислим относительно Ψ. ♦

Это построение принципиально неконструктивно. Объединить конструктивные фрагменты в единую конструктивную функцию на пространстве невозможно. Таким образом, вопрос, какая именно функция применяется к данному элементу, неразрешим относительно никакой основы. Заметим, что в выполненных построениях ни разу не использовались запросы к предикату принадлежности элемента носителю **Б**-пространства или к свойствам, выделяющих **Б**-пространство из материнского.

Пример 5. Наконец, построим функцию действительного переменного, не являющуюся счётно непрерывной. Возьмём первый ординал мощности континуум и упорядочим действительные числа по этому ординалу следующим образом. Каждому ординалу сопоставим действительное число, не имеющее конструктивного представления через ранее упорядоченные числа. Такое найдётся, поскольку мощность каждого ординала, меньшего первого континуального, меньше континуума, и через каждое число конструктивно определима лишь счётная совокупность чисел. Тогда функция, сопоставляющая каждому x_α следующее за ним $x_{\alpha\oplus 1}$, не является относительно конструктивной для любого числа и тем самым не является счётно непрерывной.

3. ОБСУЖДЕНИЕ И ПРИМЕНЕНИЕ РЕЗУЛЬТАТОВ

3.1. Взаимосвязь с другими концепциями конструктивности в топологии

Прежде всего, заметим, что в работах, использованных и описанных в [11] непрерывность конструктивных функционалов в том или ином виде заранее предполагается.

Лишь в советском конструктивизме [14] непрерывность функций действительного переменного доказывалась полностью независимо, но там функционалы были «хакерскими»: во-первых, они должны были перерабатывать лишь конструктивные функции и, во-вторых, считался известным исходный код алгоритма любой функции, с ним можно было делать всё, что угодно. Первый пункт, как было показано в работе [15], безвреден, если не предполагать второго: пространство функций в интуиционизме может состоять лишь из алгоритмически вычислимых, но доступа к их программам не должно быть.

Маленькое видоизменение определения функционалов высших типов (отсутствие исходных, внешним образом заданных функционалов высших порядков) привело к сильной «бэровской» непрерывности и одновременно к возможности обрабатывать любые функции. Эту непрерывность можно обосновать интуиционистски, если принять брауэров принцип бар-индукции [16]. Тем самым был сделан ещё один шаг. Можно предполагать знание алгоритмов функций, если разрешить использовать их лишь как в современной информатике: вызывая замкнутые модули.

В топологии следуем линии Ю. Л. Ершова [7, 8] и П. Мартин-Лёфа [9]. Сам П. Мартин-Лёф пользовался алгоритмами, а топологию вводил как бесточечную, на базе аппроксимаций, и функции на топологических пространствах как окрестностные отношения. Последователи линии П. Мартин-Лёфа ограничивались рассмотрением компактности и, как следствие, традиционных пространств действительных чисел, Кантора и Бэра, они больше склонялись к формальной топологии. Обзор работ линии П. Мартин-Лёфа дан в работе [17].

Э. Бишоп развил концепцию конструктивности, которую метко охарактеризовал в разговоре А. Г. Драгалин: пользоваться лишь алгоритмами, но никогда не подтверждать и не отрицать этого [18]. У развивавших концепцию Бишопа [19–21] понятие непрерывности вводится косвенно, через лемму Гейне – Бореля, которая эквивалентна бар-индукции Брауэра. Обзор этой линии Э. Бишопа дан в работе [13]. Понятие подпространства у обеих школ подвержено сильным ограничениям.

Используемое понятие более абстрактно и не зависит от конкретного базиса вычислимости. Оно позволило рассматривать произвольные подпространства, выделенные неконструктивным способом, а также избежать использования выделяющего предиката в конструктивных построениях. При этом производное понятие базы на **Б**-пространстве не удовлетворяет традиционным требованиям к

базе открытых множеств. В частности, пересекающиеся как базовые окрестности объекты могут иметь пустое пересечение как множества, а базовые окрестности как множества могут быть пустыми. Но они наследуются из материнского $\mathbf{Б}_0$-пространства, и это позволяет корректно работать с ними.

Удалось соединить преимущества аппроксимационного (или, что с точки зрения автора то же самое, подхода формальных топологий) и точечного подхода. Определить функции чисто функционально (через элементы) и получить как непрерывность, так и возможность обработки не заданных алгоритмами элементов. Частично роднит с советским конструктивизмом требование, что функционалы заданы программой, но эта программа используется корректно как вызываемый модуль, над которым можно лишь проводить эксперименты, применяя его для различных аргументов (в том числе заданных внешним образом), и совокупность программ не используется как целое.

Единственным существенным ограничением остаётся счётность базиса открытых множеств. Возможность переноса на неотделимые пространства и пространства с несчётной базой требует дополнительных исследований.

3.2. Связь с приложениями и метрологией

В дальнейшем *вычислительными задачами* называются задачи, связанные с действительными числами либо другими хаусдорфовыми пространствами, и они противопоставляются дискретным.

Тонкость в определении регулятора соответствует отсутствию в конструктивном анализе дихотомии $\forall x,\, y(x \le y \vee y \ge x)$ и в её замене на неточное сравнение

$$\forall \varepsilon(\varepsilon > 0 \supset \forall x,\, y(y > x - \varepsilon \vee y < x + \varepsilon)).$$

В вычислительной практике это приводит к тому, что в случае представлений действительных чисел с перекрытием операция равенства оказывается трудно вычислимой, и на этой базе они бракуются. Но с физической точки зрения *точных действительных чисел нет*. В алгоритмах сравнение чисел как равенство приводит к массе трудно выявимых ошибок и неустойчивостей. Надо было бы в каждом случае задумываться, с какой точностью достаточно сравнивать данные, но этому мешает педагогическая проблема: почти везде рациональные числа трактуются как подмножество действительных, что игнорирует их принципиально разную природу.

Соответственно, получаем ещё один важный практический вывод: часто встречающееся возражение против представления действительных чисел системами с перекрытием (пример таких систем см. в работе [22]) – сложность вычисления равенства – с прикладной и научной точек зрения неосновательно.

Полученная эквивалентность конструктивности и чисто топологического понятия является новой. Отдалённой аналогией является теорема: мощность совокупности непрерывных функций на пространстве со счётной базой окрестностей есть континуум. Ранее конструктивность была лишь частным случаем топологии, а теперь она оказывается именно топологией. Это является следствием важного методологического принципа: *любая привязка к конкретным представлениям и структурам данных ограничивает наш взгляд* (вспомним И. Канта, сказавшего: «Мы можем мыслить вещи лишь в пространстве и во времени», что несправедливо для современной логики, математики и физики). И таковыми структурами чаще всего оказываются действительные числа. Поэтому переход к максимально абстрактным представлениям многократно показал в математике и логике свою мощь. Но стоит помнить, что переход от идеальных объектов к реальным в таком случае более труден и может быть многошаговым.

Исходные «неразвитые» концепции П. Мартин-Лёфа и Н. А. Шанина оказались лучшими для обобщения и модификации. Это подтверждает ещё один методологический принцип: *оптимизация уменьшает гибкость, т. е. способность к обобщению и изменению*. В эволюции это означает вымирание хорошо приспособленных специализированных видов при изменении среды.

Первым показал на примере, что одни и те же пространства функций в конструктивной математике могут иметь разные пространства конструктивных функционалов, П. Мартин-Лёф, но он явно не отметил различие его функционалов с функционалами советского конструктивизма. В дальнейшем стало видно, что расхождения могут начаться и на более высоких уровнях. И предлагаемая концепция вычислимости – один из примеров. Она расходится на третьем уровне с концепцией П. Мартин-Лёфа.

С практической точки зрения можно сделать вывод: если программа написана в функциональном стиле без использования некорректных с конструктивной точки зрения операций над действительными числами (равенство, $\ge, \operatorname{sign}(x)$ и т. п.), то она определяет непрерывную функцию и никаких других доказательств не нужно. Заодно

принятые ограничения на вычислимость высших порядков показывают, почему в численных задачах стиль функционального программирования другой, чем в дискретных: например, не используются категорные конструкции. Категорные функционалы – более высокого порядка, чем допускаемые в основе, и они разрушают вычислимость на топологических пространствах, как показано в примере 1. Но их использование как макросов ничего не портит.

Одновременно теоретический и практический вопрос: какие преобразования программ допустимы в качестве макросов в вычислительных задачах? Представляется, что это суперкомпиляция [23]. Но до сих пор она исследовалась прежде всего для дискретных задач.

ЗАКЛЮЧЕНИЕ

Данное исследование порождает серию теоретических вопросов, касающихся использованной концепции вычислимости. Она формально слабее обычной концепции вычислимых функционалов конечного типа. При этом она избавляет от необходимости явно строить модели множеств функционалов конечных типов и позволяет решать достаточно сильные задачи. Здесь нужны дополнительные исследования, в частности, о возможности расширений, сохраняющих топологические свойства и позволяющих вычислять функционалы, потребовавшие внешней перестройки программ.

Результаты п. 2.4 частично докладывались на Тринадцатом Национальном суперкомпьютерном форуме (НСКФ-2024) и опубликованы в его интернет-материалах. Результаты работы были анонсированы в докладе на Смирновских чтениях. Полный текст статьи докладывался на семинаре ИПС РАН 10 апреля 2025 г.

ЛИТЕРАТУРА

1. *Митчелл Дж.* Основания языков программирования. – М. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2010. – 720 с. [*Mitchell, J.C.* Foundations for Programming Languages. – Cambridge: MIT Press, 1996. – 846 p.]
2. *McCarthy, J.* A Basis for a Mathematical Theory of Computation / In: Computer Programming and Formal Systems. Ed. by P. Braffort and D. Hirshberg. – Amsterdam: North-Holland Publishing Co., 1963. – P. 33–70.
3. *Barendregt, H.P.* The Lambda Calculus. Its Syntax and Semantics. Revised Edition. Studies in Logic and the Foundations of Mathematics. – Amsterdam: North-Holland Publishing Co., 1984. – 621 p.
4. *Mitchell, J.C.* Concepts in Programming Languages. – Cambridge: Cambridge University Press, 2003. – 529 p.
5. *Stark, R.W.* LISP, Lore, and Logic. – New York: Springer-Verlag, 1990. – 278 p.
6. *Strachey, C., Wadsworth, C.P.* Continuations: A Mathematical Semantics for Handling Full Jumps, with a Foreword by Christopher P. Wadsworth // Higher-Order and Symbolic Computation. – 2000. – Vol. 13, no. 1/2. – P. 135–152.
7. *Ершов Ю.Л.* Теория А-пространств // Алгебра и логика. – 1973. – Т. 12, № 4. – С. 369–416. [*Ershov, Ju.L.* Teorija A-prostranstv // Algebra i logika. – 1973. – Vol. 12, no. 4. – P. 369–416. (In Russian)]
8. *Ершов Ю.Л.* Теория нумераций. – М.: Наука, 1977. – 416 с. [*Ershov, Ju.L.* Teorija numeracij. – Moscow: Nauka, 1977. – 416 s. (In Russian)]
9. *Martin-Löf, P.* Notes on Constructive Mathematics. – Stockholm: Almqvist & Wiksell, 1970.
10. *Lacombe, A.* Quelques procedes de definition en topologie recursive / In: Constructivity in Mathematics. – Amsterdam: North-Holland, 1959. – P. 129–158.
11. *Bridges, D., Ishihara, H., Rathjen, M., Schwichtenberg, H.* Handbook of Constructive Mathematics. – Cambridge: Cambridge University Press, 2023.
12. *Шанин Н.А.* Конструктивные вещественные числа и конструктивные функциональные пространства // Проблемы конструктивного направления в математике. 2. Конструктивный математический анализ. Сборник работ. Тр. МИАН СССР. – 1962. – Т. 67. – С. 15–294. [*Shanin, N.A.* Konstruktivnye veshhestvennye chisla i konstruktivnye funkcional'nye prostranstva // Problemy konstruktivnogo napravlenija v matematike. 2. Konstruktivnyj matematicheskij analiz. Sbornik rabot. Tr. MIAN SSSR. –1962. – Vol. 67. – P. 15–294. (In Russian)]
13. *Kawai, T.* Bishop Metric Spaces in Formal Topology / In: Handbook of Constructive Mathematics. Ed. by D. Bridges, H. Ishihara, M. Rathjen, H. Schwichtenberg. – Cambridge: Cambridge University Press, 2023. – P. 395–425.
14. *Кушнер Б.А.* Лекции по конструктивному математическому анализу. – М.: Наука, 1973. – 448 с. [*Kushner, B.A.* Lekcii po konstruktivnomu matematicheskomu analizu. – Moscow: Nauka, 1973. – 448 s. (In Russian)]
15. *Kreisel, G., Troelstra, A. S.* Formal Systems for Some Branches of Intuitionistic Analysis // Ann. Math. Logic. – 1970. – Vol. 1, no. 3. – P. 229–387.
16. *Гейтинг А.* Интуиционизм: Математическое введение.– М: URSS, 2010. – 160 с. [*Heyting, A.* Intuitionism: an introduction. – Amsterdam : North-Holland Pub. Co., 1966. – 136 p.]
17. *Ciraulo, F.* Subspaces in Pointfree Topology: Towards a New Approach to Measure Theory / In: Handbook of Constructive Mathematics. Ed. by D. Bridges, H. Ishihara, M. Rathjen, H. Schwichtenberg. – Cambridge: Cambridge University Press, 2023. – P. 426–444.
18. *Bishop, E.* Foundations of Constructive Analysis. – New York: McGrawHill, 1967.
19. *Kawai, T.* Localic Completion of Uniform Spaces // Log. Meth. Comput. Sci. – 2017. – Vol. 13, no. 3. – Art. no. 22. – P. 1–39.
20. *Kawai, T.* Point-Free Characterisation of Bishop Compact Metric Spaces // J. Log. Anal. – 2017. – Vol. 9, no. 5. – P. 1–30.
21. *Kawai, T.* A Point-Free Characterisation of Bishop Locally Compact Metric Spaces // J. Log. Anal. – 2017. – Vol. 9, no. c2. – P. 1–41.
22. *Непейвода Н.Н., Григоревский И.Н., Лилитко Е.П.* О представлении действительных чисел // Программные системы: теория и приложения: электрон. научн. журн. – 2014. – Т. 5, № 4 (22). – С. 105–121. – URL http://psta.psiras.ru/read/psta2014_4_105-121.pdf [*Nepejvoda, N.N., Grigorevsky, I.N., Lilitko, E.P.* New Representation of Real Numbers // Program

Systems: Theory and Applications. – 2014. – Vol. 5, No. 4 (22). – P. 105–121. – URL http://psta.psiras.ru/read/psta 2014_4_105-121.pdf (In Russian)]

23. *Климов, А.В.* Суперкомпиляция: основные принципы и базовые понятия // Препринты ИПМ им. М.В. Келдыша. – 2018. – № 111. – 36 с. – DOI: 10.20948/prepr-2018-111 – URL: http://library.keldysh.ru/preprint.asp?id=2018-111 [*Klimov, A.V. and Romanenko, S.A.* Supercompilation: Main Principles and Basic Concepts // Preprints of the Keldysh Institute of Applied Mathematics, Moscow. – 2018. – no. 111. – DOI: 10.20948/prepr-2018-111. – URL: http://library.keldysh.ru/preprint.asp?id=2018-111. (In Russian)]

Статья представлена к публикации членом редсовета академиком РАН С. Н. Васильевым.

Поступила в редакцию 13.05.2025,
после доработки 25.08.2025.
Принята к публикации 02.10.2025.

Непейвода Николай Николаевич – д-р физ.-мат. наук, Институт программных систем РАН, г. Переславль-Залесский, ✉ nnn@nnn.botik.ru
ORCID iD: https://orcid.org/0000-0001-8845-7627

CONTINUITY AS COMPUTABILITY

N. N. Nepeivoda

Ailamazyan Program Systems Institute, Russian Academy of Sciences, Pereslavl-Zalessky, Russia

✉ nnn@nnn.botik.ru

Abstract. The relationship between computability and continuity is studied. Computability over an arbitrary initial basis of data types and functions (a base) is considered using McCarthy recursive schemata and strongly typed operators of finite types. In this case, computable operators are proved to be strongly continuous in the Baire sense: for parameter functions with any value of the other arguments, it is possible to find a finite collection of their values that uniquely determines the result. Based on relative computability, an approach to constructive topology is developed within which the pointwise approach (an element is a fundamental sequence of neighborhoods) and the approximation approach of abstract topology (a function over topological spaces is a neighborhood relation) are equivalent. The concept of **B**-spaces is formulated, allowing one to constructivize separable spaces with a countable base of neighborhoods. The continuity of pointwise constructive functions of **B**-spaces and their transformability into neighborhood relations are proved. The equivalence of the concepts of computability relative to a certain base and continuity is established. The concept of a relatively constructive function is formulated: such a function transforms each element into an element constructible relative to its argument and a fixed base. Its equivalence to the concept of a countably continuous function formed by the union of a countable family of functions continuous on subspaces is established. Since any separable space with a countable base can be described as a **B**-space, this result contains no constructive restrictions. The connection between the proposed approach and other approaches to constructive topology is discussed.

Keywords: relative computability, separability, countable base, pointwise spaces, abstract topology, approximation approach, Baire continuity, continuity on subspaces.