УДК 004.42 + 519.681.2 + 519.682.1

О логических и алгебраических основаниях классификации формальной семантики программ

Шилов Н.В. (Институт систем информатики СО РАН)

Существует определённый разрыв в уровне математической подготовки программистов-теоретиков и программистов-практиков: первые сильны в науке (абстрактной алгебре и математической логике), а вторые — в искусстве (разработке программных систем). В статье представлен достаточно простой подход к алгебраическим и логическим основаниям формальной семантики программ, ориентированный на инженеров-программистов с элементарными знаниями в абстрактной алгебре и математической логике. Для этого в статье объясняются основы *операционной*, *денотационной*, *аксиоматической* семантики, а также семантики *второго порядка* на примере "эзотерического" языка, синтаксически похожего на язык программирования.

Ключевые слова: Формальная семантика программ, операционная семантика, денотационная семантика, аксиоматическая семантика, семантика второго порядка, эзотерические языки программирования

1. Введение

Мир очень изменился за последние 30 лет. Век информационных технологий сменил век достижений технологий обычных, всюду мобильная связь, компьютеры и Интернет. Но мы обращаемся с "умной" техникой, как с волшебным реквизитом, забывая, что этот "ум" — программы, написанные людьми. Как происходят вычисления в конкретном компьютере? (Ведь в вычислительных машинах нет действительных чисел, даже и рациональных почти нет; есть, к примеру, типы INTEGER и REAL.) И ведь ошибки могут быть не только в вычислениях. Могут ошибаться специалисты, создающие математическую модель процесса или сооружения. Могут быть ошибки в алгоритме решения задачи. Могут быть ошибки в программной реализации алгоритма. Используя выросшие, как на дрожжах, информационные технологии, мы делаем выводы о реальности — а насколько эти выводы верны?

Аварии в энергетике и на транспорте, обрушения зданий и сооружений — похоже, XXI век становится веком техногенных катастроф. И во многом потому, что человечество слиш-

ком небрежно относится к тому аппарату, с помощью которого проектирует свою мощную технику и управляет ею. Программными ошибками были вызваны серьёзные аварии на заводах, катастрофы с самолётами, гибель больных при лечении. Ошибки в программном обеспечении систем управления работой электростанций всех типов, трубопроводов, энергосистем, водоочистных сооружений приводили (и не раз ещё приведут) к очень серьёзным последствиям [1, 2].

Одно из направлений повышение надёжности программного обеспечения — разработка и практическое применение математически точных методов описания "смысла" программ (или, как об этом говорят специалисты, "формальной семантики"). Минуло уже почти полвека с момента публикации Робертом В. Флойдом статьи *Assigning Meanings to Programs* [7], в которой была сделана первая (и удачная) попытка разработать такую семантику. Прошло почти 20 лет после публикации Дэвидом А. Шмидтом призыва к академической общественности [12] сделать формальную семантику программ понятной и доступной широкому кругу инженеров-программистов. За эти годы в академических кругах было разработано множество вариантов формальной семантики, но сложилась такая ситуация с использованием этих формализмов в практике программной инженерии, которую Д.А. Шмидт охарактеризовал словами[1]: "Значительно больше экспертов-теоретиков, чем практикующих программистов".

А между тем академическая активность в попытках обобщить, сделать доступной и полезной формальную семантику, представить свой "взгляд с высоты" нарастает, в неё включаются всё новые учёные.

Например, Петер Д. Мозес в 2001 г. пропагандировал свой подход к многообразию формальных семантик и их использованию [9]. В опубликованной аннотации доклада он написал[2]: "Работа даёт обзор основных парадигм семантики языков программирования... Работа рассчитана на самый широкий круг специалистов по информатике. Она не требует предварительного знакомства с техническими подробностями какой-либо определённой парадигмы, хотя предполагает понимание основных принципов формальной семантики."

В следующем 2002 г. известный учёный Патрик Кузо опубликовал журнальную статью [6], в которой построил иерархию формальных семантик. Эта иерархия включает

[1] more experts, but fewer general users (Перевод Шилова Н.В.)

[2] This paper surveys the main frameworks available for describing the dynamic semantics of programming languages. ... The paper is intended to be accessible to all computer scientists. Familiarity with the details of particular semantic frameworks is not required, although some understanding of the general concepts of formal semantics is assumed. (Перевод Шилова Н.В.)

семантику большого шага, семантику по Плоткину, “демоническую” семантику по Смиту (Smyth’s demoniac) и “ангельскую” реляционную семантику по Хоару (Hoare’s angelic relational), недетерминированную денотационную и просто денотационную семантику по Скотту, обобщённую семантику второго порядка Дейкстры, а также обобщённую семантику частичной/тотальной корректности по Хоару и пр. Все семантики, рассмотренные в этой обширной статье (56 страниц), представлены в терминах[3] “неподвижных точек, связи между семантиками — посредством связок Галуа, каждая из семантик вычисляется посредством некоторой абстрактной интерпретации по отношению к более конкретной семантике с использованием теорем Клини или Тарского о неподвижной точке”.

Но наука не стоит на месте, и одновременно с попытками бросить взгляд с высоты на картину, которую представляют сложившиеся к XXI веку семантические формализмы, продолжают появляться новые семантические парадигмы. Например, так называемая теоретико-игровая семантика Самсона Абрамского и Чих-Хао Лук Онга [5]. Авторы этой семантики утверждают, что этот формализм готов для моделирования и композиционной верификации программного обеспечения.

Но вот на фоне этой бурной академической деятельности по систематизации “старых” и разработке “новых” формальных семантик Дэвид Л. Парнас в январе 2010 г. опубликовал статью “Реальное переосмысление "формальных методов“” [10]. По его мнению, проблема состоит в следующем[4]: “Мы должны подвергнуть сомнению предположения, лежащие в основе хорошо известных современных формальных методов разработки программного обеспечения, чтобы понять, почему они не получили широкого распространения, и что в них следует изменить.” Это пожелание относится и к формальной семантике.

Настоящая статья является расширенным и переработанным вариантом сообщения [4].

2. Привести ум в порядок

Заголовок этой части статьи навеян известным афоризмом Михаила Васильевича Ломоносова: “Математику уж затем любить надо, что она ум в порядок приводит”. Это не случайно, у нас есть веская причина прибегнуть к его авторитету: формальные методы в

[3]... all the semantics are presented in a uniform fixpoint form and the correspondences between these semantics are established through composable Galois connections, each semantics being formally calculated by abstract interpretation of a more concrete one using Kleene and/or Tarski fixpoint approximation transfer theorems. (Перевод Шилова Н.В.)

[4]We must question the assumptions underlying the well-known current formal software development methods to see why they have not been widely adopted and what should be changed. (Перевод В. Кулямина, см. `http://citforum.ru/SE/quality/fm_rethinking/`.)

программировании (и формальная семантика в том числе) выполняют ту же роль — "ум в порядок приводят" в компьютерных науках. Разумеется, математика имеет множество практических приложений, и формальные методы тоже должны найти своё применение в практике программной инженерии. Но роль формальных методов и математики для образования ума тоже имеет немалое значение.

Однако тут мы наталкиваемся на хроническую аллергию программистов-практиков (студентов и инженеров) к формальным методам: программисты-практики считают формальные методы слишком "рафинированными" в теории и неэффективными на практике. На наш взгляд, основа этой устойчивой аллергии — отсутствие элементарных (как в начальной школе) курсов по формальным методам. Общепризнано, что обучение арифметике нельзя начинать с аксиоматики Пеано, сначала решают задачи о сливах, яблоках, карандашах и т.д. Никто не предлагает сразу учить манипуляциям с доказательствами и не предлагает учащимся показать, что в арифметике Пеано доказуемо утверждение

$$\forall x\ \forall y\ \forall z:\ (y \le z \to ((x + y) - z) = (x + (y - z))),$$

для начала учат, как решать задачки вроде следующей: Пете дали 5 яблок, а он отдал 2 яблока Ване; сколько яблок осталось у Пети?[5]

По нашему мнению, аллергия на формальные методы во многом объясняется тем, что эксперты не заботятся о начальном уровне обучения формальным методам. Курсы по формальным методам обычно начинаются с введения понятий *абстрактная машина состояний*, *преобразователь предикатов*, *логический вывод*, *операционная семантика*, *денотационная семантика*, *аксиоматическая семантика* — и никаких элементарных пояснений вроде яблок и карандашей... И никакого простого инструментального средства (наподобие счётных палочек) для знакомства с этими понятиями, а сразу полные версии экспериментальных верификаторов моделей (model checkers), систем поддержки доказательства (proof assistances) или автоматических верификаторов (theorem provers).

Поэтому в следующей части статьи мы приведём пример для первоначального ознакомления с понятиями операционная, денотационная, аксиоматическая семантика и семантика второго порядка; в заключительной части обсудим, как от этого начального уровня перейти к "настоящим" формальным семантикам, а также приведём вариант простого инструментального средства поддержки автоматизации работы с этими семантиками для простых программ.

[5]Между прочим, если Вы думаете, что правильный ответ "3", то ошибаетесь: правильный ответ — "не менее 3".

⟨program⟩ ::= ⟨assignment⟩ | (⟨program⟩) | ⟨program⟩ ; ⟨program⟩ |
| if ⟨condition⟩ then ⟨program⟩ else ⟨program⟩ |
| while ⟨condition⟩ do ⟨program⟩
⟨assignment⟩ ::= ⟨variable⟩:= ⟨expression⟩
⟨condition⟩ ::= ⟨(in)equality⟩ | (⟨condition⟩) | ¬⟨condition⟩ |
| ⟨condition⟩ ∧ ⟨condition⟩ | ⟨condition⟩ ∨ ⟨condition⟩
⟨(in)equality⟩ ::= ⟨expression⟩ = ⟨expression⟩ |
| ⟨expression⟩ < ⟨expression⟩ | ⟨expression⟩ ≤ ⟨expression⟩ |
| ⟨expression⟩ > ⟨expression⟩ | ⟨expression⟩ ≥ ⟨expression⟩
⟨expression⟩ ::= ⟨constant⟩ | ⟨variable⟩ | (⟨expression⟩) |
| ⟨expression⟩ + ⟨expression⟩ | ⟨expression⟩ − ⟨expression⟩ |
| ⟨expression⟩ ∗ ⟨expression⟩

Рис. 1. BNF определение синтаксиса языка TEL

Однако здесь необходимо сказать, что всё-таки есть вполне удачные примеры учебных материалов по формальным методам для начального и пользовательского уровня. Например, в 2010 г. в издательстве БХВ-Петербург вышла учебная монография Юрия Глебовича Карпова "*Model Checking: верификация параллельных и распределённых программных систем*". Эта книга является прекрасным руководством по верификатору моделей SPIN, она учит пользоваться этим инструментальным средством и понимать основы теории верификации конечных моделей последовательно,шаг за шагом, на примере решения увлекательных головоломок.

3. Эзотерический язык TEL

Язык программирования — это любой искусственный язык для автоматической обработки данных на вычислительной машине. Всякий (искусственный или естественный) язык обычно характеризуются своим *синтаксисом*, *семантикой* и *прагматикой*. Синтаксис — это правописание языка. Семантика — это правила придания смысла синтаксически правильным выражениям этого языка. А прагматика — это практика, методы, способы, пути использования грамматически правильных осмысленных (т.е. имеющих семантику) выражений языка.

Так вот, игрушечный язык TEL (Toy Esoteric Language) вообще не является языком программирования, т.к. не предназначен для обработки данных на компьютере. Он пред-

назначен (это его прагматика) для первоначального знакомства с разными видами формальной семантики. А вот синтаксис языка TEL чрезвычайно похож на синтаксис языка программирования. Контекстно-свободный синтаксис языка TEL дан на рис. 1. (Переменные и константы в этом определении — это идентификаторы и целые числа в произвольной фиксированной системе счисления). Однако синтаксис TEL проще объяснить на следующем простом примере правильного выражения, представленном на рис. 2 (которое мы обозначим S для дальнейшего использования).

```
if z<0 then z:= -1
        else (x:= 0 ; y:= 0 ;
              while y≤z do (y:= y + 2*x + 1 ; x:= x + 1) ;
              x:= x - 1)
```

Рис. 2. Пример правильно построенного TEL-выражения (“программы”)

Неформально говоря, правильные выражения языка TEL — это как бы “программы”, построенные из чисел, переменных, арифметических выражений, логических выражений, операторов присваивания при помощи разделителя “;”, условий “if-then-else” и циклов “while-do”.

Неформальная семантика языка TEL может быть описана следующим образом. Так как правильные выражения языка TEL выглядят как программы, то их можно изображать графически в виде блок-схем. (Например, на рис. 3 представлена блок-схема правильного выражения S.) Всякая блок-схема — это граф, вершины которого — операторы присваивания и логические выражения. Договоримся измерять “длину” пути по блок-схеме количеством операторов присваивания на этом пути (т.е. логические выражения не идут в счёт). Тогда условимся считать “смыслом” (семантикой) каждого правильного выражения целое (натуральное) число, равное длине кратчайшего пути по графу блок-схемы этого выражения от начала до конца. (В частности, как это видно из рис. 3, семантика правильного выражения S есть число 1.)

4. Операционная семантика: исполнимая машина состояний

Операционная семантика основана на *трансляции* правильно построенных выражений в *машину состояний* (“механическую” процедуру), действия которой преобразуют её

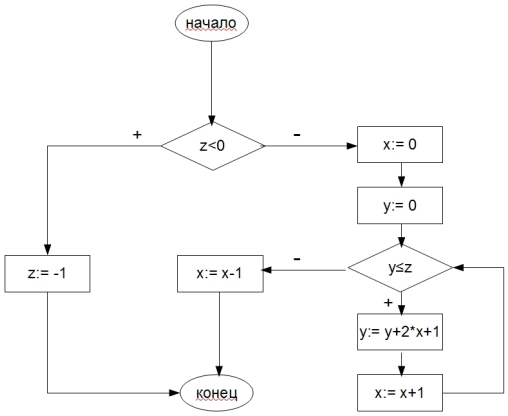


Рис. 3. Блок-схема правильного выражения S

- $F_{\langle \texttt{assignment} \rangle} = 1$;
- $F_{(\beta)} = F_\beta$;
- $F_{\beta;\gamma} = F_\beta + F_\gamma$;
- $F_{\texttt{if } \phi \texttt{ then } \beta \texttt{ else } \gamma} = \min\{F_\beta, F_\gamma\}$;
- $F_{\texttt{while } \phi \texttt{ do } \beta} = 0$,

Рис. 4. Алгоритм трансляции TEL-выражений в арифметические выражения

“состояния”. Например, для языка TEL в качестве такой машины можно принять “арифмометр” — алгоритм вычисления значения арифметического выражения, построенного из целых чисел, операций сложения и выбора минимального из двух значений. Состояния такой машины при вычислении какого-либо такого выражения — это все возможные промежуточные и заключительные стадии вычисления значения этого арифметического выражения. А в качестве алгоритма трансляции — рекурсивный алгоритм F, представленный на рис. 4, преобразующий выражения TEL в арифметические выражения; в описании этого алгоритма β, γ и ϕ — произвольные правильные выражения и условие языка TEL. Для произвольного правильного выражения α этого языка F_α — арифметическое выражение, построенное из целых чисел, операций сложения и выбора минимального из двух значений. Операционная семантика правильного выражения α языка TEL — это значение выражения F_α, которое мы будем обозначать $val(F_\alpha)$.

Например, для правильного TEL-выражения S имеем:

$F_{\texttt{if z<0 then z:=}-1\texttt{ else(x:=0;y:=0; while y}\leq\texttt{z do(y:=y+2*x+1;x:=x+1);x:=x}-1)} =$

$$= \min\{F_{\texttt{z:=-1}},\ F_{\texttt{x:=0;y:=0; while y}\leq\texttt{z do(y:=y+2*x+1;x:=x+1);x:=x-1}}\} =$$
$$= \min\{1,\ F_{\texttt{x:=0}} + F_{\texttt{y:=0; while y}\leq\texttt{z do(y:=y+2*x+1;x:=x+1);x:=x-1}}\} =$$
$$= \min\{1,\ 1 + F_{\texttt{y:=0}} + F_{\texttt{while y}\leq\texttt{z do(y:=y+2*x+1;x:=x+1);x:=x-1}}\} =$$
$$= \min\{1,\ 1+1+\dots\}.$$

Таким образом, выражению S языка TEL сопоставлено арифметическое выражение $\min\{1, 1+1+\dots\}$. Наш алгоритм-аримометр, запущенный на этом выражении, выполнит несколько сложений и одну операцию минимизации и остановится (очевидно) с заключительным значением 1; следовательно, операционная семантика $val(F_S)$ равна 1. Это заключительное значение и все промежуточные выражения, которые получались во время этого вычисления, — состояния нашей машины, которые она проходит при вычислении этого значения $val(F_\alpha)$.

Здесь уместно заметить, что наличие семантики для языка позволяет определить эквивалентность выражений: эквивалентные выражения — это выражения, у которых одинаковая семантика. (Разумеется, если определено несколько семантик, то отношения эквивалентности на выражениях могут не совпадать.) Так, для определённой выше операционной семантики языка TEL эквивалентным будут все выражения, в которых совпадают длины кратчайших (по числу присваиваний) путей от начала до конца в блок-схемах. В частности, наше выражение S (в рассмотренной операционной семантике) эквивалентно любому выражению, в котором есть путь от начала до конца по графу его блок-схемы, содержащий ровно одно присваивание, и всякий путь от начала до конца содержит хотя бы одно присваивание.

5. Денотационная семантика: алгебра для вычислений

Алгебра — это множество элементов и операций над ними. Например, множество натуральных чисел $\mathbb{N}$ с нуль-местными операциями (константами) 0 и 1, бинарными операциями “+” и “–” — это алгебра. А то же множество $\mathbb{N}$ с теми же константами 0 и 1, операцией “+”, но с бинарной операцией min вместо “–” — это уже другая алгебра, т.к. использует другой набор операций.

Денотационная семантика — это определение семантики языка в терминах подходящей алгебры путём согласованного сопоставления каждому правильному выражению языка элемента алгебры, а каждому конструктору правильных выражений — алгебраической

- $[\![\langle \texttt{assignment} \rangle]\!] = 1$;
- $[\![(\alpha)]\!] = [\![\alpha]\!]$;
- $[\![\alpha; \beta]\!] = [\![\alpha]\!] + [\![\beta]\!]$;
- $[\![if\ \phi\ then\ \alpha\ else\ \beta]\!] = \min\{[\![\alpha]\!]\ [\![\beta]\!]\}$;
- $[\![while\ \phi\ do \ldots]\!] = 0$.

Рис. 5. Определение денотационной семантики языка TEL

операции; часто такое сопоставление (функцию) обозначают посредством двойных квадратных скобок $[\![\ ]\!]$, а элемент алгебры или операция алгебры, сопоставленные функцией $[\![\ ]\!]$ правильному выражению или конструктору выражений, называют *денотантом* этого выражения или, соответственно, *денотацией* конструкции. Например, для определения денотационной семантики нашего игрушечного языка TEL возьмём следующую алгебру натуральных чисел $\mathbb{N}$ с константами 0, 1, унарной *тождественной* операцией $\lambda x.x$, бинарными операциями сложения (+) и минимизации (min), а функцию $[\![\ ]\!]$ определим так, как это показано на рис. 5

В частности, для правильного выражения S имеем:

$[\![$`if z<0 then z:= -1`
`else (x:= 0 ; y:= 0 ;`
`while y≤z do (y:= y + 2*x + 1 ; x:= x + 1) ;`
`x:= x - 1)`$]\!] =$

$= [\![$`if` $-$ `then...else...`$]\!]$
$([\![$`z:= -1`$]\!], [\![$`x:= 0 ; y:= 0 ;`
`while y≤z do (y:= y + 2*x + 1 ; x:= x + 1);`
`x:= x - 1`$]\!]) =$

$= \min\{\ 1\ , [\![;]\!]([\![$`x:= 0`$]\!], [\![$`y:= 0 ;`
`while y≤z do (y:= y + 2*x + 1 ; x:= x + 1);`
`x:= x - 1`$]\!])\} =$

$= \min\{\ 1\ , 1 + [\![;]\!]([\![$`y:= 0`$]\!],$
$[\![$`while y≤z do (y:= y + 2*x + 1 ; x:= x + 1) ;`
`x:= x - 1`$]\!])\} =$

$= \min\{\ 1\ , 1 + (1 + \ldots)\} = 1.$

Совпадение $val(F_S)$ и $[\![S]\!]$ не случайно: операционная и денотационная семантики TEL

Присваивание (Assignment): $\dfrac{}{1{\leq}x{:=}t{\leq}1}$	Цикл (Loop): $\dfrac{}{0{\leq}\texttt{while}...\texttt{do}...{\leq}0}$
Блок (Block): $\dfrac{m{\leq}\alpha{\leq}n}{m{\leq}(\alpha){\leq}n}$	Ослабление ограничений (Stretching): $\dfrac{m'{\leq}\alpha{\leq}n'}{m{\leq}\alpha{\leq}n}$, $m \leq m'$, $n' \leq n$
Композиция (Composition): $\dfrac{m'{\leq}\alpha{\leq}n' \qquad m''{\leq}\beta{\leq}n''}{m{\leq}\alpha;\beta{\leq}n}$, $m = m' + m''$, $n = n' + n''$	
Then-ветвление: $\dfrac{m{\leq}\alpha{\leq}n \qquad m{\leq}\beta{\leq}\infty}{m{\leq}\texttt{if}...\texttt{then}\ \alpha\ \texttt{else}\ \beta{\leq}n}$	Else-ветвление: $\dfrac{m{\leq}\alpha{\leq}\infty \qquad m{\leq}\beta{\leq}n}{m{\leq}\texttt{if}...\texttt{then}\ \alpha\ \texttt{else}\ \beta{\leq}n}$

Рис. 6. Аксиоматическая семантика языка TEL

тесно связаны.

Утверждение 1. *$val(F_\alpha) = [\![\alpha]\!]$ для любого правильного выражения α языка TEL.*

Доказательство легко выполнить индукцией по структуре выражения α. ■

Обычно, когда можно доказать свойство "совпадания" двух семантик (наподобие равенства в утверждении 1), говорят о *(взаимной) корректности*. Поэтому мы будем называть утверждение 1 теоремой о взаимной корректности операционной и денотационной семантики языка TEL. Заметим однако, что в общем случае две семантики не обязательно будут взаимно корректны и могут быть связаны друг с другом более сложным образом или даже вообще не связаны.

6. Аксиоматическая семантика: синтаксически управляемое доказательство

Аксиоматическая семантика — это синтаксическое исчисление для вывода ("доказательства") новых утверждений ("теорем") по правилам вывода из заранее определённых аксиом ("фактов"). (Для того, чтобы пользоваться аксиоматической семантикой, надо иметь некоторые навыки построения доказательства в аксиоматических системах.)

В частности, аксиоматическая семантика языка TEL имеет дело с утверждениями вида $m \leq \alpha \leq n$, где m — натуральное число, α — правильное выражение языка TEL, а n — или натуральное число такое, что $m \leq n$, или символ бесконечности ∞. Аксиоматическая семантика для языка TEL представлена на рис. 6 в виде аксиоматической системы. Будем

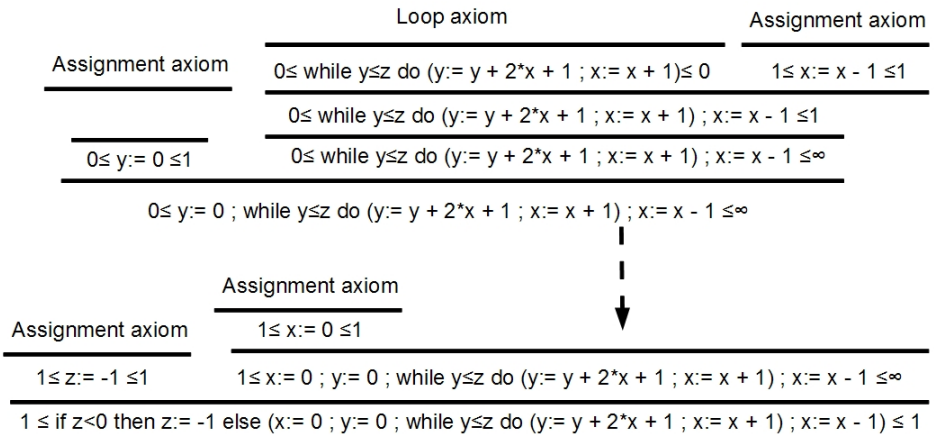


Рис. 7. Пример доказательства в аксиоматической семантике TEL

говорить, что утверждение $m \le \alpha \le n$ доказуемо и писать $\vdash m \le \alpha \le n$, если существует доказательство этого утверждения в этой аксиоматической системе.

Будем говорить, что утверждение $m \le \alpha \le n$ верно (или истинно) и писать $\models m \le \alpha \le n$, если $m \le [\![\alpha]\!] \le n$. Говорят, что аксиоматическая семантика *непротиворечива*, если любое доказуемое утверждение верно; говорят, что аксиоматическая семантика *полна*, если всякое верное утверждение доказуемо. Пример верного и доказуемого утверждения — это $1 \le S \le 1$: его истинность следует из того, что $[\![S]\!] = 1$, а доказуемость подтверждается доказательством, приведённым на рис. 7.

Совпадение $\models 1 \le S \le 1$ и $\vdash 1 \le S \le 1$ не случайно.

Утверждение 2. *Для любых $n, m \in \mathbb{N} \cup \{\infty\}$ и правильного выражения α языка TEL*

$$\models m \le S \le n \;\Leftrightarrow \vdash m \le S \le n.$$

Доказательство непротиворечивости легко получить индукцией по высоте дерева вывода, а доказательство полноты — индукцией по структуре правильного выражения. ■

Заметим опять же, что в общем случае аксиоматическая семантика не обязательно будет полной; она также может быть противоречивой (то есть некоторые доказуемые утвер-

ждения будут неверными).

7. Семантика второго порядка: преобразователи предикатов

Пусть D — произвольное множество, а $k \geq 0$ — произвольное целое число. Тогда k-местный предикат на множестве D — это произвольное подмножество D^k. В частности, одноместный предикат — это некоторое множество значений (элементов) D. Очевидно, что множество всех одноместных предикатов — это множество 2^D всех подмножеств D.

Преобразователь (одноместных) предикатов (на D) — это произвольная функция $F : 2^D \to 2^D$, то есть функция, которая преобразует "входной" предикат $S \subseteq D$ в "выходной" предикат $F(S) \subseteq D$.

Пусть D — некоторое множество ("область"). Элементы D и векторы с компонентами из D (то есть элементы пространства $D_1 = \bigcup_{k\geq 1} D^k$) называют элементами *первого порядка* над D. Подмножества D_1 (и, в частности, все функции из D^m в D^n) называются множествами (предикатами) первого порядка над D (функциями соответственно). Всякая "теория", изучающая или описывающая свойства элементов первого порядка, называется теорией первого порядка области D.

Подмножества D_1 и векторы, компонентами которых являются подмножества D_1 (то есть элементы пространства $D_2 = \bigcup_{k\geq 1}(2^{D_1})^k$), называют элементами *второго порядка* над D. Подмножества D_2 и функции из D_2 в D_2 называются множествами (предикатами) второго порядка и функциями второго порядка (*функционалами*) над D. Всякая "теория", изучающая или описывающая свойства элементов второго порядка, называется теорией второго порядка области D.

(Подобным образом можно определить элементы *элементы*, *предикаты*, *функции* и *теории* третьего и высших порядков.)

Рассмотрим с точки зрения классификации теорий по порядкам такие учебные предметы, как элементарная алгебра, изучаемая в старших классах средней школы, и математический анализ, изучаемый на первых курсах вузов.

Элементарная алгебра занимается решением (систем) уравнений и неравенств в вещественных числах. Эти уравнения заданы многочленами с параметрическими коэффициентами, которые обозначают опять же вещественные числа. Например, типичной задачей элементарной алгебры является нахождение всех вещественных корней квадратного уравне-

ния с вещественными коэффициентами $a \times x^2 + b \times x + c = 0$. В школьном курсе доказывается, что для любых вещественных значений коэффициентов уравнение

$$a \times x^2 + b \times x + c = 0$$

имеет решение в вещественных числах тогда и только тогда, когда

$$b^2 - 4 \times a \times c \geq 0,$$

то есть, что предложение

$$\forall a\ \forall b\ \forall c : \big(b^2 - 4 \times a \times c \geq 0 \leftrightarrow \exists x : (a \times x^2 + b \times x + c = 0)\big).$$

принадлежит теории первого порядка поля вещественных чисел. Можно предположить, что курс элементарной алгебры изучает теорию первого порядка поля вещественных чисел.

Математический анализ занимается изучением свойств непрерывных и дифференцируемых функций вещественной переменной. Примером может служить теорема о промежуточном значении непрерывной функции на замкнутом отрезке, авторство которой приписывается К.Т.В. Веерштрассу, Б. Больцано и О.Л. Коши (см. рис. 8): для любой функции f, непрерывной на отрезке вещественных чисел $[a..b]$, для любого числа $y \in [f(a)..f(b)]$ найдётся такое число $x \in [a..b]$, что $f(x) = y$.

Вспомним, что всякая вещественная функция — это множество пар чисел. Поэтому теорема может быть сформулирована в терминах логики второго порядка над полем вещественных чисел следующим образом:

$$\forall f \in 2^{R \times R}\ \forall a \in R\ \forall b \in R \forall c \in R\ \forall d \in R\ \forall y \in R :$$
$$(C(f,a,b)\ \wedge\ (a,c) \in f\ \wedge\ (b,d) \in f\ \wedge\ c \leq y \leq d\ \rightarrow$$
$$\rightarrow\ \exists x \in R : (a \leq x \leq b\ \wedge\ (x,y) \in f)),$$

где C — трёхместное отношение “быть непрерывной вещественной функцией на отрезке”. Поэтому можно сказать, что курс математического анализа формализуем как теория второго порядка поля вещественных чисел.

Функциональный анализ изучает свойства функционалов, то есть отображений, которые сопоставляют функциям некоторые числа или функции. Часто изучаются свойства линейных операторов, к которым относятся, например, дифференцирование и интегрирование вещественных функций. Поэтому часть функционального анализа, изучающая теорию линейных операторов на функциях вещественной переменной, может быть формализована как теория третьего порядка поля вещественных чисел.

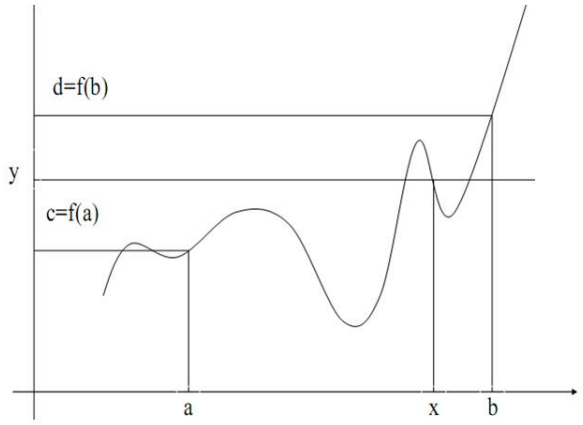


Рис. 8. Иллюстрация к теореме Веерштрасса – Больцано – Коши

Но вернёмся к формальной семантике: семантика второго порядка — это сопоставление каждому синтаксически правильному выражению некоторого преобразователя предикатов (первого порядка) над некоторой "семантической" областью D, то есть функции $F : 2^D \to 2^D$, которая преобразует "входной" $S \subseteq D$ предикат — в "выходной" предикат $F(S) \subseteq D$. В частности, для любого выражения α языка TEL определим преобразователь предикатов PT_α на натуральных числах $\mathbb{N}$ следующим образом: для любого множества натуральных чисел S пусть $PT_\alpha(S)$ — это множество таких натуральных чисел $n \in \mathbb{N}$, что для любого правильного выражения β языка TEL, если $[\![\beta]\!] = n$, то $[\![\alpha;\beta]\!] \in S$. Следующее утверждение устанвливает связь семантики второго порядка с денотационной семантикой языка TEL и является очевидным.

Утверждение 3. *Для любого выражения α языка TEL и множества натуральных чисел $S \subseteq \mathbb{N}$ имеет место равенство $PT_\alpha(S) = S \ominus [\![\alpha]\!]$, где операция $\ominus$ — операция поэлементного вычитания без перехода через 0.*

8. Заключение

Итак, в чем же польза формальной семантики языков программирования? С одной стороны, автор утверждает, что изучение её ум в порядок приводит, но тут же пишет, что хороших курсов для начального обучения нет. Кроме того, автор однозначно согласны с мнением Дэвида Парнаса[6], что "исключительно редкое применение формальных методов в производственной разработке ПО только подчеркивает, что использование таких методов не стало установившейся практикой". О чём же тогда эта статья, о какой пользе?

[6] Applications of formal methods to industrial practice remain such exceptions that they confirm that the use of formal methods is not common practice. (Перевод В. Кулямина, см. `http://citforum.ru/SE/quality/fm_rethinking/`.)

Она — всё-таки о пользе формальных методов (формальной семантики в частности) для образования и дисциплины ума программистов. Но для того, чтобы эту пользу ощутить, необходимо делать следующее.

Нельзя оглоушивать новичков непонятными и непривычными терминами, теориями и методами, обещая, что если они одолеют все эти новые понятия, то в результате создадут полностью автоматический универсальный промышленный верификатор программ.

Надо подготовить систему курсов по формальным методам, начиная с элементарных курсов с игрушечными примерами и простыми (до тривиальности) объяснениями, что есть что в формальных методах (как, например, это сделано в настоящей статье с понятиями денотационная и аксиоматическая семантика).

Нельзя новичков учить пользоваться современными экспериментальными системами, поддерживающими конкретный формализм и нотацию: при таком обучении происходит не образование ума, а только накопление навыков продвинутого пользователя одной конкретной системы.

Для начинающих надо разрабатывать специальные простые и понятные по своей архитектуре и используемым алгоритмам (хотя не очень эффективные) инструментальные средства, как, например, верификатор F@BOOL@ для модельного языка программирования NIL [3].

Верификатор F@BOOL@ использует булевские решатели в качестве средства доказательства теорем, а язык программирования NIL имеет виртуальную машину, реализационную семантику (трансляцию в виртуальную машину), структурную операционную семантику (аксиоматическую систему доказательства возможности вычисления), денотационную семантику (в алгебре бинарных отношений) и аксиоматическую семантику (для утверждений частичной корректности). Доказана корректность и полнота реализации операционной семантики на виртуальной машине, доказано свойство взаимной корректности операционной и денотационной семантики, а для аксиоматическая семантики — полнота и непротиворечивость. Для расширений языка NIL предложен метод трансформационной семантики (реализация новых конструкций средствами ядра языка).

А как быть с практической пользой от формальных методов, экономическим эффектом и т.д.? Неужели нельзя добиться отдачи, и вся польза навсегда останется "виртуальной"?

На наш взгляд, о практической пользе формальных методов можно вести речь. Только для этого, по-видимому, надо ориентироваться не на создание универсальных фор-

мальных методов (и семантик), а на проблемно-ориентированные формальные методы (и семантики). Опыт показывает, что как только специалисты пробуют разработать универсальную формальную семантику языка программирования, то дело не доходит до практического её использования в силу её громоздкости. А специализированные, проблемно-ориентированные формальные методы и семантики программ находят практическое применение. Наиболее яркий пример такого сорта — это статические анализаторы программ, основанные на использовании упрощённой (обычно посредством абстрактной интерпретации) семантики того или иного вида.

Относительно новый пример перспективной проблемно-ориентированной денотационной семантики программ с указателями — это так называемое исчисление алиасов (Calculus of Aliasing) Бертрана Мейера [8]. Эта денотационная семантика предназначена для обнаружения программных дефектов или ошибок, обусловленных тем, что несколько разных выражений указывают на одну область памяти. Аксиоматическая семантика для этих же целей известна под именем логики отделимости (Separation Logic), она разрабатывается уже более 15 лет и имеет несколько экспериментальных реализаций. Её основы были заложены Джоном С. Рейнольдсом с соавторами [11]. Вопрос о связи этих двух формальных семантик для программ с указателями открыт, хотя в работе [13] сделана попытка построить исчисление алиасов для языка программирования, используемого для задания семантики логики отделимости.

Список литературы

1. Аджиев В. Мифы о безопасном ПО: уроки знаменитых катастроф // Открытые системы. 1998. N 6. [Электронный ресурс]. URL: `http://www.osp.ru/os/1998/06/179592` (дата обращения: 20.12.2015).
2. Живич М., Каннингэм Р. Истинная цена программных ошибок // Открытые системы. 2009. N 3. [Электронный ресурс]. URL: `http://www.osp.ru/os/2009/03/8158133/` (дата обращения: 20.12.2015).
3. Шилов Н.В. Пример верификации в проекте FBOOL, основанном на булевских решателях // Моделирование и анализ информационных систем. 2010. Т. 17, N 4. С.111-124.
4. Шилов Н.В., Шилова С.О. О преподавании логических и алгебраических оснований формальной семантики программ // Информатикии информационные технологии в образовании: теория, приложения, дидактика. Материалы Всероссийской научной школы-конференции с международным участием. Новосибирск: ФГБОУ ВПО НГПУ, 2012. Т. 2. С.64-71.
5. Abramsky S., Ghica D.R., Murawski A.S., and Ong C.-H. L. Applying game semantics to compositional software modeling and verification. // Proceedings of 10th International Conference

"Tools and Algorithms for the Construction and Analysis of Systems" TACAS-2004. Lecture Notes in Computer Science. Springer. 2004. Vol. 2988. P. 421-435.

6. Cousot P. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. // Theoretical Computer Science. Springer. 2002. Vol. 277, N 1-2. P. 47-103.
7. Floyd R.W. Assigning Meanings to Programs // Proc. Symp. Applied Mathematics. Am. Math. Soc. 1967. P. 19-31.
8. Meyer B. Steps Towards a Theory and Calculus of Aliasing // International Journal of Software and Informatics. 2011. Vol., N 1-2. P. :77-115.
9. Mosses P.D. The Varieties of Programming Language Semantics And Their Uses // Perspectives of System Informatics. Lecture Notes in Computer Science. Springer. 2001. V. 2244. P. 165-190.
10. Parnas D.L. Really Rethinking "Formal Methods" // Computer (IEEE Computer Society). 2010. Vol. 43, N 1. P. 28-34.
11. Reynolds J.C. Separation Logic: A Logic for Shared Mutable Data Structures // IEEE Symposium on Logic in Computer Science. 2002. P. 55-74.
12. Schmidt D.A. On the Need for a Popular Formal Semantics // ACM SIGPLAN Notices. 1997. Vol. 32. P. 115-116.
13. Shilov N.V., Satekbayeva A., Vorontsov A.P. Alias calculus for a simple imperative language with decidable pointer arithmetics // Bulletin of the Novosibirsk Computing Center. 2014. Vol. 37. P.131-148.

УДК 004.912

Подход к описанию жанра текста на основе его формальной жанровой структуры

Сидоров В.В.

В статье представлено описание математической модели представления документов. Модель включает в себя описания различных типов сегментов, которые определяются в тексте с помощью маркеров. Для данной модели приведен алгоритм сегментирования текста документа.

Ключевые слова: *жанровая сегментация, жанровая модель, жанровый сегмент.*

1. Введение

Важной задачей на сегодняшний день являются обработка и извлечение информации из текста. Разработчики современных систем, предназначенных для анализа различных документов, стараются включить в них программные блоки, отвечающие за более глубокий анализ текста. Причина расширения анализирующих систем такими программными блоками заключается в стремлении находить и извлекать наиболее точную информацию из них.

Каждый текстовый документ обладает определенными чертами [4, 5], которые определяются спецификой контекста этого документа и формируются благодаря сходству тематического содержания, единству стиля и композиционного построения, что соответствуют классическому определению жанра речевого произведения, сформулированному М.М. Бахтиным [1]. Жанр – это типовая модель построения речевого целого. Для каждого документа должна быть определена конкретная жанровая модель, представляющая его «типичную воспроизводимую жанровую форму».

Каждая жанровая модель представляет собой общую структуру документов, принадлежащих этой модели [3, 6]. Таким образом, каждый документ может быть разбит на структурные части, называемые жанровыми сегментами, ограничивающие определенную логическую область в тексте с помощью маркеров [2]. Эти сегменты, в свою очередь, могут быть также представлены своей жанровой моделью, еще сильнее структурируя документ.

В данной статье будет рассмотрен способ описания формальных жанровых структур документа и предложен алгоритм структурно-жанровой сегментации текста относительно данных структур.

2. Структурно-жанровая модель текста

Структура каждого текста определенного жанра может быть представлена тремя логическими уровнями: уровень жанровой модели, уровень жанровых сегментов и уровень маркеров. Маркеры представляют собой конечные последовательности символов, которые могут быть найдены тексте. Найденные маркеры (экземпляры) описываются начальной и конечной позицией в тексте.

Жанровый сегмент определяется наборами начальных и конечных маркеров. Экземпляр жанрового сегмента так же, как и экземпляр маркера, определяет границы данного сегмента в тексте. Жанровые сегменты могут быть нескольких типов. Для каждого типа сегмента задан набор аксиом и определено отображение, задающее правило построения экземпляров сегментов.

Жанровая модель является корнем иерархической жанровой структуры. Она содержит в себе набор главных сегментов, но не имеет границ, в отличие от сложного сегмента. Имея жанровую модель и документ, можно определить, принадлежит ли данный документ этой модели. Имея набор моделей, можно попытаться определить жанр документа, последовательно проверив его на соответствие этим жанровым моделям.

Разбиение документа на различные логические сегменты позволяет локализировать искомые блоки текста и рассматривать в конкретном контексте только необходимые области документа, игнорируя его оставшиеся части, являющиеся лишними в данном контексте. Применение этого метода при анализе больших объемов текстовых данных позволяет значительно сократить время выполнения специфичных анализирующих операций за счет уменьшения предназначенного для анализа массива данных.

2.1 Модель документа

Модель $Model$ является верхним уровнем жанровой структуры документа, она агрегирует в себе основные сегменты, которые, в свою очередь, имеют внутреннюю сложную структуру из других сегментов и маркеров, определяющих их границы (рис. 1).

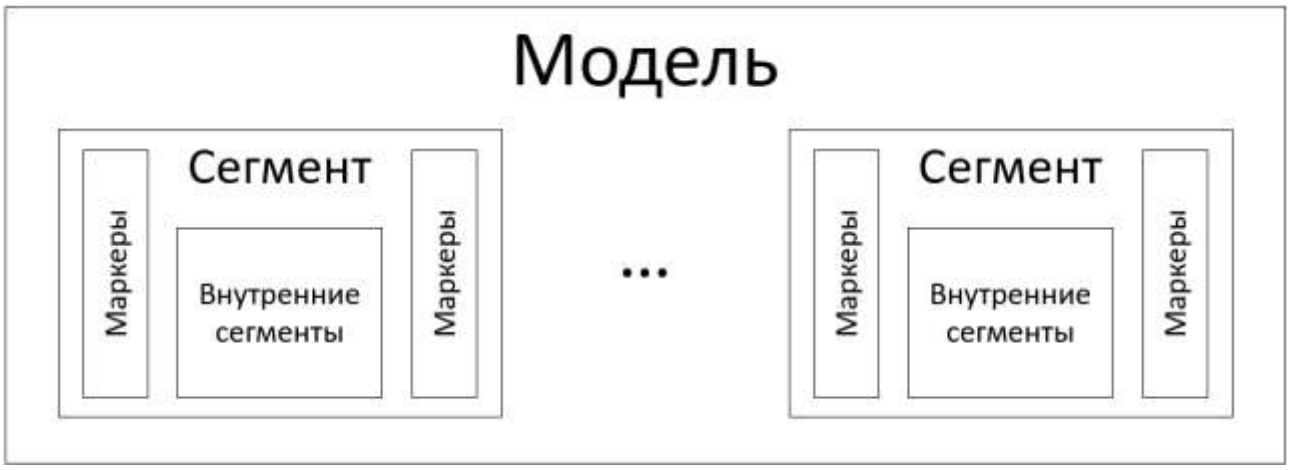


Рис. 1. Модель документа

$Model = < I, P^I >$, где I – множество сегментов, $P^I \subset I \times I$ – нерефлексивное, транзитивное, антисимметричное бинарное отношение частичного порядка.

2.2 Маркер

Алфавит A содержит в себе все заглавные и строчные буквы языка, а также специальные символы, такие как знаки препинания или символы конца и начала текста.

$A^* = \bigcup_{i=0}^{\infty} A^i$, где $A^i = \{\omega\upsilon \mid \omega \in A^{i-1}, \upsilon \in A\}, i \in \mathbb{N}$ – последовательности символов длины i, $A^0 = \{\varepsilon\}$, ε – пустой символ.

A^* - множество всех конечных последовательностей символов алфавита ($*$ - звезда Клини).

Текст $T \in A^*$ – произвольная конечная последовательность символов алфавита.

Фрагмент текста T – это $FR_T = \tau \in A^* \Leftrightarrow \exists\, \omega \in A^* \exists\, \upsilon \in A^*: \omega\tau\upsilon = T$.

Маркер M представляет из себя набор последовательностей символов. Можно считать, что он выделяет определенные части текста, совпадающие с этими последовательностями.

$M = < D >$, где $D \subset A^*$.

Примеры маркеров:

- «Начало текста» = <{¶$^{\text{H}}$}>; «Конец текста» = <{¶$^{\text{K}}$}> - состоят только из одного символа;

- «Конец предложения» = <{«. », «? », «! », «… »}> - можно составить полный список всех знаков препинания, на которые предложение может заканчиваться;
- «Адрес» = <{«Адрес», «Место проживания», «Город»}> - содержит основные слова, которыми можно обозначить место проживания;
- «Обращение» = <{«Уважаемый», «Уважаемая», «Здравствуйте,», «Добрый день»}> - этот список можно пополнить различными дополнительными определениями, такими как «Добрый вечер», «Доброго времени суток», однако полный перечень таких словосочетаний в данной статье приводить нецелесообразно;
- «Имя» = <{«Иван», «Андрей», «Мария», «Елена»,..}> - к сожалению, в данном случае нельзя составить полный список всех существующих имен или определить их какой-либо формулой, можно только перечислить наиболее распространенные;

2.3 Сегмент

В ходе исследования удалось выделить два основных типа сегментов: простой сегмент и сложный сегмент, который, в отличие от простого, содержит в себе набор внутренних сегментов. Более того, сложный сегмент может быть представлен следующими вариациями:

- последовательные сегменты;
- повторяющийся сегмент;
- альтернативные сегменты;
- комбинация сегментов;
- факультативный сегмент;
- отрицание сегмента.

В общем виде жанровый сегмент S описывается следующей системой:

$$S = < M_B, M_E, i_B, i_E, I, P^I, l_L, l_R, Type >,$$

где $M_B, M_E \in M$ – начальные и конечные маркеры,

$i_B, i_E \in \{0; 1\}$ – булевы значения, определяющие, требуется ли включить в экземпляр сегмента начальные и конечные маркеры,

$I \subset S$ – множество вложенных сегментов,

$P^I \subset I \times I$ – нерефлексивное, транзитивное, антисимметричное бинарное отношение частичного порядка над I,

$l_L, l_R \in \{0; 1\}$ – булевы значения, определяющие, должны ли вложенные сегменты быть размещены строго в начале и конце искомого сегмента,

$Type \in \{Type_{Simple}, Type_{Complex}, Type_{Sequence}, Type_{Repeatable}, Type_{Alternative}, Type_{Combination}, Type_{Optional}, Type_{Negative}\}$ – тип сегмента.

Кроме того, каждый тип сегмента имеет определенный набор аксиом, определяющих этот тип.

Жанровый экземпляр сегмента – это $S_{ex} = (b, e), b \in \mathbb{N}, e \in \mathbb{N}$.

Для каждого сегмента и для каждого текста определен конкретный экземпляр сегмента, определяющий положение этого сегмента в тексте. b, e – позиции начала и конца сегмента, нумерующиеся с 0. Таким образом, если $T' \in A^i$, то экземпляр сегмента S'_{ex} для T' – это пара чисел (b', e') таких, что $0 \leq b' \leq e' < i$ и $\tau \in FR_{T'}: \omega\tau\upsilon = T', \omega \in A^{b'}, \upsilon \in A^{i-e'-1}$ – искомый фрагмент текста.

Для каждого типа сегмента задано определенное отображение, определяющее правила построения F экземпляров сегментов:

$F: FR_T \times S \to S_{ex}$, где S_{ex} – экземпляр сегмента $s \in S$, построенного по фрагменту текста $fr_T \in FR_T$.

2.3.1 Простой сегмент

Простой сегмент является основным атомарным типом сегмента.

$S_{Simple} \subset S, F_{Simple}: FR_T \times S_{Simple} \to S_{ex}$.

Для данного типа сегмента определены следующие аксиомы:

1. $I = \emptyset$;
2. $P^I = \emptyset$;
3. $l_L = 0$;
4. $l_R = 0$;
5. $Type = Type_{Simple}$.

Пример:

«Предложение» = <{«Начало текста», «Конец предложения»}, {«Конец текста», «Конец предложения»}, 0, 1, $\emptyset, \emptyset, 0, 0, Type_{Simple}$>.

Тогда из текста «¶[H]*На комоде лежала какая-то книга. Он каждый раз, проходя взад и вперед, замечал ее; теперь же взял и посмотрел. Это был Новый завет в русском переводе. Книга старая, подержанная, в кожаном переплете.*¶[K]» (Ф.М. Достоевский, "Преступление и наказание") могут быть выделены следующие сегменты «Предложение»:

- *На комоде лежала какая-то книга.*
- *Он каждый раз, проходя взад и вперед, замечал ее; теперь же взял и посмотрел.*
- *Это был Новый завет в русском переводе.*
- *Книга старая, подержанная, в кожаном переплете.*

2.3.2 Сложный сегмент

Сложный сегмент, в отличие от простого, имеет структуру, которая содержит в себе вложенные сегменты. Для всех вариаций сложного сегмента подразумевается наличие следующей аксиомы: $I \neq \emptyset$.

$S_{Complex} \subset S, F_{Complex}: FR_T \times S_{Complex} \rightarrow S_{ex}$.

Для сложного сегмента определены следующие аксиомы:

1. $I \neq \emptyset$;
2. $Type = Type_{Complex}$.

Пример:

Расширим пример сегмента «Предложение». Пусть этот сегмент теперь должен содержать в себе слово «книга».

Определим новый простой сегмент, а также новые маркеры:

Маркер «Книга (м)» = <{«Книга», «книга»}>.

Сегмент «Книга» = <{«Книга (м)»}, {«Книга (м)»}, 1, 1, $\emptyset, \emptyset, 0, 0, Type_{Simple}$> - блок текста, состоящий только из слова «Книга» или «книга».

Теперь преобразуем простой сегмент «Предложение» в сложный, добавив в него вложенный сегмент «Книга»:

«Предложение» = <{«Начало текста», «Конец предложения»}, {«Конец текста», «Конец предложения»}, 0, 1, {«Книга»}, ∅, 0, 0, $Type_{Complex}$>.

Тогда в тексте «¶[H]*На комоде лежала какая-то книга. Он каждый раз, проходя взад и вперед, замечал ее; теперь же взял и посмотрел. Это был Новый завет в русском переводе. Книга старая, подержанная, в кожаном переплете.*¶[K]» можно найти следующие сегменты «Предложение»:

- *На комоде лежала какая-то книга*.
- *Книга старая, подержанная, в кожаном переплете*.

Заметим, что не важно, в начале или в конце располагается искомое слово, однако, если изменить параметр l_L на true, то сегмент «Предложение» обязан будет начинаться с этого слова, и подойдет только такой вариант:

- *Книга старая, подержанная, в кожаном переплете*.

2.3.3 Последовательные сегменты

Последовательные сегменты являются вариацией сложного сегмента. Их отличие от сложного сегмента заключается в том, что между ними не может быть разрывов, а также первый вложенный сегмент должен располагаться строго в начале искомого сегмента, а последний строго в конце. Таким образом, тип последовательные сегменты является своего рода контейнером, содержащим в себе некоторые сегменты и предотвращающем появление дополнительного текста вокруг них.

$S_{Sequence} \subset S, F_{Sequence}: FR_T \times S_{Sequence} \to S_{ex}$.

Для данного типа сегмента определены следующие аксиомы:

1. $I \neq \emptyset$;
2. $l_L = 1$;
3. $l_R = 1$;
4. $Type = Type_{Sequence}$.

В предыдущем примере сложный сегмент «Предложение» содержал в себе (кроме сегмента «Книга») дополнительный текст – «*старая, подержанная, в кожаном переплете*.». Сегмент типа последовательные сегменты не может содержать в себе лишний текст, поэтому в примере, предложенном выше, нельзя найти ни один сегмент последовательного типа:

«Предложение» = <{«Начало текста», «Конец предложения»}, {«Конец текста», «Конец предложения»}, 0, 1, {«Книга»}, ∅, 1, 1, $Type_{Sequence}$>. На самом деле, существует только два «Предложения», подходящие под это определение: предложение, состоящее из слова «Книга» (с заглавной буквы) и из слова «книга» (со строчной).

2.3.4 Повторяющийся сегмент

Повторяющийся сегмент также является вариацией сложного сегмента. Он описывает сегмент, который может быть встречен в тексте подряд один или более раз. Таким образом, этот тип сегмента является «надстройкой» над другим сегментом, означающий возможное повторение оного.

$S_{Repeatable} \subset S,\ F_{Repeatable}: FR_T \times S_{Repeatable} \to S_{ex}.$

Для данного типа сегмента определены следующие аксиомы:

1. $I = \{s\}, s \in S;$
2. $P^I = \emptyset;$
3. $Type = Type_{Repeatable}.$

В вышеописанном примере делового письма упоминался блок «Адресат». Также утверждалось, что адресатов может быть несколько. Такой случай можно описать сегментом повторяющегося типа:

«Адресаты» = <Ø, Ø, 0, 0, {«Адресат»}, Ø, 0, 0, $Type_{Repeatable}$>.

Пустые множества начальных и конечных маркеров означают, что данный сегмент не имеет определенных границ и может иметь произвольное положение в тексте. Его границы в таком случае будут определены границами внутренних вложенных сегментов.

2.3.5 Альтернативные сегменты

Тип сегмента альтернативные сегменты может обозначить те сегменты, которые являются взаимозаменяемыми, или те, для которых не важно, какой из них должен быть найден в данном контексте.

$S_{Alternative} \subset S,\ F_{Alternative}: FR_T \times S_{Alternative} \to S_{ex}$.

Для данного типа сегмента определены следующие аксиомы:

1. $I \neq \emptyset$;
2. $P^I = \emptyset$;
3. $Type = T_{Alternative}$.

Пример:

Имеются следующие сегменты: «E-mail адрес», «Телефон».

Можно описать сегмент «Контактная информация», содержащий либо адрес электронной почты, либо телефон, следующим образом:

«Контактная информация» = <Ø, Ø, 0, 0, {«E-mail адрес», «Телефон»}, Ø, 0, 0, $Type_{Alternative}$>.

2.3.6 Комбинация сегментов

Для тех случаев, когда порядок взаиморасположения нескольких сегментов неизвестен, но известно, что они должны присутствовать в рассматриваемом жанре документа, предусмотрен тип комбинация сегментов.

$S_{Combination} \subset S, F_{Combination}: FR_T \times S_{Combination} \to S_{ex}$.

Для данного типа сегмента определены следующие аксиомы:

1. $I \neq \emptyset$;
2. $P^I = \emptyset$;
3. $Type = Type_{Combination}$.

Пример:

Многие документы содержат раздел общей информации, содержащий в себе такие данные как ФИО, Возраст, Пол, Город. Предположим, структуры этих сегментов уже определены. Однако, неизвестно в каком порядке эти сегменты будут располагаться в документе. Поэтому, их можно объединить как комбинацию сегментов.

«Общая информация» = <Ø, Ø, 0, 0, {«ФИО», «Возраст», «Пол», «Город»}, Ø, 0, 0, $Type_{Combination}$>.

2.3.7 Факультативный сегмент

Факультативный сегмент, также, как и повторяющийся сегмент, является надстройкой над сегментом. Он позволяет пометить тот сегмент, присутствие которого в тексте необязательно.

$S_{Optional} \subset S, F_{Optional}: FR_T \times S_{Optional} \to S_{ex}$.

Для данного типа сегмента определены следующие аксиомы:

1. $I = \{s\}, s \in S$;
2. $P^I = \emptyset$;
3. $Type = Type_{Optional}$.

Предположим, что в предыдущем примере сегмент «Город» может быть факультативным.

Определим факультативный сегмент «Город*» = <Ø, Ø, 0, 0, {«Город»}, Ø, 0, 0, $Type_{Optional}$>.

Тогда новый сегмент «Общая информация» будет выглядеть так:

«Общая информация» = <Ø, Ø, 0, 0, {«ФИО», «Возраст», «Пол», «Город*»}, Ø, 0, 0, $Type_{Combination}$>.

2.3.8 Отрицание сегмента

Отрицание сегмента также является надстройкой над конкретным сегментом. Добавление этой надстройки к сегменту означает, что этого сегмента в данном блоке текста быть не должно.

$S_{Negative} \subset S, F_{Negative}: FR_T \times S_{Negative} \to S_{ex}$.

Для данного типа сегмента определены следующие аксиомы:

1. $M_B = \emptyset$;
2. $M_E = \emptyset$;
3. $i_B = 1$;
4. $i_E = 1$;
5. $I = \{s\}, s \in S$;
6. $P^I = \emptyset$;
7. $l_L = 0$;
8. $l_R = 0$;
9. $Type = Type_{Negative}$.

Предположим, имеется база текстов, полученных из разных источников, включая Интернет. Если требуется проанализировать все тексты, которые не являются web-страницами, можно добавить отрицание сегмента «HTML»:

«HTML» = <{«<html>»}, Ø, 1, 1, $\emptyset, \emptyset, 0,0, Type_{Simple}$> - простой сегмент.

«-HTML» = <Ø, Ø, 1, 1, «HTML», $\emptyset, 0,0, Type_{Negative}$> - его отрицание.

Добавив отрицание к любой структуре сегментов, например, сформировав сложный сегмент, содержащий «-HTML» на первом месте в множестве вложенных сегментов, можно распознавать тексты, не являющиеся web-страницами.

2.4 Пример «Резюме»

Общая структура резюме (рис. 2) состоит из трех разделов: раздела общей информации, который может быть представлен сегментом сложного типа, раздела основной информации, представляемого комбинацией сегментов, и раздела дополнительной информации, представляемого простым сегментом с факультативной надстройкой (штриховая пунктирная линия).

Раздел общей информации является сложным сегментом и содержит подразделы главной и общей информации. Стоит отметить, что резюме не может начинаться с обращения, поэтому перед подразделом главной информации находится отрицание простого сегмента «Обращение» (точечная пунктирная линия).

В начале подраздела главной информации находится ФИО составителя резюме, и сразу за ним должна быть указана целевая должность. Таким образом, удобнее всего определить сегмент подраздела главной информации как сегмент типа последовательные сегменты. Подразделы ФИО и Должность являются простыми сегментами.

Подраздел общей информации представим комбинацией факультативных сегментов над сегментами Возраст, Адрес и E-mail. Так как в резюме могут быть указаны несколько записей Адреса и E-mail, лучше всего представить их как повторяющиеся сегменты. Сегмент Возраст имеет следующую структуру: он является сегментом альтернативного типа, показывая, что на его месте может быть обнаружен как сегмент Возраст составителя, так и сегмент Дата рождения.

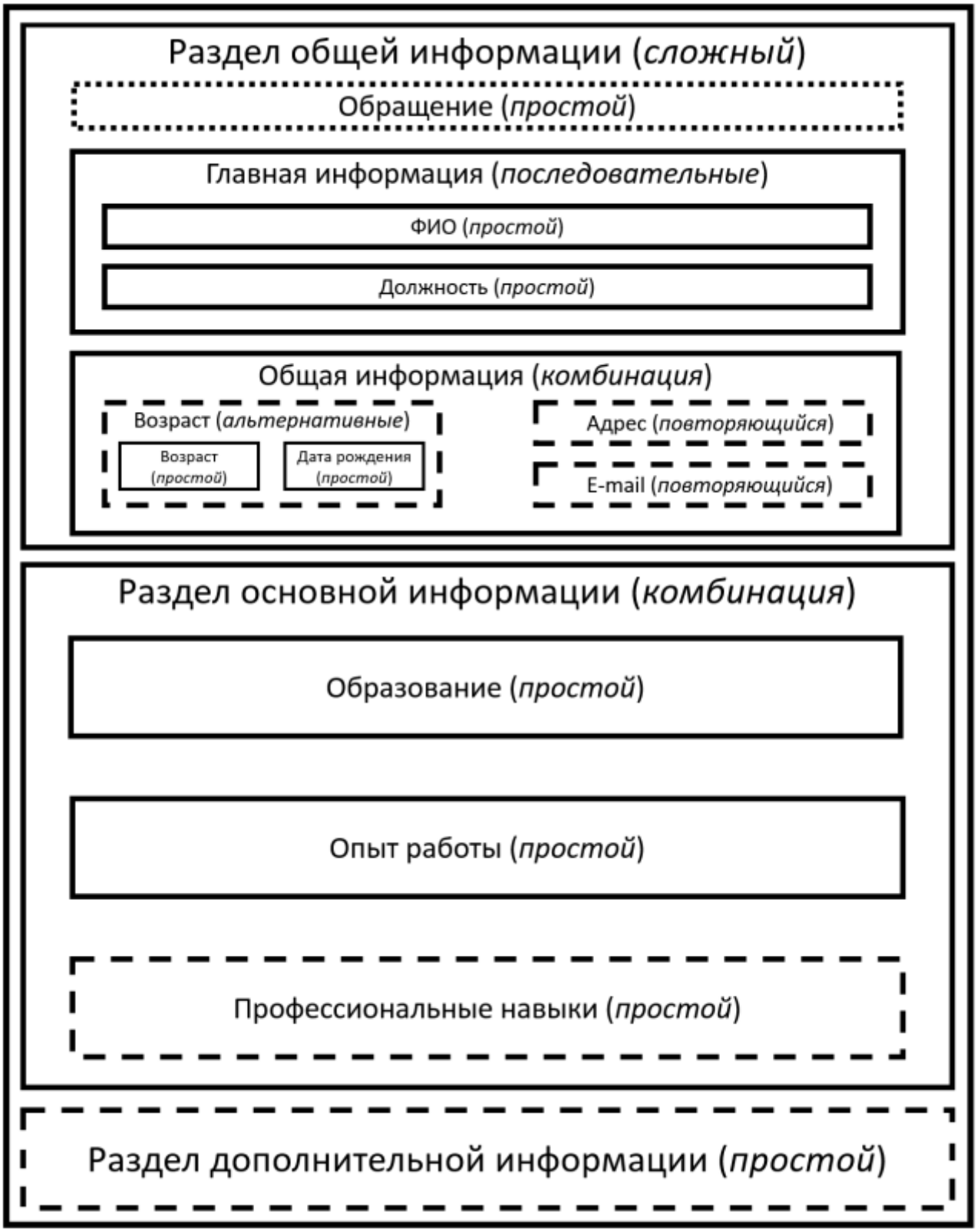


Рис. 2. Структура резюме

Раздел основной информации является комбинацией сегментов и включает в себя такие обязательные подразделы как Образование и Опыт работы (простые сегменты) и раздел Профессиональные навыки (факультативный сегмент).

Раздел дополнительной информации является факультативной надстройкой над простым сегментом.

3. Алгоритм построения жанровой модели текста

Построение жанровых моделей документов позволяет проводить углубленный анализ, синтез текстов. Ниже приведен пример алгоритма сегментирования текста, также позволяющий определять принадлежность документа определенному жанру.

Изначально имеется иерархическая структура сегментов (заданная в XML формате) и текст, для которого будет выполняться поиск. Структура сегментов представляет собой жанровую модель, содержащую в себе список внутренних сегментов. Соответственно, каждый сегмент также содержит в себе вложенные сегменты (если таковые имеются) и множества начальных и конечных маркеров.

Изначально полагается, что модель является корневым сегментом. Тогда для корневого сегмента:

1. Берется фрагмент текста (в начале фрагмент полагается равным тексту).
2. Создается два вектора экземпляров начальных и конечных маркеров (экземпляр маркера содержит индексы начала и конца данного маркера в тексте, вектор экземпляров содержит все возможные позиции в тексте для всех заданных маркеров). Этот шаг пропускается для корневого сегмента, так как модель содержит только список внутренних сегментов и не имеет границ.
3. Если сегмент содержит внутренние сегменты:

    3.1. Относительно векторов экземпляров маркеров находятся максимальные начало и конец сегмента, определяя внутреннюю область, в которой будет производиться поиск внутренних сегментов.

    3.2. Для каждого сегмента:

    3.2.1. Алгоритм рекурсивно запускается с первой операции. Фрагмент текста выбирается через ранее определенную внутреннюю область. Если для данного внутреннего сегмента уже был определен предыдущий соседний внутренний

сегмент, то начало анализируемого текста смещается до конца найденного сегмента.

3.2.2. У сегмента берется его экземпляр. Он определяет область, в которой был найден этот сегмент. Эта область соединяется с областью, полученной ранее для предыдущих сегментов.

3.3. Вектор экземпляров маркеров фильтруется таким образом, чтобы его экземпляры не попадали в область найденных сегментов.

4. Относительно векторов экземпляров маркеров определяются наилучшие минимальные границы сегмента.

5. Создается экземпляр сегмента – структура, содержащая позиции начала и конца данного сегмента. Если создать экземпляр не удалось, то алгоритм завершает работу.

После завершения работы основного алгоритма для всех сегментов формируется XML структура, содержащая облегченную иерархию сегментов (без векторов маркеров и без сегментов, которые не имеют конкретной позиции в тексте, например, отрицание сегмента), в которой определены позиции начал и концов этих сегментов. Если анализ текста окончился неудачей, и основной алгоритм преждевременно завершил работу, в файле вывода это будет отражено, но структура все равно будет сформирована, определяя те сегменты, которые были успешно найдены до завершения работы.

4. Заключение

В работе приведено математическое описание модели жанра документа, ориентированное на автоматическую обработку текста. Данная модель может применяться как для задач анализа жанровой структуры текста и жанровой классификации, так и для синтеза, где жанровая модель позволяет выстроить повествовательную структуру текста и оформить ее в соответствии с заданным форматом. Жанр в предложенном подходе сопоставляется формальной структуре текста, снабженной жанровыми маркерами. Приведенный алгоритм демонстрирует применимость рассмотренной модели для задач жанровой сегментации текста, что является важным компонентом при решении широкого класса задач, связанных с анализом текста. Апробация данного алгоритма на корпусе текстов жанра резюме показало его работоспособность, и, следовательно, корректность предложенного формализма описания модели. В дальнейшем планируется провести исследование применимости модели для более широкого набора жанров, чтобы выявить ее различительную способность для решения классификационных задач.

Список литературы

1. Бахтин М.М. Проблема речевых жанров // Эстетика словесного творчества. – М.: Искусство, 1986. – С. 250–296.
2. Блюменау Д.И., Гендина Н.И., Добронравов И.С., Лахути Д.Г., Леонов В.П., Федоров Е.Б. Формализованное реферирование с использованием словесных клише (маркеров) // Научно-техническая информация. Сер. 2. – 1981. – № 2. – С. 16–20.
3. Кибрик, А. А. Модус, жанр и другие параметры классификации дискурсов // Вопросы языкознания. – 2009. – №2. – С. 3–21.
4. Кононенко И.С., Сидорова Е.А. Жанровые аспекты классификации веб-сайтов // Программная инженерия № 8. 2015. С. 32–40.
5. Кононенко И.С., Сидорова Е.А. Обработка делового письма в системе документооборота // Труды международного семинара Диалог’2002 по компьютерной лингвистике и ее приложениям. Т.2. – Москва: Наука, 2002. –С. 299–310.
6. Щипицина Л.Ю. Жанры компьютерно-опосредованной коммуникации. – Архангельск: Поморский университет, 2009. – 238 с.

УДК 51-77

Формализация представления целенаправленно развивающихся процессов

Скопин И.Н. (Институт вычислительной математики и математической геофизики СО РАН, Новосибирский государственный университет)

Обсуждаются возможности изучения процессов на основе представления их с помощью спиралей развития. Вводятся геометрические понятия, связанные со спиралью, позволяющие моделировать развитие проекциями траекторий на различные плоскости пространства факторов.

Ключевые слова: *целенаправленный процесс, траектория развития, факторы развития, критерий развития; S-кривая, улитка развития, моделирование.*

1. Введение

Моделирование развития является весьма актуальным методом изучения в самых разных прикладных областях. От точности прогноза естественных, социальных и иных процессов зависит качество управления, надежность работы проектируемых приборов и агрегатов, с одной стороны, а с другой — успешность выполнения социальных, экономических и иных проектов. В то же время, инструментальная поддержка моделирования развития как средства управления соответствует потребности далеко не всегда. Так, вполне удовлетворительно представляют развитие модели, основанные на выявленных закономерностях, которые описываются математическими методами.1 Однако, если процесс развивается под воздействием разнонаправленных факторов, механическое соединение закономерностей, связанных с каждым из них, неадекватно реальному поведению. Это особенно заметно, когда исследователь имеет дело с так называемыми открытыми системами, при изучении которых дифференциальные уравнения, статистические зависимости и иные соотношения дают лишь ориентиры, помощью которых можно принимать те или иные разумные решения.

Для класса управляемых процессов, которые характеризуются наличием цели развития, часто рекомендуется использовать так называемые S-кривые. Этот метод предлагается, например, в документе международной организации Project Management Institute «Руководство к своду знаний по управлению проектами (Руководство PMBOK®)», где он отнесен к разряду «лучших практик» (The Best Practices) проектного менеджмента. Вот

[1] Очень точно представил этот вопрос еще в 1997 на семинаре при Президентском совете РФ В.И. Арнольд в своем докладе, посвященном практическому использованию так называемых “жестких” и “мягких” математических моделей [1].

как определяется S-кривая в PMBOK: «Это график зависимости от времени итоговых затрат, трудозатрат, процента выполнения работ или других количественных показателей. Название получено от характерной S-образной (более пологой в начале и конце и более крутой в середине) формы кривой развития проекта, имеющего плавное начало, более быстрое развитие и плавное окончание. Термин также используется для обозначения кривой распределения вероятности, получаемой в результате моделирования, которое применяется в количественном анализе рисков» [2]. Использование S-кривых для анализа развития процессов хорошо себя зарекомендовало в самых разных исследовательских областях. Существуют различные методики их построения. Но, к сожалению, все они приводят лишь к качественным оценкам [3]. Слабое место S-кривых как аналитического инструмента в том, что они не дают средств, которые позволили бы выявлять влияние отдельных факторов на развитие процессов.

В работе [4] была предложен подход к анализу экономических систем, который за счет специальных построений позволяет преодолеть указанный недостаток S-кривых. Его идея связана с использованием тройной спирали развития, отражающей главные факторы эффективности как экономики в целом, так и ее отраслей. Проекция спирали на плоскость фиксированного времени образует улитку, разбиваемую на секторы, которые представляют в модели развития последовательность фаз процесса. Для каждой фазы в множестве факторов, влияющих на развитие процесса, выявляются те, влияние которых, наиболее существенно. Эта модельная структура дает возможность анализа многофазных процессов в разных аспектах, используя общие для всех фаз критерии оценки качества развития. При моделировании экономических процессов такие критерии в конечном итоге сводятся к стоимостным характеристикам и инвестиционной привлекательности проектов.

Моделирование физических процессов, на которых основывается функционирование приборов и агрегатов реальных производств также может рассматриваться как развитие, и во многих случаях для них также могут быть выполнение построения, аналогичные спирали развития.

Идея улитки весьма продуктивна и может быть распространена на исследования в других областях, для которых понятие развития процессов рассматривается как одно из ключевых положений. Улитка привлекательна для так называемых междисциплинарных исследований, когда приходится совместно моделировать системы влияющих друг на друга автономных процессов, поведение которых подчинено разнородным правилам и законам. В этом случае она имеет хорошие перспективы по отношению к решению сложной задачи согласования поведения автономных моделей процессов системы, что весьма актуально, например, при проектировании и разработке приборов и устройств.

Обязательным условием адекватности использования улитки является корректное определение общего для модели критерия качества развития системы, которому подчинены критерии моделей составляющих процессов.

В связи с тем, что потребность изучения таких систем весьма высока, представляется важным точное описание подхода, которое в дальнейшем можно конкретизировать для тех или иных исследований. В основе такого описания лежат понятия, характеризующие целенаправленно развивающийся процесса как геометрический объект в пространстве факторов, называемый далее конусом траекторий. Целенаправленность в этих построениях связывается с соответствием траектории критерию развития, который рассматривается в качестве аналога S-кривой. Построение проекций траекторий на различные плоскости отражает картину допустимого и оптимального поведения процесса во времени.

В последующих разделах дается мотивация предлагаемого подхода, формально определяются только что обозначенные понятия, а также процедуры оперирования, необходимые при построении моделей развития и анализе их использования. Возможности подхода иллюстрируется условными примерами, относящимися к области проектирования приборов и устройств. Предлагаемые материалы рассматриваются как концептуальная основа программного инструментария поддержки исследований моделей развития. Необходимыми элементами поддержки являются средства геометрического представления процессов как объектов многомерного пространства факторов, а также функции задания параметров этих объектов и извлечения информации о связях факторов развития с геометрией. В заключении поводятся итоги и намечаются перспективы дальнейшего исследования.

Работа поддержана грантом РНФ № 14-07-0048.5

2. Критерий и траектории развития

Процессы как объекты моделирования используются в различных исследованиях и технологиях очень давно. В соответствии с направлениями использования выделяются различные аспекты процессов. Довольно распространенным является бихевиористское понимание процесса как черного ящика, перерабатывающего входную информацию и ресурсы в выходные продукты. Эта позиция позволяет говорить о целенаправленности развития процесса производства некоего продукта и определять системы связанных между собой процессов, изучать их взаимодействия с системных позиций. Поведение таких систем часто представляют в виде интегрирующего процесса со своими входами и

выходами. Рассматривая выходные продукты системы в качестве цели ее развития, можно ставить задачи оптимизации интегрирующего процесса.

К сожалению, такая позиция не может быть принята для всех систем хотя бы потому, что цель интегрирующего процесса не складывается из целей составляющих процессов. Не менее существенно, что желательные цели меняются со временем, а зачастую просто не определены для системы в целом. Это характерно, в частности, для развивающихся процессов, для которых последствия реализации идеи внедрения результатов разработки часто не предсказуемы. Точнее было бы задавать критерии развития, характеризующие текущие предпочтения и отражающие прогнозы развития.

Формально *критерий развития* можно рассматривать как функцию (функционал) F, зависящую от времени и совокупности *факторов* $a_1, \ldots, a_n$, которые влияют на процесс, и на которые процесс влияет, вырабатывающую значения из множества с отношением полного порядка. Для определенности далее в качестве области значений F рассматривается **R**.

Зависимость критерия от времени характерна для развивающихся процессов. Она практически всегда проявляет себя при переходе от одной фазы процесса к другой. Определяя фазовые переходы как смену факторов, наиболее существенных для развития, изменения критерия во времени естественно ограничивать фазовыми переходами, а в стабильные по отношению к оцениванию качества периоды развития использовать не зависящиеся от времени критерии.

Считается, что для фиксированных начального момента зарождения процесса $t = 0$ и времени его окончания $t = E$ любая *траектория развития* есть направленная кривая $\mathfrak{C}$ в n+2-мерном пространстве (время, область значений F плюс n факторов, изменяющихся с течением времени), точки которой задаются формулой:

$$R = \left(t, F\left(t, a_1(t), \ldots, a_n(t)\right), a_1(t), \ldots, a_n(t)\right), 0 \le t \le E.$$

Любая кривая $\mathfrak{C}$, для которой определены начало и конец, определяет естественный порядок $\lessdot$ на множестве точек, из которых она состоит. Для траекторий развивающихся процессов этот порядок должен соответствовать временному упорядочиванию. Таким образом при моделировании развития нужно рассматривать только такие кривые, для которых справедливо:

(1) $\quad \forall R', R'' \in \mathfrak{C}\ (R' \lessdot R'') \equiv (R'_t < R''_t)$.

Здесь и далее нижним индексом t обозначена временная координата точки кривой.

Следующее ограничение запрещает траектории иметь пропуски во времени, что вполне соответствует интуитивному пониманию развития процесса:

(2) $\quad \forall\, t \in [0, E]\ \exists R (R \in \mathfrak{C})$.

Далее, некорректными считаются кривые, которые имеют возвраты по времени:

(3) $\forall R', R'' \in \mathfrak{C}\,(R' \neq R'') \supset \neg(R'_t = R''_t)$.

Для некоторых прикладных исследований возврат по времени, в принципе, может трактоваться как срыв текущего процесса и образование нового. Но в нашем рассмотрении такое допущение только усложнило бы понимание.

Соотношения (1 – 3) называются условиями корректности траектории развития. Они вполне соответствуют тому, что можно считать развитием процесса, оставаясь на формальном уровне рассмотрения. Здесь стоит отметить, что мы разрешаем временной координате траектории иметь разрывы первого рода: если она имеет предел слева от точки разрыва (отличный от предела справа), то образующуюся ступеньку следует трактовать как внешнее воздействие на модельный и/или реальный процесс, результатом которого становится изменение состояния.

Условия корректности траектории в дальнейшем специально не оговариваются, поскольку другие кривые в качестве представления траекторий не рассматриваются. Последнее замечание на этот счет, важное в дальнейшем, — следующее легко выводимое свойство: корректная траектория пересекается с любой плоскостью константного времени $\tau \in [0, E]$ в точности в одной точке; если константа времени τ находится вне указанного интервала, то с такой плоскостью траектория не пересекается.

Понятно, что далеко не все кривые соответствуют реально осуществимым траекториям. Взаимосвязи и зависимость факторов определяют множество тех кривых, из которых можно выбирать отвечающие критерию развития процесса. Информация об этом извлекается из априорных соглашений, из закономерностей, характерных для прикладной области и для решаемой задачи. Продуктивный источник данных о соотношениях между факторами в динамике их развития — автономные модели процессов, которые отражают разные аспекты отдельных фаз главного целенаправленно развивающегося процесса. Такие модели далее называются аспектными.

Для адекватности использования аспектного моделирования важно, чтобы аспектные модели выполнялись не независимо от расчетов модели развития, а как запросы на поставку ей данных о факторах, влияющих на развитие. Для реализации аспектных моделей можно рекомендовать весь арсенал средств, который сегодня используется в практике математического моделирования процессов и явлений. Концептуальный обзор таких средств можно найти в работе [5]. В терминах этой работы аспектные модели следует рассматривать как привнесенную в решение задачи моделирования развития интеллектуальность.

Вопросы, связанные с аспектными моделями и, в частности, с их взаимодействиями, выходят за рамки настоящей работы. Мы сосредоточиваем внимание на основных принципах моделирования развития, а потому считаем, что вся необходимая информация о факторах предоставляется до выполнения расчетов по главной модели и по требованиям, в ходе этих расчетов. Модель развития, которая рассматривается как главная, играет роль организующего и управляющего процесса активизации междисциплинарной системы моделей всех процессов, используемых для получения информации о развитии. Это соглашение позволяет считать, что в качестве представления траекторий используются только те кривые из n+2-мерного пространства, точки которых удовлетворяют условиям, связанным с априорными соглашениями и с соотношениями, получаемыми при аспектном моделировании.

Траектория начинается в точке плоскости $t = 0$, называемой *начальной*, которая соответствует значениям факторов в начальном состоянии моделируемого процесса:

$$Begin = (0, F(0, a_1(0), \dots, a_n(0),),\ a_1(0), \dots, a_n(0)),$$

и заканчивается в *конечной* точке:

$$end = (E, F\big(E, a_1(E), \dots, a_n(E)\big), a_1(E), \dots, a_n(E)).$$

Траектория процесса отслеживается до того момента, когда она достигает плоскости $t = T$ — *конец отслеживания*:

$$End = (T, F\big(T, a_1(T), \dots, a_n(T)\big), a_1(T), \dots, a_n(T)).$$

Фиксируя $Begin$, разработчик указывает на все процессы, траектории которых начинаются в этой точке. В общем случае выбор начальной точки неоднозначен, и она назначается из некоторого множества $DBegin$, лежащего в плоскости $t = 0$. Это множество называется *областью допустимости начала траектории*.

Пусть DE — множество конечных точек траекторий, начинающихся в $DBegi$n, на которых достигаются целевые значения факторов. Это множество разбивается на три класса, соответствующие трем вариантам завершения процесса:

- $DEnd >$. В этот класс попадают точки из DE, для которых $E > T$, что означает, что процесс завешается позже окончания отслеживания.
- $DEnd$. В этот класс, равный пересечению DE с плоскостью $t = T$, попадают точки, для которых $E = T$, что означает одновременность завершения и отслеживания процесса.
- $DEnd <$. В класс попадают точки из DE, для которых $E < T$, что означает, что процесс завершается раньше окончания отслеживания.

Траектории, заканчивающиеся в $DEnd <$ и их процессы в зависимости от специфики моделирования развития можно трактовать по-разному:

- Для одних процессов завершение раньше окончания отслеживания *невозможно* или *запрещено*, и тогда соответствующие траектории должны быть исключены из множества допустимых.
- Для других оно *нейтрально*, т.е. не влияет на качество развития процесса, и тогда при формальном рассмотрении эти траектории можно продлить до плоскости $t = T$, задавая процессу, выполняющемуся в соответствии какой-либо из них, константное, т.е. не меняющее значений факторов поведение.
- Для третьих раннее прекращение процесса является *дополнительным бонусом* в его оценке.

Следующие примеры поясняют это разграничение.

Разрабатывая прибор электронно-лучевой сварки, среди прочего необходимо обеспечить фокусировку испускаемого пучка электронов, согласованную по времени с подготовкой свариваемого материала. Более раннее выполнение фокусировки некорректно. Поэтому при моделировании ее допустимая траектория не должна завершаться до окончания процесса подготовки и наоборот. Как следствие, для обоих процессов нужно зафиксировать общее время завершения выполнения и отслеживания. В этом примере траектория фокусировки из $DEnd <$ приведет к снижению качества процесса сварки из-за простоя подготовленного свариваемого материала, а завершение подготовки в $DEnd >$, скорее всего, недопустимо, но это проявится в оценке как нарушение условия $End \in DEnd$ для траектория при достижении ею плоскости $t = T$.

Примером раннего завершения процесса, оцениваемого при прочих равных условиях как заслуживающего бонус, для сварки может служить моделирование охлаждения. Если использованию сваренной детали в других процессах препятствует высокая температура, то анализ траекторий охлаждения может подсказать разработчику, что в приборе целесообразен кулер.

Бонусная трактовка траекторий из класса $DEnd <$ предпочтительна, когда есть возможность перераспределения высвобождающихся ресурсов и организации использующего процесса в точке E. Поскольку денежные критерии качества очевидно удовлетворяют этому условию, такая трактовка характерна для большинства экономических моделей.

Считается, что модель процесса задана правильно, если выполняется одно из трех условий:

(1) $DEnd \neq \emptyset$;

(2) $DEnd > \neq \emptyset$;

(3) $DEnd < \neq \emptyset$ и досрочное завершение процесса правомерно.

Это утверждение мотивируется следующим образом:

- Если выполнено условие (1), то для соответствующих траекторий осмыслена оценка качества развития процесса в конце отслеживания.
- Условие (2) означает, что каждая траектория, заканчивающаяся в $DEnd > \neq \emptyset$, содержит точку R_T пересечения с плоскостью $t = T$. Множество всех таких точек, обозначаемое как DR, может использоваться для промежуточных оценок качества траекторий, заканчивающихся в $DEnd >$.
- Выполнение условия (3) может пополнить множество DR, если следующим образом модифицировать траекторию процесса:
  - для нейтральной оценки досрочного завершения она продлевается до плоскости $t = T$ с константным поведением, т.е. с сохранением значений факторов, равными тем, которые они получили при завершении;
  - для бонусной оценки кроме того нужно модифицировать критерий, корректируя оценку качества на константном окончании траектории.

Таким образом, для правильно заданной модели процесса можно определить *область результатов допустимых траекторий*, которая обозначается как $DResult = DEnd \cup DR$.

Следует отметить, что с формальной точки зрения можно исключить из рассмотрения траектории класса $DEnd >$. Это достигается объявлением T временем завершения одного процесса и заданием другого, который представляет в модели остаток траектории, разбиваемой на две части. Учитывая это замечание в дальнейшем мы считаем, что область $DResult$, лежащая в плоскости окончания отслеживания $t = T$, содержит конечные точки кривых, представляющих все такие траектории.

При моделировании развития процесса целесообразно выделить на плоскости $t = T$ точку наиболее предпочтительного завершения процесса. Для такой точки вектор $(Begin, End)$ рассматривается как особая идеальная траектория, которая ведет из заданной начальной точки в выбираемую конечную. Понятно, что выбор точки End зависит от критерия, но на формализованном уровне рассмотрения фиксировать эту зависимость не следует. Моделирование развития предполагает анализ различных вариантов траекторий, и первый шаг в подготовке расчетов связан с экспертной оценкой того, какие точки начала и конца отслеживания целесообразно выбрать для дальнейшего анализа. Это главные ориентиры изучения процесса. В дальнейшем вектор $(Begin, End)$ называется *трендом траекторий развития*.

Реальная траектория, разумеется, существенно отклоняется от тренда и даже не столько потому, что критерий может оказаться не определенным, например, в начале процесса — эта техническая трудность преодолима формальным доопределением *F*. Дело в том, что в адекватном критерии оценки целенаправленного развития процесса необходимо отражать не только получаемые результаты и прогресс в достижении цели, но также затрачиваемые на это ресурсы и другие факторы.

3. Оценка качества траектории развития

Содержательной основой оценки качества выполнения процесса является степень приближения его траектории к цели. Это положение конкретизируется, во-первых, выбором функции-критерия *F*, а во-вторых, процедурой, запускаемой для вычисления этой функции. Как уже отмечалось, далеко не всегда достижением цели можно считать равенство *F* целевому, т.е. априорно фиксируемому показателю. Так, целевым показателем разогрева парового котла является определенная температура пара при фиксированном давлении. Но нагревание не должно происходить слишком быстро — для котла известна максимально допустимая скорость последующего разогрева, когда достигнута определенная температура. Поэтому, формируя функцию критерия, необходимо наряду с целевым показателем предусмотреть оценивание качества процесса в ситуациях риска нарушения допустимости. Что же касается запуска функции *F*, то это действие должно определяться, исходя из текущей температуры и режима работы нагревателя. Функция *F* всегда должна быть готова к осуществлению замера критерия с тем, чтобы заблокировать возможный перегрев котла. Иными словами, при описанной стратегии разогрева запуск этой функция для замера заданного критерия может быть осуществляться в любой точке траектории процесса.

Представленная иллюстрация указывает на то, что оценку качества развития процесса не всегда следует связывать с вычислением функции-критерия *F* лишь в точке End, т.е. в конце отслеживания траекторий. В зависимости от специфики решаемой задачи для анализа может требоваться анализ качества и в других точках траектории R_{τ_i} в моменты τ_i, $1 \le i \le k$, где k— число отслеживаемых точек:

$$R_{\tau_i} = \left(\tau_i, F\left(\tau_i, a_1(\tau_i), \dots, a_n(\tau_i)\right), a_1(\tau_i), \dots, a_n(\tau_i)\right), 0 \le \tau_i \le T,$$

Оценивание в точке $R_{\tau_1} = End$ при $k = 1$, называется *финальным*. Оно связывается с итогами развития. Это тот случай, когда дополнительные замеры не предусматриваются.

Когда требуется получать информацию в ходе выполнения процесса в фиксированных точках траектории, т.е. при $k > 1$, оценивание называется *расширенным*. Оно полезно,

например, для процессов существенным аспектом качества которых является равномерность развития, когда скачки тех или иных факторов не допускаются или нежелательны. Это характерно для моделирования при разработке приборов и устройств. В том или ином виде степень неравномерности должна отражаться в критерии.

Если есть потребность и возможность получать оценки, вычисляя их в любой точке, то оценивание называется *непрерывным*. Пример с паровым котлом иллюстрирует один из вариантов такого оценивания. Из него видно, что для анализа качества процесса не обязательно передавать всю информацию, извлекаемую при непрерывном оценивании, — она может использоваться, в частности, для локальных корректировок траектории. Отметим также, что непрерывное оценивание позволяет контролировать ход модельного выполнения процесса с заданным уровнем точности.

Финальное, расширенное и непрерывное оценивания могут сводиться к друг другу. Расширенное оценивание есть частный случай непрерывного, финальное — вырожденный случай расширенного, а потому и непрерывного. Имея расширенное или непрерывное оценивание, легко подобрать новый критерий развития, имитирующий получение оценочной информации в промежуточных точках в ходе вычисления финальной оценки по новому критерию.

К сожалению, как часто бывает, принципиальная эквивалентность ситуаций никак не связана с возможностью одинакового оперирования в них на практике. Поэтому при реализации оценивания нужен тот вид критерия, который адекватен задаче моделирования и предметной области. Так, для решения оптимизационных задач финальные критерии более удобны, а когда требуется следить за колебаниями значений некоторых факторов, разумно расширенное или непрерывное оценивание.

Поскольку мы ограничиваемся формальным рассмотрением подхода, последнее замечание позволяет в дальнейшем говорить преимущественно о финальном оценивании, имея ввиду возможность подмены функции-критерия, т.е. использования вместо F, функции F_e, которая требуемую информацию извлекает из всей траектории, но предъявляет ее для анализа лишь в конце отслеживаемого развития процесса.

4. Конусы траекторий

Множество всех возможных траекторий, исходящих из точки начала тренда $Begin$, можно рассматривать как конус с вершиной в этой точке и с основанием, содержащим точку End и лежащим на плоскости $t = T$, перпендикулярной временной оси. Вектор $(Begin, End)$ задает ось этого конуса.

Это слишком большой конус, т.к. он включает в себя заведомо избыточные кривые. По этой причине разумно выделить в нем часть, огибающую все допустимые траектории. В результате для фиксированного начала всех таких траекторий образуется конус, в качестве основания которого на плоскости $t = T$ можно выбрать область $DResult$ окончаний допустимых траекторий. Поскольку начальная точка может выбираться в области $DBegin$ произвольно, рассматривается объединение всех таких конусов. В результате строится усеченный конус, который содержит все допустимые траектории развития процесса. Его основаниями являются $DBegin$ и $DResult$,[2] лежащие на плоскостях $t = 0$ и $t = T$. В качестве оси этого конуса естественно выбрать луч, которому принадлежит вектор тренда $(Begin, End)$. Вершина оси имеет отрицательную временную координату, что допускает вполне разумную трактовку: отсекаемая основанием $DBegin$ часть конуса рассматривается как область результатов допустимых траекторий некоторого уже завершенного процесса, в дальнейшем именуемого *предысторией отслеживаемого процесса развития*.

Продолжения траекторий за пределы плоскости $t = T$ ни при каких обстоятельствах не могут влиять на продвижение их к области допустимости. Не всякое развитие приводит к тому, что точка End достигается: возможно выполнение процесса, которое не приводит в область результатов, что означает недопустимость траектории. При моделировании имеет смысл рассматривать их также и, в частности, выяснять степень приближения траекторий к цели.

Задать вид оснований конусов и их поверхностей не представляется возможным, поскольку они существенно зависят от интерпретации модели. Но это и не нужно — для анализа достаточно знать соотношения и интервалы значений факторов $a_1, \ldots, a_n$, которые влияют на развитие процесса. То же можно сказать о допустимых траекториях. Для анализа требуется знать не их конкретный вид, а лишь периоды их *стационарного поведения* и смены тенденций, т.е. перехода от одного стационарного периода к другому. Если траекторию можно было бы считать достаточно гладкой функцией, то стационарность означала бы постоянство всех частных производных, а переходы между периодами стационарности — нарушениями этого свойства. Это можно сделать, используя проецирования траекторий на различные плоскости. В следующем разделе обсуждается проецирование, которое может приводить к образованию S-кривых развития.

[2] Точнее было бы говорить о минимальных связных областях, содержащих все точки из $DBegin$ и $DResult$, соответственно. Минимальность можно понимать, как выбор области с самой короткой границей, с самой малой площадью или в каком-либо ином смысле.

5. Проекция траектории развития на плоскость (время, критерий)

Как показано в [4], цикл развития процесса можно представлять в виде S-кривой, на которой выделяются участки *вживания* (детство), *роста* (расцвет) и *исчерпания ресурсов* (старость). Эта участки соответствуют секторам различного влияния факторов, определяющих критерий развития. Если кривая траектории является витком спирали, раскручивающимся в конусе допустимых траекторий между основаниями $DBegin$ и $DResult$, то ее проекция на плоскость (t, F) оказывается S-кривой. Правильная S-кривая получается, когда траектория не допускает возвратов по времени, что соответствует интуитивному представлению развития процесса.

На S-кривой определены точки ее начала, минимума, максимума и окончания. Их прообразы на траектории разделяют указанные выше участки:

- начало спирали, прообраз минимума — *вживание,*
- прообразы минимум и максимум — *рост,*
- прообраз максимум и конец витка — *исчерпание ресурсов.*

Участок роста естественным образом делится на участки *первичного* и *затухающего роста*, которые разделяются прообразом пересечения S-кривой с проекцией тренда $(Begin, End)$, т.е. идеальной траектории. Плоскости, построенные в указанных точках перпендикулярно оси t, делят траекторию на четыре части.

Представленные положения иллюстрирует рис. 1, показывающий проекцию траектории процесса в соответствии с введенными выше обозначениями. Заштрихованным прямоугольником выделен образ значений функции F на основаниях $DBegin$ и $DResult$ конуса всех допустимых траекторий, начинающихся и заканчивающихся на этих основаниях. Серым цветом окрашен конус допустимых траекторий, начинающихся в точке $Begin$ тренда — его проекция выделена жирным отрезком. Область допустимости таких траекторий указана прямоугольником с точечной заливкой (он наложен на проекцию основания $DResult$ конуса и для наглядности несколько сдвинут вправо). Проекция некоторой допустимой траектории показана жирной кривой. Вертикальными штриховыми линиями отмечены моменты бифуркации, т.е. фазовых переходов процесса. Конус предыстории выделен пунктиром.

Представление цикла развития процесса в виде S-кривой, т.е. как витка спирали не является ограничением, поскольку процессы с более чем одним витком можно формально, а чаще всего и по существу считать составными, т.е. состоящими из двух и более процессов. Рассмотрение составных процессов полезно еще и потому, что реальные

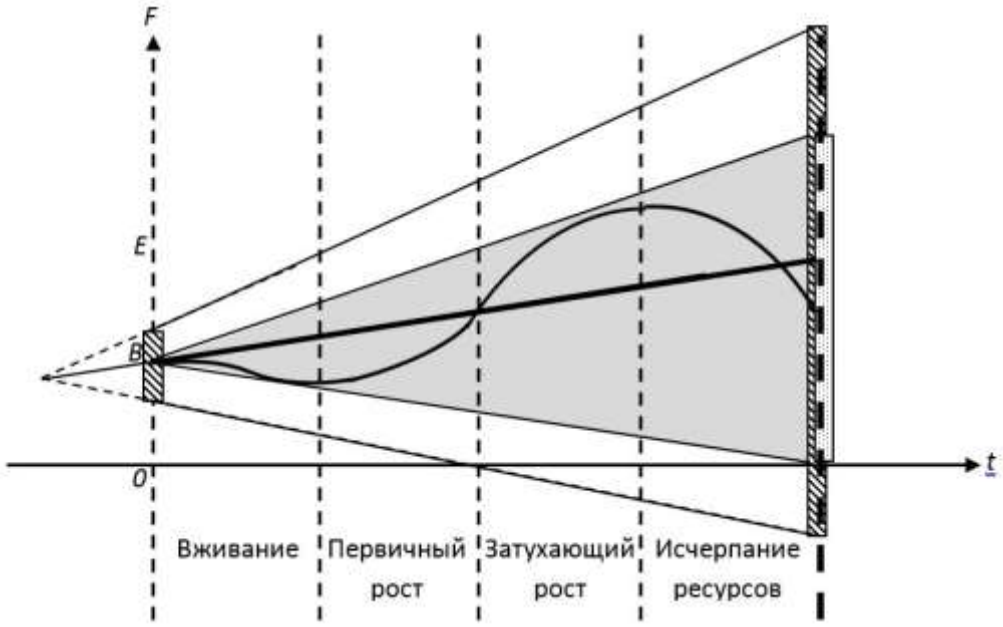


Рис. 1. Проекция траектории развития проекта на плоскость (*t*, *F*)

процессы всегда состоят из других процессов, т.е. подпроцессов, со своими осями развития, факторами влияния и критериями развития. Они формируют данный процесс, поддерживая его или противодействуя его развитию. В свою очередь, рассматриваемый процесс является подпроцессом системы, развитие которой естественно трактовать как интегрирующий процесс. Вопрос о том, как формируются системы процессов, относится к области конкретизирующей интерпретации модели. На абстрактном уровне можно указать лишь на следующие положения:

1. Интегрирующий процесс имеет те же стадии, периоды развития, которые определяются (своей) S-кривой. Если для него осмысленно определение оси развития, то период роста разбивается на первичный и затухающий рост.
2. S-кривая интегрирующего процесса может быть представлена как огибающая семейства составляющих его подпроцессов, или, что то же, можно подобрать критерий развития, выстраивающий процесс в соответствии с положением 1.
3. Системы процессов, возникающие в связи с декомпозицией изучаемого процесса и включением его в системы более высоких уровней, обладают фрактальными свойствами, что позволяют использовать для их анализа общую методологическую основу фрактального анализа.

Рассмотрение S-кривых как проекций конусов траекторий развития может оказаться полезным для преодоления главного недостатка этого инструмента — неразличимость значимости факторов для развития на разных фазах. В то же время, не следует ожидать от

этого слишком многого. Вопрос о механизмах, благодаря которым интеграция процессов приводит к образованию процесса, подчиняющегося указанным условиям, остается открытым. По-видимому, его уместно ставить лишь по отношению к области применения модели развития.

S-кривая является проекцией траектории специального вида, соответствующего четырем указанным выше периодам (участкам) развития. Ничто не препятствует рассмотрению процессов, траектории которых отражают другие соглашения о развитии. Так, при проектировании любого прибора или устройства в траектории его работы различаются три фазы: включение, стационарное функционирование и выключение. Каждая из этих фаз имеет свои группы факторов, влияющих на качество использования. Структура фаз, а значит, и вид траектории развития зависят как от особенностей элементарных актов работы устройства, так и от целей моделирования.

Для иллюстрации траектории развития, проекция которой отлична от S-кривой, рассмотрим задачу определения срока эксплуатации лампы накаливания. Здесь процесс развития — серия элементарных актов, каждый из которых состоит из включения, горения и выключения, связанных с ростом усталости — фактором, интегрирующим физические свойства материалов, от которых зависит работоспособность лампы. Усталость резко нарастает при включении и выключении и плавно растет на фазе горения. Когда этот фактор становится больше порогового значения, лампа перегорает. Процесс нарастания усталости лампы выходит за рамки нашего обсуждения — можно считать, что данные, необходимые в решении задачи о сроках эксплуатации, получены путем аспектного моделирования.

Адекватная модель для данной задачи связывается с построением траектории процесса развития как многократного повторения элементарных актов, пока усталость не достигнет порогового значения. Критерием качества развития может служить время работы лампы, накапливаемое в ходе выполнения элементарных актов с финальным оцениванием. Для анализа хода процесса более информативный критерий — уровень накопленной усталости в каждой точке траектории или разность порогового значения с текущей усталостью. В этом случае появляется возможность отслеживать фазовые переходы траектории. Тренд развития начинается в точке с нулевой временной координатой и начальными значениями параметров аспектной модели усталости. Конечная точка тренда задается с пороговым значением усталости и варьируется вокруг желаемого гарантийного срока эксплуатации. Проекция траектории развития на плоскость (время, критерий) для первого критерия мало информативна, тогда как следуя второму критерию, появляется возможность сравнивать рост усталости на разных элементарных актах.

6. Технологические аспекты моделирования развития

Задача изучения развития представлена в предыдущих разделах без привлечения содержательной информации о моделируемых процессах. Введенные понятия отражают в точности то, что может быть сделано, оставаясь на уровне формализованной абстракции. В то же время, эта абстракция позволяет указать на архитектурные решения, которые при надлежащем наполнении средствами, связанными с содержанием исследования, могут стать основой практического комплекса инструментальной поддержки изучения развития. В данном разделе обсуждаются главные из таких решений.

Основная концепция инструментального комплекса заключается в следующем. Развитие процесса связывается с геометрическим представлением его траекторий, направляемых функцией критерия развития. Для отслеживания траекторий используется содержательная информация, которая определяется априорно постулируемыми закономерностями и аспектным моделированием. В комплексе должна быть обеспечена возможность связи геометрической и содержательной информации для свободного доступа к ней. Эти положения определяют главные требования к архитектуре комплекса.

Средства комплекса, предлагаемые для модельного исследования, подразделяются на *базовые*, которые используются во всех моделях, и *специализированные*, отражающие специфику предметной области. В состав базовых средств включены заготовки в виде абстрактных классов, методов и интерфейсов, которые для специализированных средств играют роль технологических шаблонов: с помощью механизма наследования абстрактные заготовки конкретизируются и, тем самым, становятся пригодными для использования в программах моделей.

Комплекс предусматривает следующие возможности описания поведения процесса:

- Поддержка определения факторов развития процесса и пространства факторов посредством соглашений о них как об информационных объектах.
- Абстрактные средства геометрического оперирования представлением процесса как конуса его траекторий в многомерном пространстве факторов, т.е. построение оснований конусов, трендов траекторий и конкретных траекторий, а также сечений конусов и различных фигур, допускающих их трактовку как свойств объектов предметной области. Набор средств геометрического оперирования может пополняться в зависимости от потребностей конкретного моделирования. Он регламентируется заданным критерием развития, трендом, известными соотношениями между факторами и другими особенностями.

- Абстрактные шаблоны для задания аспектных моделей, которые служат регламентирующим стандартом представления зависимостей между факторами и изменения факторов во времени в содержательных аспектных моделях.

Для реального использования этих и других средств моделирования развития инструментальный комплекс должен обеспечивать ввод, вывод и хранение информации, стандартизованные элементы управления и другие общесистемные возможности. В частности, в качестве очень важных компонентов комплекса рассматривается подсистема сопоставления результатов модельных расчетов для разных вариантов развития процесса. Обсуждение общесистемного аспекта инструментария моделирования выходит за рамки настоящей работы — это особая тема, имеющая лишь косвенное отношение к задаче формализованного представления моделей развития.

Завершая обсуждение эскиза технологических аспектов моделирования развития, отметим, что предлагаемый подход хорошо сочетается с методами имитационного моделирования динамики взаимодействия процессов. При их применении нужно на ряду с основным процессом развития определить аспектные модели как систему процессов, для которых основной процесс рассматривается в качестве интегрирующего компонента. В этой системе управление задается с помощью событийного механизма, который согласуется с моделью развития следующим образом.

Для аспектных процессов определяются события, генерируемые ими, и реакции на них других процессов в связи с фазовыми переходами. При имитации производства и потребления ресурсов, к таким событиям относятся начало и конец потребления некоторого ресурса, начало и конец его производства, приостановка активности процесса при исчерпании ресурса, начало активности процесса и его завершение. Примеры реакций — возобновление активности процесса при появлении ресурса, выбор нового поведения. Общая для всех процессов реакция — пересчет критериев развития. Предметом анализа при имитационной динамической стратегии являются протоколы событий процессов с целью выяснения фазового поведения интегрирующего процесса. Вопросы организации имитационного моделирования, основанного на событийном взаимодействии процессов, можно найти в работе [6].

7. Заключение

Предлагаемая формализация моделирования развития процессов построена на их геометрическом представлении. Предпосылкой подхода является традиция использовать S-кривые при анализе развития. В отличие от обычных S-кривых, предложенный подход позволяет учитывать отдельные факторы и их сочетания, в той или иной степени

влияющие на развитие. Это достигается путем рассмотрения факторов как элементов построения конуса возможных траекторий и отслеживания динамики критерия развития. Другие варианты проецирования траекторий и, в частности, построение проекций на плоскости различных факторов дают возможность отслеживания значимости влияния факторов на качество развития процесса. Эта задача выходит за рамки рассмотрения настоящей работы. Учитывая ее особую важность, мы намерены посвятить другим вариантам проецирования специальное исследование.

Геометрическое представление развивающихся процессов согласуется с использованием иных моделей процессов и явлений различной природы, которые продуцируют информацию о факторах развития изучаемого процесса. В частности, правомерно применение методов статистического анализа и математического моделирования. Предоставление этой информации возможно не только до проведения расчетов по модели развития, но и в динамике выполнения расчетов.

Все построения, связанные с предлагаемым подходом, могут быть использованы при изучении развития процессов, которые подчиняются закономерностям, отличным от S-кривых, но сходных с ними с точки зрения управления. В частности, по этой причине подход можно рекомендовать для использования при проектировании приборов и устройств, рассматривая такие критерии качества развития как стабильная работоспособность за гарантированные сроки эксплуатации, потребление ресурсов и др.

Список литературы

1. Арнольд В. И. «Жесткие» и «мягкие» математические модели. // М.: МЦНМО, 2004. — 32 с.
2. S-кривая (S-Curve) // Глоссарий PMBOK. URL: http://www.pm-glossary.com/ (дата обращения: 6.01.2016).
3. Крамер А. Размышления *над S-кривой* URL: http://www.metodolog.ru/01490/01490.html (дата обращения: 6.01.2016).
4. Бадулин Н.А. «Улитка инноваций», «тройная спираль» и другие круговые процессы в экономике // В кн.: Инноватика-2012: Сборник материалов VIII Всероссийской школы-конференции студентов, аспирантов и молодых ученых с международным участием (25-28 апреля 2012 г.). Т.1, 2012. — с. 7 – 29.
5. Ильин В.П., Скопин И.Н. О производительности и интеллектуальности суперкомпьютерного моделирования. — Программирование, 2016, № 1. — с 10 – 25.
6. Скопин И.Н. Иерархичность и моделирование развивающихся систем // В кн.: Проблемы системной информатики: Сб. науч. тр. / Под ред. В.Н. Касьянова. — Новосибирск: Ин-т систем информатики им. А.П. Ершова СО РАН, 2010. — с. 188 – 214.

УДК 004.82

Сравнение способов представления неточных значений в методе недоопределённых вычислений

Сидоров В.А. (Институт систем информатики СО РАН)

Метод недоопределённых вычислений является одним из подходов для решения задачи удовлетворения ограничений. Его базовым понятием является понятие вида недоопределённости, которое представляет собой способ задания неточного значения в виде множества. В статье рассматриваются различные виды недоопределённости, приводятся их формальные определения, выполняется сравнение и выделяются их основные свойства.

***Ключевые слова**: задача удовлетворения ограничений, недоопределённые вычисления, интерввальное значение, мультиинтервал.*

1. Введение

Задача удовлетворения ограничений (**ЗУО**) является популярным подходом для решения нестандартных и сложных вычислительных задач. ЗУО базируется на понятиях «*ограничение*» и «*переменная*», с помощью которых можно описать широкий класс задач. Впервые задача удовлетворения ограничений была сформулирована в начале 70х годов [11, 14]. К настоящему времени разработано множество методов решения ЗУО и с их помощью реализованы как отдельные приложения, так и целые системы программирования, успешно применяемые на практике.

В начале 80х годов А.С. Нариньяни был предложен *метод недоопределённых вычислений* [2]. Метод недоопределённых вычислений является одним из методов решения задачи удовлетворения ограничений и может быть описан следующим образом:

- Модель (описание задачи) состоит из набора переменных и ограничений.
- С каждой переменной связывается множество допустимых значений (*недоопределённое значение*).
- Ограничения связывают переменные и позволяют вычислять новые (недоопределённые) значения для своих аргументов.
- Алгоритм вычислений имеет потоковый характер, выражающийся в том, что изменение значения переменной активирует (вызывает к исполнению) ограничения,

связанные с данной переменной. В свою очередь, исполнение (удовлетворение) ограничения может вызывать изменение связанных с ним переменных.

Важнейшей особенностью метода недоопределённых вычислений является его универсальность — метод позволяет решать задачи, содержащие как непрерывные, так и дискретные значения переменных. Более того, тип переменных может быть полностью произвольным; для работы метода достаточно определить операцию пересечения для выбранного представления множества значений этой переменной. Таким образом переменная описывается не только её типом данных, но и способом представления множества её значений, который мы будем называть *видом недоопределённости*.

От выбора вида недоопределённости сильно зависит как скорость вычисления отдельных ограничений, так и точность представления множества значений. Однако сравнение видов недоопределённости обычно ограничивается утверждениями «интервальное представление грубее перечисления», «перечисление слабее мультиинтервального представления» и т.д. (например, в статье [6]) - которые часто некорректны.

В данной работе приводятся формальные определения основных видов недоопределённости, используемых при решении задачи удовлетворения ограничений и проводится их сравнительный анализ.

1. Задача удовлетворения ограничений и метод недоопределённых вычислений

Опишем область применения видов недоопределённости: приведём краткое описание метода недоопределённых вычислений и задачи удовлетворения ограничений.

1.1 Задача удовлетворения ограничений

Задача удовлетворения ограничений может быть сформулирована с различной степенью детализации [3, 9, 10]. Далее в статье будем использовать следующие определения:

Задача удовлетворения ограничений M – это тройка $\langle Var, D, C \rangle$, где:

- $Var = \{v_1, \ldots, v_n\}$ – множество переменных.
- $D = \langle D_{v1}, \ldots, D_{vn} \rangle$ – набор множеств значений переменных.
- $C = \{c_1, \ldots, c_k\}$ – множество ограничений. Ограничение $c\left(R_c, \langle v_{c1}, \ldots, v_{cm} \rangle\right)$ задаётся некоторым отношением $R_c(D_1, \ldots, D_m) \subseteq D_1 \times \ldots \times D_m;\ D_i \subseteq D_{vi}$, которое определено

над множествами значений переменных $v_{c1},\ldots,v_{cm}$ и является подмножеством картезианского произведения этих множеств.

Означивание переменной – это функция присваивания (множественного) значения переменной: $val(v,d):Var\times D\to D_v;\ D_v\in D$.

Набор множеств значений $\langle A_1,\ldots,A_m\rangle, A_i\subseteq D_i$ ***совместен*** на ограничении $c=\left(R_c,\langle v_{c1},\ldots,v_{cm}\rangle\right)$, если $\exists d_1\in A_1,\ldots,\exists d_m\in A_m$, такие что $\langle val(v_{c1},d_1),\ldots,val(v_{cm},d_m)\rangle\in R_c$.

Локально-совместное решение задачи удовлетворения ограничений – множество значений всех переменных, совместное на каждом ограничении модели.

Решение (глобальное) задачи удовлетворения ограничений — множество значений всех переменных, совместное на всех ограничениях модели одновременно.

Точное решение задачи удовлетворения ограничений — это глобальное решение $\langle A_1,\ldots,A_n\rangle$, состоящее из одно-элементных множеств: $A_i=\{x_i\};x_i\in D_{vi}$.

Отметим, что большинство алгоритмов решения ЗУО ищет локально-совместное решение, представленное множеством значений переменной, в то время как пользователю обычно нужно точное решение.

1.2 Метод недоопределённых вычислений

Для описания метода недоопределённых вычислений уточним ЗУО для множественного представления значений переменных.

Недоопределённое значение D_v^* переменной v является подмножеством множества значений D_v.

Экстенсиональное задание ограничения (в виде множества всех допустимых комбинаций значений аргументов), приведённое в определении ЗУО, заменяется на интенсиональное. Для этого для ограничения $c\left(R_c,\langle v_{c1},\ldots,v_{cm}\rangle\right)$ определяются ***функции интерпретации*** $f_{c,i}:D_{v1}^*\times\ldots,\times D_{vm}^*\to D_{vi}^*$, которые вычисляют новое значения каждого i-ого аргумента ограничения $x'_i:=f_{c,i}\left(x_1,\ldots,x_m\right)$, и удовлетворяют следующим свойствам:

- $f_{c,i}$ сужает множество значений x_i: $x'_i\subseteq x_i$.
- x'_i - максимальное совместное подмножество x_i. То есть, если $\exists y\subseteq D_{vi}^*$ такое, что $\langle x_1,\ldots,x_{i-1},y,x_{x+1},\ldots,x_m\rangle\subseteq R_c$ и $y\subseteq x_i$, то $y\subseteq x'_i$.

Заметим, что способ представления множества (а все аргументы и результат функции интерпретации являются множествами) сильно влияют как на реализацию самой функции, так и на её результат.

Для описания алгоритма работы метода недоопределённых вычислений введём дополнительные обозначения:

- $d(t) \subseteq \langle D_{v1}^*, \ldots, D_{vn}^* \rangle$ - множество значений всех переменных на шаге t.
- $d(t,i) \subseteq D_{vi}^*$ - множество значений i-ой переменной на шаге t.
- $Arg(c) = \{v_{c1}, \ldots, v_{cm}\}$ - множество переменных, являющихся аргументами ограничения c.
- $d(t)|_c = \langle d(t,c1), \ldots, d(t,cm) \rangle$ - множество значений всех переменных, являющихся аргументами ограничения $c(R_c, \langle v_{c1}, \ldots, v_{cm} \rangle)$ на шаге t.

Вход

Множество ограничений $C = \{c_1, \ldots, c_k\}$, заданных с помощью функций интерпретаций; множество переменных $Var = \{v_1, \ldots, v_n\}$; начальные значения переменных $\langle D^*_{v1}, \ldots, D^*_{vn} \rangle$.

Выход

Новые значения переменных $\{v_1, \ldots, v_n\}$.

Алгоритм

$t := 0$

// Инициализация начальных значений переменных d(t) и очереди ограничений Q(t)

$d(t) := \langle D^*_{v1}, \ldots, D^*_{vn} \rangle$; $Q(t) := C$

// Проверка условия останова:

// 1. Очередь ограничений не пуста

// 2. Значения всех переменных не пусты

DO WHILE $Q(t) \neq \varnothing \wedge \forall i : d(t,i) \neq \varnothing$

// Выбор ограничения из очереди

$c \in Q(t)$

// Удовлетворение ограничения

$$d(t+1,i) := \begin{cases} d(t,i)\ if v_i \notin Arg(c) \\ f_{c,j}(d(t)|_c)\ if v_i \in Arg(c) \wedge i = cj \end{cases}$$

// Выделение изменившихся переменных

$\{v'_i\} : v'_i \in Var;\ d(t+1,i) \neq d(t,i)$

// Выделение связанных с ними ограничений

$\{c'_j\} : c'_j \in C;\ Arg(c'_j) \cap \{v'_i\} \neq \varnothing$

// Добавление новых ограничений в очередь

$Q(t+1) := Q(t) \cup \{c'_j\} \setminus \{c\}$

// Следующий шаг

$t := t + 1$

END DO

Таблица 1. Алгоритм решения ЗУО методом недоопределённых вычислений.

Если в результате работы алгоритма множество значений любой из переменных стало пусто, то система несовместна. Иначе, множество текущих значений переменных является локально-совместным.

2. Виды недоопределённости

Так как метод недоопределённых вычислений работает с множествами значений, то важнейшим вопросом является способ представления множества, называемый *видом недоопределённости*.

Вид недоопределённости SD переменной v это множество подмножеств значений из области допустимых значений D_v, удовлетворяющее следующим свойствам:

1. $SD(D_v) \subseteq 2^{D_v}$.
2. $\varnothing \in SD(D_v),\ D_v \in SD(D_v),\ \forall x \in D_v, \{x\} \in SD(D_v)$.
3. $x_1, x_2 \in SD(D_v) \Rightarrow x_1 \cap x_2 \in SD(D_v)$.

Свойство 1 говорит, что вид недоопределённости является способом представления множества.

Свойство 2 задаёт необходимый набор подмножеств:

- пустое множество (для индикации несовместности текущего значения переменной с одним из ограничений);
- всё множество допустимых значений переменной (для задания начального значения и области определения переменной);
- все точные значения (для возможности нахождения любого точного решения из области определения переменной).

Свойство 3 ограничивает набор используемых алгоритмов решения ЗУО такими, которые монотонно сужают множество допустимых значений переменных.

Для видов недоопределённости естественным образом задаётся частичный порядок:

Вид недоопределённости $SD_1(D)$ ***более полный***, чем $SD_2(D)$ ($SD_2(D) \leq_P SD_1(D)$), если $SD_2(D) \subseteq SD_1(D)$.

Виды недоопределённости $SD_1(D)$ и $SD_2(D)$ ***эквивалентны*** ($SD_2(D) =_P SD_1(D)$), если $SD_2(D) = SD_1(D)$.

Вид недоопределённости $SD(D)$ ***полный***, если $\forall X \subseteq D \Rightarrow X \in SD(D)$.

Свойство полноты влияет на эффективность вычислений:

- Нахождение точного решения задачи удовлетворения ограничений является NP-сложной задачей.
- Для нахождения локально-совместного решения разработан ряд эффективных алгоритмов с полиномиальной сложностью [13].
- Соответственно, более полный вид недоопределённости позволяет более точно представлять локально-совместное решение и, как следствие, выполнять более медленный поиск точного решения ЗУО на более узком множестве значений.

Рассмотрим некоторые виды недоопределённости, реализованные в системах Nemo+ и NemoNext [1, 4].

3.1 Одиночное значение

Вид недоопределённости $Single$:

$Single(D) = \{\varnothing, D, \{x\} \mid x \in D\}$.

Single является минимальным видом недоопределённости: любое дальнейшее уменьшение количества элементов множества приведёт к нарушению свойства 2 определения вида недоопределённости. Соответственно, *Single* является подмножеством любого другого вида недоопределённости.

На практике, *Single* часто используется в следующих случаях:

- Работа с узкими областями значений: $|D| \approx 2$.
- Представление точных констант.
- Моделирование специфических сетевых задач, таких как вычислительные сети Тыугу [7].

3.2 Интервал

В системах, решающих задачу удовлетворения ограничений для непрерывных областей [3, 8, 12], обычно используется интервальное представление множества значений.

Пусть на D задано отношение частичного порядка. Обозначим $[low, upp]_D = \{x \mid low, upp, x \in D; low \le x \le upp\}$. Тогда вид недоопределённости *Interval* определяется следующим образом:

$$Interval(D) = \{\varnothing, [low, upp]_D \mid \forall low, upp \in D; low \le upp\} =$$
$$= \{\varnothing, \{X \mid X \subseteq D; \exists low, upp \in X : \forall x \in X \quad low \le x \le upp\}\}.$$

Отметим, что представление в виде интервала возможно, только если D является решёткой: для каждого подмножества $X \subseteq D$ существует точная верхняя и нижняя грани:

$$\forall X \subseteq D; \exists \sup(X) \in D, \exists \inf(X) \in D : \forall x \in X \ \inf(X) \le x \le \sup(X).$$

Это свойство необходимо для замкнутости *Interval* относительно операции пересечения:

Утверждение. *Interval* замкнут относительно операции пересечения.

Пусть $x, y \in Interval(D)$, $x = \lfloor x_{low}, x_{upp} \rfloor$, $y = [y_{low}, yupp]$. Тогда:

$$x \cap y = \{a \mid a \in D, x_{low} \le a \le x_{upp}\} \cap \{b \mid b \in D, y_{low} \le b \le y_{upp}\} = \{c \mid c \in x, c \in y\} =$$

$$= \{c \mid c \in D, x_{low} \le c \le x_{upp}, y_{low} \le c \le y_{upp}\} =$$

$$= \{c \mid c \in D, \inf(x_{low}, y_{low}) \le c \le \sup(x_{upp}, y_{upp})\} \in Interval(D).$$

Последнее равенство вытекает из определения точной верхней и нижней грани; $\inf(x_{low}, y_{low}) \in D$, $\sup(x_{upp}, y_{upp}) \in D$, т. к. D – решётка.

3.3 Перечисление

В системах, решающих задачу удовлетворения ограничений для дискретных областей [15], допустимое значение обычно представляют в виде прямого перечисления значений.

Вид недоопределённости *Enum*:

$$Enum(D) = \{\varnothing, X \mid X \subseteq D\}.$$

Вид недоопределённости *Enum* является полным (так как содержит все подмножества D).

Теоретически, *Enum* может применяться к любым, не только к счётным D. Но на практике, *Enum* не применим как для непрерывных, так и для больших дискретных множеств значений из-за больших затрат на хранение и обработку значений. Поэтому при реализации перечисления размер множества принудительно ограничивают, получая другой вид недоопределённости - *ограниченное перечисление*.

3.4 Ограниченное перечисление

Если количество элементов *Enum* ограничиваются некоторым числом N, то получим следующий вид недоопределённости:

Вид недоопределённости $Enum_N$:

$$Enum_N(D) = \{\varnothing, D, X \mid X \subseteq D, |X| < N\}$$

Заметим, что $Enum_N$ совпадает с $Enum$ при $N \geq |D|$. В противном случае, $Enum_N$ не является полным, так как любое подмножество из более чем N элементов (кроме всего D) не представимо в $Enum_N$.

3.5 Мультиинтервал

Представление значения в виде мультиинтервала было введено в работах [5, 6] для того, чтобы совместить преимущества *Interval* (компактность, эффективность, работа с непрерывными областями) и *Enum* (полнота представления).

Вид недоопределённости *Multiinterval*:

$$MultiInterval(D) = \{X \mid X = \cup X_i,\ X_i \in Interval(D)\}.$$

Вид недоопределённости *Multiinterval* является полным (так как $Enum \subseteq Multiinterval$)

Несмотря на то, что мультиинтервал является более компактным представлением, чем $Enum$, на практике также приходится ограничивать количество составляющих его подинтервалов.

3.6 Ограниченный мультиинтервал

Множественное представление $Multiinterval_N$:

$$MultiInterval_N(D) = \{X \mid X = \cup X_i,\ X_i \in Interval(D),\ i \leq N\}.$$

Такое представление не является видом недоопределённости, так как не замкнуто относительно операции пересечения при $N > 1$: $x, y \in MultiInterval_N(D) \Rightarrow x \cap y \in MultiInterval_{2*N-1}(D)$. Этот факт является причиной принципиальных проблем мультиинтервального представления:

- Автоматическое увеличение количества подинтервалов в процессе решения ЗУО.
- Неоднозначность представления множества значений с помощью $Multiinterval_N$.

3.7 Смешанный вид недоопределённости

Альтернативный способ совместить преимущества интервального представления и перечисления представляет собой комбинацию двух значений, одно из которых является $Interval(D)$, другое – $Enum_N(D)$:

Вид недоопределённости $Mixed_N$:

$$Mixed_N(D) = \{I, E \mid I \in Interval(D),\ E \in Enum_N(D)\} = \{X \mid X \subseteq I \cap E\}.$$

3.8 Вид недоопределённости для составных значений

Отдельно следует рассмотреть случай, когда значение переменной содержит внутри себя отдельные поля (является составным): $D = D_1 \times \ldots \times D_n = \{d_1, \ldots, d_n \mid d_1 \in D_1, \ldots, d_1 \in D_n\}$.

Для таких значений можно определить специальный вид недоопределённости *Struct* :

$Struct(D) = SD_1(D_1) \times \ldots \times SD_n(D_n) = \{x_1, \ldots, x_n \mid x_i \in SD_i(D_i)\}$.

То есть, недоопределённое составное значение является комбинацией недоопределённых значений его элементов.

3. Свойства видов недоопределённости

Рассмотрим некоторые свойства видов недоопределённости.

4.1 Мощность видов недоопределённости

Если D - конечное множество, то для каждого вида недоопределённости можно явно выписать его мощность:

- $|Single(D)| = |D| + 2$.
- $|Interval(D)| = 1 + |D| * (|D| + 1)/2$.
- $|Enum(D)| = 2^{|D|}$.
- $|Enum_N(D)| = 2 + \sum_{i=1}^{i=N} C_{|D|}^i$.
- $|Mixed_N(D)| = 2 + \sum_{i=1}^{i=N} C_{|D|}^i + |D| * (|D| - N)/2$.
- $|Multiinterval(D)| = 2^{|D|}$.

Заметим, что при $|D| = 2$ (например, $D = \{false, true\}$) мощности всех видов недоопределённости совпадают.

4.2 Полнота видов недоопределённости

В общем случае мощность вида недоопределённости оценить невозможно, но можно сравнить виды недоопределённости по полноте. Построим иерархию видов недоопределённости по этому критерию.

Раннее мы уже отметили некоторые свойства видов недоопределённости:

- Вид недоопределённости *Single* является минимальным по полноте.
- Вид недоопределённости *Enum* является полным.

- Вид недоопределённости *Multiinterval* также является полным.
- Виды недоопределённости $Enum_N$ и *Interval* не сравнимы по полноте:

Пусть $D_x \subset ; x = \{1,2,4,5,6\}$. Тогда $Interval\ (x) = [1,6\ ]$; $Enum_4(x) =$; $Enum_5(x) = x$. И, соответственно, $Enum_5(x) \subset Interval(x) \subset Enum_4(x)$.

Для различных пар видов недоопределённости можно выписать соотношения:

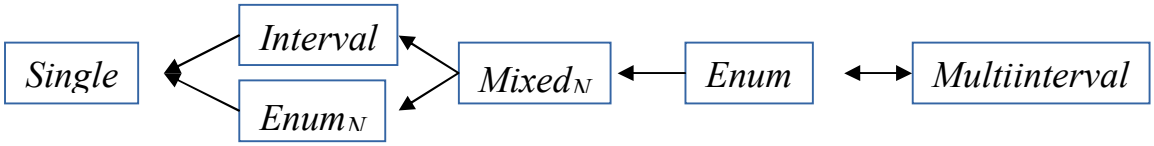
- $Single(D) \le_P Interval(D)$.
- $Interval(D) \le_P Enum(D)$.
- $Enum_N(D) \le_P Enum(D)$.
- $N \le M \Leftrightarrow Enum_N(D) \le_P Enum_M(D)$.
- $N = |D| \Rightarrow Enum_N(D) =_P Enum(D)$.
- $Interval\,(D) \le_P MuliInterval\,(D)$.
- $MultiInterval(D) =_P Enum(D)$.
- $Enum_N(D) \le_P Mixed_N(D)$.
- $Interval(D) \le_P Mixed_N(D)$.
- Для вида недоопределённости *Struct* в работе [16] приведены следующие соотношения:
  - $Single(D_1 \times \ldots \times D_n) \le_P Single(D_1) \times \ldots \times Single(D_n)$.
  - $Interval(D_1 \times \ldots \times D_n) =_P Interval(D_1) \times \ldots \times Interval(D_n)$.
  - $Enum(D_1 \times \ldots \times D_n) \ge_P Enum(D_1) \times \ldots \times Enum(D_n)$.

На основе приведённых отношений выстраивается иерархия:

- Если $|D| = 2$, то:

- Если $|D| = N$, то:

- В общем случае:



На схемах однонаправленная стрелка "$\longleftarrow$" соответствует отношению $\leq_P$, а двунаправленная стрелка "$\longleftrightarrow$" - отношению $=_P$,

5. Заключение

В статье были рассмотрены различные способы представления значения переменных для задачи удовлетворения ограничений. Построена иерархия способов представления по полноте.

Также отмечены некоторые практически значимые свойства видов недоопределённости:

- Для логического типа данных ($|D| = 2$) все способы представления множества эквивалентны.
- Представление множества значений в виде перечисления и в виде мультиинтервала эквивалентны.
- Представления множества в виде перечисления и в виде ограниченного перечисления обладают существенно разными свойствами. То же верно для мультиинтервала и ограниченного мультиинтервала.
- Иногда применяемый на практике способ представления в виде ограниченного мультиинтервала является некорректным и его рекомендуется избегать.

Полученные результаты могут использоваться как при формулировке задач, решаемых с помощью метода недоопределённых вычислений, так и при реализации программных продуктов на основе данного метода.

Список литературы

1. Загорулько Г.Б., Сидоров В.А., Телерман В.В. и др. НеМо+: Объектно-ориентированная среда программирования в ограничениях на основе недоопределённых моделей // КИИ'98. Шестая национальная конференция с международным участием. Сборник научных трудов в трёх томах. Том I. — Пущино, 1998. — С. 524–530.
2. Нариньяни А.С. Недоопределенность в системе представления и обработки знаний // Изв. АН СССР. Техническая кибернетика. – 1986. – № 5. – С. 8 – 11.
3. Семенов А.Л. Методы распространения ограничений: основные концепции // PSI'03/ИМРО. – Интервальная математика и методы распространения ограничений, 2003.

4. Сидоров В.А. Программирование в ограничениях с чёрными ящиками. — Новосибирск, 2003. — 39 с. (Препр. / ЗАО Ледас; N2).
5. Телерман В.В. Использование мультиинтервалов в недоопределённых моделях. // Тез. докл. X всесоюз. семинара " Параллельное программирование и высокопроизводительные системы: Методы представления знаний в информационных технологиях". — Киев, 1990.
6. Телерман В.В., Сидоров В.А., Ушаков Д.М. Интервальные и мультиинтервальные расширения в недоопределённых моделях // Вычислительные технологии №1. — Т. 2. — 1997. — С. 62–70.
7. Тыугу Э.Х. Концептуальное программирование. — Москва, 1984. — 255 с.
8. Хансен Э., Уолстер Дж.У. Глобальная оптимизация с помощью методов интервального анализа: Пер. с англ. / Под ред. С. Кумков. — М.-Ижевск: Регулярная и хаотическая динамика, 2012. — 520 с.
9. Щербина О.А. Удовлетворение ограничений и программирование в ограничениях. // Интеллектуальные системы. 2011. — Т. 15, вып. 1-4. — С. 54–73.
10. Christophe Lecoutre. Constraint Networks: Techniques and Algorithms. — ISTE Ltd and John Wiley & Sons Inc, 2009. — 573 p.
11. Huffman D.A. Impossible objects as nonsence sentences // Machine Intelligence. — 1971. — Vol. 6. — P. 295–323.
12. Hyvonen E. Constraint Resoning Based on Interval Arithmetic. // Proc. IJCAI. 1989. — P. 193–199.
13. Lecoutre C., Cardon S. A greedy approach to establish singleton arc consistency // Proc. IJCAI. 2005. — P. 199–204.
14. Mackworth A.K. Consistency in Networks of Relations. // Artificial Intelligence. — 1977. — N 8. — P. 99–118.
15. Tsang E. Foundation of Constraint Satisfaction. — London: Academic Press Ltd., 1993. — 405 p.
16. Ushakov D. Some Formal Aspects of Subdifinite Models. — Novosibirsk, 1998.— 23 p. (Preprint / A. P. Ershov Institute of Informatics Systems, Siberian Division of Russian Academy of Sciences; N 49).