УДК 004.4'2+811.93

КОНТЕКСТНАЯ ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

В.С. Выхованец
Институт проблем управления им. В.А. Трапезникова РАН
Россия, 117997, Москва, Профсоюзная ул., 65
E-mail: valery@vykhovanets.ru

Ключевые слова: методология программирования, технология программирования, грамматический разбор, понятийный анализ, система программирования
Key words: programming methodology, programming technology, parsing, conceptual analysis, programming system

Рассматривается контекстная технология программирования, основанная на создании специализированного языка для решения заданного класса прикладных задач путем понятийного анализа предметной области и отражения ее понятийной структуры в понятиях языка. Описываются средства для задания семантики определяемого языка.
CONTEXT PROGRAMMING TECHNOLOGY / V.S. Vykhovanets (V.A. Trapeznikov Institute of Control Sciences, 65 Profsoyuznaya, Moscow 117997, Russia, E-mail: valery@vykhovanets.ru). The paper considers a context programming technology based on developing a specialized language for a specific class of applications by concept analysis of problem domain and reflecting its conceptual structure in language concepts. The tools for specifying the semantics of the determined language are described.

1. Введение

Современные информационные технологии характеризуются возрастающей сложностью информационных систем, создаваемых в различных областях и предназначенных для хранения, обработки, поиска, распространения, передачи и предоставления информации. Накопленный опыт проектирования информационных систем показывает сложность и трудоемкость этого процесса, длительного во времени и требующего высокой квалификации участвующих в нем специалистов.

До сих пор проектирование информационных систем выполняется в основном на интуитивном уровне с применением неформализованных методов, основанных на искусстве проектировщиков, их практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках получаемых результатов. Кроме того, в процессе жизненного цикла информационная система подлежит постоянному изменению в соответствии с изменяющимися потребностями пользователей и развитием их представлений о предметной области, что еще более усложняет разработку и сопровождение таких систем.

Особую сложность при разработке имеют программные системы, реализующие тем или иным способом параллельную обработку данных на многих вычислительных узлах. Идея распараллеливания вычислений базируется на том, что большинство задач может быть разделено на набор меньших задач, которые

могут быть решены одновременно. Однако писать программы для параллельных систем сложнее, чем для последовательных, так как конкуренция за ресурсы представляет новый класс потенциальных ошибок в программном обеспечении, среди которых состояние гонки является самой распространенной. Взаимодействие и синхронизация между процессами представляют большой барьер для получения высокой производительности параллельных систем.

Известным недостатком современных технологий программирования является наличие семантического (смыслового) разрыва между содержательными представлениями относительно предметной области и теми средствами, которые заложены в языке программирования для решения прикладных задач. Иными словами семантический разрыв – это явление несоответствия решаемых задач тем средствам, которые используются для их решения.

Семантический разрыв является следствием различия понятий, реализуемых системой программирования и понятий, используемых при постановке и решении задач. По своей сути любой универсальный язык программирования навязывает разработчику информационной системы некоторую систему понятий, в то время как содержательные представления о предметной области с этими понятиями согласуются плохо или не согласуются вообще. Высокая стоимость информационных систем, их низкая надежность, объективная трудоемкость, интеллектуальная и технологическая сложность в большой степени являются следствиями семантического разрыва.

Для сокращения семантического разрыва используется повышение уровня абстракции языков программирования. Однако последнее снимает только часть проблем, не затрагивая существенным образом систему понятий языка программирования.

Настоящая статья посвящена контекстной технологии программирования, которая зародилась как объединение объектно-ориентированной и фортоподобной технологий [1, 2], осуществленное путем контекстной интерпретации слов языка программирования и на основе понятийного анализа предметной области. С целью сокращения семантического разрыва средствами контекстной технологии создаются специализированные (проблемные) языки, отражающие понятийную структуру предметной области в аспекте решаемых задач, на которых и осуществляется решение этих задач [3].

2. Содержательная постановка задачи

Семантический разрыв между формальной системой и содержательными представлениями относительно предметной области будем рассматривать как следствие различия понятий, предоставляемых этим формализмом, и понятий, используемых при постановке и решении прикладных задач. Например, широко известно, что любой универсальный язык программирования навязывает разработчику информационной системы некоторую систему понятий, в то время как содержательные представления о предметной области с этими понятиями согласуются плохо или не согласуются вообще.

Однако, если предоставить разработчику возможность выражать необходимые для него понятия и найденные им декомпозиционные схемы в виде специализированной формальной системы, которая предназначена для содержательного описания предметной области и заданного класса прикладных задач, а также описать семантику этой системы, то следует ожидать сокращения семантиче-

ского разрыва между предметной областью и средствами формальной спецификации этой области.

Понятно, что основной семантический разрыв между содержательными представлениями и вычислительным средством, используемым для решения прикладных задач, устранен быть не может. Однако предоставление возможности разработчику на каждом уровне спецификации предметной области и решаемых в ней задач выражать требуемую ему систему понятий и найденные декомпозиционные схемы в форме, близкой постановке и решению стоящих прикладных задач, позволит значительно облегчить преодоление основного семантического разрыва.

Для преодоления семантического разрыва между содержательными представлениями и языком моделирования (программирования) применим контекстную технологию, в рамках которой для каждого класса задач (и для каждого уровня описания) и в процессе решения этих задач будем создавать свой, присущий только этим задачам (и уровню описания) проблемный язык.

Для описания семантики проблемного языка применим принципы семантического замыкания и математической индукции, которые заключаются в использовании семантических категорий, которые определяются по мере необходимости, в процессе описания семантики определяемого языка и средствами самого языка. Для привязки самого нижнего уровня описания к целевой вычислительной платформе будем использовать базовые семантические категории, реализуемые командами и встроенными структурами данных этой платформы.

Таким образом, в результате приближения выразительных средств формальной системы к постановке и решению прикладных задач следует ожидать повышение качества и надежности программных систем, созданных по контекстной технологии. Предлагаемый подход основывается на допущении о том, что уже в процессе изучения предметной области и специфики прикладных задач, еще до начала формализации, создается система понятий и декомпозиционные схемы, наиболее приспособленные для постановки и решения этих задач.

3. Семантическое замыкание

В области современных информационных технологий до сих пор одной из самых трудных является задача преодоления семантического разрыва между содержательными представлениями относительно предметной области и используемыми для формализации этих представлений математическими моделями, выраженными, например, на некотором универсальном языке моделирования или программирования (рис. **1**).

Применяемые языки не имеют развитых средств определения новых языковых конструкций, задания их синтаксиса, семантики и прагматики. Последнее объясняется рядом методологических проблем, среди которых следует выделить:

- отсутствие единого теоретического базиса для структурирования знаний и несовершенство используемых для этого математических моделей (их дескриптивный, а не конструктивный характер);
- невозможность точной формальной спецификации предметной области и решаемых в ней задач (не разработаны средства определения семантики и прагматики формальных языков и высокоуровневых спецификаций предметной области).

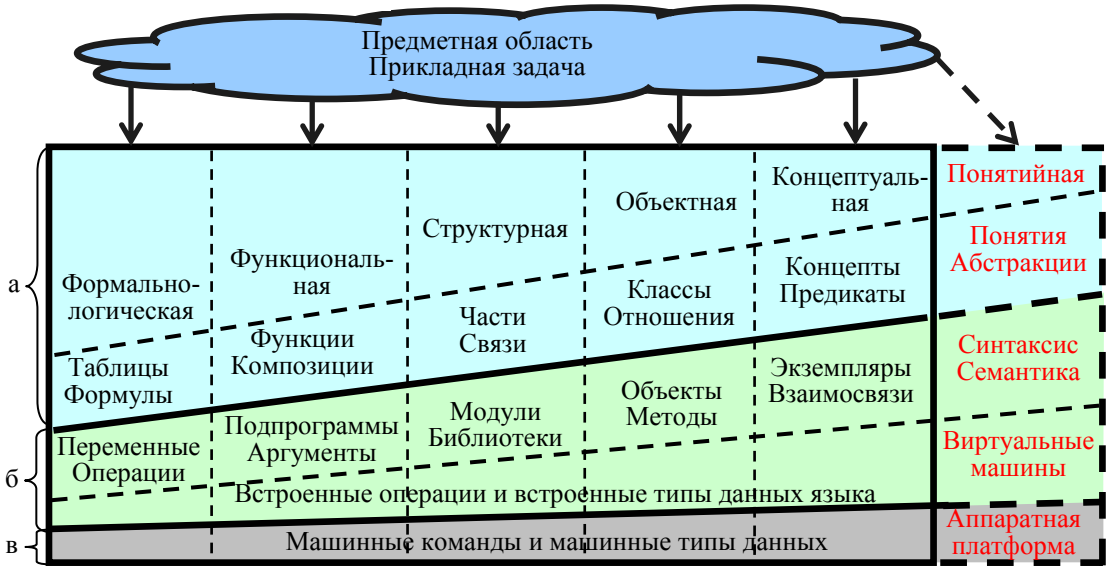


Рис. 1. Преодоление семантического разрыва (а – методология программирования, б – технология программирования, в – вычислительное средство)

3.1. Формальная семантика

С теоретической точки зрения формальный язык это произвольное множество строк конечной длины (рис. **2**).

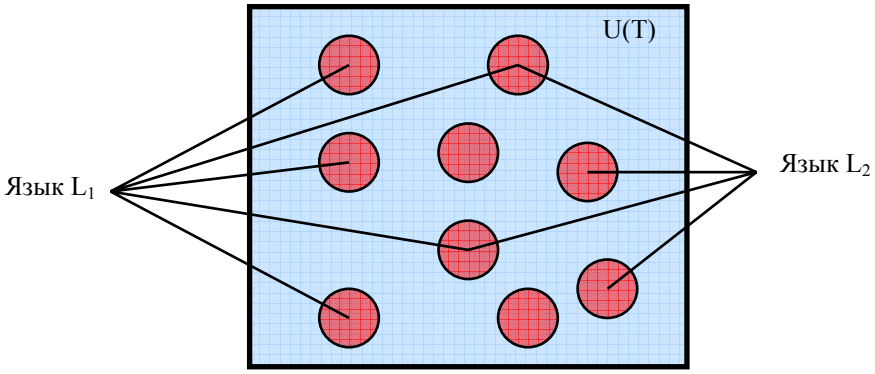


Рис. 2. Формальный язык.

Для описания семантики формального языка необходимо задать интерпретирующее отображение его строк в элементы некоторой модели, возможно в строки другого языка (рис. **3**). В основе описания семантики формальных языков лежит принцип композиционности Г. Фреге [4], заключающийся в определении семантики целого через семантику частей, т.е. между семантическими правилами интерпретации и синтаксическими правилами образования выражений языка устанавливается соответствие. Синтаксис формальных языков традиционно задается формальными грамматиками.

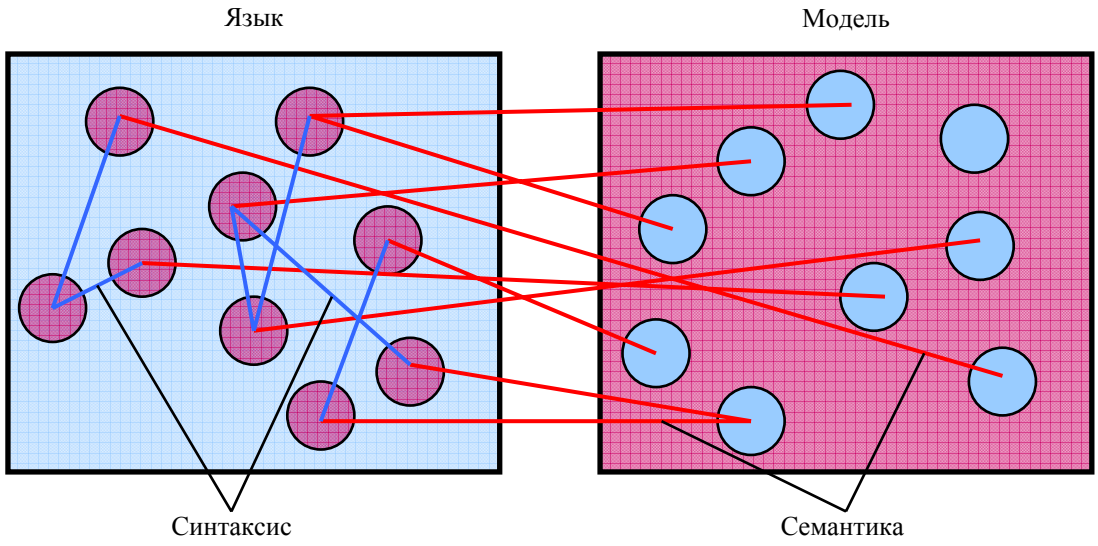


Рис. 3. Синтаксис и семантика

Задача формального определения семантики рассматривается теоретиками так же долго, как и задача определения синтаксиса, но до сих пор удовлетворительного решения не найдено. Для описания семантики формальных языков разработано множество различных моделей и методов:

- грамматические модели, основанные на добавлении расширений к грамматике, определяющей этот;
- аппликативные модели, в которых непосредственно конструируется определение функции, которую вычисляет каждая программа, написанная на этом языке;
- модели спецификаций, в которых описываются отношения между различными функциями языка и, если программа реализует эти отношения, то она корректна по отношению к спецификации.

Во всех перечисленных методах описание семантики осуществляется путем выделения некоторой типологии смыслов – категорий значений (семантических категорий, примитивов), через которые и посредством которых на некотором семантическом языке задается более сложный смысл.

При описании семантики языков программирования упоминаются три наиболее распространенных метода [5]: продукционной (дедуктивной, аксиоматической), денотационной (математической) и операционной семантики, которые являются фактически единственными потенциально пригодными. Однако ни один из перечисленных методов не оказался эффективным ни для пользователя, ни для разработчика языка [6].

Операционный метод достаточно удобный для реализации и может быть полезен разработчику, но для пользователя этот метод не имеет большого значения, так как порождаемые им описания содержат слишком много ненужных ему подробностей. Разработчик вряд ли сможет руководствоваться функциональным и денотационным методами, а для пользователя они, как правило, оказываются слишком сложными, чтобы их можно было использовать непосредственно. Пользователю легче понять аксиоматический метод, но при попытке составить полное определение языка он порождает чрезвычайно сложное описание, а для разработчика этот метод и вовсе непригоден.

3.2. Метаязык

Несмотря на некоторые относительные успехи в описании семантики формальных языков задача определения семантики естественного языка не может быть решена по причине его «семантической замкнутости». Семантически замкнутым называется язык, который содержит в себе как выражения, относящиеся к некоторым внеязыковым объектам, так и выражения, относящиеся к характеристике самого языка. Последнее приводит к возникновению в нем противоречий и парадоксов. Например, пытаясь ответить на вопрос, истинно или ложно высказывание «Данное предложение ложно», мы приходим к противоречию.

Однако понятие истинности присуще не объективной действительности, а оно выражает не более чем одно из возможных свойств высказываний. Следовательно, смысл «противоречивых» высказываний может быть различен, в зависимости от контекста их применения. Например, ранее использовалось предложение, которое в одной из своих интерпретаций порождает антиномию. Но это высказывание не разрушило семантическую структуру настоящего текста, так как было применено в особом контексте и для выражения особого смысла.

Таким образом, проблемы описания семантики естественных языков, появляющиеся вследствие возникновения противоречий, вызваны тем, что язык оказывается настолько широким, что позволяет определять внутренне противоречивые ситуации и понятия, например такое, как «множество всех множеств». Заметим, что любую формальную теорию, путём введения новых понятий или чрезмерного расширения объема существующих понятий можно сделать противоречивой. Поэтому расширение понятийного базиса не только в формальной, но и содержательной теории должно сопровождаться тестом на непротиворечивость.

Чтобы избежать возникновения противоречий Тарским предложено разделить язык на две части: объективный (предметный) язык, служащий для выражения высказываний, и метаязык, предназначенный для описания семантики этих высказываний [7]. На предметном языке говорят о той или иной предметной области, а на метаязыке обсуждают свойства предметного языка. Благодаря этому разделению в языке не могут появляться предложения, говорящие о самих себе.

В отличие от предметных языков, метаязык является, как правило, некоторым специальным образом созданным языком, не содержащим всякого рода двусмысленностей, препятствующих использованию его в качестве орудия для создания точных высказываний о предметном языке. Так как метаязык по своей природе должен быть более выразительным, чем предметный язык, то при использовании метаязыкового подхода возникли трудности, связанные с необходимостью динамического расширения категориального аппарата в зависимости от того, какие выражения предметного языка встретились при анализе [8]. С другой стороны, метаязык, как и любой язык, также нуждается в описании. Для определения семантики метаязыка требуется метаязык более высокого порядка, что приводит к некоторой иерархической структуре метаязыковых средств.

Таким образом, метаязыковый путь описания семантики языков видится неконструктивным, так как приводит к бесконечной рекурсии. По этой причине любая формальная семантика задается путем определения некоторого универсального метаязыка, который замыкает иерархию, строится на основе конечного множества первичных неопределяемых семантических категорий и содержит

формальный аппарат, с помощью которого и через первичные категории которого выражается более сложный смысл.

3.3. Объединенный подход

Для решения возникающих проблем формального описания семантики выполним замыканием метаязыка на его предметный язык, суть которого заключается в том, что, во-первых, множество первичных семантических категорий оставляется открытым и пополняется на этапе решения конкретной прикладной задачи, во-вторых, используется индуктивная грамматическая форма определения более сложного смысла через первичные семантические категории, а также через семантические категории, определенные ранее.

Для этого предметный язык будем рассматривать как тройку, состоящую из предметного синтаксиса, или правил построения высказываний; предметной онтологии, или системы понятий языка; предметной семантики, или правил интерпретации высказываний (рис. 4).

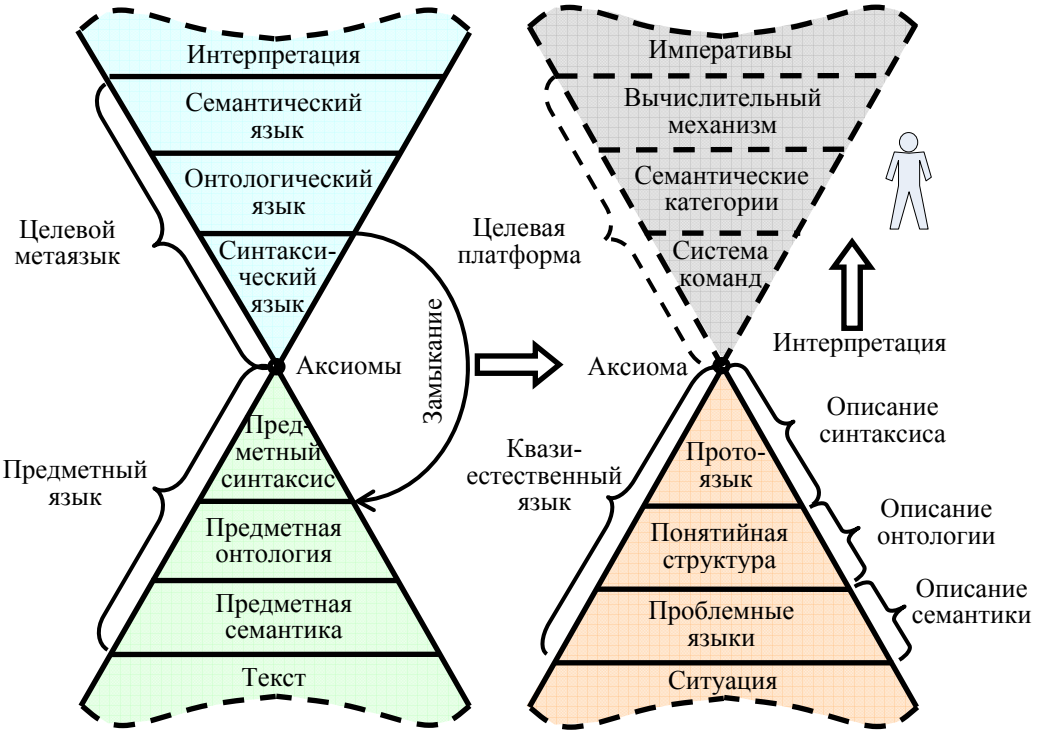


Рис. 4. Семантическое замыкание.

Для описания предметного языка необходим метаязык. В соответствии с определенным выше делением предметного языка, в метаязыке выделим три вида выразительных средств: для описания предметного синтаксиса (синтаксический язык); для определения предметной онтологии (онтологический язык); для описания предметной семантики (семантический язык). Связь метаязыка и предметного языка осуществляется через некоторую систему аксиом, регламентирующих правила представления, структурирования и кодирования текстов.

Отождествим составные части метаязыка с соответствующими частями предметного языка, а семантику предметного языка будем определять на самом предметном языке, который, в этом случае, играет роль своего метаязыка, а то

общее, что присутствует при определении произвольных предметных языков вынесем в протоязык. Так как внутри замкнутой семиотической системы семантику определить нельзя, воспользуемся внешним семантическим интерпретатором – целевой вычислительной платформой, состоящей, как и целевой метаязык, из следующих частей:

- алфавита метазнаков, задающих синтаксические средства метаязыкового уровня (система команд);
- семантических категорий, сопоставленных метазнакам (действия, реализуемые командами);
- механизма интерпретации метавысказываний (вычислительный механизм исполнения императивов).

В итоге получаем квазиестественный язык для описания предметной области, который состоит из аксиомы, протоязыка, понятийной структуры и множества проблемных языков.

Таким образом, для каждого класса задач и в процессе их решения будем создавать свой, присущий только этим задачам проблемный язык, отражающий понятийную структуру предметной области. Выявленные в процессе анализа понятия включим в множество понятий создаваемого языка, а способы выражения понятий положим в основу его синтаксиса. Описание понятийной структуры и синтаксиса создаваемого проблемного языка выполним на протоязыке, который имеет фиксированный синтаксис и семантику, минимально достаточные для определения синтаксиса проблемного языка и пополнения его множества базовых семантических категорий.

Для описания семантики проблемного языка применим метод математической индукции, заключающийся в использовании семантических категорий, которые определяются по мере необходимости, в процессе описания синтаксиса проблемного языка и средствами этого языка. Базу индукции, или первичные семантические категории, объявим с помощью аксиомы и реализуем средствами целевой вычислительной платформы.

На содержательном уровне семантическое замыкание заключается в использовании создаваемого проблемного языка для описания его семантики. Последнее возможно ввиду отрытого множества базовых семантических категорий и наличия механизма его пополнения посредством использования единственно необходимой для этого аксиомы.

4. Квазиестественный язык

Квазиестественный язык как категория в научный обиход введена Н.Н. Непейводой [9]. Под квазиестественным языком понимается язык, принимающий естественную форму, но имеющий искусственную сущность. Наиболее важное их практическое приложение – системы запросов к базам данных для специализированных нужд. Считается, что синтаксис квазиестественного языка является ограниченным (язык выглядит как не полностью определенный), а семантика остается точной.

В контекстной технологии программирования совокупность языков, используемых для описания предметной (протоязык, онтологический язык, проблемные языки) образует некоторое интегральное языковое средство, которое ввиду ограниченного, но расширяемого синтаксиса и точного определения семантики можно отнести к классу квазиестественных языков.

4.1. Протоязык

Протоязык, или первичный язык, предназначен для описанием синтаксиса проблемных языков и реализует аксиому, которая необходима для объявления базовых семантических категорий при индуктивном описания семантики. Протоязык является формальным языком с фиксированными синтаксисом и семантикой.

На рис. **5** приведена формальная грамматика протоязыка, где нетерминальные знаки грамматики обозначены строками над терминальным алфавитом, например, cognition, essences и др., а терминальные знаки заключены в одинарные кавычки, например ′()′, или двойные кавычки.

```
cognition  →  essences [cognition]
essences   →  ′( )′ notion ′( )′ [intension]
intension  →  sentence [intension]
sentence   →  syntax ′{ }′
syntax     →  item [axiom] [syntax]
item       →  notion [alias] | lexeme
axiom      →  ′[′ [terminals] ′]′
alias      →  ′`′ terminals ′`′
lexeme     →  pattern | term
pattern    →  ′"′ [terminals] ′"′
term       →  ′'′ [terminals] ′'′
terminals  →  terminal [terminals]
```

Рис. 5. Грамматика протоязыка.

Грамматика задана с точностью до пробелов и обозначенных курсивом нетерминальных знаков notion, служащих для выражения определяемых нетерминальных знаков проблемных языков. В квадратных скобках указаны части продукций, которые могут быть опущены, а альтернативные правые части правил грамматики разделены вертикальной чертой.

Описание cognition понятийной модели состоит из описаний сущностей ее предметной области essences. Сущностям присваивается имя notion нетерминального понятия определяемого проблемного языка.

Содержание intension нетерминального понятия notion состоит из предложений sentence, которые служат для выражения синтаксиса syntax предложений, выражающих это понятие в тексте.

Синтаксис предложения syntax выражается последовательностью элементов item: понятий notion и лексем lexeme. Лексема является терминальным понятием определяемого языка и задается в виде последовательностью терминальных знаков terminals. Для выражения лексем используются как терм term, так и множества термов, задаваемых на языке регулярных выражений в виде шаблонов pattern.

После любого элемента item может быть указано применение аксиомы axiom, выражаемое текстом, возможно пустым, заключенным в квадратные скобки. Парные круглые скобки используются для выделения определяемых нетерминальных знаков notion проблемного языка, а фигурные – как разделители предложений sentence. Для задания идентификаторов понятий в конструкции syntax могут использоваться их алиасы alias, представленные терминальными знаками terminals, заключенными в обратные апострофы.

Пример 1. Опишем на протоязыке синтаксис языка целых десятичных чисел (рис. **6**). Множество строк, принадлежащих языку описывается нетерминальным понятием Целое. Целое выражается как 0 или Натуральное, возможно со знаком. Натуральное, в свою очередь, представлено как последовательность десятичных цифр, начинающаяся не с нуля. ♦

```
() Целое ()
        '0' { }
        Натуральное { }
        '-' Натуральное { }
() Натуральное ()
        "[1-9]" { }
        Натуральное "[0-9]" { }
```

Рис. 6. Синтаксис языка целых чисел.

4.2. Проблемные языки

Проблемный язык – это формальный язык, предназначенный для описания своей семантики, предметной области и решаемых в ней задач. Для описания семантики предложений sentence проблемного языка будем использовать конструкцию semantic, которую определим множеством прагматик pragmatic с именами aspect и текстом text на уже определенном до этого проблемном подъязыке (рис. 7).

```
sentence  →  syntax semantic
semantics →  pragmatic [semantic]
pragmatic →  [aspect] '{' [text] '}'
text      →  terminals | [aspect] '{' [text] '}'
```

Рис. 7. Правила описания семантики.

Таким образом, для каждого предложения имеется возможность описания одной или нескольких именованных семантик. Для указания на ту или иную семантическую интерпретацию некоторого фрагмента текста text используются фигурные скобки, перед которыми указывается имя соответствующей прагматики aspect.

Пример 2. Рассмотрим язык для дифференцирования выражений, включающих целочисленные константы, переменную x и алгебраические операции: изменение знака -, сложение +, вычитание –, умножение *. Свяжем с каждым предложением, выражающим нетерминальный знак Exp, две прагматики: прагматику по умолчанию (без имени) и прагматику с именем dif. Прагматика по умолчанию указывает на то, что выражение не дифференцируется, а именованная прагматика – что выражение необходимо продифференцировать. Таким образом, формальная производная некоторого выражения e – это dif{e}.

Описание семантики языка дифференцирования приведено на рис. **8**, где предполагается, что каким-то образом удалось описать семантику предложений (показана многоточием) нетерминального знака Str, который предназначен для представления (первое предложение) и конкатенации (второе предложение) строк.

```
() Str ()
        "'.*'" { … }
        Str '&' Str { … }
() Exp ()
        "[09]+" `e`
                { e }
                dif { '0' }
        'x' `e`
                { e }
                dif { '1' }
        '(' Exp `e` ')'
                { '(' & e & ')' }
                dif { '(' & dif { e } & ')' }
        '-' Exp `e`
                { '-' & e }
                dif { '-' & dif { e } }
        Exp `e1` '*' Exp `e2`
                { e1 & '*' & e2 }
                dif { '(' & e1 & '*' & dif { e2 } & '+'
                        & dif { e1 } & '*' & e2 & ')' }
        Exp `e1` "+|–" `o` Exp `e2`
                { e1 & o & e2 }
                dif { dif { e1 } & o & dif { e2 } }
```

Рис. 8. Язык дифференцирования.

Рассмотрим предложение Exp `e1` '*' Exp `e2`. В этом предложении заданы алиасы (синонимы) нетерминальных знаков, которые необходимы для описания семантики этого предложения. Неименованная прагматика предложения тривиальна – если выражение не дифференцируется, то оно остается неизменным, e1 & '*' & e2. Для описания именованной прагматики использовано известное правило дифференцирования произведения функций,

'(' & e1 & '*' & dif {e2} & '+' & dif {e1} & '*' & e2 & ')'.

В итоге, при стандартном описании семантики проблемного подъязыка, соответствующего нетерминальному знаку Str, текст ситуации dif{x*x+5*(x-3)} преобразуется в строку -(x*1+1*x)+(5*(10)+0*(x3)).

Следует заметить, что порядок перечисления предложений задает их относительный приоритет: предложение, определенное позже (ниже), при грамматическом разборе текста text просматривается раньше. Это позволяет, в частности, задать естественный приоритет используемым в примере алгебраическим операциям и порядок их интерпретации (слева направо или справа налево).

Для тождественных преобразований выражений, получаемых после дифференцирования, возможно определение прагматики с именем equ, в результате применения которой к ситуационному описанию equ{dif{x*x+5*(x-3)}} получим -2*x+5. ♦

5. Интерпретация

Для описания семантики квазиестественного языка было использовано отображение его строк в строки этого же языка. Однако не решенной осталась за-

дача интерпретации квазиестественного языка в объекты реального мира, которая возможна как посредством мыслительного аппарата человека, так и с помощью технического устройства (машины).

В примере 2 использовалась мысленная интерпретация языка дифференцирования, заключающаяся в анализе и преобразовании ситуационного описания в соответствии с заданным определением проблемного языка. Для это потребовалось только знание синтаксиса и семантики протоязыка, а также семантики первых двух предложений. Следует заметить, что их семантика была определена метаязыковыми средствами, для чего использовался естественный язык.

5.1. Аксиома

В случае машинной интерпретации языка дифференцирования, в фигурных скобках, где приводится описание прагматик первых двух предложений, необходимо записать некоторую последовательность команд (операций), которую машина будет выполнять всякий раз, когда соответствующее предложение будет распознано во входном тексте.

В этом случае система команд целевой машины реализует множество первичных семантических категорий, а сама интерпретация осуществляется вычислительным механизмом машины, который выполняет последовательности команд, составляющие эти императивы. Для создания императивов используется аксиома. Для записи в императив некоторой команды применяются пустые квадратные скобки (рис. **5**). Пустые квадратные скобки, будучи приведенные после некоторого элемента предложения, вызывают запись значения этого элемента в тело императива.

Пример 3. Опишем семантику языка исчисления высказываний для его машинной интерпретации. Первое предложение в тексте программы на рис. **9** выражает пустое понятие и задает выразительные средства для формирования императивов, а именно запись байта, задаваемого двумя шестнадцатеричными цифрами.

В следующем тексте (рис. **10**) первое предложение осуществляет запись в тело императива мнемонических обозначений команд целевой машины (строки целевого языка программирования). В последнем случае подразумевается, что вычислительный механизм перед выполнением императива вызывает ассемблер (целевую систему программирования) для получения соответствующей последовательности команд времени исполнения.

Следует обратить внимание на последние предложения рассматриваемого языка. Для определения семантики этих предложений использованы выразительные средства описанного до этого подъязыка исчисления высказываний. ♦

```
() ()
        '#' "[0-9A-F][0-9A-F]" [ ] { }
() Boolean ()
        'false'
                { #B8 #00 #00 #00 #00 #50 }
        'true'
                { #B8 #FF #FF #FF #FF #50 }
        '(' Boolean ')' { }
        'not' Boolean
                { #58 #F7 #D0 #50 }
        Boolean 'and' Boolean
                { #58 #5A #0B #C2 #50 }
        Boolean `a` 'or' Boolean `b`
                { not (not a and not b) }
```

Рис. 9. Низкоуровневая семантика исчисления высказываний.

```
() ()
        '<' ".+" [ ] '>' { }
() Boolean ()
        'false'
                { <mov eax, 0; push eax;> }
        'true'
                { <mov eax, -1; push eax;> }
        '(' Boolean ')' { }
        'not' Boolean
                { <pop eax; not eax; push eax> }
        Boolean 'and' Boolean
                { <pop eax; pop edx; and eax, edx; push eax> }
        Boolean `a` 'or' Boolean `b`
                { not (not a and not b) }
```

Рис. 10. Высокоуровневая семантика исчисления высказываний.

В итоге получаем, что текст, описывающий семантику предложений квазиестественного языка, преобразуется (компилируется) в императив – некоторую последовательность действий, которую целевая машина выполняет всякий раз, когда при грамматическом разборе входного текста распознается это предложение. По своей сути императивы – единицы вызова целевой машины, в то время как синтаксис предложений – описание структуры таких вызовов.

5.2. Семантическая индукция

Суть предложенного подхода к описанию семантики проблемных языков заключается также в использовании метода математической индукции.

Базой индукции в этом случае выступают первичные семантические категории, которые непосредственно реализуются целевой вычислительной платформой и объявляются перед использованием, для чего служит аксиома – единственное необходимое и достаточное правило, включаемое в грамматику всех определяемых языков.

Предположением индукции служат все ранее определенные семантические категории (все ранее определенные нетерминальные понятия языка). Совокуп-

ность правил вывода грамматики, выражающая одно и то же нетерминальное понятие, используется для обозначения выражения в тексте семантической категории, связанной с этим понятием.

Индуктивный переход осуществляется в процессе добавления нового правила в грамматику определяемого языка и описания семантики этого правила текстом, выражаемым на уже определенном до этого подъязыке квазиестественного языка.

Заключением индукции является определение новой или уточнение (доопределение) уже существующей семантической категории.

Таким образом, в процессе описания квазиестественного языка создается и семантический язык, необходимый для описания его семантики.

6. Разнесенный разбор

Определение семантики квазиестественного языка осуществлено путем последовательного описания правил вывода формальной грамматики этого языка и индуктивного выражения семантики каждого такого правила через базовые и ранее определенные семантические категории.

Отсюда, в частности, следует, что описание квазиестественного языка и тексты на этом языке должны быть подвергнуты грамматическому разбору таким образом, чтобы обработанные ранее императивы предложений могли быть использованы при грамматическом разборе и компиляции следующих.

Для решения этой задачи применим метод разнесенного грамматического разбора [10], заключающийся в разделении определения применимости предложения при анализе текста на две части – на контекстное сопоставление предложений, осуществляемое при просмотре текста назад, и структурное распознавание, выполняемое при просмотре вперед. Для этого разделим описание синтаксиса предложений на четыре области: контекст, лексему, область разбора и результат (рис. 11).

6.1. Контекст

Контекстом предложения назовем последовательность понятий, предшествующих первой лексеме. Под лексемой предложения будем понимать его первую лексему. Область разбора определим как часть предложения, которая непосредственно следует за первой лексемой. И, наконец, результат предложения – это нетерминальное понятие языка, выражаемое этим предложением.

Пусть имеется структура данных, которую назовем текущим контекстом и реализуем в виде стека контекста (рис. 11) – запоминающего устройства с дисциплиной доступа «первым вошел, последним вышел». Текущий контекст содержит последовательность понятий, которые уже распознаны, а соответствующий этим понятиям текст извлечен из входного потока. Таким образом, в каждый момент времени контекстный анализатор имеет некоторое текущее состояние, определяемое состоянием стека контекста и текущей позицией во входном потоке.

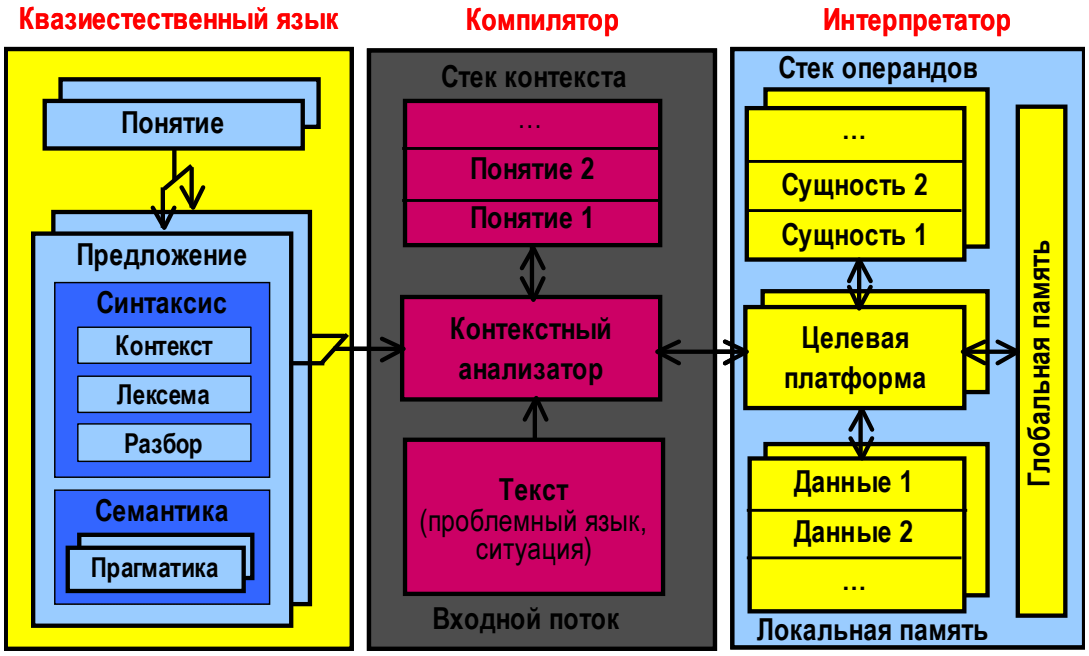


Рис. 11. Грамматический разбор.

6.2. Сопоставление

Поиск предложений, применимых в текущем состоянии анализатора, осуществим путем сравнения контекста предложений с текущим контекстом. Только предложения, имеющие сопоставимый контекст, могут использоваться в текущем состоянии. Здесь под сопоставимостью понимается соответствие нетерминальных понятий на вершине стека контекста понятиям из контекста предложений. Из сопоставимых предложений следует выбрать только применимые, лексема которых представляется текущим термом из входного потока.

6.3. Распознавание

После выявления применимого предложения наступает этап разбора оставшейся его части, которая еще не сопоставлена входному потоку. Для этого контекстный анализатор запоминает свое состояние и выполняет просмотр вперед при пустом контексте.

Если элементом разбора является лексема, то выполняется ее сравнение с текущим термом из входного потока. При успешном сравнении терм извлекается из входного потока, а в случае неудачи – состояние анализатора восстанавливается, а анализируемое предложение считается нераспознанным.

Если требуется распознать понятие, то для анализа выбираются только те предложения, которые имеют это понятие в качестве своего результата, т.е. выражают это понятие. При удачном распознавании анализатор переходит к следующему элементу из области разбора. В противном случае анализируемое предложение считается нераспознанным, состояние анализатора восстанавливается, и он переходит к разбору следующего применимого предложения.

Если все элементы области разбора сопоставлены входному потоку, то предложение считается распознанным, контекст предложения в стеке контекста

замещается на его нетерминальное понятие-результат, а полученное состояние входного потока объявляется текущим.

6.4. Реализация

Покажем выполнение разнесенного грамматического разбора на примере.

Пример 4. Рассмотрим язык исчисления высказываний (рис. 8), для которого выполним грамматический разбор текста «true or false and true».

Следует заметить, что при описании языка использовано единственное нетерминальное понятие Boolean, а предложения расположены в порядке естественного приоритета, т.е. константы false и true имеют наибольший приоритет, в то время как логическая связка or имеет наименьший приоритет

При грамматическом разборе текста «true or false and true» с пустым начальным контекстом сопоставимыми являются первые четыре предложения. Однако применимо только второе предложение 'true', т.к. его лексема находится в текущей позиции входного потока. У рассматриваемого предложения область разбора отсутствует, следовательно, это предложение помечается как распознанное, терм true извлекается из входного потока, а в стек контекста заносится понятие Boolean.

Продолжим грамматический разбор оставшейся части входного потока «or false and true» при текущем контексте Boolean. В указанном состоянии сопоставимыми являются последние два предложения, контекст которых сравним с текущим. Однако применимо только последнее предложение Boolean 'or' Boolean, в виду того, что только его лексема может быть выражена термом or из входного потока.

После контекстного сопоставления предложения Boolean 'or' Boolean наступает фаза его структурного распознавания. Теперь из входного потока при пустом контексте необходимо извлечь терм, выражающий нетерминальное понятие Boolean.

Находим, что в текущем состоянии сопоставимым является предложение 'false'. После его распознавания состояние входного потока становится «and true», а в стек контекста заносится искомое понятие Boolean. Однако, в указанном состоянии применимо предложение Boolean 'and' Boolean. Ввиду того, что текущее анализируемое предложение Boolean 'or' Boolean расположено после применимого (т.е. является менее приоритетным), продолжаем грамматический разбор. В процессе распознавания предложения Boolean 'and' Boolean получаем пустой входной поток и понятие Boolean в стеке контекста.

Теперь возвращаемся к анализу предложения Boolean 'or' Boolean. Так как нетерминальное понятие Boolean извлечено из входного потока, завершаем распознавание этого предложения. В итоге имеем пустой входной поток и понятие Boolean в стеке контекста.

Отсюда заключаем, что входной текст «true or false and true» распознан как выражающий понятие Boolean, а при его распознавании были выполнены императивы следующих предложений: 'true', 'false', 'true', Boolean 'and' Boolean и Boolean 'or' Boolean, что соответствует традиционной интерпретации этого текста, при которой связка and имеет больший приоритет, чем or. ♦

6.5. Эффективность

Эффективность разнесенного грамматического разбора по сравнению с другими известными методами определяется тем, что для принятия решения о применимости в заданном состоянии грамматического анализатора той или иной

продукции (правила вывода) не требуется обращение к другим продукциям. Для принятия решения оказывается достаточным только знание текущей позиции во входном потоке, текущего контекста и, конечно же, анализируемой лексемы.

6.6. Неоднозначность

Метод разнесенного грамматического разбора позволяет повысить эффективность синтаксического и семантического анализа. Последнее необходимо ввиду возможности обработки текстов, описываемых неоднозначными грамматиками.

Для учета неоднозначности в выборе предложений анализатор перед началом разбора каждого нового предложения сохраняет свое состояние (создает точку отката назад) и начинает грамматический разбор. В каждом новом состоянии анализатор производит поиск применимых предложений. Если таковых предложений не найдено, восстанавливается сохраненное ранее состояние (производится откат назад) и анализируется следующее предложение.

Описанная процедура реализуется рекурсивным вызовом анализатора всякий раз, когда начинается контекстное сопоставление предложений. Учитывая то, что квазиестественный язык создается для описания некоторой предметной области естественным образом, доля неоднозначностей в описании предметной области не должна быть высока. Как бы то ни было, при реализации грамматического разбора может быть предусмотрено некоторое критическое количество точек отката назад. При достижении последнего делается вывод о недостаточной проработке квазиестественного язык или об ошибке в тексте программы.

7. Понятийный анализ

Главной содержательной особенностью словесно-логической традиции в описании понятий является его понимание как некоторого мысленного «слепка» множества сущностей, обобщенно репрезентирующим его в сознании человека в виде определенно организованной совокупности существенных признаков. Последнее приводит к тому, что в рамках этой традиции сложную организацию предметного содержания понятия тщетно пытаются описать уже довольно давно. Причины неудач, видимо, кроются в том, что понятие является не просто статической репрезентацией реальности, а сложно устроенным когнитивным феноменом, позволяющим изменять (перестраивать) свои собственные репрезентации в зависимости от познавательных целей субъекта [11].

Однако, как бы то ни было, внешняя репрезентация понятия в словесно-логической форме видится единственной и принципиально необходимой формой представления и обработки знаний. По этой причине описываемый далее проблемный подход и пополняемое множество форм многоаспектного выражения понятий является некоторым неизбежным компромиссом между понятием как когнитивным феноменом и понятием как сложно организованной совокупностью существенных признаков, используемой для внешней репрезентации понятия в словесно-логической форме.

Для формальной спецификации предметной области будем использовать две формальные системы. Первую формальную систему – исчисление понятий, применим для выражения инвариантных свойств предметных областей. К инвариантным свойствам отнесем свойства понятийных структур предметных областей. Вторую формальную систему – проблемный язык, будем строить для каждой предметной области и использовать для описания специфических ее свойств.

Понятия, выявленные в процессе анализа предметной области, условно разделим на две группы: терминальные, или сигнификативные, выражаемые последовательностью знаков терминального алфавита проблемного языка, и нетерминальные, или денотационные, соответствующие нетерминальным знаками порождающей грамматики этого языка. Разделение понятий на денотационные и сигнификативные осуществим с учетом некоторой фиксированной проблематики, задающей класс задач, для которых определяется проблемный язык.

На основе выявления способов абстрагирования денотационных понятий построим понятийную структуру предметной области, где под абстракцией понимается одно из четырех отображений вида одних понятий в другие, которые соответствуют четырем фундаментальным способам образования понятий: обобщению, типизации, агрегации и ассоциации. Для каждой такой абстракции дадим формальное и семантически прозрачное (инвариантное) определение, не требующее предметной интерпретации, как это имеет место в других концептуальных моделях, где используется множество связей между понятиями, несущими различную семантическую нагрузку.

Выявленные в процессе анализа предметной области денотационные понятия включим в множество понятий специализированного предметного языка, а найденные декомпозиционные схемы преобразуем в его языковые конструкции. Языковые конструкции будем рассматривать как формы выражения денотационных понятий в тексте и задавать последовательностью знаков денотационных и сигнификативных понятий.

7.1. Основные определения

Проблемную область будем рассматривать как совокупность предметной области и решаемых на ней задач (проблем), где под предметной областью понимается фрагмент реальной (виртуальной) действительности, представляемый некоторой совокупностью принадлежащих ему сущностей.

Сущность определим как устойчивое и уникальное представление предметной области, воспринимаемое некоторой совокупностью признаков. Признак, как именованная сущность, характеризуется множеством своих проявлений (значений) и имеет некоторую проблемную интерпретацию (семантическую роль). Признаки будем воспринимать как элементарные сущности, с точностью до которых осуществляется описание предметной области.

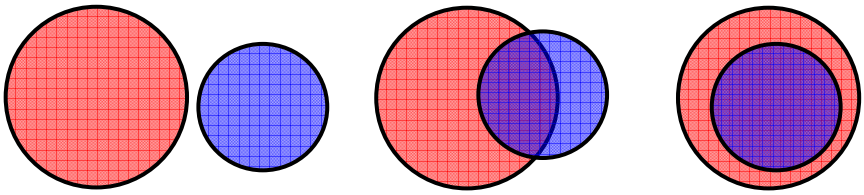
Понятие представим множеством сущностей, объединенных на основе сходства своих признаков. Понятия будем именовать и задавать схемой (shm), интенсионалом (int) и экстенсионалом (ext).

Имя, или знаковое представление понятия, будем рассматривать как языковую единицу, отражающую некоторый смысл в семантическом плане, и некоторую конкретную сущность – в плане синтаксическом. Схему понятия зададим набором признаков, на которых понятие определено.

Признаки будем интерпретировать как понятия, на которых определяются схемы. Интенсионал, или содержание понятия, представим набором значений взаимосвязанных признаков, позволяющим отличать сущности, принадлежащие понятию, от других сущностей предметной области. Экстенсионал, или объем понятия, будем рассматривать как множество сущностей, принадлежащих понятию.

7.2. Абстрагирование

Абстрагирование – форма мышления, при которой происходит образование понятия. При абстрагировании между понятиями выявляются отношения независимости, дифференциации и интеграции (рис. **12**). Понятия независимы, если их признаки не пересекаются. Если у двух понятий имеются общие признаки, то наблюдается дифференциация понятий. Если все признаки одного понятия являются признаками другого понятия, то происходит их интеграция.



а) независимость б) дифференциация в) интеграция

Рис. 12. Пространство признаков.

Известны следующие абстракции [12]: обобщение (специализация), типизация (конкретизация), агрегация (декомпозиция) и ассоциация (индивидуализация). Обобщение и типизация, и обратные им специализация и конкретизация, выражают общность понятий, проявляющуюся при дифференциации. Агрегация и ассоциация, и обратные им декомпозиция и индивидуализация раскрывают интеграцию понятий.

При обобщении происходит порождение нового понятия на основе одного или нескольких подобных понятий, когда порождаемое понятие сохраняет общие признаки исходных понятий, но игнорирует их различия (рис. **13**). Обобщение – порождение понятия на основе пересечения схем обобщаемых понятий и расширенного объединения их экстенсионалов. При специализации, наоборот, из понятия-обобщения выделяется одно из обобщенных в нем понятий.

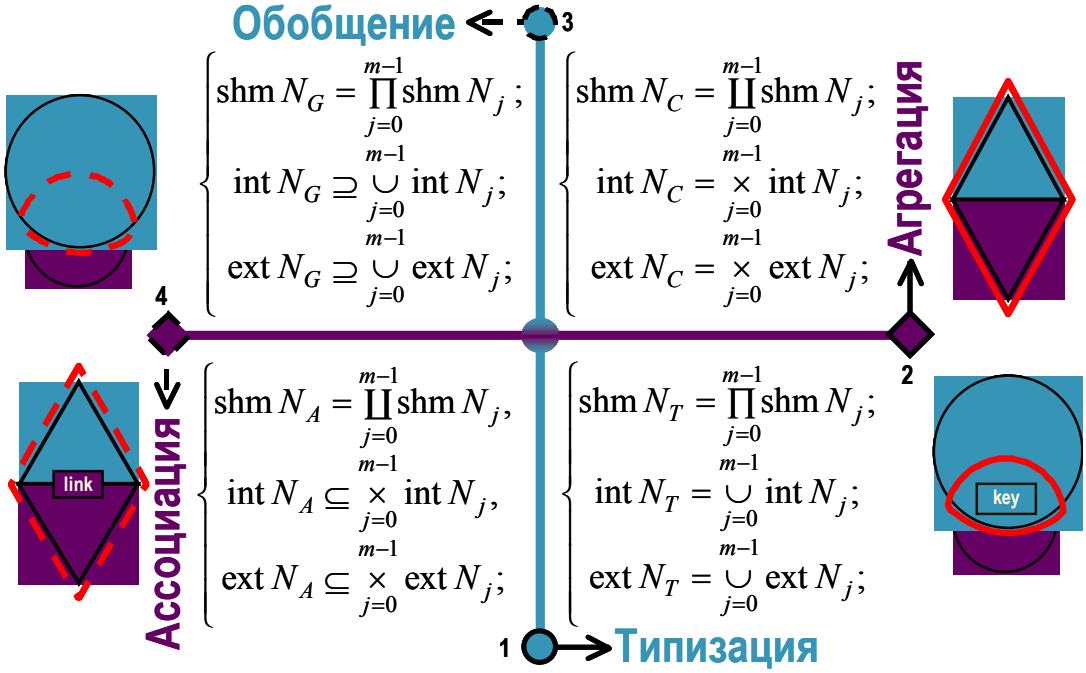


Рис. 13. Абстракции.

Типизация является частным случаем обобщения (рис. **13**). В отличие от обобщения, при типизации имеется возможность для каждой сущности из экстенсионала понятия-типа узнать ее исходное понятие. Для этого используется множество признаков, называемое ключом. Таким образом, типизация – порождение понятия на основе пересечения схем типизируемых понятий и объединения их экстенсионалов. При конкретизации понятия-типа фиксируется одно из типизированных в нем понятий, для чего необходим ключ.

При ассоциации устанавливается взаимосвязь между сущностями одного и того же или разных понятий (рис. **13**). Ассоциация выражает специфическое соединение сущностей. Это соединение позволяет от сущности одного понятия перейти к одной или нескольким сущностям других понятий. Ассоциация – порождение понятия на основе объединения схем ассоциируемых понятий и ограниченного декартового произведения их экстенсионалов. При индивидуализации из понятия-ассоциации выделяются ассоциированные в нем понятия. Для перехода между сущностями этих понятий используется набор признаков, называемый связью.

При агрегации понятие строится как совокупность других понятий (рис. **13**). Процесс, противоположный агрегации называется декомпозицией. Агрегация – порождение понятия на основе объединения схем агрегируемых понятий и декартового произведения их экстенсионалов. При декомпозиции понятие-агрегат разделяется на входящие в него агрегированные понятия.

Агрегация является предельным случаем ассоциации. В отличие от ассоциации, где между сущностями устанавливаются только часть связей, при агрегации присутствуют все возможные связи: на одном и том же множестве понятий можно задать несколько ассоциаций, в то время как их агрегация единственна.

7.3. Понятийная структура

Под понятийной структурой будем понимать совокупность понятий, для которых заданы способы их образования (абстрагирования). Носителем понятийной структуры является множество понятий, а ее сигнатурой – множество отображений обобщения, типизации, агрегации и ассоциации.

В отличие от семантических сетей и концептуальных схем, где на понятиях задаются различные виды отношений, понятийная структура определена множеством понятий с четырьмя типами отображений, единственное назначение которых – показать способы образования понятий. Понятийная структура близка расширенной модели данных «сущность-связь» [12], однако в этой модели элементами являются не понятия, а типы данных, причем для детализации модели используется трудно формализуемая семантическая разметка в виде дополнительных нотаций и ограничений.

Пример 5. Рассмотрим предметную область, которую условно назовем «Компьютер». Предположим, исходя из содержательной постановки задачи нам необходима некоторая морфологическая модель сущностей предметной области. Одна из таких моделей, выраженная в виде понятийной структуры, приведена на рис. **14**.

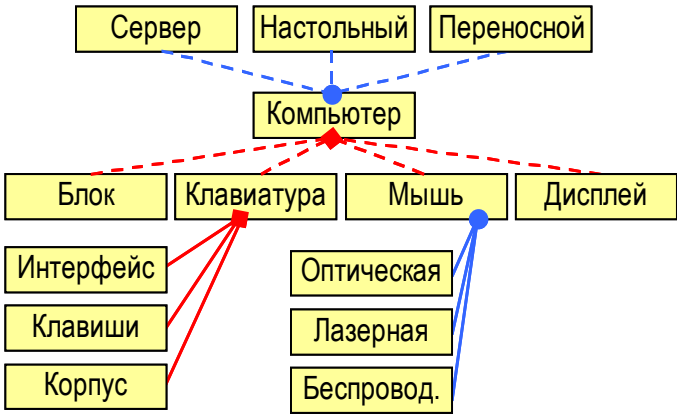


Рис. 14. Понятийная структура

Из рисунка видно, что понятие Мышь есть типизация понятий Оптическая, Лазерная и Беспроводная. В этом случае использование абстракции типизации отражает тот факт, что имея некоторую Мышь всегда можно по ее признакам установить, является ли она оптической, лазерной или беспроводной.

Клавиатура в примере описана как агрегация понятий Интерфейс, Клавиши и Корпус. Последнее означает, что существуют сущности понятия Клавиатура, имеющие в своем составе произвольные комбинации сущностей понятий Интерфейс, Клавиши и Корпус.

Самым сложным понятием является Компьютер. Он одновременно описан как обобщение и ассоциация других понятий. С одной стороны, компьютер как понятие определено как обобщение таких понятий как Сервер, Настольный и Переносной. В этом случае понятие Компьютер содержит общие признаки обобщаемых понятий. С другой стороны, Компьютер есть ассоциация понятий Блок, Клавиатура, Мышь и Дисплей. Последнее означает, что не любая комбинация сущностей перечисленных понятий может присутствовать в сущностях понятия Компьютер. Например, предполагается, что Переносной компьютер не

строится на базе дисплея с электронно-лучевой трубкой, а признак дисплея, связанный с технологией воспроизведения изображений, входит в связь понятия Компьютер и позволяет выделить те сущности понятий Блок, Клавиатура и Мышь, которые «совместимы» с заданной сущностью этого признака. ♦

7.4. Онтологический язык

Для описания понятийных структур в контекстной технологии используются специальные средства – онтологический язык (рис. **15**).

```
cognition      →   essences [cognition]
essences       →   differentiation notion integration [intension]
differentiation →  ′(′ [notions] ′)′
integration    →   ′(′ [notions] ′)′
notions        →   notion [alias] [notions]
```

Рис. 15. Грамматика онтологического языка.

Понятийная модель состоит из описаний сущностей проблемной области essences. Каждой сущности присваивается имя notion нетерминального понятия определяемого языка, а само понятие задается как находящееся в отношениях дифференциации differentiation и интеграции integration с системой других, ранее определенных понятий notions.

Определение сущностей essences содержит указание как на обобщение или типизацию, так и на агрегацию или ассоциацию определяемого понятия notion, что служит для выражения понятийной структуры. Для указания на отсутствие у понятия обобщения (типизации) используются пустые круглые скобки. В этом случае понятие считается обобщенным от пустого понятия. Аналогично, при отсутствии понятий в списке интеграции, понятие считается агрегирующим (ассоциирующим) пустое понятие.

Для ссылок на понятия в списках notions могут использоваться алиасы alias. Алиасы служат для реализации обратных абстракций: декомпозиции (индивидуализации) – для интегрированных понятий, и конкретизации (специализации) – для дифференцированных. Необходимость использования обратных абстракции возникает при описании семантики предложений.

Для выражения перекрестных и рекурсивных связей между понятиями используется их предварительное объявление, которое не включает конструкцию intension.

Пример 6. Описание на онтологическом языке понятийной структуры из примера 5 приведено на рис. 16.

() Сервер ()
() Настольный ()
() Переносной ()
() Интерфейс ()
() Клавиши ()
() Корпус ()
() Оптическая ()
() Лазерная ()
() Беспроводная ()
() Блок ()
() Дисплей ()
(Оптическая Лазерная Беспроводная) Мышь ()
() Клавиатура (Интерфейс Клавиши Корпус)
(Сервер Настольный Переносной)
Компьютер
(Блок Клавиатура Мышь Дисплей)

Рис. 16. Онтологическое описание предметной области.

7.5. Методика понятийного анализа

Необходимость анализа предметной области до начала ее формализованного описания осознана давно и используется при разработке масштабных программных проектов. В этом случае процесс разработки существенно отличается от написания программ для решения вычислительной задачи. Например, главное отличие проектирования баз данных заключается в том, что осуществляется предварительная разработка концептуальной схемы, которая отражает взаимосвязи типизированных сущностей проблемной области и особенности организации представляющих их данных.

В отличие от известных подходов, основанных на построении объектной модели, понятийный анализ определим как методику построения и верификации понятийной структуры проблемной области.

Понятийный анализ выполняется с учетом некоторой активной проблематики и состоит из следующих этапов:

- разделение сущностей предметной области на сигнификативные и денотационные;
- означивание сигнификативных сущностей и выявление существенных признаков у денотационных сущностей;
- сопоставление сущностей и определение их общих и различающихся признаков;
- образование новых или определение уже существующих понятий на основе интеграции и дифференциации признаков;
- создание понятийной структуры предметной области путем описания отображений одних понятий на другие;
- уточнение способа абстрагирования понятий (обобщения или типизации, ассоциации или агрегации);
- верификации понятийной структуры путем проверки выполнения ее свойств;
- вычисление схем понятий и задание ключей – для типизации, связей – для ассоциации.

7.5.1. Разделение сущностей. Разделение сущностей предметной области на сигнификативные и денотационные представляет, в общем случае, нетривиальную задачу. Сложность данной задачи проявляется в неформальном характере правил, которые можно применять для этой цели:

- сущности, составляющие объемные понятия, относят к денотационным, а единичные – к сигнификативным;
- часто выражаемые сущности относят к денотационным, редко выражаемые – к сигнификативным;
- сущности, имеющие сложную семантику относят к денотационным, а простую – к сигнификативным.
- сущности, используемые в множествах контекстов относят к денотационным, а в одном или нескольких контекстах – к сигнификативным;
- сущности, являющиеся основными носителями смысла относят к денотационным, а вспомогательные – к сигнификативным.

7.5.2. Означивание сущностей. Означивание сигнификативных сущностей осуществляется путем присвоения им индивидуальных имен, которые образуют базовый словарь проблемной области. Выявление существенных признаков у денотационных сущностей производится путем перечисления тех признаков, которые являются значимыми в рассматриваемой проблемной области. Заметим, что на этом этапе происходит и выявление самих признаков, или первичных понятий проблемной области, которые рассматриваются как денотационные сущности, не имеющие признаков.

7.5.3. Сопоставление сущностей. Следующим этапом понятийного анализа является сопоставление сущностей и определение их общих и различающихся признаков. Это позволяет выполнить первичную группировку сущностей на основе анализа тождества и различия множества признаков, на которых эти сущности определены.

7.5.4. Образование понятий. Образование новых или определение уже существующих понятий осуществляется на основе выявления интеграции и дифференциации признаков у выявленных на предыдущем этапе сущностей. Очевидно, сравнению подвергаются только те сущности и их группы, которые видятся связанными по условию решаемой задачи.

7.5.5. Создание понятийной структуры. Понятийная структура призвана в формализованном виде отразить результаты предыдущих этапов и выражает отображения одних понятий в другие.

7.5.6. Уточнение абстрагирования. В случае необходимости производится уточнение способа образования понятий: дифференциальные понятия помечаются как понятия-обобщения или понятия-типы, а интегральные понятия – как понятия-ассоциации или понятия-агрегаты.

7.5.7. Верификация. Методы верификации понятийной структуры непосредственно следуют из следующих ее свойств:

- любая понятийная структура содержит не более чем одно обобщение для каждого включенного в нее понятия;
- в любой понятийной структуре типизируемые и типизированное понятия имеют одинаковые схемы;
- любая понятийная структура содержит не более чем одну агрегацию для каждого включенного в нее понятия;
- в любой понятийной структуре у каждого объемного понятия-ассоциации пересечение схем ассоциированных понятий не пусто.

7.5.8. Вычисление схем. По определению схема понятия – это набор простых понятий, на которых это понятие определено. Схему произвольного понятия будем вычислять на основе отображений абстрагирования, заданных в понятийной структуре. Для этого воспользуемся следующей рекуррентной процедурой:

- схема простого понятия N равна (N);
- схема понятия, полученного в результате дифференциации, равна пересечению схем дифференцируемых понятий;
- схема понятия, полученного в результате интеграции, равна объединению схем интегрируемых понятий;
- схема понятия, полученного в результате дифференциации и интеграции, равна объединению схем интегрируемых понятий, принадлежащему пересечению схем дифференцируемых понятий.

Операции объединения и пересечения множеств в рассматриваемом случае выполняются с учетом повторения элементов. Одновременная интеграция и дифференциация понятий является выразительным средством, которое позволяет уточнить схему определяемого понятия путем ограничения пересечения схем дифференцируемых понятий указанием некоторого подмножества этого пересечения. Очевидно, если схема, получаемая через интеграцию понятий, не содержится в схеме, вычисленной на основе дифференциации, то последнее говорит о недопустимом описании понятийной структуры.

7.6. Объектный и понятийный анализ

Целью объектного анализа является разработка объектной модели проблемной области, содержащей описания объектов, а также различные зависимости между ними. Считается, что декомпозиция проблемы на объекты – творческий, плохо формализуемый процесс. Процесс построения объектной модели традиционно включает в себя следующие этапы:

- определение объектов и их классов;
- подготовка словаря данных;
- определение зависимостей между объектами;
- определение атрибутов объектов и связей;
- организация и упрощение классов при использовании наследования;
- дальнейшее исследование и усовершенствование модели.

Объединение объектов в классы позволяет ввести абстракцию типизации и рассмотреть проблему в более общей постановке. Классам приписываются атрибуты. Атрибут – это значение простого типа данных, характеризующее объект в его классе. Текущие значения атрибутов характеризуют текущее состояние объекта. В объектном анализе используется понятие операции, которая определяется как функция, или преобразование, которую можно применять к объектам данного класса. Каждой операции соответствует метод – реализация этой операции для объектов данного класса. Таким образом, операция – это спецификация метода, а метод – это реализация операции.

Так как в модели используемая сложная семантическая разметка, то важным этапом объектного анализа является построение словаря данных, который содержит четкие и недвусмысленные определения объектов (классов), атрибутов, операций и ролей, задаваемые на естественном языке.

Между объектами устанавливаются связи, которые выражают отношения между классами указанных объектов. Связи, как и классы, могут иметь атрибу-

ты. В объектном анализе различают шесть видов связей: зависимость, агрегация, ассоциация, композиция, генерализация и реализация.

Прежде всего из классов исключаются атрибуты, являющиеся явными ссылками на другие классы; такие атрибуты заменяются зависимостями. Зависимость задает отношение зависимости и характеризуется специальными атрибутами, называемыми ролью и квантификатором, где роль несет семантическую нагрузку, а квантификатор определяет кратность связи.

Ассоциация определяется как отношение взаимодействия и, как все оставшиеся связи, описывается двумя ролями и двумя квантификаторами. Агрегация выражает отношение «часть-целое», а композиция рассматривается как частный случай агрегации, при которой жизненный цикл частей и целого совпадает. Генерализация задает отношение «частное-общее», а реализаций – отношение выполнения соглашения.

В конечном итоге, объектная модель представляется в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования, где под наследованием понимается абстракция конкретизации классов, при которой производный класс приобретает свойства и поведение базового класса. Иерархия наследования является основной формой представления объектной модели: остальные виды связей наносятся на уже полученную иерархию классов.

В основе проверки правильности объектной модели лежат внешние признаки, объединены в следующие группы:

- признаки пропущенного объекта (класса);
- признаки ненужного (лишнего) класса;
- признаки пропущенных зависимостей;
- признаки ненужных (лишних) зависимостей;
- признаки неправильного размещения зависимостей;
- признаки неправильного размещения атрибутов.

В таблице **1** показано сравнение терминов и процедур понятийного и объектного анализа. Из таблицы видно, что понятийный анализ по сравнению с объектным является некоторым обобщением последнего, так как базируется на более общих принципах и подходах. Заметим, что в отличие от объектного, результаты понятийного анализа могут быть легко формализованы и подвергнуты проверке на полноту и непротиворечивость. В конечном итоге понятийный анализ позволяет на основе фундаментальных абстракций понятий получить декомпозиционную схему проблемной области в виде семантически прозрачной формальной спецификации ее понятийной структуры.

Таблица 1. Сравнение понятийного и объектного анализа.

Понятийный анализ	**Объектный анализ**
Сущность (единичное понятие):	**Объект:**
– обладает уникальностью; – различается признаками; – выражает смысл	– обладает идентичностью; – имеет состояние; – проявляет поведение
Признак (элементарное понятие):	**Свойство (атрибут):**
отличает одну сущность от другой; имеет имя, домен и семантическую роль	принимает различные значения и характеризует состояние объекта
Понятие:	**Класс:**
множество сущностей, образованное на основе абстрагирования	объекты, имеющие общую структуру и общее поведение
Описание понятия:	**Спецификация класса:**
– имя (уникальность), схема (признаки); – интенсионал (содержание); – экстенсионал (объем)	– имя (идентичность); – свойства (состояние); – методы (поведение)
Взаимосвязь понятий:	**Взаимосвязь классов:**
– обобщение (есть некоторый); – агрегация (есть часть); – ассоциация (есть участник); – типизация (есть экземпляр)	– общее и частное (наследование); – целое и часть (агрегация); – зависимость (ассоциация).
Обобщение:	**Наследование:**
расширенная типизация, объединение сущностей дифференцируемых понятий	наследуемый класс повторяет структуру и поведение базового класса
Агрегация:	**Агрегация:**
соединение сущностей интегрируемых понятий	отношения целого и части, приводящие к иерархии объектов
Ассоциация:	**Ассоциация:**
ограниченная агрегация, связывание сущностей интегрируемых понятий	зависимость классов, обеспечивающая переход между объектами этих классов
Типизация:	**Виртуальные классы:**
идентификация сущностей дифференцируемых понятий	объект базового класса замещается объектами различных классов
Понятийная структура:	**Диаграммы:**
множество понятий с отображениями абстрагирования	иерархия классов, диаграммы состояний и поведения объектов

8. Система программирования

Описываемая далее система контекстного программирования является одной из реализаций технологии контекстной обработки данных на основе спецификации предметной области в форме языка-письма. Однако не видится препят-

ствий для построения на тех же принципах, например, системы обработки речевых данных, реализующей другой способ обработки – в форме языка-речи.

8.1. Компиляция императивов

Семантика предложений понятийной модели определяется их описаниями и задается в виде императивов – именованных последовательностей действий, выраженных текстом text на определенном до этого проблемном языке (рис. 7). Если при определении императива некоторого предложения будут распознаны другие предложения, то необходимо некоторым образом организовать выполнение этих императивов при вызове императива определяемого предложения.

Пример 7. Семантика предложения Boolean 'imp' Boolean исчисления высказываний (см. пример 3) может быть описана как низкоуровневыми средствами на языке ассемблера:

```
Boolean 'imp' Boolean
    { <pop eax; not eax; pop edx; or eax, edx; push eax> },
```

так и на определенном до этого проблемном языке:

```
Boolean `a` 'imp' Boolean `b`
    { not a or b }.
```

В последнем случае при построении императива этого предложения необходимо предусмотреть вызов императивов тех предложений, которые распознаны при грамматическом разборе текста «not a or b».

Пусть имеются императивы, описывающие семантику распознаваемых предложений:

```
'not' Boolean { … }
Boolean 'or' Boolean { … }
```

Представим императив рассматриваемого предложения в виде вызовов распознанных предложений, условно обозначенных как (not) и (or), и в порядке их распознавания:

```
Boolean `a` 'imp' Boolean `b`
        { a (not) b (or) }.
```

Так как рассматриваемое предложение содержит два нетерминальных понятия Boolean, то открывающаяся фигурная скобка извлекает из стека две сущности Boolean и записывает их в локальную память (см. рис. 11). Для обеспечения возможности использования этих идентификаторов в тексте, понятие Boolean дополняется двумя временными предложениями:

```
'a' { }
'b' { },
```

семантика которых реализуется компилятором и состоит в извлечения из локальной памяти соответствующих сущностей понятия Boolean. Тогда терм a, который встретился первым после фигурной скобки, приведет к генерации кода извлечения и записи сущности с именем a в стек операндов.

Далее императив (not) обрабатывает операнд a, который уже находится в стеке операндов. В процессе исполнения этого императива сущность a извлекается из стека, инвертируется и полученный результат опять заносится в стек.

Перед вызовом императива (or) предложения Boolean 'or' Boolean сущность b записывается на вершину стека, где к этому времени уже находился результат, созданный императивом (not). После выполнения императива (or), который извлекает два операнда с вершины стека и выполняет операцию дизъюнкции, на вершине стека имеем искомый результат: not a or b.

Как и следовало ожидать, первое и второе определения семантики предложения Boolean 'imp' Boolean оказались эквивалентными. ♦

Таким образом, во время грамматического разбора текста создаются императивы предложений, состоящее из последовательности вызовов императивов тех предложений, которые распознаны при грамматическом разборе и в порядке этого распознания. Сам императив, в свою очередь, можно трактовать как некоторую единицу вызова, требующую для своего выполнения как передачи параметров, так и возврата результата. Формальными параметрами императива являются понятия, которые встречаются при описании синтаксиса предложения, а фактическими параметрами – сущности этих понятий.

8.2. Прямая подстановка

Как видно из примера 7 вызов императивов сопровождается достаточно большими накладными расходами. Для повышения эффективности генерируемого кода вместо компиляции кода вызова императива разрешим использование прямой подстановки кода, составляющего тело небольших императивов.

Для задания прямой подстановки кода воспользуемся выразительными средствами, предоставляемыми аксиомой. В этом случае текст, заключенный в квадратные скобки будем интерпретировать как текст времени компиляции. Этот текст компилируется во время определения предложения, а записывается в тело формируемого императива во время его распознавания. Очевидно, что текст времени компиляции подлежит такому же грамматическому разбору, как и описание прагматик предложений.

Пример 8. Переопределим предложения из примера 7 следующим образом:

```
'not' Boolean
        [ <pop eax; not eax; push eax> ] { }
Boolean 'or' Boolean
        [ <pop eax; pop edx; or eax, edx; push eax> ] { }
Boolean `a` 'imp' `b` Boolean
       { not a or b }
```

где семантика первых двух предложений задана в виде прямой подстановки. Тогда для императива последнего предложения будет сгенерирован код, эквивалентный

```
{ <push eax; not eax; push eax>
<pop eax; pop edx; or eax, edx; push eax > } ♦
```

8.3. Фазы

В процессе своего функционирования система контекстного программирования может находиться в одной из трех фаз:

- в фазе разбора;
- в фазе компиляции;
- в фазе выполнения.

В фазе разбора выполняется грамматический разбор текста (лексический, синтаксический и семантический анализ), заданного на протоязыке контекстной технологии. Если в структуре предложения встретится текст времени компиляции, то этот текст компилируется, а порожденный им код сохраняется в структуре анализируемого предложения для использования в фазе компиляции.

В фазе компиляции происходит грамматический разбор текста, заданного на определенном в понятийной модели проблемном языке, осуществляемый путем контекстного сопоставления и структурного распознавания предложения, находящегося в текущей позиции входного потока. При этом используется внутреннее представление предложений, ранее обработанных и сохраненных в

понятийной модели. Если в структуре распознаваемого предложения встретится откомпилированный в фазе разбора текст времени компиляции, то код, порожденный этим текстом, записывается в тело компилируемого императива.

В фазе выполнения исполняется код, порожденный описанием решения прикладной задачи (ситуационная часть situation) или код, порожденный описанием семантики предложений (императивы конструкции semantic). Код ситуационной части выполняется непосредственно после компиляции, а код императивов – в местах его вызова.

В итоге получаем следующий важный вывод: в процессе определения проблемного языка и одновременно с ним определяется и специализированный компилятор для этого языка.

8.4. Динамическая семантика

В языках программирования различают статическую и динамическую семантику. Семантика, описываемая при задании правил вывода грамматики языка, называется статической, а семантика последовательности команд, получаемой в результате компиляции программы – динамической.

Однако разделение семантики квазиестественного языка на статическую и динамическую в классическом смысле не представляется возможным по причине отсутствия в системе контекстного программирования явного разделения компиляции описания языка и компиляцией текста, написанного на этом языке. По этой причине под динамической семантикой будем понимать временное создание в фазе компиляции вспомогательных предложений, которые удаляются при выходе из области видимости компилируемого фрагмента текста.

Сущности, с которыми оперирует система контекстного программирования, являются, по своей сути, данными, выражающими соответствующие этим сущностям понятия. Сущности как глобальной, так и локальной видимости могут использоваться внутри других единиц вызова. В этом случае сущности передаются в качестве операндов, например, через стек операндов, а синтаксис и семантика их использования определяются описанием соответствующих предложений.

Для описания сущностей, создаваемых во время компиляции текста, написанного на проблемном языке, необходимо предусмотреть механизм динамического описания их семантики. Семантическая роль таких сущностей определяется понятием, к которому эти сущности принадлежат, а синтаксис их выражения задается некоторым предложением, в частном случае, выражаемом терминальной строкой – бесконтекстным именем сущности.

Таким образом, для задания синтаксиса и семантики динамически создаваемых сущностей необходимо предусмотреть механизм добавления и удаления предложений, состоящих, например, из терма-имени и понятия-результата. Такие предложения в языках программирования соответствуют описаниям переменных, создаваемых в процессе компиляции текста программы и сохраняемых в таблице идентификаторов компилятора.

В системе контекстного программирования переменные создаются во время объявления сущности и используются синтаксическим анализатором при грамматическом разборе текста в виде предложения, временно добавленного в список предложений соответствующего понятия. После уничтожения сущности это предложение должно быть удалено.

Хранение предложений для временных сущностей-переменных организовано в понятийной модели, например, в специальном словаре динамических сущ-

ностей, который выполняет ту же роль, что и таблица идентификаторов у компиляторов языков программирования. В этом случае, предложения словаря используются при грамматическом разборе текста и имеют наивысший приоритет по отношению к другим предложениям модели.

Если в области действия одной сущности создается другая сущность с тем же синтаксисом и различающейся семантической ролью, необходимо добавить еще одно предложение с большим приоритетом, чем первое (расположенное после первого). Последнее позволяет создать область действия для второй сущности-переменной путем ограничения области видимости первой.

8.5. Структура

Структурно система контекстного программирования может быть представлена состоящей из следующих частей (рис. **17**):

- входной поток;
- лексический анализатор;
- синтаксический анализатор;
- семантический анализатор;
- интерпретатор (виртуальные машины);
- квазиестественный язык (понятийная модель).

Входной поток организован в виде файлов, содержащих текст обрабатываемой программы. Результатом работы системы контекстного программирования является понятийная модель предметной области, представленная в компилированном виде. Если во входном файле содержится текст ситуационной части, то после его компиляции в код целевой платформы происходит выполнение этого кода.

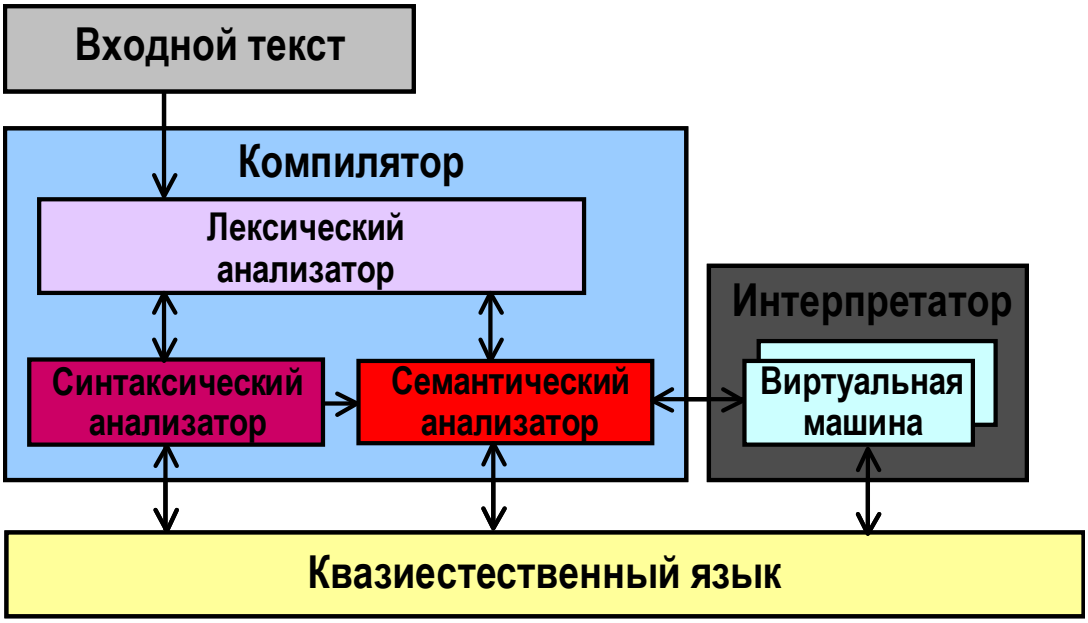


Рис. 17. Система контекстного программирования

Лексический анализатор в контекстной технологии отличается по своей организации от лексических анализаторов других известных систем программирования. В частности, в нем реализовано контекстное выделение лексем, задавае-

мых текущим термом анализируемого предложения. Иными словами, в процессе работы лексического анализатора правила разделения текста на лексемы изменяются. Особенностью лексического анализатора является его способность к сохранению и восстановлению не только указателя во входном потоке, но и состояния препроцессора текста.

Основное назначение синтаксического анализатора – грамматический разбор той части входного текста, которая описывается грамматикой протоязыка. Входными данными синтаксического анализатора является текущая понятийная модель предметной области и файл с текстом программы.

Синтаксис части входного текста определяется грамматикой протоязыка и обрабатывается синтаксическим анализатором. Синтаксис другой части текста задается описанием предложений, которые определены ранее и содержатся в понятийной модели.

Для исполнения кода, генерируемого семантическим анализатором, используется целевая вычислительная платформа. Со стороны системы программирования целевая платформа представляется в виде одной или нескольких виртуальных машин, исполняющих некоторый набор команд (операторов). В свою очередь виртуальная машина представляется в виде некоторого интерфейса, декларируемого средствами протоязыка.

На вход виртуальной машины подается код, порожденный семантическим анализатором, организованный в виде последовательности команд. Этот код является единицей вызова виртуальной машины и соответствует императиву некоторого предложения понятийной модели. В зависимости от типа целевой платформы каждая единица вызова может представляться:

- последовательностью команд непосредственного исполнения;
- последовательностью интерпретируемых операторов;
- текстом на входном языке другой системы программирования.

В последних двух случаях виртуальная машина генерирует исполняемый код, который в виде идентификатора образа памяти возвращается системе программирования для сохранения в понятийной модели. Тем самым реализуется однократное преобразование промежуточного кода в исполняемый. При повторном выполнении той же единицы вызова не требуется ее повторная компиляция. Так как виртуальных машин может быть несколько, то может существовать и несколько образов одного и того же промежуточного кода.

Таким образом квазиестественный язык, создаваемый в процессе обработки входного текста, используется для задания текстов, которые являются описанием:

- действий, которые необходимо выполнить во время синтаксического анализа предложения;
- действий, которые система программирования осуществляет всякий раз, когда будет распознано предложение;
- прагматик, представленных в виде одного или нескольких именованных императивов;
- ситуационной части программы, в которой декларируются исходные данные и осуществляется решение некоторой прикладной задачи.

9. Демонстрационный пример

На примере реальной задачи покажем преодоление семантического разрыва между описанием предметной области на естественном языке и ее описанием на создаваемом проблемном языке, а также определим эффективность полученного решения. В качестве примера рассматривается задача управления лифтом, ставшая уже традиционной для демонстрации различных подходов к программированию [13, 14].

9.1. Условие задачи

Дано 9-этажное здание с одним лифтом. Этаж с номером 0 – подвальный, 1 – первый, 2 – второй, и т.д.

На каждом этаже – две кнопки для вызова лифта на движение вверх и вниз. На нулевом этаже кнопка «Вниз» заблокирована, как и кнопка «Вверх» на 9-м этаже.

В кабине лифта имеется панель с кнопками для перемещения на конкретный этаж. В ней также размещены индикаторы движения лифта. Первоначально лифт находится на втором этаже в режиме ожидания.

Разработать программу управления лифтом.

9.2. Описание работы лифта

Рассмотрим описание работы лифта, выполненное экспертом на естественном языке и следующее из содержательных представлений о предметной области.

9.2.1. Ожидание вызова. Если вызовов нет, то ожидать вызов (9.2.1). Если вызов со второго этажа, то перейти к открытию дверей (9.2.2), иначе – на принятие решения о направлении движения (9.2.4).

9.2.2. Открытие дверей. Открыть двери. Если дверь открылась, то перейти на принятие решения о направлении движения (9.2.4), иначе повторить открытие двери (9.2.2).

9.2.3. Закрытие дверей. Закрыть дверь. Если дверь не закрылась, то повторить закрытие двери (9.2.3), иначе перейти к определению направления движения (9.2.4).

9.2.4. Принятие решения.

Продолжение движения: если лифт двигался вверх (вниз) и имеются вызовы на движение вверх (вниз), то начать движение вверх (9.2.5) (вниз 9.2.6).

Изменение направления: если вызовов на движение вверх (вниз) нет, но есть вызовы на движение вниз (вверх), то начать движение вниз (9.2.6) (вверх 9.2.5).

Возврат в начало: если вызовов нет и при этом лифт выше (ниже) второго этажа, то начать движение вниз (9.2.6) (вверх 9.2.5), иначе перейти в состояние ожидания (9.2.1).

9.2.5. Движение вверх. Начать движение вверх. Если при проходе этажа имеется вызов на движение вверх, то остановить лифт и открыть дверь (9.2.2), иначе продолжить движение вверх (9.2.5).

9.2.6. Движение вниз. Начать движение вниз. Если при проходе этажа имеется вызов для движения вниз, то остановить лифт и открыть дверь (9.2.2), иначе продолжить движение вниз (9.2.6).

9.3. Понятийный анализ задачи

В качестве основных сигнификативных сущностей будем использовать такие понятия как ожидать, открыть, закрыть, вверх, вниз, останов, двигать, выше, ниже, начать, продолжить. В этом случае денотационными сущностями, используемыми при описании задачи, следует выбрать следующие понятия: Здание, Этаж, Лифт, Дверь, Вызов, Команда и Движение. Из анализа предметной области заключаем, что выделенных понятий достаточно для описания работы лифта.

9.3.1. Этаж, Вызов, Движение. Здание имеет несколько Этажей. Этаж характеризуется номером (от 0 до 9). На Этаже имеются кнопки Вызова, задающие направление Движения. Понятие Движение будем рассматривать как совокупность сущностей «вверх», «вниз», «нет». В итоге Вызов может быть определен как агрегат понятий Этаж и Движение.

9.3.2. Лифт, Дверь, Команда. Для описания Лифта используются такие понятия как Этаж, на котором Лифт находится, состояние Движения и управление Движением, состояние Двери (открыта, закрыта) и управление Дверью (открыть, закрыть). Команда на Движение выдается внутри кабины нажатием на кнопку соответствующего Этажа. Вызов Лифта осуществляется нажатием на кнопки Вызова на Этажах.

9.3.3. Управление лифтом. Определим проблемный язык, максимально близкий описанию работы лифта, данному на естественном языке.

Анализ текста 9.2 показывает, что последний структурирован и разделен на пункты (секции). На секции имеются ссылки. Предусмотрим в языке определение таких секций и реализуем механизм перехода из одной секции к другой по ссылке. Для этого введем такие понятия как Метка и Переход.

В описании работы лифта имеются общеупотребительные языковые конструкции, такие как условное предложение и элементы исчисления высказываний. Для обеспечения наглядности не будем использовать подгружаемые подмодели, описывающие эти области, а определим соответствующие языковые конструкции в создаваемой понятийной модели. В итоге получаем понятийную структуру, приведенную на рис. **18**.

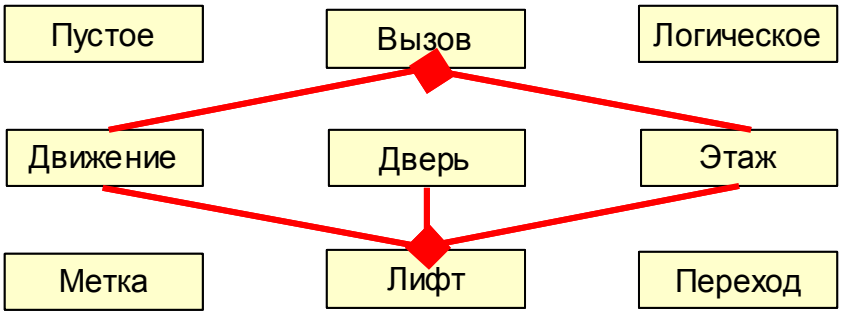


Рис. 18. Понятийная структура для задачи «Лифт».

9.4. Понятийная модель для управления лифтом

```
() Движение ()                                  \# Понятие движения:
      ″[Дд]вижение″ {}                          \#     направление движения лифта.
() Дверь ()                                     \# Понятие двери лифта:
      ″[Дд]верь″ {}                             \#     состояние двери.
() Этаж ()                                      \# Понятие этажа:
      'этаж' I[0-9]I {}                         \#     выражение этажа по номеру;
      'этаж' 'вызова' ″ | вверх | вниз″ {}      \#
      'этаж' 'лифта' {}                         \#     этаж, где находится лифт
() Вызов (Этаж Движение)                        \# Понятие вызова:
      'вызов' {}                                \#     состояние вызова лифта.
() Лифт (Этаж Движение Дверь)                   \# Понятие лифта:
      'лифт' {}                                 \#     запуск системы управления
() Метка ()                                     \# Понятие метки секции:
      ″[А-Яа-я][А-Яа-я0-9]* ″ {}                \#     имя (адрес) перехода.
() Переход ()                                   \# Понятие перехода на метку:
      Метка {}                                  \#     перехода по имени (адресу).
() Логическое ()                                \# Понятие высказывания
      Этаж ″выше|ниже|равен″ Этаж {}            \#     сравнение этажей по высоте
      Вызов ″вверх|вниз|нет″ {}                 \#     проверка направления вызова
      Дверь ″открыта|закрыта″ {}                \#     состояние двери лифта
      'не ' Логическое {}                       \#     логическое отрицание
() ()                                           \# «Пустое» понятие
      '. ' {}                                   \#     точка как знак пунктуации
      Метка ': ' {}                             \#     метка секции описания
      Движение ″вверх|вниз|останов″{}           \#     команды управления лифтов
      Дверь ″открыть|закрыть″ {}                \#     команды управления дверью
      'Если ' Логическое ', ' 'то ' Переход {}
      'Если ' Логическое ', ' 'то ' Переход ', ' 'иначе ' Переход {}
```

где \# используется для комментирования текста программы.

В приведенной выше модели все предложения только объявлены, т.е. определен их синтаксис. Описание семантики этих предложений может следовать далее по тексту. Будем предполагать, что такое описание выполнено одним из возможных способов:

- на дополнительно определенном в модели проблемном подъязыке;
- последовательностью команд целевой вычислительной платформы;
- на целевом языке программирования.

9.5. Ситуационное описание работы лифта

Ожидание:
Если вызов нет, то Ожидание.
Если этаж вызова равен этаж 2, то Открыть, иначе Решение.
Открыть:
Дверь открыть.
Если дверь открыта, то Решение, иначе Открыть.
Закрыть:
Дверь закрыть.
Если дверь закрыта, то Решение, иначе Закрыть.
Решение:
Если лифт вверх и вызов вверх, то Вверх.
Если лифт вниз и Вызов вниз, то Вниз.
Если не вызов вверх и вызов вниз, то Вниз.
Если не вызов вниз и вызов вверх, то Вверх.
Если вызов нет и этаж лифта выше этаж 2, то Вниз.
Если вызов нет и этаж лифта ниже этаж 2, то Вверх, иначе Ожидание.
Вверх:
Движение вверх.
Если этаж лифта равен этаж вызова вверх, то Открыть, иначе Вверх.
Вниз:
Движение вниз.
Если этаж лифта равен этаж вызова вниз, то Открыть, иначе Вниз.

Из текста ситуационной части видно, что описание работы лифта имеет легко читаемую форму и не превосходит по числу знаков исходное описание на естественном языке. Последнее позволяет надеяться на высокое качество и надежность порожденного исполняемого кода. Если в рассматриваемой реализации и имеются ошибки, то они вызваны или неточным (противоречивым) исходным описанием, или неверным описанием семантики.

9.6. Реализация системы управления

Для реализации системы управления лифтом в виде параллельного процесса мультизадачной вычислительной системы перенесем ситуационное описание из 9.5 в императив одного из предложений понятия Лифт.

```
() Лифт ()                          \# Понятие лифт (доопределение)
    'лифт'
        {
            \# Текст ситуационной части из 9.5
        }
```

После компиляции этого предложения получаем исполняемый код, представленный неименованным императивом. Для запуска системы управления лифтом определим еще одно предложение.

```
() ()
    'Запустить' 'лифт'
        {
            \# Создать вируальную машину в новом процессе и
            \# передать ей для исполнения указатель на императив
            \# предложения ‘лифт’
        }
```

При описании этого предложения используются подгружаемые понятийные модели параллельных процессов и используемой виртуальной машины (в настоящем примере не определены).

9.7. Анализ эффективности системы управления

В основу определения эффективности положим измерение семантического разрыва между исходным описанием задачи управления лифтом, выполненном на естественном языке, и формализованным описанием, реализованным средствами контекстной технологии.

Базовой характеристикой исходного описания примем ее длину, равную 1545 знакам. Заметим, что исходный текст не раскрывает семантику терминов и понятий, на основе которого строится описание работы лифта. Поэтому для сравнения выберем текст понятийной модели, также не включающий описании семантики. Суммарное количество знаков в понятийной модели и ситуационном описании равно 1606. Отсюда получаем предварительную эффективность реализации системы управления лифтом, которая равна 1606/1545, что в итоге дает 1,04.

Таким образом, семантический разрыв между текстом исходного описания и текстом программы фактически отсутствует. Небольшое превышение длины текста программы над длиной исходного описания связано с присутствием в программе понятийной структуры и синтаксиса выражения понятий в тексте, в то время как в исходном описании эти сведения не содержатся, а подразумеваются или выявляются в процессе понятийного анализа предметной области.

Учет описания семантики не должен изменить полученную эффективность в худшую сторону. Последнее основано на следующих соображениях. Семантика исходного текста, подразумевающаяся и необходимая для реализации реальной системы управления лифтом, может быть получена только после детального изучения технической документации на лифт и на те подсистемы ввода-вывода вычислительной системы, которые задействованы для управления лифтом и определения его состояния. Эти же сведения должны быть представлены в виде текстов описания семантики соответствующих предложения понятийной модели. Если предположить, что эти частные подзадачи, полученные в результате понятийной декомпозиции предметной области и не превышающие исходную по сложности, будут описаны средствами контекстной технологии с той же эффективностью, что общая задача, то и итоговая эффективность всего решения существенно не измениться.

9.8. Определение семантики низкоуровневыми средствами

Как в исходном описании работы лифта, так и в понятийной модели не содержится описание семантики. Семантику исходного описания следует выявлять в процессе изучения технической документации на лифт и вычислительную системы, предназначенную для управления лифтом.

Может оказаться, что в результате такого изучения будет установлено, что для реализации той или иной функции системы управления лифтом требуется выполнение дискретной обработки данных, для которой нет адекватных средств в используемой вычислительной системе. А если такая реализация и возможна, то она неэффективна в силу специфичности требований к этой обработке. Более того, применение аппарата понятийного анализа и контекстной технологии для реализации этой обработки не представляется возможным ввиду отсутствия содержательной интерпретации выполняемого преобразования данных. Например, такая ситуация может возникнуть при чтении показания датчиков, когда данные, необходимые для определения состояния лифта, получаются из поступающих данных и задаются на основе табличных форм.

Табличная форма преобразования данных используется в случае, когда отсутствует понятийное описание и содержательная интерпретация выполняемой обработки данных. Для формализации такой обработки следует применять аппарат функциональной декомпозиции, а реализацию осуществить или в виде низкоуровневой программы, или аппаратурно, в виде отдельного устройства.

Как при программно-аппаратной, так и при аппаратурной реализации табличной обработки данных декомпозицию необходимо выполнять в базисе операций, реализуемых системой команд процессора, или в базисе операций, реализуемых системой логических элементов.

9.9. Семантическое замыкание понятийной модели

Заметим еще одно немаловажное обстоятельство, проявившиеся в рассматриваемом примере: для интерпретации исходного описания задачи необходимо знание грамматики используемого для этого естественного языка, в то время как в тексте программы содержатся все необходимые сведения для полного грамматического разбора этого текста. Иными словами понятийная модель и ситуационное описании являются синтаксически замкнуты, в противоположность этому текст исходного описания синтаксически разомкнут.

После определения семантики предложений понятийной модели получаем квазиестественный язык, который является и семантически замкнутым. Семантическое замыкание осуществляется через аксиому, которая вводит в использование необходимые базовые семантические категории. В рассматриваемом примере такими категориями будут команды для аппаратных средств лифта на выдачу управляющих воздействий и получения состояния управляемых устройств.

10. Заключение

Контекстная технология программирования, будучи основанной на понятийном анализе, позволяет объективировать (формализовать) описание предметной области в более устойчивых формах, чем это предполагает ее объектно-ориентированный прототип.

Понятийный анализ как предельная форма абстракции познавательной деятельности позволяет ввести в использование и определить систему понятий, через которую знания о некоторой предметной области выражаются наиболее точно и компактно. Для этого найдены средства задания синтаксиса понятий, их семантической интерпретации, а также средства для определения связи понятий между собой, что позволяет задать различия между понятием как категорией и его языковым выражением.

Контекстная технология позволяют создавать базы знаний для различных предметных областей, которые могут как пополняться, так и использоваться для создания других баз знаний в качестве их составных частей. Формализация знаний о предметной области осуществляется путем построения ее понятийной модели. Понятийная модель является выражением понятийной структуры как некоторой концептуальной схемы решения задачи и включает определение квазиестественного языка, предназначенного для описания предметной области и этого решения. В отличие от других парадигм в области программирования, например структурной и объектно-ориентированной, контекстная парадигма определяет методологию постановки и решения задач специфическими языковы-

ми средствами, эквивалентными по выразительным возможностям формализму контекстных грамматик.[1]

Следует упомянуть о новых возможностях, которые появляются при использовании контекстной технологии программирования для реализации параллельных вычислений. В связи с тем, что описание предметной области и решаемой задачи формулируется на языке, наиболее близком содержательным представлениям, то появляется возможность адекватным образом выразить и реализовать естественный параллелизм задачи, а не использовать для этого «неестественные» средства распараллеливания на уровне элементов данных, инструкций процессоров или потоков команд.

Таким образом, принципиальными отличиями контекстной технологии программирования от других аналогов является:

- контекстная интерпретация фрагментов текста;
- обобщение и ассоциация нетерминальных понятий;
- описание семантики на создаваемом для этого проблемной языке;
- расширяемость протоязыка и проблемных языков;
- множество используемых виртуальных машин;
- пополняемость набора базовых семантических категорий.

Существенное отличие системы контекстного программирования заключается в возможности определения семантики квазиестественного языка на самом квазиестественном языке и использование в процессе интерпретации текстов открытого множества базовых семантических категорий.

Вычислительный эксперимент, выполненный в системе контекстного программирования, подтвердил обоснованность базовых принципов, использованных в основе понятийного анализа и контекстной технологии.

Список литературы

1. Баранов С.Н., Ноздрунов Н.Р. Язык Форт и его реализации. Л.: Машиностроение, 1988.
2. Brodie L. Thinking FORTH. A Language and Philosophy for Solving Problems. Englewood Cliffs N.J.: PrenticeHall, Inc., 1984.
3. Выхованец В.С., Иосенкин В.Я. Понятийный анализ и контекстная технология программирования // Проблемы управления. 2004. № 4. С. 325.
4. Борщев В.Б. Естественный язык – наивная математика для описания наивной картины мира // Московский лингвистический альманах. 1996. № 1. С. 203-225.
5. Себеста Р.У. Основные концепции языков программирования. М.: Вильямс, 2001.
6. Пратт Т., Зелковиц М. Языки программирования: разработка и реализация. СПб.: Питер, 2002.
7. Tarski A. Logic, Semantics, Metamathematics. Oxford, 1956.
8. Смирнова Е.Д. Формализованные языки и проблемы логической семантики. М., 1982.
9. Непейвода Н.Н. Квазиискусственные объекты // Логические исследования. 2002. № 8. С. 23-25.
10. Выхованец В.С. Разнесенный грамматический разбор // Проблемы управления. 2006. № 3. С. 32-43.

[1] В настоящей статье не описаны используемые в контекстной технологии программирования средства для проверки контекстных условий. В качестве таких средств используется конструкция вида '<' text '>', которая задает текст, компилируемый во время обработки (ввода) предложения и исполняемый во время его грамматического разбора. Последнее позволяет проверять произвольные контекстные условия во время структурного распознавания предложений, что делает проблемный язык контекстным.

11. Кравцов Л.Г. Методологические проблемы психологического анализа мышления в понятиях // Материалы Первой российской конференции по когнитивной науке. Казань, Казанский гос. ун-т, 2004 (http://www.ksu.ru/ss/cogsci04/).
12. Brodie M.L., Mylopoulos J., Schmidt J.W. On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases and Programming Languages. New York: Springer-Verlag, 1984.
13. Крут Д. Искусство программирования. В 3-х томах. Т. 1: Основные алгоритмы. М.: Вильямс, 2001.
14. Наумов Л.А., Шалыто А.А. Искусство программирования лифта. Объектно-ориентированное программирование с явным выделением состояний // Информационно-управляющие системы. 2003. № 6. С. 38-49.