ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

ПрИн 8 2013

Учредитель: Издательство "НОВЫЕ ТЕХНОЛОГИИ"

Издается с сентября 2010 г.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана, ОАО "Концерн "Сириус".

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru E-mail: prin@novtex.ru

Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

УДК 004.432.2

Н. И. Вьюкова, стар. науч. сотр., **В. А. Галатенко**, д-р физ.-мат. наук, стар. науч. сотр.,
С. В. Самборский, стар. науч. сотр.
НИИ системных исследований РАН, г. Москва,
e-mail: niva@niisi.msk.ru

О стандарте C11

Статья посвящена новому стандарту языка программирования C ISO/IEC 9899:2011, который был принят в декабре 2011 г. Представлен краткий исторический обзор предыдущих стандартов языка C и рассмотрены основные нововведения, вошедшие в стандарт C11, за исключением средств поддержки многопоточности, которым авторы планируют посвятить еще две статьи.

Ключевые слова: *язык программирования C, C11, средства анализа*

Введение

Язык C за время своего существования, начиная с 1969—1972 гг., претерпел значительные изменения, которые отражают тенденции в развитии информационных технологий в целом. В начале статьи рассмотрены история стандартов ANSI и ISO C, а также эволюция идей и принципов, лежащих в основе языка. Дальнейшие разделы содержат общий список новшеств, вошедших в стандарт C11, сводку так называемых выборочных (необязательных) возможностей и информацию о текущем состоянии поддержки C11 в компиляторе GCC.

Об истории стандартов языка C

В 1983 г. Американский национальный институт стандартов (ANSI) сформировал комитет X3J11 для разработки стандартной спецификации языка C. В 1989 г. стандарт языка был утвержден как "Язык программирования C" ANSI X3.159—1989 [1]. Эту версию принято называть ANSI C или C89. В 1990 г. стандарт ANSI C был принят с небольшими изменениями Международной организацией по стандартизации (ISO) как ISO/IEC 9899:1990 [2], он известен как C90. Дальнейшие версии стандарта разрабатывались уже этой организацией, но они принимались также и как стандарты ANSI. Поэтому стандарт до сих пор часто называют ANSI C, а не ISO C.

Одной из целей этого стандарта была разработка надмножества K&R C[1], включающего многие языковые средства, созданные позднее. Комитет по стандартизации включил в него также некоторые новые возможности, такие как прототипы функций и более сложный препроцессор.

После выхода C90 в течение долгого времени спецификация языка C оставалась относительно неизменной. В конце 1990-х гг. стандарт подвергся пересмотру, и в 1999 г. была опубликована спецификация ISO 9899:1999 [3]. Этот стандарт обычно называют C99. Среди новых возможностей, вошедших в C99, отметим inline-функции, объявление локальных переменных не обязательно в начале функции, новые типы данных: `long long int`, `_Bool`, комплексные типы, массивы переменной длины.

Хартия C1X

В апреле 2007 г. на собрании рабочей группы ISO WG14, состоявшемся в Лондоне, было принято решение о начале работы над новой ревизией стандарта языка C, получившей предварительное название C1X. В связи с этим обстоятельством в июне того же года был выпущен документ под названием "Хартия C1X" [4]. В хартии сформулированы "идеологические основы" языка C, а также принципы и направления его дальнейшего развития.

При разработке стандарта 1990 г. за основу были приняты следующие принципы.

- **Сохранение имеющегося большого объема кода.** Были приложены все усилия к тому, чтобы обеспечить работоспособность имеющегося кода при использовании реализации языка, удовлетворяющей стандарту.
- **Код может быть портируемым.** Хотя изначально язык C разрабатывался для ЭВМ серии DEC PDP-11

[1] Язык C в том виде, как он описан в книге Кернигана и Ричи "Язык программирования Си", вышедшей в 1978 г.

под управлением UNIX, в дальнейшем появились реализации для широкого круга процессоров и операционных систем, в том числе средства кросс-компиляции, генерирующие код для встроенных систем. Соответственно, спецификация языка и библиотечных средств разрабатывалась с учетом возможности реализации для максимально широкого спектра аппаратно-программных конфигураций.

♦ **Код может быть непортируемым.** Сильной стороной языка С является возможность его применения в качестве "высокоуровневого ассемблера". Программа может использовать (и это стандарт допускает) особенности и расширения конкретных реализаций языка.

♦ **Запрет на "молчаливые" изменения.** Разработчики стандарта стремились избегать модификаций языка, которые могут привести к изменению семантики существующих корректных программ.

♦ **Стандарт — это своего рода договор между программистом и реализацией.** В спецификацию языка были включены числовые параметры, определяющие минимальные требования, которым обязана удовлетворять реализация языка, претендующая на соответствие стандарту (такие как число аргументов функции, размер строковых констант, уровень вложенности блоков в функции и т. п.).

♦ **Принципы, постулирующие дух языка С:**
- доверие программисту;
- гибкость;
- компактность и простота;
- единственность способов для выполнения операции;
- эффективность даже в ущерб портируемости.

При разработке стандарта С99 приведенный выше набор пополнился следующими принципами.

▪ Поддержка интернационального программирования.

▪ Включение в стандарт только тех средств, которые необходимы для исправления явных недостатков языка и осознаются как насущная потребность значительного числа программистов.

▪ Минимизация отличий от С90. Возможность постепенной миграции проектов к С99.

▪ Минимизация расхождений с языком С++.

▪ Сохранение концептуальной простоты.

На совещании рабочей группы ISO WG14 в 2007 г. приведенный выше список был пересмотрен и дополнен перечисленными далее принципами, которые легли в основу разработки стандарта С1X.

▫ Принцип доверия программисту в свете растущего осознания проблем надежности и безопасности программ был признан устаревшим. Хотя он не должен полностью отвергаться, при разработке С1X следует учесть тот факт, что программисты должны иметь средства для проверки своей работы.

▫ В стандарте не должно быть изобретений — в него могут быть включены только средства, прошедшие достаточную апробацию в существующих коммерческих реализациях. Их спецификация при этом должна быть совместима с имеющимися реализациями.

▫ Необходимо максимально учитывать аспекты миграции существующего кода к новой версии стандарта, а также аспекты совмещения кода, удовлетворяющего стандартам С90, С99 и С1X.

Что нового в С11?

Опубликованный в декабре 2011 г. стандарт ISO/IEC 9899:2011 [5] получил неформальное название С11. В него вошли отмеченные далее новые возможности, которые поддерживаются в широко распространенных современных компиляторах.

- *Средства обеспечения надежности и безопасности программ*:
  - ♦ библиотечные функции с интерфейсами, обеспечивающими проверку ограничений (*bounds checking interfaces*);
  - ♦ поддержка средств анализа (*analyzability*);
  - ♦ статические утверждения;
  - ♦ дополнительные макросы для получения характеристик чисел с плавающей точкой;
  - ♦ функция **`gets`** изъята из стандарта.
- *Средства поддержки многопоточности*:
  - ♦ класс хранения **`_Thread_local`** — объекты, локальные для потока;
  - ♦ библиотека **`<threads.h>`** — средства управления потоками, локальными данными потоков и средства синхронизации (мьютексы и условные переменные);
  - ♦ спецификатор **`_Atomic`**, описывающий атомарные типы данных;
  - ♦ библиотека **`<stdatomic.h>`**, включающая атомарные операции записи, чтения, условного и безусловного обмена, чтения—модификации—записи и тестирования—установки;
  - ♦ **`quick_exit`** — новая функция для завершения программы.
- *Прочие возможности*:
  - ♦ выражения, не зависящие от типа (*type-generic expressions*);
  - ♦ выравнивание данных;
  - ♦ спецификатор функции **`_Noreturn`**;
  - ♦ анонимные структуры и объединения;
  - ♦ макросы для создания комплексных чисел;
  - ♦ улучшенная поддержка Unicode;
  - ♦ режим создания и открытия файлов для эксклюзивного доступа.

В настоящей статье кратко представлены все перечисленные выше нововведения за исключением средств поддержки многопоточности, которым авторы планируют посвятить еще две работы.

Выборочные возможности

Поскольку большинство существующих компиляторов не обеспечивают полной поддержки С99, было решено ввести в стандарт С11 больше выборочных (необязательных) возможностей, включая как новые

Средство	Макрос	Статус средства в стандарте C99
Средства анализа	`__STDC_ANALYZABLE__`	Отсутствует
Интерфейсы с проверкой ограничений	`__STDC_LIB_EXT1__`	Отсутствует
Типы комплексных чисел и библиотека `<complex.h>`	`__STDC_NO_COMPLEX__`	Обязательное
Операции над вещественными числами по стандарту IEC 60559	`__STDC_IEC_559__`	Выборочное
Арифметика комплексных чисел, совместимая со стандартом IEC 60559	`__STDC_IEC_559_COMPLEX__`	Выборочное
Массивы переменной длины	`__STDC_NO_VLA__`	Обязательное
Многопоточное программирование, библиотека `<threads.h>`	`__STDC_NO_THREADS__`	Отсутствует
Атомарные типы данных и библиотека `<stdatomic.h>`	`__STDC_NO_ATOMICS__`	Отсутствует

языковые средства C11, так и средства, считавшиеся обязательными в C99. В таблице представлен набор выборочных средств с указанием соответствующих макросов и статуса в C99.

Поддержка C11 в компиляторе GCC

Несмотря на то что стандарт принят совсем недавно, в GCC [6] версий 4.6.x и 4.7.x уже есть поддержка нескольких возможностей, описанных в стандарте. К их числу относятся:

- статические утверждения;
- переопределение деклараций `typedef`;
- анонимные структуры и объединения;
- функции, не возвращающие управление (спецификатор `_Noreturn` и заголовочный файл `<stdnoreturn.h>`).
- строки Unicode;
- средства выравнивания данных (ключевые слова `_Alignas`, `_Alignof`, заголовочный файл `<stdalign.h>`).

Некоторые возможности (инициализация объектов комплексных типов, атомарные операции библиотеки `<stdatomic.h>`) присутствуют в gcc-4.7.x на уровне встроенных (*builtin*) функций.

Библиотека GNU C (*glibc*), начиная с версии 2.16, поддерживает стандарт C11, исключая некоторые необязательные возможности.

Средства обеспечения надежности и безопасности программ

В этом разделе рассмотрены новые возможности языка, предназначенные для повышения надежности и безопасности программ. К их числу относятся средства анализа, позволяющие диагностировать некоторые распространенные программные ошибки, библиотечные функции, обеспечивающие проверку ограничений, статические утверждения, при помощи которых можно проверять свойства реализации языка, а также дополнительные макросы, описывающие характеристики представления чисел с плавающей запятой.

Средства анализа

Средства анализа (*analyzability*) являются одной из выборочных возможностей, наличие которой можно проверить при помощи макроса `__STDC_ANALYZABLE__`. Требования к ним сформулированы в нормативном приложении (Annex) L.

Смысл данной части стандарта состоит в том, чтобы стимулировать создание средств анализа программ, позволяющих диагностировать распространенные ошибки, которые возникают в ситуациях *неопределенного поведения* программы и приводят к некорректному состоянию памяти.

Стандарт выделяет несколько видов неоднозначного поведения программы.

- *Поведение, зависящее от реализации* — неспецифицированное поведение, которое должно быть документировано в каждой конкретной реализации. Например, точность, с которой выполняются операции и библиотечные функции над вещественными значениями.
- *Поведение, зависящее от локализации* — поведение, которое зависит от национальных, языковых или культурных традиций и должно быть документировано в конкретной реализации. Пример: значение, возвращаемое функцией `islower` для входных символов, отличных от 26 строчных букв латинского алфавита.
- *Неспецифицированное поведение* — поведение реализации для корректной программы, на которое стандарт не накладывает определенных требований и которое не обязательно должно быть документировано в конкретной реализации. Например, порядок, в котором вычисляются входные аргументы функций.
- *Неопределенное поведение* — поведение некорректной или непортируемой программной конструкции, или поведение при наличии некорректных данных, которое не регламентируется стандартом. Пример: поведение при целочисленном переполнении. Поведение программы в такой ситуации может варьироваться от полного игнорирования некорректности с порождением произвольного результата до строго определенной документированной реакции во время выполнения или компиляции (с выдачей диагностики

или без нее). Заметим, что неопределенное поведение имеет место в случаях, которые явно оговорены в стандарте как ситуации, порождающие неопределенное поведение, а также во всех случаях, которые не описаны стандартом явно.

Приложение L выделяет относительно небольшой подкласс ситуаций *критического неопределенного поведения*, при котором может произойти некорректная запись в память (например, запись за границу массива). Для volatile[2]-объекта критическим также считается некорректный доступ на чтение. Эти ситуации приводят к наиболее "разрушительным" последствиям для приложения и являются наиболее сложными для отладки.

Прочие ситуации относятся к разряду *ограниченного* (*bounded*) *неопределенного поведения*. Пример ограниченного неопределенного поведения — преобразование к целому типу, если результат не может быть представлен в диапазоне значений данного типа. В случаях ограниченного неопределенного поведения на практике обычно обеспечивается некоторый приемлемый результат, например, целочисленное переполнение может привести к вычислению неверного значения или возникновению исключительной ситуации.

Реализация, поддерживающая средства анализа, должна тем или иным образом доопределить поведение для всех ситуаций критического неопределенного поведения, за исключением 12 явно перечисленных в приложении L случаев, включая, например, следующие:

- доступ к объекту за пределами интервала жизни;
- попытка модифицировать строковый литерал;
- некорректный аргумент библиотечной функции с переменным числом аргументов;
- неверное значение операнда оператора.

Данная реализация может обеспечить диагностику во время компиляции или появление исключительной ситуации при выполнении с предоставлением соответствующего обработчика. Пример критического неопределенного поведения, для которого должна быть предоставлена документированная реакция — доступ к массиву с индексом вне диапазона. Таким образом, поддержка данного требования стандарта требует реализации развитых средств статического анализа и/или средств контроля во время выполнения.

Библиотечные функции с проверками ограничений

Стандартная библиотека С традиционно содержит набор функций, которые рассчитаны на то, что программист предоставил массивы достаточного размера для записи результата. Во многих случаях такая функция не имеет информации чтобы проверить, достаточен ли размер выходного массива. Разумеется, несмотря на такого рода недостатки, вполне можно писать безопасные, надежные и устойчивые к внешним воздействиям программы, используя существующую спецификацию библиотеки. Однако сложившаяся практика такова, что интерфейсы и реализации библиотек фактически поддерживают стиль программирования, ведущий к возникновению "необъяснимых" сбоев, если размер результирующего массива превысил ожидания программиста.

Подобный стиль программирования создает угрозу безопасности компьютеров и сетей. Известно, что переполнение буфера часто используется для выполнения злоумышленного кода с правами программы, в которой это переполнение произошло. Если же программист использует динамические проверки размеров данных при вызовах библиотечных функций, то они зачастую дублируются проверками, выполняемыми в самих функциях.

Приложение K стандарта C11 описывает альтернативные библиотечные функции, интерфейсы которых способствуют продвижению более надежного и безопасного стиля программирования. Кроме вопросов, связанных с переполнением буферов, это приложение решает также вопросы нереентерабельности[3] некоторых библиотечных функций, поскольку они могут возвращать результат в статической памяти. В этом случае ранее возвращенный результат может быть затерт при последующих вызовах этой функции, выполненных, возможно, в другом потоке.

Данное библиотечное расширение не использует дополнительных заголовочных файлов — декларации безопасных функций находятся в тех же заголовочных файлах, что и декларации соответствующих исходных функций. Декларации безопасных функций доступны, если в программе, которая включает заголовочный файл, определен макрос **`__STDC_WANT_LIB_EXT1__`** со значением 1. При этом безопасные функции не замещают соответствующие функции из библиотеки базового стандарта, а предоставляются в дополнение к ним (за исключением безопасной функции **`gets_s`**, которая введена взамен изъятой из стандарта **`gets`**).

В описаниях интерфейсов безопасных функций используют следующие новые типы и макросы:

- **`errno_t`** (файл **`<errno.h>`**) — тип результата многих безопасных функций, соответствует множеству значений переменной **`errno`**;
- **`rsize_t`** (файл **`<stddef.h>`**) — тип аргументов, задающих размеры входных и выходных массивов;
- **`RSIZE_MAX`** (файл **`<stdint.h>`**) — макрос, задающий максимальное значение для аргументов типа **`rsize_t`** (отметим, что рекомендуемое значение для данного макроса — **`SIZE_MAX`** (или меньше); безопасные функции проверяют, что значения аргументов типа **`rsize_t`** не превышают **`RSIZE_MAX`**, этот факт позволяет избежать некоторых ошибок, когда, например, небольшое отрицательное целое при преобразо-

[2] Объект, декларированный с квалификатором volatile, может изменять свое значение независимо от выполнения программы, и операции доступа к нему могут порождать дополнительные побочные эффекты. Обычно такие объекты используют в программе для взаимодействия с портами ввода-вывода.

[3] Функция называется реентерабельной (или повторно входимой), если она разработана таким образом, что одна и та же копия инструкций программы в памяти может быть совместно использована несколькими потоками или процессами.

вании к беззнаковому типу становится очень большим положительным значением).

Спецификация каждой безопасной функции включает список ограничений, которые она должна проверять. При нарушении хотя бы одного ограничения выполняется обработчик, либо предопределенный в данной реализации, либо установленный при помощи функции **set_constraint_handler_s** из библиотеки **<stdlib.h>**. Обработчик имеет три аргумента — строка сообщения об ошибке, некоторый указатель и значение типа **errno_t**. Как правило, обработчик не возвращает управление в вызывающую библиотечную функцию; если же управление возвращается, то функция возвращает результат, указывающий на наличие ошибки. Важно, что при нарушении отмеченных ограничений запись результата в выходной буфер не происходит.

Сравним в качестве примера интерфейсы функций, генерирующих уникальное имя файла — **tmpnam** и ее безопасного аналога **tmpnam_s**, декларации которых приведены ниже.

```
#include <stdio.h>
errno_t tmpnam_s (char *s, rsize_t maxsize);
char* tmpnam (char *s);
```

Ограничения функции **tmpnam_s**:

- **s! = NULL;**
- **maxsize <= RSIZE_MAX;**
- **maxsize** > *размер_имени* (максимальный размер имени определяется значением макроса **L_tmpnam_s**).

Таким образом, функция **tmpnam_s** отличается от традиционной **tmpnam**, во-первых, тем, что она никогда не возвращает результат в статической области памяти (**tmpnam** делает это, если указатель **s** нулевой). Во-вторых, она не запишет результат в выходной буфер, если длина результата превышает размер буфера, заданный аргументом **maxsize**. В-третьих, она проверяет, не превышает ли размер буфера максимальное допустимое значение.

Далее приведен список функций, вошедших в приложение K, для того чтобы дать представление об объеме данного библиотечного расширения.

- *Ввод-вывод* **<stdio.h>**:

tmpfile_s, tmpnam_s, fopen_s, freopen_s, fprintf_s, fscanf_s, printf_s, scanf_s, snprintf_s, sprintf_s, sscanf_s, vfprintf_s, vfscanf_s, vprintf_s, vscanf_s, vsnprintf_s, vsprintf_s, vsscanf_s, gets_s.

- *Общие утилиты* **<stlib.h>**:
  - Функция установки обработчика, выполняемого при нарушении ограничений: **set_constraint_handler_s.**
  - Стандартные обработчики: **abort_handler_s, ignore_handler_s.**
  - Аналоги функций базового стандарта: **getenv_s, bsearch_s, qsort_s, wctomb_s, mbstowcs_s, wcstombs_s.**
- *Функции работы со строками* **<string.h>**:

memcpy_s, memmove_s, strcpy_s, strcat_s, strncat_s, strtok_s, memset_s, strerror_s, strerrorlen_s, strnlen_s.

- *Функции работы со временем* **<time.h>**:

asctime_s, ctime_s, gmtime_s, localtime_s.

- *Функции работы с широкими символами* **<wchar.h>**:

fwprintf_s, fwscanf_s, snwprintf_s, swprintf_s, swscanf_s, vfwprintf_s, vfwscanf_s, vsnwprintf_s, vswprintf_s, vswscanf_s, vwprintf_s, vwscanf_s, wprintf_s, wscanf_s, wcscpy_s, wcsncpy_s, wmemcpy_s, wmemmove_s, wcscat_s, wcsncat_s, wcstok_s, wcsnlen_s, wcrtomb_s, smbsrtowcs_s, wcsrtombs_s.

Статические утверждения

Статические утверждения позволяют проверять свойства реализации (компилятора и среды выполнения). Традиционно для этой цели используют директивы препроцессора, например:

```
#if __STDC__ ! = 1
# error "Not a standard compliant compiler"
#endif
```

Статические утверждения позволяют статически проверять истинность произвольных константных выражений, в том числе таких, вычисление которых не поддерживается препроцессором, например:

```
_Static_assert(sizeof(size_t) > = 8,
        "size_t must be at least 64-bits");
```

Если значение выражения ложно, то это расценивается как ошибка компиляции. Текст сообщения об ошибке задает второй аргумент статического утверждения. Другие примеры использования этого средства можно найти в подразд. "Обобщенные выражения".

Характеристики чисел с плавающей запятой

Стандарт C11 включает несколько дополнительных макросов, описывающих характеристики представления вещественных чисел. Макросы **FLT_TRUE_MIN**, **DBL_TRUE_MIN** и **LDBL_TRUE_MIN** (файл **<float.h>**) задают минимальные значения, представимые в виде денормализованных значений

типов **float**, **double** и **long double** соответственно. Пример:

```
#define FLT_MIN 1.17549435E-38F
#define FLT_TRUE_MIN 1.40129846E-45F
#define DBL_MIN 2.2250738585072014E-308
#define DBL_TRUE_MIN 4.9406564584124654E-324
#define LDBL_MIN 2.2250738585072014e-308
#define LDBL_TRUE_MIN 4.9406564584124654E-324
```

Макросы **FLT_DECIMAL_DIG, DBL_DECIMAL_DIG, LDBL_DECIMAL_DIG** задают минимальное число десятичных цифр, которые можно хранить без потери точности.

Другие нововведения

В настоящем разделе рассмотрены прочие нововведения, явно перечисленные в преамбуле стандарта. Следует иметь в виду, что в текст стандарта внесено также довольно большое число частных изменений (по сравнению с версией C99), многие из которых связаны с поддержкой многопоточности и атомарных операций.

Обобщенные выражения

В стандарте C99 был введен заголовочный файл **tgmath.h**, содержащий обобщенные (не зависящие от типа) имена математических функций. Компилятор должен генерировать вызовы специализированных функций в зависимости от типов параметров. Вместе с тем механизм порождения специализированных имен из обобщенных оставался закрытым для программиста. В стандарте C11 введен механизм обобщающих селекторов (*generic selection*), позволяющий на стадии компиляции выбирать выражение, зависящее от типа входного выражения.

Синтаксис этого механизма аналогичен селектору **switch**, где выбор проводится по именам типов. Типовое применение обобщенных выражений — определение макросов, позволяющих абстрагироваться от типов выражений, в том числе описывать обобщенные имена функций. Приведем в качестве примера описание макроса, задающее обобщенное имя библиотечной функции **cbrt** (извлечение кубического корня):

```
#define cbrt(X) _Generic((X), \
   long double: cbrtl, \
   default: cbrt, \
   float: cbrtf \
)(X)
```

Макровызов **cbrt(x)** преобразуется в **cbrt(x)**, **cbrtl(x)** или **cbrtf(x)** в зависимости от типа аргумента.

Ниже приведен чуть более сложный пример макроса с двумя аргументами, задающего обобщенное имя функции возведения в степень.

```
#define pow(x, y) _Generic((x), \
   long double: powl, \
   double: _Generic((y), \
     long double: powl, \
     default: pow), \

   float: _Generic((y), \
     long double: powl, \
     double: pow, \
     default: powf)\
)(x, y)
```

Применение обобщающих селекторов не ограничивается описанием обобщенных имен функций. Они могут быть полезны везде, где нужно делать выбор в зависимости от типа выражения. Следующие макросы позволяют выбрать подходящий формат печати для выражений различных типов.

```
#define fmt(x) _Generic((x), \
   char: "%c", \
   signed char: "%hhd", \
   unsigned char: "%hhu", \
   signed short: "%hd", \
   unsigned short: "%hu", \
   signed int: "%d", \
   unsigned int: "%u", \
   long int: "%ld", \
   unsigned long int: "%lu", \
   long long int: "%lld", \
   unsigned long long int: "%llu", \
   float: "%f", \
   double: "%f", \
   long double: "%Lf", \
   char *: "%s", \
   void *: "%p")
#define prnt(x) printf(#x "=" fmt(x), x)
#define prnt_nl(x) printf(#x "=" fmt(x)
"\n", x);
```

Селекторы **_Generic** вычисляются на стадии компиляции, поэтому их можно использовать в статических утверждениях. Следующие макроопределения позволяют проверять тип выражения и выдавать ошибку компиляции, если тип не соответствует ожидаемому. Это может быть полезно для проверки корректности параметров функций (с переменным числом аргументов) или значений макросов.

```
#define is_compatible(x, T) _Generic((x),
T:1, default: 0)
#define ensure_type(p, t) \
   _Static_assert(is_compatible(p, t), \

   "incorrect type for parameter '" #p "',
   expected " #t)
```

Выравнивание данных

Каждому типу данных в реализации языка соответствуют требования по выравниванию объектов в памяти. Выравнивание — это положительное значение, являющееся целой степенью числа 2, которому должен быть кратен адрес объекта. Стандарт C11 позволяет задать для объекта или типа данных выравнивание, большее чем обычное значение, предусмотренное для него реализацией. Для этого в декларации следует использовать спецификатор **_Alignas**, за которым в скобках необходимо указать целое константное выражение или имя типа:

```
int _Alignas (8) data;
float eps _Alignas (double);
```

Стандарт также вводит оператор **_Alignof**, позволяющий узнать выравнивание объекта или типа. Оператор **_Alignof** так же как **sizeof**, можно использовать в статических утверждениях.

Заголовочный файл **<stdalign.h>** содержит определения макросов **alignas** и **alignof**, значениями которых являются ключевые слова **_Alignas** и **_Alignof** соответственно.

Реализация должна предоставлять тип **max_align_t**, имеющий максимальное выравнивание, которое должно поддерживаться в любом случае. Выравнивания, большие чем **_Alignof (max_align_t),** могут не поддерживаться реализацией или поддерживаться только в определенных контекстах. Например, если кадры стека выравниваются по границе 8 байт, то для автоматических переменных нельзя обеспечить более строгое выравнивание.

Спецификатор функции _Noreturn

Новый спецификатор функций **_Noreturn** позволяет описывать функции, которые не должны возвращать управление в точку вызова. Использование спецификатора преследует две цели — исключение предупреждения о том, что функция не возвращает управление, и оптимизация. Дополнительная информация о функции позволяет компилятору точнее анализировать потоки управления в программе, а следовательно, выдавать более точную диагностику и лучше оптимизировать код.

Стандарт требует, чтобы компилятор выдавал предупреждение, если функция, продекларированная со спецификатором **_Noreturn** может вернуть управление, например:

```
_Noreturn void g (int i) {
  if (i > 0) abort();
}
```

Заголовочный файл **<stdnoreturn.h>** содержит определение макроса **noreturn** со значением **_Noreturn**.

Анонимные структуры и объединения

Стандарт C11 позволяет определять анонимные структуры и объединения как элементы объемлющего агрегатного типа, например, следующим образом:

```
struct v {
  union {
    struct { int i, j; };
    struct { long k, l; } w;
    };
  int m;
} v1;
```

Обращаться к элементам анонимной структуры или объединения можно напрямую, как если бы они были элементами объемлющего агрегатного типа. Это делает текст программы более простым и кратким.

```
v1.i = 2;    // верно
v1.k = 3;    // неверно, т. к. структура, содержащая
             // элемент k, не анонимная
v1.w.k = 5; // верно
```

Естественно, описания анонимных агрегатных типов не должны создавать коллизий по именам элементов.

Макросы для создания комплексных чисел

В стандарте C99 был определен удобный и наглядный синтаксис для констант комплексного типа:

```
_Complex float c01 = 0.0f + 1.0if;
```

Недостатком этой конструкции является то, что она неприменима для создания комплексных значений с компонентами NaN ("не число") и бесконечность. По этой причине в новом стандарте введены макросы **CMPLX**, **CMPLXF**, **CMPLXL** для создания вещественных комплексных значений с произвольными компонентами. Эти макросы вводятся в заголовочном файле **<complex.h>,** где также содержатся декларации соответствующих функций:

```
double complex CMPLX (double x, double y);
float complex CMPLXF (float x, float y);
long double complex CMPLXL(long double x,
  long double y);
```

Пример инициализации переменной:

```
complex double c1inf = CMPLX (1.0, INFINITY);
```

Улучшенная поддержка Unicode

В стандарт C11 включена поддержка Unicode, основанная на техническом отчете C Unicode Technical Report ISO/IEC TR 19769:2004. Заголовочный файл **<uchar.h>** содержит описания типов **char16_t** и **char32_t** для хранения данных в кодировках UTF-16 и UTF-32, а также функции преобразования из UTF-16 и UTF-32 в строки многобайтных символов и наоборот.

Префиксы **u**, **U** строковых литералов служат для описания строк символов UTF-16/UTF-32; префикс **u8** служит для обозначения строк UTF-8.

Эксклюзивный режим создания и открытия файлов

Функция открытия файла **fopen** теперь позволяет создавать или открывать файлы для эксклюзивного доступа.

```
#include <stdio.h>
FILE *fopen(const char * restrict filename,
     const char * restrict mode);
```

Для этого в качестве последней буквы режима (**mode**) следует задать **'x'**.

Заключение

В стандарте C11 введены существенные дополнения по сравнению со стандартом C99. Хотелось бы отметить то внимание, которое в C11 уделено средствам повышения надежности и безопасности программ, что отражает общую тенденцию к осознанию значимости и развитию этого аспекта информационных технологий.

Некоторые из прочих новшеств C11 существенно связаны с введением многопоточности. Так, селекторы **_Generic** необходимы для реализации стандартной библиотеки **<stdatomic.h>**, содержащей перегруженные функции, а также для реализации пользовательских перегруженных функций, работающих с атомарными данными. Библиотечные функции с проверкой ограничений также приобретают особую значимость именно в контексте разработки многопоточных приложений.

Как отмечается в небольшом, но содержательном обзоре [7], в стандарте C11 исправлены важные недостатки предыдущего стандарта. Некоторые средства, введенные в C99, оказались сложными для реализации на ряде платформ. В результате даже в настоящее время трудно найти реализацию, удовлетворяющую этому стандарту. По этой причине в отдельных организациях были приняты решения об использовании C++ вместо C. С учетом этого обстоятельства в C11 многие новые средства, а также часть средств, введенных в C99, имеют статус выборочных, т. е. необязательных, что значительно облегчает создание реализаций, удовлетворяющих стандарту.

Следует также отметить, что при разработке C11 было уделено особое внимание обеспечению максимальной совместимости с C++11. Согласованность стандартов упрощает как разработку реализаций этих двух языков, так и создание приложений.

Список литературы

1. **ANSI X3.159-1989.** American National Standard for Information Systems — Programming Language C.
2. **ISO/IEC 9899:1990** — Programming Languages — C.
3. **ISO/IEC 9899:1999** — Programming Languages — C.
4. **WG14 N1250.** The C1X Charter. URL: http://www.open-std.org/JTC1/SC22/wg14/www/docs/n1250.pdf.
5. **ISO/IEC 9899:2011** — Information technology — Programming Languages — C.
6. **GCC** the GNU Compiler Collection. URL: http://gcc.gnu.org.
7. **Kalev D.** C11: A New C Standard Aiming at Safer Programming. URL: http://blog.smartbear.com/software-quality/bid/173187/C11-A-New-C-Standard-Aiming-at-Safer-Programming.

УДК 004.052, 004.056

В. В. Липаев, д-р техн. наук, проф., гл. науч. сотр.,
Институт системного программирования РАН,
e-mail: vlip28@mail.ru

Надежность и функциональная безопасность комплексов программ реального времени

Рассматриваются основные факторы, определяющие надежность и безопасное использование программных продуктов реального времени в составе автоматизированных систем. Анализируются характеристики таких систем и среды их функционирования, для которых должна обеспечиваться функциональная безопасность. Представлены требования к проектным решениям, обеспечивающим функциональную пригодность сложных программ. Обсуждаются вопросы организации и планирования жизненного цикла таких программ, в том числе процессы разработки требований к их безопасности. Выделены основные международные стандарты технологических процессов, которые поддерживают функциональную безопасность в жизненном цикле сложных комплексов программ. Значительное внимание уделено испытаниям систем на функциональную безопасность используемых в их составе программных продуктов.

Ключевые слова: *надежность, функциональная безопасность, функциональная пригодность, жизненный цикл, реальное время*

Безопасность и надежность — широкие понятия, относящиеся к объектам в сферах экономики, техники, экологии, государственного управления, социальных процессов, а также ко многим другим областям человеческой деятельности. Эти понятия отражают процессы и состояния, при которых отсутствует недопустимый риск (определяется вероятностью неблагоприятного события) причинениям ущерба техническим системам, жизни и здоровью граждан, имуществу физических и юридических лиц, государственному или муниципальному имуществу, окружающей среде. При этом под надежностью и функциональной безопасностью программных продуктов, процессов производства таких продуктов и поддерживаемых с их помощью систем автоматизации различного назначения, понимается их работоспособное состояние и функционирование в соответствии с требованиями заказчика и документацией (регламентом), при которых ***отсутствуют опасные отказы и недопустимый ущерб*** жизни и здоровью граждан, государственному имуществу и окружающей среде, собственности физических и юридических лиц.

Понятия и характеристики функциональной безопасности автоматизированных систем и программных продуктов (ПП) в их составе близки к соответствующим понятиям надежности. По этой причине далее для сокращения часто под термином "функциональная безопасность" будет подразумеваться и "надежность". Основные различия этих терминов (сущностей) состоят в том, что в показателях надежности учитываются все реализации опасных отказов, а в характеристиках функциональной безопасности регистрируются и учитываются только те отказы, которые приводят к столь большому (включая катастрофический) ущерб, который отражается на безопасности применения системы и/или информации для потребителей. Статистически таких отказов может быть в несколько раз меньше тех, которые учитываются в показателях надежности. Однако методы, влияющие факторы и реальные значения показателей надежности ПП могут служить ориентирами при оценке функциональной безопасности критических систем автоматизации. Для краткости изложения далее вместо "автоматизированной системы" будем использовать термин "система".

В критических системах и ПП, в которых результаты обработки информации и управляющие воздействия непосредственно определяют работоспособность и качество выполнения системой функций, не исключены, а иногда происходят ***аварии и катастрофы, причинами которых является недостаточная функциональная безопасность ПП***.

Основные требования к функциональной безопасности

Любые программные комплексы прежде всего должны иметь экономическую, техническую, научную или социальную эффективность применения, которая должна определяться основной ***целью их использования на всех этапах жизненного цикла*** системы. Эта, именуемая ***системной, эффективность*** может быть описана количественно или качественно в виде набора полезных свойств и характеристик программ, их отличий от имеющихся у других комплексов программ (КП), а также факторов и источников, которые могут определять другие, потенциально возможные показатели эффективности использования программ в составе системы. В результате должна быть формализована ***цель использования ПП и набор требований заказчика и пользователей*** при создании ПП или приобретении, а также предполагаемое назначение и сфера применения [2, 4, 11].

Системная эффективность применения ПП обусловливается степенью удовлетворения потребностей определенных лиц, заказчиков и/или пользователей, которую желательно измерять экономическими категориями: прибылью, стоимостью, трудоемкостью, предотвращенным ущербом, длительностью применения. В стандартах эта эффективность отражается основной, обобщенной характеристикой качества — ***функциональной пригодностью***. Данная характеристика связана с тем, какие функции и задачи решает КП в составе системы для удовлетворения потребностей ее пользователей. При этом другие, конструктивные характеристики (в том числе безопасность) главным образом связаны с тем, как и при каких условиях, заданные функции могут выполняться в соответствии с обозначенными в требованиях показателями качества. Номенклатура и значения всех показателей качества определяются требованиями к функциям ПП в составе системы и в той или иной степени влияют на выполнение этих функций. По этой причине ***выбор и формализация требований функциональной пригодности***, подробное и корректное описание характеризующих ее свойств являются ***основными исходными данными на стадии проектирования*** для установления требуемых значений всех остальных стандартизированных показателей качества.

Для обеспечения эффективности ***КП в составе системы целесообразно придерживаться следующих общих принципов:***

- безопасность применения аппаратуры системы, функциональных программ и данных должна быть комплексной и многоуровневой, ориентированной на все виды угроз с учетом их опасности для потребителя;
- стоимость (трудоемкость) создания и эксплуатации системы обеспечения информационной безопасности должна быть меньше, чем размеры ущерба — риска, наиболее вероятного или возможного (в среднем), неприемлемого потребителям системы при реализации любых потенциальных угроз;
- комплекс программ обеспечения безопасности должен иметь целевые, индивидуальные компоненты контрмер, предназначенные для безопасного функционирования каждого отдельно взятого компонента и задачи, которую призвана выполнять система в целом, с учетом уязвимостей компонентов и ее степени влияния на безопасность;
- реализация комплекса программ обеспечения безопасности не должна приводить к ощутимым трудностям, помехам и снижению эффективности решения основных задач пользователями системы в целом.

Следует заметить, что в требованиях технических заданий реализуемых и уже выполненных проектов создания сложных систем, ***систематически умалчиваются и/или недостаточно формализуются*** понятия и метрики надежности и безопасности ПП. Это замечание, как правило, касается выдаваемой пользователям информации о том, какими характеристиками эти понятия и метрики должны описываться, как их следует измерять и сравнивать с требованиями, изложенными в контракте, техническом задании или в спецификациях. Кроме того, некоторые характеристики безопасности часто вообще отсутствуют в требованиях заказчика и согласованных документах на систему и соответствующий ей КП. Это обстоятельство приводит к их произвольному толкованию и учету, к недостаточному вниманию при проведении испытаний и, соответственно, в ходе анализа их результатов. Такому положению дел может способствовать ограниченность ресурсов, необходимых для достижения и оценивания, описанных в требованиях и де-факто реализованных показателей безопасности КП, а также недостаточная степень формализации и документирования всего процесса их выбора и анализа.

Расширение спектра сфер применения компьютеров и, как следствие, необходимость их корректной работы, зачастую являются определяющими факторами для качественного управления объектами, для успешной эксплуатации этих объектов на предприятии или для безопасности человека. По этой причине тщательная спецификация требований, оценивание и повышение безопасности целевых систем, ПП и обработанной для потребителей информации являются ***ключевым фактором обеспечения их адекватного и эффективного применения***. Такого положения дел можно добиться на основе четкого выделения, строгого определения и обеспечения на практике необходимых характеристик качества ПП. При этом в первую очередь следует отталкиваться от целей использования и подлежащих решению с их помощью задач, а также от назначения систем, в которых такие ПП используют. В качестве приоритета при таком подходе должно рассматриваться использование стандартизированных и формализованных методов.

Непрерывно возрастающая сложность и вследствие этого ***уязвимость систем и ПП от случайных деструктивных воздействий*** выдвинули в число актуальных вопросы исследовательского и инженерного характера, связанные с обеспечением надежности и безопасности систем и ПП в их составе. К числу таких продуктов в первую очередь относятся ***стратегически важные***, которые предназначены для применения в системах органов государственного управления, в промышленных объектах и в изделиях военного назначения. На направлениях разрешения отмеченных вопросов выделились две научно-технические области: информационная безопасность и функциональная

безопасность. В сферу ***информационной безопасности*** входят вопросы, связанные в основном с защитой от деструктивных воздействий на ресурсы информационных систем. К вопросам ***функциональной безопасности*** относятся обусловленные отказами отдельных элементов, потерей работоспособности систем в целом и их ПП вследствие проявления в реальном времени непредумышленных дефектов программ и отрабатываемых ими данных, случайных отказов аппаратуры и воздействий внешней среды. Доминирующие категории обеспечения безопасности автоматизированных систем, их программных средств и баз данных можно условно разделить на ***два крупных класса***:

- системы, поддерживающие процессы сбора и обработки, хранения и анализа больших объемов информации из внешней среды с активным участием в их использовании большого числа пользователей, для реализации которых должна обеспечиваться конфиденциальность, целостность и доступность данных потребителям, что отражается требованиями к ***высокой степени их информационной безопасности***;
- системы и объекты автоматизации, в аппаратуру которых ***встроены КП*** для управления и обработки информации в реальном времени, основные задачи которых состоят в обеспечении достоверной реализации эффективного и устойчивого управления объектами внешней среды при относительно малом участии пользователей в их решении и высоких требованиях к ***функциональной безопасности***.

Значения функциональной безопасности тесно ***коррелированы с характеристикой корректности ПП***. Однако можно добиваться высокой безопасности функционирования программ при относительно невысокой их корректности ***за счет автоматизации процессов оперативного сокращения ущерба путем восстановления программ при отказах***. Степень покрытия тестами структуры функциональных компонентов и КП в целом при разработке и испытаниях может служить ориентиром для прогнозирования их потенциальной безопасности.

Заказчиком совместно с пользователем в требованиях к каждой конкретной системе должны быть установлены ***пороговые значения***, разделяющие виды и размеры ущерба, а также ***характеристики восстановления***. Такие характеристики следует относить либо к нарушению функциональной безопасности системы, либо только к снижению уровня ее надежности. Эти пороговые значения зависят от назначения и базовых характеристик, ***отражающих функциональную пригодность*** конкретной системы или программного комплекса. При малом (ниже принятого порога) ущербе вследствие отказа и при быстром восстановлении системы такие события следует относить к ***снижению уровня ее надежности***. Если последствия отказов, представляющие собой-какой либо вид ущерба, превышают заданный критический порог, то такие события относятся к ***нарушению функциональной безопасности***.

При создании сложных КП одна из основных задач на этапе их проектирования состоит в выявлении факторов, от которых они зависят, в создании методов и средств уменьшения их влияния на безопасность. Необходимо ***оценивать уязвимости*** функциональных компонентов системы для различных деструктивных воздействий и степень их влияния на основные характеристики ее безопасности. В зависимости от результатов такого оценивания следует ***распределять ресурсы*** для создания системы и ее компонентов, обладающих равновеликими уровнями ***безопасности*** функционирования с минимальным обобщенным риском при любых негативных внешних воздействиях. Исходя из этого, уже на ранних стадиях проектирования должны быть сформулированы соответствующие методы и контрмеры, которые, в свою очередь, определяют необходимые функции и механизмы средств обеспечения должного уровня работоспособности и безопасности.

Для обеспечения необходимого уровня функциональной безопасности систем предпринимают ***соответствующие контрмеры*** и создают специализированные средства и системы. Такие меры включают подготовку набора взаимосвязанных нормативных документов, организационно-технических мероприятий и реализацию соответствующих им методов и программных компонентов, предназначенных для предупреждения и/или ликвидации негативных последствий отказов, угроз безопасности, для их выявления и локализации. Создание таких комплексов повышения безопасности предусматривает ***планирование и реализацию целенаправленной политики*** комплексного обеспечения функциональной безопасности системы, а также ***эффективное распределение ресурсов на контрмеры и средства обеспечения безопасности***. Разработчики и заказчики должны анализировать возможные угрозы, чтобы решать, какие из них действительно адекватны внешней среде, системе и программному продукту.

При анализе характеристик ***функциональной безопасности автоматизированных систем, которые поддерживаются комплексами программ реального времени,*** можно выделять ***два класса таких систем. Первый класс*** составляют системы, имеющие ***встроенные комплексы программ реального времени***, в автоматизированном режиме управляющие объектами или процессами. Время необходимой реакции на отказы таких систем обычно исчисляется секундами или долями секунды. Процессы восстановления их работоспособности должны оперативно проводиться за это время в автоматизированном режиме (бортовые системы в авиации, на транспорте, в некоторых средствах вооружений, системы управления атомными электростанциями).

Системы второго класса применяют для управления процессами сбора, хранения и обработки деловой информации из внешней среды, в которых активно участвуют специалисты-операторы (банковские, системы в органах государственного управления, штабные военные системы). Допустимое время реакции на опасные отказы в этих системах может составлять десятки секунд и минуты. Операции по восстановлению их работоспособности частично могут быть доверены специалистам-администраторам по обеспечению надежности и функциональной безопасности.

Заказчиком, как правило, ***не полностью, с недостаточной степенью детализации определяются и описываются требования и особенности функционирования взаимодействующих с системой объектов вне-***

шней среды. Сложность и, как следствие, неопределенность их представления адекватны сложности всей системы, функциональная безопасность которой должна обеспечиваться в течение ее жизненного цикла (ЖЦ). Заметим, что квалификация, знания и опыт потенциальных пользователей изменяются по мере освоения ими функциональных возможностей системы и оценки уровня ее работоспособности. Как следствие, отмеченные обстоятельства не позволяют точно определить ее реальную надежность и уровень безопасности. Различия в знаниях и опыте персонала, эксплуатирующего систему и ее программное обеспечение, создают дополнительные трудности в определении уровня безопасности. Они усугубляются большим числом ***субъективных факторов, определяющих знания и опыт различных специалистов, которые участвуют в разработке и эксплуатации системы***.

В процессе проектирования, разработки и эксплуатации системы ее функциональные возможности, операционная среда, аппаратурное обеспечение с течением времени развиваются. Это обстоятельство приводит к объективной необходимости соответствующего изменения методов и средств обеспечения функциональной безопасности. Таким образом, ***обеспечение функциональной безопасности сложных систем должно проводиться с учетом перманентного развития знаний о свойствах системы.*** Такой сложный, непрерывный, многофакторный процесс трудно реализовать на практике единовременно, поэтому его целесообразно осуществлять поэтапно, итерационным образом.

Современные технологии производства программного обеспечения способствуют повышению функциональной безопасности систем, снижению потенциального ущерба от ее нарушения. Однако практически всегда открытым остается вопрос, ***насколько применяемые методы оправдывают затраты*** на их реализацию. По этой причине оценить эффективность методов и средств обеспечения функциональной безопасности в общем случае, строго количественно по заранее заданному многопараметрическому критерию не удается. Для конкретных ПП и систем приходится устанавливать частные критерии эффективности с учетом интуитивных оценок допустимых затрат на их реализацию. ***Испытания, эксплуатация и сертификация КП*** способствуют снижению неопределенности оценок их эффективности и итерационному приближению к практически приемлемому уровню функциональной безопасности систем.

Роль ***негативных воздействий и их разрушительные последствия быстро возрастают*** в связи с ростом сложности разработки и применения современных компьютерных систем и повышением уровня ответственности решаемых с их помощью задач. ***Возрастает сложность внешней и операционной среды,*** в которой функционирует прикладное программное обеспечение, и повышается уровень ответственности функций, реализуемых системой. Объективное повышение сложности функций, реализуемых программами в современных системах, непосредственно приводит к увеличению объема их кода и трудоемкости создания. В соответствии с повышением сложности программ ***возрастают относительное и абсолютное количества выявляемых и остающихся в них дефектов и ошибок***. Это обстоятельство прямо отражается на снижении потенциальной безопасности их функционирования. По мере увеличения сложности задач, которые решаются с помощью программ, возрастает число потенциальных ошибок, которые могут угрожать катастрофическими последствиями в системах, выполняющих критические функции управления крупными, дорогими объектами или процессами особой важности. Необходимо, чтобы общие цели безопасности системы были согласованы с определенными ранее целями применения и характеристиками функциональной пригодности системы или программ как продукта, а также со всеми известными сведениями о внешней среде, в которой система работает.

Внешняя среда комплексов программ реального времени

Функции автоматизированных систем и их ПП реального времени реализуются на базе некоторых аппаратных средств, в операционном окружении и в пользовательской ***внешней среде***. Характеристики внешней среды существенно влияют на функциональную пригодность программ. Для выполнения обозначенных в технических требованиях к системе функций КП (далее для краткости — требуемых функций), необходима ***адекватная исходная информация от объектов внешней среды***, содержание которой должно полностью обеспечивать реализацию этих функций. Полнота формального описания номенклатуры, структуры и качества входной информации для выполнения требуемых функций является одним из важных факторов при определении функциональной пригодности ПП в соответствующей внешней среде. Так как без дефектов и ошибок на практике невозможно создать и применять сложные КП, эти вопросы должны подробно изучаться и обобщаться в некоторую совокупность знаний о внешней среде. В ходе такого анализа и обобщения следует оценивать возможные уязвимости, содержание и характеристики дефектов и ошибок в программах, их проявление и негативные последствия для потребителя, угрозы и ущерб при нарушении безопасности функционирования системы в целом.

Характеристики внешней среды, а именно прикладные сферы применения КП, цели и задачи пользователей, уровень автоматизации их функций и многие другие факторы в значительной степени определяют методы и свойства средств обеспечения безопасности автоматизированных систем. При последующем анализе внимание акцентируется на определении требований к сложным аппаратно-программным системам, на вопросах их применения для поддержки ***функциональной безопасности***. Соответственно, ущерб при отказах определяется уязвимостью и нарушением режима корректного (регламентированного) выполнения системой функциональных требований по ее назначению при ограниченных ресурсах на их реализацию. В отмеченном выше смысле среда (средства и методы) обеспечения функциональной безопасности системы и ее программного обеспечения относится к внешней среде, являясь ее составной частью.

Среда обеспечения функциональной безопасности программ включает политики и программы обеспечения безопасности предприятий и систем, опыт, специальные навыки и знания, определяющие контекст предполагаемого применения системы. Среда включает также возможные угрозы безопасности, присутствие которых в этой среде установлено или предполагается. В процессе систематизации и формального описания среды функциональной безопасности следует принимать во внимание:

- предназначение системы и комплекса программ, включая функции ПП и предполагаемую сферу его применения;
- активы — программы и данные для решения основных задач, реализующие функции системы, к которым применяются требования и/или политики безопасности; компоненты, которые подчинены требованиям безопасности реализации системы;
- внешнюю среду в той ее части, которая определяет все аспекты, обеспечивающие эксплуатацию системы, включая мероприятия, относящиеся к средствам физической защиты и к персоналу.

На основании разработанных политик безопасности, оценок угроз и рисков их реализации следует сформировать исходные положения, ***относящиеся к функциональной безопасности системы и основного КП***. К их числу относятся:

- факторы, которым должна удовлетворять среда для того, чтобы система или ПП становились безопасными;
- угрозы безопасности компонентов системы, в числе которых должны быть идентифицированы все угрозы к каждому компоненту со стороны внешней среды, относящиеся к предмету функциональной безопасности;
- угрозы, которые раскрываются через понятия источника угроз, предполагаемого метода их реализации, предпосылки для отказов, и идентификация компонентов, которые являются объектами таких отказов;
- политика безопасности, в которой были бы достаточно точно идентифицированы и описаны цели, методы и правила реализации действий, направленных на обеспечение функциональной безопасности системы.

Испытания и обеспечение функциональной безопасности программных продуктов

Работоспособность автоматизированных систем и поддерживающих их ПП может быть обеспечена при выполнении исходных требований, которые формализуются и соблюдаются при их разработке, отладке и испытаниях таких систем. Вместе с тем ***исходные значения реальных параметров подконтрольной подсистемы и ее программного обеспечения могут отличаться от заданных техническим заданием и от используемых при эксплуатации программ и баз данных.*** При таких условиях режим функционирования программ трудно предсказать заранее. Как следствие, весьма вероятны различные аномалии, завершающиеся отказами, отражающимися на безопасности системы. ***Требования по функциональной безопасности*** систем, содержащих программы реального времени, поддерживают путем использования ***современных регламентированных технологических процессов*** и инструментальных средств обеспечения таких процессов на всех этапах ЖЦ создаваемых систем. Они должны быть поддержаны группой международных стандартов, определяющих состав и процессы выполнения требований к заданному уровню функциональной безопасности систем и программ.

Ограниченность ресурсов различных видов влияет на технико-экономические показатели, качество и функциональную безопасность производства всей системы и ПП. В результате сложность программ и баз данных, а также доступные ресурсы для их реализации становятся косвенными критериями или факторами, влияющими на выбор методов разработки и на качество, которого можно при этом добиться. Разработку систем и ПП должны завершать ***комплексные испытания и удостоверение, подтверждающее достигнутый уровень функциональной безопасности*** систем с используемыми в их составе ПП. Результаты испытаний должны предоставлять возможность совершенствования их характеристик и уровень функциональной безопасности путем соответствующих корректировок программ. Повышение уровня функциональной безопасности систем целесообразно осуществлять на основе анализа выявленных дефектов и ***оперативного восстановления вычислительного процесса, программ и данных (рестарта)*** после обнаружения аномальных режимов и отказов ПП. Этому могут способствовать накопление, мониторинг и хранение данных о выявленных дефектах, сбоях и отказах в процессе исполнения программ и обработки данных.

Степень некорректности программ определяется вероятностью попадания реальных исходных данных в область, которая задана требованиями спецификации и технического задания, однако не была проверена при испытаниях. В реальных условиях по различным причинам исходные данные могут попадать в области значений, вызывающих сбои, не проверенные при испытаниях, а также не заданные требованиями спецификации и технического задания на ПП. Если в этих ситуациях происходит достаточно быстрое восстановление, при котором не фиксируется отказ, то такие события не влияют на основные показатели безопасности — наработку на отказ и коэффициент готовности. Следовательно, ***безопасность функционирования программ является понятием динамическим, изменяющимся во времени и существенно отличается от понятия корректности программ.***

Отсутствие возможности в процессе эксплуатации предсказать вид, место и время проявления деструктивных воздействий, обусловленных дефектами ПП, приводит к необходимости создания специальных дополнительных систем оперативной защиты от непредумышленных и/или случайных искажений вычислительного процесса, программ и данных. Системы оперативной защиты предназначены для выявления и блокирования процессов распространения негативных последствий проявления дефектов и уменьшения их влияния на надежность функционирования ПП до устранения их источников. Для этого в ПП должна вводиться временная, программная и информационная избыточность, осуществляющая оперативное обнаружение дефектов

функционирования, их идентификацию и автоматическое восстановление (рестарт) регламентированного режима функционирования ПП.

Классификация сбоев и отказов в программах при отсутствии их физического разрушения может проводиться по размеру причиненного ими ущерба или по временному показателю ***длительности восстановления*** после любого искажения программ, данных или вычислительного процесса, которые зарегистрированы как нарушение работоспособности [3, 7, 12]. При длительности восстановления меньшей заданного порога, дефекты и аномалии при функционировании программ следует относить к ***сбоям***. При восстановлении, превышающем по длительности заданное пороговое значение, происходящее искажение соответствует ***отказу***. Временная зона перерыва нормальной выдачи информации и потери работоспособности, которую следует рассматривать как зону сбоя, тем шире, чем более инертный объект находится под воздействием сообщений, подготовленных данным ПП. Пороговое время восстановления работоспособного состояния системы, при превышении которого следует фиксировать отказ, близко ко времени решения задач для подготовки информации соответствующему абоненту. Соответствующая этому отклонению временная зона перерыва выдачи информации потребителю позволяет установить ***границу допустимой длительности нарушения работоспособности, которая разделяет зоны сбоев и отказов.***

Чем более инерционным является потребитель информации, тем больше может быть время отсутствия у него результатов функционирования ПП без катастрофических последствий нарушения работоспособности, соответствующего отказу [2, 10]. Такое ***допустимое отклонение результатов после перерыва функционирования ПП зависит в основном от динамических характеристик источников и потребителей информации.*** Таким образом, установив в результате системного анализа динамических характеристик системы размер порогового значения, можно определить интервал времени функционирования ПП при отсутствии выдачи данных потребителю, которые разделяют события сбоя и отказа без физического разрушения программ.

Безопасность функционирования ПП наиболее широко характеризуется способностью к безотказному функционированию и ***восстанавливаемостью (временем восстановления)*** работоспособного состояния после произошедших сбоев или отказов. ***Восстанавливаемость*** характеризуется полнотой и длительностью восстановления регламентированного (нормального) режима функционирования программ в процессе ***перезапуска (рестарта)***. Для перезапуска необходимы вычислительные ресурсы и время. По этой причине полнота и длительность восстановления регламентированного режима функционирования системы после сбоев отражают качество и безопасность ПП и возможность его использования по прямому назначению.

Показатели безопасности ПП в значительной степени адекватны аналогичным характеристикам, принятым для других технических систем. Наиболее широко используется ***критерий длительности наработки на отказ***. Для определения этого параметра измеряется время работоспособного состояния системы между последовательными отказами или моментами времени, когда система начинала нормально функционировать после них. Вероятностные характеристики этой величины в нескольких формах используют как разновидности критериев функциональной безопасности. Критерий безопасности восстанавливаемых систем учитывает возможность многократных отказов и восстановлений.

Обобщенным показателем процесса восстановления является ***длительность восстановления*** и ее вероятностные характеристики. Этот критерий учитывает возможность многократных отказов и восстановлений. Обобщение характеристик отказов и восстановлений осуществляется путем применения критерия ***коэффициент готовности***. Этот показатель отражает вероятность иметь восстанавливаемую систему в работоспособном состоянии в произвольный момент времени. Значение коэффициента готовности соответствует доле времени полезной (в соответствии с регламентом по назначению) работы системы на достаточно большом интервале времени, содержащем отказы и восстановления.

Наработка на отказ учитывает ситуации ***потери работоспособности***, когда длительность восстановления велика и превышает пороговое значение времени, разделяющее события сбоя и отказа. При этом большое значение имеют средства оперативного контроля и восстановления (рестарта). Качество проведенной отладки программ более полно отражает значение интервала времени между потерями их работоспособности — ***наработка на отказы***, независимо от того, насколько быстро произошло восстановление. По этой причине при оценке времени, необходимого на отладку, целесообразно измерять и контролировать наработку на отказы. При этом объем и время тестирования в ряде случаев следует устанавливать по наработке на отказы, принимая во внимание ***эффективности средств рестарта***.

Для программ, используемых в системах обработки информации и управления в реальном времени, наработка на отказ измеряется десятками и сотнями часов. При этом для особенно важных или широко тиражируемых систем она может достигать десятков тысяч часов. Реально очень трудно добиться наработки на отказы на уровне сотен часов. Поэтому для получения высокого уровня безопасности программ нельзя ограничиваться тестированием и отладкой без использования средств рестарта. Априори нельзя обеспечить абсолютное отсутствие дефектов проектирования и производства в сложных ПП. Как следствие, ***безопасность их функционирования имеет всегда конечное, ограниченное значение***.

Для безопасного использования и модификации ПП квалифицированными специалистами-пользователями документация должна точно отражать назначение, функции, характеристики и требования к ним. С этой целью эксплуатационные документы необходимо тестировать (проверять) на полное соответствие всей ***совокупности перечисленных требований на ПП***, согласованных разработчиками и заказчиком. В результате такие документы можно использовать как отдельный, независимый при разработке ***эталон и вид тестов***, направленный на реализацию требований к функциям и

характеристикам ПП. Разработчики документов должны обеспечивать безопасное, комфортное и корректное применение КП пользователями. Для этого в документах следует ясно и непротиворечиво излагать положения концептуального характера, содержание технологических процедур и операций, которых нужно придерживаться для штатного функционирования и получения ***требуемых результатов***. Организация документирования должна определять стратегию, стандарты, процедуры, распределение ресурсов и планы создания, изменения и применения документов на КП. Для этого в общем случае должны быть выделены ***специалисты***, которые обязаны планировать, утверждать, выпускать, распространять и сопровождать комплекты апробированных и утвержденных эксплуатационных документов. Они должны стимулировать разработчиков программных комплексов, осуществлять непрерывное, регламентированное документирование процессов и результатов своей деятельности, а также контролировать полноту и качество утвержденных эксплуатационных и отчетных документов перечисленных далее двух классов.

Первый класс составляют ПП систем автоматизированного управления динамическими объектами и процессами в реальном масштабе времени. При их применении допускается минимальное вмешательство пользователями в процедуры управления и, соответственно, необходим небольшой объем эксплуатационных документов, выделяемых из стандартизированного комплекта. Для продуктов ***второго класса*** пользователи могут применять широкий набор процедур управления, которые должны быть регламентированы достаточно полным и подробным содержанием эксплуатационных документов. Пользователей таких ПП можно разделить на две крупные группы, каждая из которых должна быть обеспечена ***комплектной эксплуатационной документацией***:

- администраторы, занимающиеся подготовкой ПП к эксплуатации и обеспечивающие его функционирование и использование по прямому назначению;
- операторы — пользователи, реализующие применение ПП в системе, их функционирование, обработку и анализ результатов.

Стандарты **ISO 15910:1999** и **ISO 18019:2004** являются наиболее полными нормативными документами, регламентирующими процессы ***создания безопасной, эксплуатационной документации для пользователей крупных ПП***. В них изложены детальные структуры планов документирования, ориентированных на разработчиков документов, которые соответствуют требованиям на ПП, и процедуры контроля реализации плана. Эксплуатационная документация должна проходить ***испытания на достоверность***. Такие испытания должны быть спланированы и реализованы на базе требований к функциям и характеристикам, а также к безопасному применению эксплуатационных процедур реального ПП.

Ресурсы для обеспечения функциональной безопасности программных продуктов

Для прямых, количественных измерений значений параметров, определяющих состояние функциональной безопасности, необходимы ***инструментальные средства, которые могут быть встроены в аппаратное обеспечение системы, в операционную систему, и в программные компоненты, выполняющие другие функции.*** Эти средства должны автоматически в режиме реального времени определять и регистрировать отказы, дефекты и искажения вычислительного процесса, программ и данных. Накопление и систематизация проявлений отказов при исполнении программ позволяют оценивать основные показатели уровня функциональной безопасности ПП, помогают определять причины сбоев и отказов, дают возможность подготавливать данные для реализации действий по его повышению. Регулярная регистрация и обобщение таких данных способствуют устранению ситуаций, негативно влияющих на функциональную пригодность и другие базовые характеристики ПП.

Задействованные на поддержание безопасности вычислительные ресурсы и их эффективное распределение по отдельным видам потенциальных уязвимостей оказывают значительное влияние на достижения высокого уровня комплексной безопасности системы [3, 11]. Важными ресурсами, которые приходится учитывать при проектировании, являются ***финансовые затраты. Они определяют трудоемкость разработки и стоимость работ по сопровождению комплекса средств обеспечения безопасности системы***. При проектировании такого работающего в режиме реального времени комплекса должна быть ***предусмотрена программная и информационная избыточность***. Такой подход, как правило, предполагает использование дополнительных ресурсов внешней и внутренней памяти базового компьютера. Для эффективного функционирования комплекса средств обеспечения функциональной безопасности необходимо обеспечить определенное временное упреждение, которое достигается высокой производительностью базового компьютера. Перечисленные ***вычислительные ресурсы в процессе обеспечения функциональной безопасности*** используют также для:

— контроля и корректировки искажений информации, поступающей от внешней среды и источников данных;

— оперативного контроля и обнаружения дефектов исполнения программ и обработки данных при использовании ПП по основному назначению;

— размещения и обеспечения регламентированного режима функционирования применяемых средств защиты от различных видов угроз безопасности системы;

— генерации тестовых наборов или хранения тестов для контроля работоспособности, сохранности и целостности ПП при функционировании системы;

— накопления, хранения и мониторинга данных о выявленных инцидентах, попытках несанкционированного доступа к информации, о дефектах, сбоях и отказах в процессе исполнения программ и обработки данных, влияющих на безопасность;

— реализации процедур анализа и мониторинга выявленных дефектов и оперативного восстановления вычислительного процесса, программ и данных (рестарта) после обнаружения дефектов и отказов функционирования системы.

Имитаторы среды для обеспечения и испытаний функциональной безопасности программных продуктов

Экспериментальные исследования и прямые измерения значений параметров, определяющих ***реальный уровень функциональной безопасности*** сложных систем и КП, в ряде случаев затруднены. Как правило, это связано с необходимостью учитывать и оценивать достаточно редкие, но тем не менее вероятные отказы, которые приводят к катастрофическим последствиям. Для этого приходится проводить длительные эксперименты с ***имитацией критических и опасных ситуаций, которые могут возникать в процессе эксплуатации*** систем. Такие исследования на реальных системах могут быть связаны с большим риском и большими ресурсозатратами, включая финансовые, вычислительные и др. Кроме того, редкие и одиночные отказы не позволяют оценить статистически достоверные значения длительностей наработки на отказы и размер вероятного ущерба. По этой причине высокий уровень функциональной безопасности ПП достигается и определяется преимущественно за счет технологий обеспечения качества на всех этапах ЖЦ систем и ПП. Для этого в стандартах, регламентирующих функциональную безопасность систем и КП, значительное ***внимание уделяется технологическим процессам и инструментальным системам обеспечения безопасности*** и качества КП на всех этапах их ЖЦ.

Ограниченные ресурсы времени реализации проектов ПП являются одним из ***факторов***, влияющих на потенциально достижимое качество и уровень безопасности КП. При реальном допустимом времени реализации проекта ограничивают затраты на все промежуточные этапы работ, и тем самым на качество их выполнения. При современных технологиях ***полностью новые, сложные КП***, которые, как правило, создают в течение 2...4 лет, ограничены объемом 10^6...10^7 строк текста. Вместе с тем даже относительно небольшие программы высокого качества и уровня безопасности в сотни тысяч строк по полному технологическому циклу с сертификационными испытаниями продукции редко создают за время, меньшее, чем полгода — год [7, 12].

Границу снизу на время создания программного комплекса определяют естественный технологический процесс коллективной разработки и необходимость выполнения ряда взаимоувязанных работ на его последовательных этапах. Такие работы в рамках сложившегося регламента их выполнения обеспечивают получение сложных программных комплексов требуемого качества. Даже большие усилия по автоматизации технологии и организации производства программ приводят к сокращению длительностей только на десятки процентов, в то время как трудоемкость может уменьшаться на порядок и больше.

Крупные затраты, достигающие 30 % от полных затрат на разработку сложных программных комплексов, приходятся на ***верификацию и тестирование их компонентов***. Такой подход призван обеспечить корректность и функциональную безопасность ПП в целом. Затраты на тестирование и испытания ПП обычно могут быть достаточно четко выделены из остальных затрат, так как в этих процессах могут непосредственно участвовать заказчик и пользователи. Размер этих затрат без учета ресурсов, необходимых для имитации внешней среды, может составлять около 10 % от общих затрат на разработку. При этом практически невозможно разделять затраты на оценивание отдельных стандартизированных характеристик.

Затраты на функциональную безопасность ПП определяются требуемым уровнем защищенности и сложностью (объемом) специализированных программ для ее реализации. При наличии особенно высоких требований к безопасности критических ПП эти затраты могут даже в 2—3 раза превышать затраты на решение задач, которые обеспечивают выполнение базовых функций системы. Для типовых систем трудоемкость создания специализированных программных комплексов обеспечения функциональной безопасности обычно составляет 20...40 % от затрат на решение основных задач при требованиях наработки на отказ в десятки тысяч часов и для обеспечения автоматического рестарта в системах.

Возможные затраты трудовых и временных ресурсов ***на развитие и совершенствование качества и безопасности КП*** в процессе сопровождения зависят не только от внутренних свойств программ. Такие затраты зависят также от потребностей пользователей в новых функциях и от готовности заказчика и разработчика удовлетворить эти потребности. Размер этих затрат определяют рыночная конъюнктура для данного ПП и коммерческая целесообразность его модификации и развития [7, 12].

При создании сложных ПП, одной из больших составляющих затрат могут быть ***ресурсы на генерацию тестов***. В ряде случаев они ***соизмеримы*** с затратами на реализацию основных функциональных возможностей КП, в том числе для обеспечения безопасности. Такие затраты определяются принципиальным соответствием ***сложности необходимых наборов тестов*** и тестового покрытия программ, а также ***сложностью функций***, которые призваны реализовать подлежащий испытаниям ПП. Создание представительных совокупностей тестов возможно путем использования реальных объектов внешней среды или с помощью ***программных имитаторов, адекватно представляющих эти объекты по реализуемым функциям и генерируемой информации***. При этом возникает вопрос: какой метод и в каких случаях более выгоден по затратам на генерацию тестов и по обеспечению необходимой степени покрытия тестами подлежащих испытаниям ПП. Затраты ресурсов на натурные эксперименты для генерации тестов при проведении разработки, испытаний и определении качества пропорциональны реальному времени функционирования ПП и затратам на применение привлекаемых для испытаний ресурсов реальной внешней среды.

Имитаторы тестов необходимы не только для оценивания характеристик безопасности и качества КП, которых удается добиться. Они также важны для их комплексной отладки, испытаний и при создании отдельных версий программного обеспечения. По этой причине затраты на программные имитаторы и их экономическую эффективность целесообразно рассматривать с учетом всего комплекса задач, которые они

способны и должны решать в ЖЦ программного комплекса. Затраты на программную имитацию тестовых данных определяются ресурсами, которые необходимы на проектирование и эксплуатацию сложных комплексов программ. К их числу относятся затраты:

— на разработку комплекса программ для имитации информации внешних объектов и среды их функционирования;

— на эксплуатацию программ имитации за время проведения тестирования, испытаний и/или определения характеристик безопасности и качества тестируемого ПП;

— на первичную установку и эксплуатацию моделирующего компьютера и вспомогательного оборудования, используемого в имитационном стенде.

Стандарты функциональной безопасности программных продуктов

Формальному описанию показателей качества программных комплексов посвящена группа нормативных документов. Выделены характеристики, которые позволяют оценивать ПП с позиций пользователя, разработчика и управляющего проектом. Стандартизированные в документах **ISO 9126:1-4:2002** и в **ISO 25010:2008** характеристики качества ПП различают по степени их влияния на эффективность применения КП по прямому назначению. Высшие приоритеты при этом, естественно, должны присваиваться свойствам и атрибутам функциональной пригодности, которые напрямую влияют на достижение основных ***стратегических целей*** использования ПП. Все остальные стандартизированные характеристики должны способствовать обеспечению ***тактических целей*** реализации выбранных требований. Решение этих задач должно быть направлено на обеспечение высокой функциональной пригодности ПП ***путем сбалансированного улучшения характеристик качества в условиях ограниченных ресурсов, которые выделяются на сопровождение продукта в течение его ЖЦ***.

Прямое измерение параметров, определяющих уровень функциональной безопасности объектов и систем, сопряжено с большими трудностями. Они, как правило, обусловлены необходимостью фиксировать редкие, непредсказуемые отказы с неожиданным ущербом. По этой причине ***основное содержание стандартов*** в области функциональной безопасности в основном отражают ***методы, процессы и технологии обеспечения высокой безопасности*** систем в ЖЦ. При этом очень мало внимания уделяется определению уровню безопасности, которого удается добиться. Соответственно, усилия разработчиков систем и ПП оказываются сосредоточены на методах регламентирования и обеспечения качества ПП и систем на всех этапах их ЖЦ.

Встроенные ПП, в частности, реализующие функции обеспечения безопасности управления динамическими объектами и системами, которые используют, например, в авиакосмическом комплексе, на транспорте, создают с применением технологий и инструментальных средств, аналогичных тем, что применяют при разработке других классов сложных ПП реального времени. Наиболее общими и широко применяемыми стандартами, которые регламентируют процессы ЖЦ крупных программных комплексов высокого качества, являются стандарты **ISO 12207:2008**. Следует однако заметить, что в этих стандартах непосредственно безопасности уделяется относительно мало внимания, а специфика обеспечения функциональной безопасности сложных программных комплексов обычно не выделяется и не комментируется. В то же время эти стандарты целесообразно адаптировать и применять при создании ПП, реализующих функции безопасности. Последнее обстоятельство, в частности, поддерживается ссылками на них в стандартах, посвященных непосредственно безопасности ПП и систем, реализуемых на базе средств вычислительной техники.

Основным задачам обеспечения безопасности ПП и систем с использованием средств вычислительной техники посвящена значительная группа стандартов, среди которых следует выделить **IEC 61508-3**, **IEC 60880, ISO 15408, ISO 13335, ГОСТ Р 51904**. Рекомендуемые процессы и структура ЖЦ программных комплексов в этих стандартах несколько различаются, однако составы процессов достаточно близки. Значительная часть положений этих стандартов посвящена требованиям, методам и процессам производства и всего ЖЦ программных комплексов и систем, которые, в частности, используют для обеспечения безопасности. Рекомендации этих стандартов могут быть адаптированы и полезны для обеспечении ***функциональной безопасности*** встроенных ПП [9, 13]. Положения большинства представленных стандартов ***близки по существу, однако значительно различаются по терминологии и структуре их изложения.***

Список литературы

1. **Блэк Р.** Ключевые процессы тестирования. Пер. с англ. М.: ЛОРИ, 2006.

2. **Вигерс К. И.** Разработка требований к программному обеспечению. Пер. с англ. М.: Русская редакция, 2004.

3. **Галатенко В. А.** Основы информационной безопасности. Курс лекций. М.: ИНТУИТ, 2003.

4. **Дастин Э., Рэшка Д., Пол Д.** Автоматизированное тестирование программного обеспечения. Внедрение, управление и эксплуатация. Пер. с англ. М.: ЛОРИ, 2003.

5. **Концепция** защиты средств вычислительной техники и автоматизированных систем от несанкционированного доступа к информации. Руководящий документ. Гостехкомиссия России. М.: Военное издательство, 1999.

6. **Липаев В. В.** Надежность программных средств. М.: СИНТЕГ, 1998.

7. **Липаев В. В.** Методы обеспечения качества крупномасштабных программных средств. М.: СИНТЕГ, 2003.

8. **Липаев В. В.** Сертификация программных средств, Учебник. М.: СИНТЕГ, 2010.

9. **Трубачев А. П., Долинин М. Ю., Кобзарь М. Т.** и др. Оценка безопасности информационных технологий. Общие критерии. Пер. с англ. / Под ред. В. А. Галатенко. М.: СИП РИА, 2001.

10. **Устинов Г. Н.** Основы информационной безопасности систем и сетей передачи данных. М.: СИНТЕГ, 2000.

11. **Boehm B. W.** Software risk management. Washington: IEEE Computer Society Press, 1989.

12. **Jones C.** Applied software measurement, assuring productivity and quality. NY.: McGraw-Hill, 1996.

13. **Smith D., Simpson K.** Functional Safety. A Straightforward Guide IEC 61508 and Related Standards. Oxford: Planta Tree, 2001.

УДК 65.015.13

К. Е. Селезнёв, канд. техн. наук, доц., Воронежский государственный университет,
e-mail: skostik@relex.ru

Синтез целевых информационно-поисковых систем

Излагается способ формулирования количественных требований к информационно-поисковым системам. Предложенные математические модели имеют теоретическое значение, также они позволяют сравнивать различные информационно-поисковые системы между собой с точки зрения их эффективности. Кроме того, предложенный подход дает возможность описывать широкий класс информационно-поисковых систем и применим для задания ограничений на характеристики ИПС при построении узкоспециализированных поисковых систем.

Ключевые слова: *информационный поиск, жизненный цикл ПО*

Введение

К настоящему времени во многих предметных областях накоплены большие объемы различной информации, представленные в виде электронных архивов. В самом общем виде их функции сводятся к двум основным операциям: занесение новой информации в архив и ее поиск в архиве. Ранее с этими задачами хорошо справлялись информационно-поисковые системы (ИПС), основные принципы работы которых были сформулированы достаточно давно.

При анализе ИПС основное внимание отводится "пользовательской" стороне, которая касается вопросов практического использования системы, эффективности ее работы, точности поиска и т. д. Сейчас возникает большое число различных целевых ИПС, функционирующих не в сети Интернет, а рамках небольшой организации и ее внутренней корпоративной структуры. Такие системы полностью подчиняются общим законам функционирования ИПС, однако учет специфики организации (предметная область информации, стандарты в оформлении документов, принятая классификация документов и т. д.) позволяет существенно повысить эффективность системы в целом.

Для решения поставленной задачи необходимы теоретическая база и практическая методология разработки и внедрения целевых ИПС. Сюда входят, как минимум, три основополагающих вопроса. Во-первых, чем отличаются различные ИПС друг от друга; во-вторых, каким образом ИПС могут быть описаны параметрически; и, в-третьих, каких сочетаний параметров можно добиться при практической реализации системы (и каким образом), а каких добиться нельзя.

Упомянутые параметры и характеристики ИПС можно разделить на две группы. Первая группа описывает ИПС со стороны пользователя и во многом повторяет существующие модели их описания, связанные с точностью поиска, релевантностью, охватом и т. д. Вторая группа параметров характеризует ИПС со стороны разработчиков и обслуживающего персонала.

Главной целью исследования, результаты которого представлены в данной статье, является формирование перечня основных параметров и характеристик ИПС, а также определение их взаимосвязей в составе формальной модели системы.

Математическая модель использования ИПС

С точки зрения классического подхода, ИПС функционируют согласно широко известным принципам [1—4], продемонстрированным на рис. 1. В систему поступает массив документов. В общем случае под документом следует понимать некоторые данные, для которых известно только то, к какому типу информации они относятся (формализованный/неформализованный текстовый документ, графика, звуковая информация, формализованная информация в некотором внутреннем формате и т. д.). В ходе процесса загрузки документов в ИПС каждый из них обрабатывается независимо от других. Результатом обработки документа является некоторая информация о нем, помещаемая в базу данных. Соот-

Рис. 1. Классическая информационно-поисковая система

ветственно, при удалении документа из ИПС эта информация удаляется из базы данных.

После того как документы загружены в систему, она приобретает способность обрабатывать пользовательские запросы и генерировать ответы на основе накопленного содержимого базы данных. Фактически каждый запрос представляет собой документ некоторого специального типа, обрабатываемый модулем выполнения запросов. Более того, результат выполнения запроса — это также документ специального типа. В большинстве практических реализаций ответами являются сравнительно небольшие выборки документов, формируемые на основе функции релевантности, позволяющей оценить, насколько содержимое документа соотносится с запросом. Согласно распространенным подходам к информационному поиску, функция релевантности принимает только два аргумента: запрос и документ. Этот факт иллюстрирует еще один базовый принцип работы ИПС: при выполнении запросов все документы обрабатываются независимо друг от друга.

Наконец следует отметить, что и документы, и запросы обрабатываются на основе словарей системы, содержащих различные вспомогательные данные. Перечень и содержимое словарей зависят уже от специфики предметной области, в которой используют ИПС, а их содержимое можно рассматривать как совокупность обладающих определенной спецификой документов.

В подавляющем большинстве случаев ИПС используют как систему поддержки принятия решений (СППР), когда для обоснования того или иного решения необходим некоторый документ. Например, для обоснования того или иного утверждения в научной статье необходимо найти соответствующие документы в сборниках научных статей, книгах, сети Интернет и других источниках. Для оценки работы ИПС, как и для любых других систем управления, можно использовать понятие эффективности [3]. Можно считать, что эффективность ИПС определяют исходя из того, насколько быстрее пользователь находит требующийся ему документ с использованием ИПС, чем без ее использования.

Изложенные выше понятия и определения можно представить формально следующими обозначениями:

D_{ALL} — множество документов, которые могут быть загружены в ИПС;

$D \subset D_{ALL}$ — множество документов, которые реально загружены в ИПС в данный момент;

Q_{ALL} — множество запросов, которые могут быть обработаны ИПС;

$d \in D_{ALL}$ — конкретный документ;

$q \in Q_{ALL}$ — конкретный запрос;

W_{ALL} — множество всех возможных наполнений вспомогательных словарей системы;

$w \in W_{ALL}$ — текущее содержимое словарей системы;

$\rho(d, q)$ — функция релевантности;

$R_q \subset D$ — результат выполнения запроса q;

$Q \subset Q_{ALL}$ — множество запросов, которые возвращают непустую выборку;

$DDB(d, w)$ — функция возвращает информацию о документе d, которую необходимо поместить в базу данных ИПС;

$QDB(q, w)$ — функция возвращает информацию о запросе q, которую необходимо искать в базе данных ИПС.

На функцию релевантности $\rho(d, q)$ накладывают следующие ограничения.

1. Функция определена для всех возможных документов и запросов, т. е. $(d, q) \in D_{ALL} \times Q_{ALL}$.

2. Функция принимает вещественные значения из отрезка [0, 1], т. е. $\forall(d, q)$: $\rho(d, q) \in [0, 1]$. При этом равенство $\rho(d, q) = 1$ означает, что документ полностью соответствует запросу, а $\rho(d, q) = 0$ — что документ не соответствует запросу.

3. Для любого документа есть запрос, полностью релевантный данному документу, т. е. $\forall d$: $\exists q$: $\rho(d, q) = 1$.

4. Из рассмотренной архитектуры ИПС следует $\rho(d, q) = \rho DB(DDB(d, w), QDB(q, w))$. Здесь некоторая функция ρDB вычисляет релевантность не по исходным текстам запроса и документа, а на основе информации о запросе $QDB(q, w)$ и информации о документе $DDB(d, w)$, получаемым из базы данных.

Множество R_q может быть определено следующим образом:

$$R_q = \{d \in D, \rho(d, q) > \alpha\}.$$

Здесь α задает ограничение на значение функции релевантности и, как правило, фиксировано. На практике ИПС позволяют задавать в самом запросе некоторое число β, с помощью которого происходит перерасчет значения релевантности следующим образом:

$$\rho^*(d, q) = \rho(d, q)^{\lambda}, \text{ где } \lambda = \log_{\beta}((\alpha + 1)/2).$$

Очевидно, что $\rho^*(d, q) > \alpha$ тогда и только тогда, когда $\rho(d, q) > \beta$. Если при работе ИПС для формирования множества R_q использовать функцию $\rho^*(d, q)$ вместо $\rho(d, q)$, то в тексте запроса можно будет указывать различные значения β и тем самым неявно регулировать значение α. Наконец, множество Q может быть определено так:

$$Q = \{q \in Q_{ALL}, R_q \neq \varnothing\}.$$

Для оценки эффективности ИПС в рассматриваемом подходе используют понятие релевантности поиска и достаточно абстрактное понятие точности поиска. Это некоторая величина, показывающая вероятность того, содержит ли полученная выборка требующийся пользователю документ. Однако с точки зрения эффективности необходимо учитывать еще и размер получаемой выборки.

Оценка качества поиска

Формально требования к точности ИПС можно представить в виде множества векторов $EX = \{(q, d)\}$, каждый из которых задает поведение системы при выполнении конкретного запроса и состоит из следующих компонент:

$q \in Q_{ALL}$ — обрабатываемый запрос;

$d \in D_{ALL}$ — один из документов получаемой выборки.

На практике в ходе своей работы ИПС выдает результаты, отличные от множества EX, представимые в виде множества EX_{REAL}, которое состоит из аналогичных по своему строению кортежей. Точность поиска ИПС может быть определена исходя из множеств EX и EX_{REAL} следующим образом:

$$ACC = |EX \cap EX_{REAL}|/|EX \cup EX_{REAL}|.$$

Точность ИПС максимальна (равна 1), если множества EX и EX_{REAL} совпадают и, наоборот, точность минимальна (равна 0), если EX и EX_{REAL} не имеют общих элементов. Указанная формализация точности поиска является исключительно теоретической моделью по ряду причин. При практической реализации поисковых систем множество EX вводится не простым перечислением, а неявно, на основе рассмотренной выше функции релевантности, т. е.:

$$EX = \{(q, d) | q \in Q_{ALL}, d \in D_{ALL}, \rho(d, q) > \alpha\};$$

$$EX_{REAL} = \{(q, d) | q \in Q_{ALL}, d \in D_{ALL}, \rho_{REAL}(d, q) > \alpha\}.$$

В приведенных формулах $\rho(d, q)$ — абстрактная релевантность документа и запроса, которая может быть определена пользователем-экспертом, а $\rho_{REAL}(d, q)$ — реальная релевантность, которая вычисляется ИПС в ходе выполнения запроса.

Посредством множества EX_{REAL} может быть формально определен столь немаловажный параметр как средний размер выборки. Пусть

$$\text{SIZE}(q) = |\{(q, d) \in EX_{REAL}\}|.$$

Функция $\text{SIZE}(q)$ задает размер выборки, возвращаемой при обработке q. Соответственно, ее среднее значение по всем $q \in Q$ является средним значением выборки.

Общая эффективность использования ИПС зависит от двух параметров: точности поиска и размера выборки. Очевидно, чем выше точность поиска, тем эффективнее будет использоваться ИПС. В идеале точность поиска должна быть равна 1, при этом размер выборки будет определяться функцией релевантности $\rho(d, q)$. Если она подобрана правильно, то выборка должна содержать небольшое число документов, которое пользователь может просмотреть за обозримое время и выбрать необходимый.

В случае, когда точность поиска не равна 1, пользователь вручную просматривает полученную выборку и находит (или не находит) в ней необходимый документ. Если он присутствует в выборке, то эффективность использования ИПС зависит от размеров выборки: чем больше выборка, тем меньше эффективность. Когда же документ отсутствует в выборке, эффективность ИПС автоматически снижается на очень большое значение, но продолжает при этом зависеть от размеров выборки.

Обобщая изложенное выше, можно сделать следующие выводы.

- Рассмотренные показатели описывают взаимодействие пользователя и ИПС, типы информационных потоков между пользователем и ИПС, а также внутреннюю обобщенную архитектуру системы.

- Данная математическая формализация архитектуры ИПС малоприменима на практике, однако пригодна в качестве основы для дальнейших теоретических исследований и формализаций.
- Достоинством описанной модели является тот факт, что она включает все аспекты взаимодействия системы и пользователя, в том числе и оценку эффективности использования системы.

Рассмотренная математическая модель во многом повторяет классические принципы информационного поиска, изложенные в работе [4], но во главу ставит эффективность использования системы с точки зрения пользователя.

Критика модели использования ИПС

Перечисленные характеристики позволяют оценить ИПС только со стороны конечного пользователя, осуществляющего поиск. Для получения более разносторонней оценки эффективности системы необходимо учитывать сложность ее создания, внедрения и сопровождения. Например, вполне допустимы ситуации, когда возможно создание ИПС с заданными "пользовательскими" параметрами, но ее сопровождение будет требовать неоправданно высоких затрат, что сведет суммарную эффективность использования системы к нулю.

Таким образом, необходимо отметить ряд недостатков, присущих "классическим" принципам построения и оценки систем информационного поиска. К их числу относятся перечисленные ниже.

- Подход не включает в себя ни качественную, ни количественную оценки информационных потоков между ИПС и пользователем. Он не рассматривает возможную типизацию обрабатываемых документов, запросов, наполнения словарей, а также видов и способов задания функции релевантности. Также не принимается во внимание возможность ручной обработки или корректировки документов на этапе загрузки в систему.
- Считается, что ИПС полностью автоматическая, т. е. при загрузке документов и выполнении запросов она не запрашивает у пользователя дополнительной информации. Однако такой информацией может быть уточнение содержимого документов для устранения возможных неоднозначностей, уточнение запросов и т. п. В некоторых случаях использование автоматизированных ИПС, требующих участия пользователя на этапе ввода информации, окажется более эффективным.
- Не делается оценка необходимых для работы ИПС вычислительных ресурсов — процессорного времени и объема памяти.
- Нет никаких способов оценки реальной нагрузки на СУБД, используемую в ИПС для хранения поисковой информации. В том числе нет методики оценки возможности или невозможности реализации ИПС, удовлетворяющей указанным характеристикам, а в случае возможности нет методики построения ИПС с заданными характеристиками.

Перечисленные недостатки свидетельствуют о том, что классические модели информационного поиска малоприменимы при разработке целевых ИПС. Необходимы более "глубокие" методологии, затрагивающие разработку, внедрение и использование ИПС.

Модель синтеза целевых ИПС

На рис. 2 приведена предлагаемая схема синтеза целевой ИПС с заданными параметрами. В блоках схемы указаны группы взаимосвязанных параметров, а стрелки демонстрируют зависимость одних групп параметров от других. Зависимые параметры могут быть определены только после того, как полностью определены все остальные группы параметров, от ко-

Рис. 2. Характеристики информационно-поисковой системы

торых они зависят. Если некоторые параметры оказываются недостижимыми на практике, то необходим откат и пересмотр их значений. Таким образом, отправной точкой при построении ИПС является описание предметной области, в которой будет использоваться система, а результирующей точкой — структура базы данных для хранения поисковых характеристик.

Наиболее важными (с точки зрения операции поиска) характеристиками документов являются объем, степень точности информации и уровень ее формализованности (формального описания, структурированности). Объем представляет вполне очевидную, широко используемую характеристику и может быть задан как функция $V(d)$, где d — загружаемый в систему документ. Другие параметры (степень нечеткости и неструктурированности) тесно взаимосвязаны между собой и требуют, во-первых, пояснения, во-вторых, строгого математического описания.

Под нечеткостью понимается следующее. Пусть документ описывается некоторым множеством своих параметров, и, соответственно, в запросах указываются ограничения на значения этих параметров. Между запросами может быть введено отношение "частности": запрос q_1 является частным случаем другого запроса q_2, если $\forall d$: $d \in R_{q1} \Rightarrow d \in R_{q2}$. Тогда может быть найдено множество QE наиболее узких запросов, не имеющих частных случаев. Каждый запрос $q \in QE$ возвращает сравнительно небольшую выборку документов. Однако сам по себе q может быть при этом сложным и состоять из множества условий, связанных логическими предикатами.

Если есть два запроса q_1 и q_2, возвращающие непустые результаты R_{q1} и R_{q2}, то очевидным образом могут быть построены запросы "q_1 и q_2", "q_1 или q_2" и "$q_1 - q_2$". Результаты выполнения построенных запросов получают вычислением соответствующих теоретико-множественных операций над R_{q1} и R_{q2}.

Утверждение 1. Множество QE состоит из непересекающихся запросов (их результаты не содержат общих элементов).

Доказательство. Предположим противное: запросы q_1 и q_2 входят в QE и содержат общие элементы. Тогда можно построить "$q_1 - q_2$" и "q_1 и q_2", которые будут являться сужением запроса q_1. Это значит, что q_1 имеет сужения и не входит в QE. Возникло противоречие, откуда следует, что все запросы из QE не пересекаются.

Определение 1. Документ d нечеткий, если он релевантен одновременно нескольким запросам из QE.

Смысл данного определения иллюстрируется следующим свойством.

Утверждение 2. Если документ является нечетким, то невозможно построить запрос, отбирающий только этот документ из множества всех возможных документов D_{ALL}.

Доказательство. Предположим противное: документ d нечеткий и есть запрос q, возвращающий только d. Тогда, поскольку результат запроса q содержит единственный элемент, то q не имеет более узкого запроса. Следовательно, запрос q содержится в QE. Далее, документ d релевантен запросу q и не может быть релевантным никакому другому запросу из QE, так как все запросы в QE не пересекаются с q. Таким образом, по определению 1 документ d не является нечетким. Получено противоречие, которое означает справедливость доказываемого утверждения.

Изложенные выше положения можно формально представить следующим образом:

1) запрос q_1 является сужением запроса q_2, если $R_{q1} \subset \subset R_{q2}$;

2) $QE = \{q_1 ... q_N | q_i$ не имеет более узкого запроса$\}$;

3) $\text{Fuzzy}(d) = |\{q | q \in QE$ и $\rho(d, q) > \alpha\}|$.

Функция $\text{Fuzzy}(d)$ формально определяет степень нечеткости: $\text{Fuzzy}(d) = 1$ для четкого документа и $\text{Fuzzy}(d) > 1$ для нечеткого. На практике каждый документ имеет уникальный идентификатор, и тогда запросы из QE представляют собой ограничения вида "документ с идентификатором таким-то". Эти запросы не имеют сужений по следующим причинам. Во-первых, если запрос "документ с идентификатором таким-то" возвращает два и более документа, то идентификатор не является уникальным. Во-вторых, если каждый запрос вида "документ с идентификатором таким-то" дает выборку из одного и только одного документа, то невозможно построить сужение этого запроса, возвращающее часть этой выборки.

Неформализованность документа можно рассматривать как *нечеткость* содержащихся в нем сведений, которую можно устранить путем редактирования и внесения необходимых уточнений без смены формата документа. Например, в текстовом документе формулировка "новая машина" является нечеткой, но вместо "новая машина" можно написать "машина такого-то года выпуска", в результате чего нечеткость будет снижена, при этом сам документ останется текстовым. Таким образом, данный документ следует считать неформализованным, поскольку содержащиеся в нем сведения можно уточнить. Другой пример: в жалобе больного указана "сильная боль". Эту формулировку как-либо уточнить и выразить в числовом виде нельзя, поэтому документ следует считать нечетким.

Нечеткость и неформализованность тесно взаимосвязаны и проявляются внешне одинаково, однако имеют различную "природу": нечеткость основана на невозможности формально отобразить некоторую информацию, а неформализованность — на приближенном указании информации, которая может быть задана более строго и формально.

Таким образом, любой документ характеризуется тремя параметрами: размером, степенью нечеткости и степенью неформализованности.

С формальной точки зрения как нечеткость, так и неформализованность, характеризуются неравенством $\text{Fuzzy}(d) > 1$. Можно считать, что

$$\text{Fuzzy}(d) = \text{Fuzzy}_F(d) + \text{Fuzzy}_S(d),$$

где $\text{Fuzzy}_F(d)$ — степень неустраняемой нечеткости, а $\text{Fuzzy}_S(d)$ — степень неформализованности (нечет-

кости, которую можно устранить путем редактирования документа).

Соответственно для оценки всех однотипных документов, загружаемых в ИПС, можно использовать средние значения размера документа, степени его нечеткости и степени неформализованности.

Словари целевых ИПС можно рассматривать как специфические документы и оценивать их с помощью тех же параметров: объем, нечеткость и неформализованность. В то же время на практике словари представляют собой предельно четкую и формализированную информацию.

Наконец, запросы являются специфическими документами, но характеризуются только объемом и степенью нечеткости, которая может быть определена как

$$\text{Fuzzy}(q) = |\{q_1 \in QE | q_1 \text{ является частным случаем } q\}|.$$

Функция Fuzzy(q) показывает, сколькими различными способами может быть трактован один и тот же запрос. Это определение нечеткости запроса может быть дополнительно обосновано тем обстоятельством, что если запрос q накладывает условия на содержимое документа, то он не может возвращать только часть выборки некоторого запроса $q_1 \in QE$. Если же такой запрос q существует, то множество QE построено неверно. Таким образом, любой пользовательский запрос можно рассматривать как описание некоторого подмножества запросов из QE. Однако пользователь практически всегда способен уточнить запрос, т. е. уменьшить это подмножество. Отсюда можно заключить, что запрос не нечеткая, а именно неформализованная информация. Например, если пользователь ищет "новая машина", то он может конкретизировать запрос и указать "машина такого-то года выпуска" или даже "машина с такой-то датой выпуска". Если же запрос нельзя конкретизировать (например, "сильная боль"), то это скорее проблема предметной области данных, и документы также не содержат более детальных сведений.

По аналогии с документами, при задании требований к ИПС необходимо отталкиваться от среднего размера словаря, среднего размера запроса, а также средних степеней их нечеткости и неформализованности. Таким образом, можно выделить следующий набор параметров ИПС:

V_D — средний размер документа;

N_D — число документов;

F_D — средняя степень нечеткости документа;

S_D — средняя степень неформализованности документа;

SR_D — трудоемкость работы пользователя при формализации документа;

V_Q — средний размер запроса;

F_Q — средняя степень нечеткости запроса;

FR_Q — трудоемкость работы пользователя при уточнении запроса;

V_W — размер вспомогательных словарей системы.

После того как заданы перечисленные характеристики документов, запросов и словарей, могут быть определены требования к функциям релевантности. Сюда следует отнести, во-первых, классические требования к точности и полноте поиска, непосредственно связанные с функциями релевантности. Во-вторых, функции релевантности работают на основе некоторой индексной информации о документах, и, следовательно, необходимо заранее оценивать и ограничивать суммарные объемы такой информации. Как правило, высокие точность и полнота поиска требуют больших объемов индексной информации и большего времени выполнения запросов. В-третьих, необходимо рассматривать варианты ИПС, когда пользователи вручную индексируют поступающую информацию, и необходимо оценивать трудоемкость их работы.

Наконец, скорость работы ИПС при выполнении различных операций может быть оценена следующими параметрами:

SP_D — средняя скорость загрузки документов;

SP_Q — средняя скорость выполнения запросов;

SP_W — средняя скорость обновления словаря системы.

Только после того как определены все перечисленные выше параметры (входные данные, запросы, словари, скорость работы, точность поиска и т. д.), становится доступным определение структуры базы данных и используемой СУБД. Не исключена ситуация, когда требующиеся характеристики ИПС не могут быть обеспечены с технической точки зрения. Например, может оказаться недостижимой заданная скорость загрузки документов или их поиска.

Заключение

В статье рассмотрены основные параметры ИПС, позволяющие оценить ее работу не только с позиции конечного пользователя, как это делают классические подходы, но также с позиции построения и сопровождения, что не менее важно. Возможны ситуации, когда создание высокоточной поисковой системы потребует неоправданно высоких затрат на ее сопровождение.

Перечисленные характеристики позволяют описывать достаточно широкий класс ИПС, сравнивать эти системы между собой и оценивать их эффективность. Взаимосвязь представленных характеристик основывается на внутренней архитектуре ИПС, а ее обсуждение выходит за рамки настоящей статьи.

Список литературы

1. **Лахути Д. Г.** Автоматический анализ естественно-языковых текстов // НТИ. Серия 2. 2003. № 11. С. 18—25.

2. **Насыпный В. В., Насыпная Г. А.** Зашифрованные поисковые системы // Открытые системы. 2004. № 11. С. 49—53.

3. **Селезнёв К. Е.** Системы поиска // Открытые системы. 2005. № 10. С. 78—79.

4. **Сэлтон Г.** Автоматическая обработка, хранение и поиск информации. Пер. с англ. / под ред. А. И. Китова. М.: Советское радио, 1973. 560 с.

УДК 004.82

К. И. Костенко, канд. физ.-мат. наук, зав. каф., e-mail: kostenko@kubsu.ru,
А. П. Лебедева, аспирант, e-mail: lebedeva@fpm.kubsu.ru,
Кубанский государственный университет, г. Краснодар

О формализованных описаниях пространств знаний для интеллектуальных программных систем

Уточняются элементы универсального языка описания систем знаний в произвольных предметных областях (пространств знаний). Такой язык основан на форматах, близких к алгебраическим системам. Модели пространств знаний составляют классы, группируемые в разделы данных, морфизмов, предикатов и процессов. Модели допускают разный уровень полноты и абстрактности, а также возможности трансформации и расширения.

Ключевые слова: *интеллектуальная система, пространство знаний, алгебраическая модель, представление знаний, онтология, формальный язык*

Введение

Создание эффективных программных систем автоматизации процессов в произвольных областях деятельности в значительной степени зависит от возможностей конструирования формализованных логико-семантических моделей этих областей [1]. Указанными моделями представляются описания конкретных многообразий знаний, а основанные на них программные системы оказываются интеллектуальными. Примеры таких моделей связаны с задачами интеллектуального анализа тематических текстовых и графических данных в науке, образовании, медицине, а также с задачами управления сложными социальными, экономическими и технологическими системами. Стремление к созданию эффективных универсальных механизмов построения интеллектуальных программных систем способствует появлению потребности в специальных инструментах формального описания (далее используется термин "формализм"). Многообразие предложенных к настоящему времени формализмов характеризует значительное разнообразие используемых в них форматов и атрибутов. Построение универсального формализма знаний, дедуктивно порождающего существующие и возможные формализмы, признается важным для развития соответствующих формальных теорий, но слишком общим для возможности практического использования в качестве основы унифицированной технологии создания интеллектуальных систем. Большое многообразие разработанных формализмов, востребованность их совместного использования, возрастание уровня и сложности задач управления знаниями, а также появление продуктивных абстрактных моделей составляют объективные предпосылки для поиска универсального и практически применимого логико-математического формализма знаний. Универсальность и полнота указанного формализма достигается путем предварительного уточнения такой системы абстрактных инвариантов, которые могут быть адаптированы и развиты в информационные и функциональные структуры конкретных интеллектуальных систем. Другое отличие универсального логико-математического формализма от многочисленных эмпирических подходов к моделированию интеллектуальных систем состоит в возможности построения на его основе абстрактных и прикладных моделей таких систем с помощью специальных преобразований [2]. Для этого в число инструментов программной разработки интеллектуальных систем включают специальные преобразования элементов абстрактных моделей, основанных на универсальном формализме. Данные операции реализуют процесс разработки прикладных интеллектуальных систем, обеспечивающий наполнение моделей прикладным содержанием и сохранение существенных свойств абстрактных прототипов.

В настоящей работе определены элементы унифицированной структуры описаний интеллектуальных

систем, основанные на формализме пространства знаний [2, 3]. В данной структуре использованы общие алгоритмические и алгебраические инварианты, позволяющие создавать описания как известных, так и новых моделей знаний для различных этапов жизненных циклов интеллектуальных систем [4]. Такие модели представляют собой формальные системы, компонентами которых являются вычислимые классы данных, морфизмов, предикатов. Одним из способов организации управления знаниями при решении задач предметной области является включение в модели дополнительных описаний процессов и жизненных циклов таких пространств. Для представления процессов удобно использовать классы специальной структуры. В формате описаний классов процессов представлены универсальные характеристики процессов. К ним относятся: время реализации процесса, начальные данные, условия активации, продолжения и завершения, схема формирования результата процесса.

Общий формат моделей имеет вид $\Sigma = (T, F, P, E)$, где T, F, P и E — это классы данных, морфизмов, предикатов и процессов соответственно, составленные из подклассов, структурированных отношениями вложения ($\subseteq$), агрегирования ($\triangleleft$), эквивалентности ($\equiv$) и гоморфного расширения ($\ll$).

Рассматриваемая в работе система конструктов основана на общих форматах составления теоретико-множественных, логических и алгебраических выражений, схемах их интерпретации и обработки. Каждый класс формальной модели обладает собственным уникальным именем и представляется набором (*Ns*; *Fs*; *Ps*; *As*). Здесь *Ns* — область описаний имен, *Fs* — область описаний форматов, *Ps* — область описаний свойств; *As* — область описаний алгоритмов, поставленных в соответствие определяемому классу. Интерпретация символов переменных в выражениях локализована границами определений классов. Определения классов моделей с помощью четырехкомпонентных наборов сходны со схемами теории содержания имен, основанной на идеях Г. Фреге (основные конструкты: имя, концепт, денотат), а также концепции квадрата Д. Поспелова (синтаксис, семантика, прагматика, денотат) [5].

Основы формализованных описаний пространств знаний

Конструкции, применяемые при описании пространств знаний, определяет система специальных лексических, синтаксических и семантических правил. Они обеспечивают возможность формализации необходимого уровня общности, полноты и точности для систем знаний в конкретных областях.

Система символов, используемых в описаниях

Алфавит символьных описаний классов включает: строчные и прописные буквы латинского и русского алфавитов; цифры от 0 до 9; символ '_'; набор специальных символов (варианты использования приведены в таблице); прочие символы, применяемые в комментариях.

Основные правила употребления специальных символов

Символы	Варианты использования
.	Конец определения класса
	Разделение целой и дробной частей вещественного числа
	Разделение элементов в составных именах классов
,	Разделение нескольких форматов в области *Fs*
	Разделение свойств определяемого класса в списке *Ps*
	Разделение алгоритмов в области *As*
...	Указание диапазона
: и |	Используют в логических формулах
;	Разделение областей в определениях классов
-, +, *, /	Арифметические знаки
=, < и >	Знаки соответствующих отношений
{ , }	Ограничение комментариев в описаниях
	Используют в определениях множеств

Словарь ключевых слов

Ключевыми словами являются идентификаторы, наделенные уникальным смыслом и составляющие отдельный словарь. Прочие идентификаторы называют именами. Они служат для обозначения классов, переменных, функций, предикатов и процессов. В качестве ключевых слов зарезервированы следующие символьные последовательности:

- *begin*, *end* — обозначают начало и конец определения класса;
- *section*, *subsection* — ключевые слова определения раздела и подраздела;
- *DT*, *DF*, *DP*, *DE* — зарезервированные имена разделов классов данных, морфизмов, предикатов и процессов в пространствах знаний соответственно;
- *Basic*, *Universal*, *Special* — зарезервированные имена разделов общеупотребительных, универсальных и специальных классов разных типов соответственно.

Синтаксис описаний пространств знаний

Для описания синтаксиса конструкций в определениях пространств знаний будем использовать следующие средства расширенной нотации Бэкуса-Наура (*EBNF*) [5]:

- нетерминалы, значениями которых являются цепочки основных символов языка, изображаются словами, заключенными в угловые скобки (<...>);

• терминалы (в частности, ключевые слова), которые записываются в кавычках, например, "*begin*";

• вертикальная черта (|), используемая для определения альтернатив;

• круглые скобки, применяемые для группирования символов;

• квадратные скобки, используемые для определения возможного вхождения символа или группы символов;

• фигурные скобки, используемые для обозначения возможного повторения символа или группы символов;

• знак равенства, который разделяет определяемую конструкцию и составляющие ее ранее определенные базовые конструкции;

• символ "точка", который применяется для обозначения конца правила;

• комментарии, которые размещаются между символами "звездочка" (*).

Семантика синтаксических правил представляется диаграммами в виде нагруженных деревьев с четырьмя специальными типами вершин:

♦ Локальные вершины-нетерминалы, изображаемые прямоугольниками белого цвета с закругленными углами.

♦ Глобальные вершины-нетерминалы, изображаемые прямоугольниками серого цвета с закругленными углами.

♦ Вершины-терминалы, изображаемые параллелограммами серого цвета и представляющие фиксированные последовательности символов.

♦ Вершины-примеры, изображаемые параллелограммами белого цвета.

Для связывания вершин используют следующие разметки ребер.

○ *RequiredLink*. Обязательная связь, которая считается задаваемой по умолчанию.

○ *OptionalLink*. Соответствует необязательной связи.

○ *Alternative*. Относится к зависимости, соединяющей нетерминал с группой вершин. При обходе диаграмм переход осуществляется ровно в одну вершину.

○ *HasExample*. Отражает иерархическое отношение нетерминала и примера.

В диаграммах допускаются петли, переход по которым может осуществляться произвольное конечное число раз.

Унифицированная структура описания пространства знаний

Общими структурными элементами описания пространства знаний (*KS*, Knowledge Space) являются разделы (*sections*). Структура раздела приведена на рис. 1.

Структура описания всякого раздела задается выражением:

<Раздел> = "*section*" <Имя> "*begin*"
{<Определение класса>|<Подраздел>} "*section*" <Имя> "*end*".

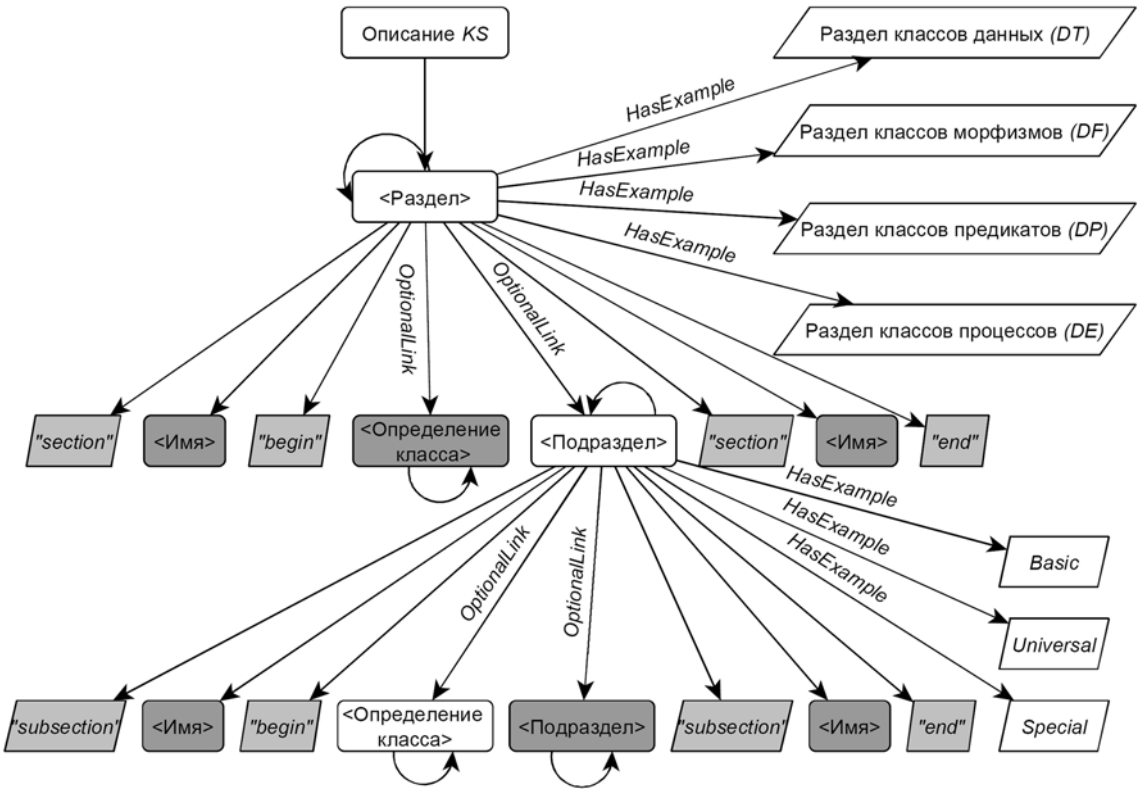

Рис. 1. Структура раздела пространства знаний

Каждый раздел составляют определения классов. Раздел может включать несколько вложенных подразделов (*subsection*), группирующих определения классов. Описания подразделов соответствуют правилу

<Подраздел> = "*subsection*" <Имя> "*begin*"
{<Определение класса>|<Подраздел>} "*subsection*" <Имя> "*end*".

Стандартная система типов подразделов всякого раздела включает подразделы базовых, универсальных и специальных классов, задаваемые специальными именами *Basic*, *Universal* и *Special*. Базовые классы связаны с общими инвариантами логико-математического языка. Поэтому основными классами раздела *Basic* являются предопределенные типы данных, общеупотребительные классы морфизмов и предикатов на указанных типах данных. Универсальные классы являются развитием системы классов абстрактного пространства знаний [2]. Специальные классы составляют системы данных, морфизмов и предикатов, отражающих специфические особенности модели или предметной области.

Общая структура описания пространства знаний приведена на рис. 2.

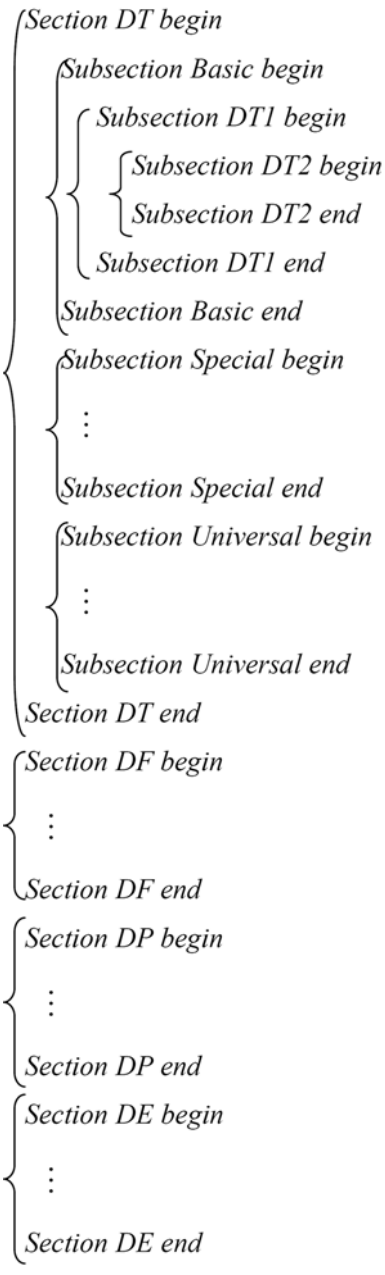

Рис. 2. Общая структура описания пространства знаний

Определение класса пространства знаний

Определения отдельных классов представляются нетерминалом <Определение класса>, структура которого приведена на рис. 3.

В определениях классов допускаются пустые области. Выражения $G(A)$ и $R(A)$, где A — имя класса, применяют для обозначения алгоритмов перечисления класса A и распознавания его элементов. Записи $NG(A)$ и $NR(A)$ обозначают отсутствие указанных алгоритмов. Последнее свойство возможно для абстрактных моделей пространств знаний, в которых не всегда обеспечивается существование вычисляющих алгоритмов.

Рассмотрим пример определения класса для множества конфигураций абстрактного пространства знаний [2]. Данный класс составляет вычислимое и разрешимое множество конфигураций M. Его элементы являются образами отдельных знаний. Существуют абстрактные алгоритмы $G(M)$ и $R(M)$, перечисляющие M и распознающие его элементы.

Множество конфигураций содержит специальную пустую конфигурацию Λ. Определение M имеет вид

Dt1. {Класс конфигураций}

$$\underbrace{(M;}_{Ns}\ \underbrace{\{z^i | i \in N\};}_{Fs}\ \underbrace{(z^0 = \Lambda);}_{Ps}\ \underbrace{G(M), R(M))}_{As},$$

где N — класс целых неотрицательных чисел, а запись $\{z^i | i \in N\}$ задает основной формат представления элементов M, в котором z^0 — пустая конфигурация.

Правило для нетерминала <Определение класса> имеет вид

<Определение класса> = <Идентификатор класса>
". {" <Описание класса> "}" "(" <*Ns*> [";" <*Fs*>] [";" <*Ps*>] [";" <*As*>]").".

Поясним конструкции структуры, изображенной на рис. 3.

Блок <Идентификатор класса> обозначает последовательности, состоящие из букв, цифр и символов подчеркивания, начинающиеся с буквы. Нетерминал <Идентификатор класса> определяют следующие выражения:

<Цифра> = "0" | "1" | ... | "9".
<Знак> = "+" | "–".

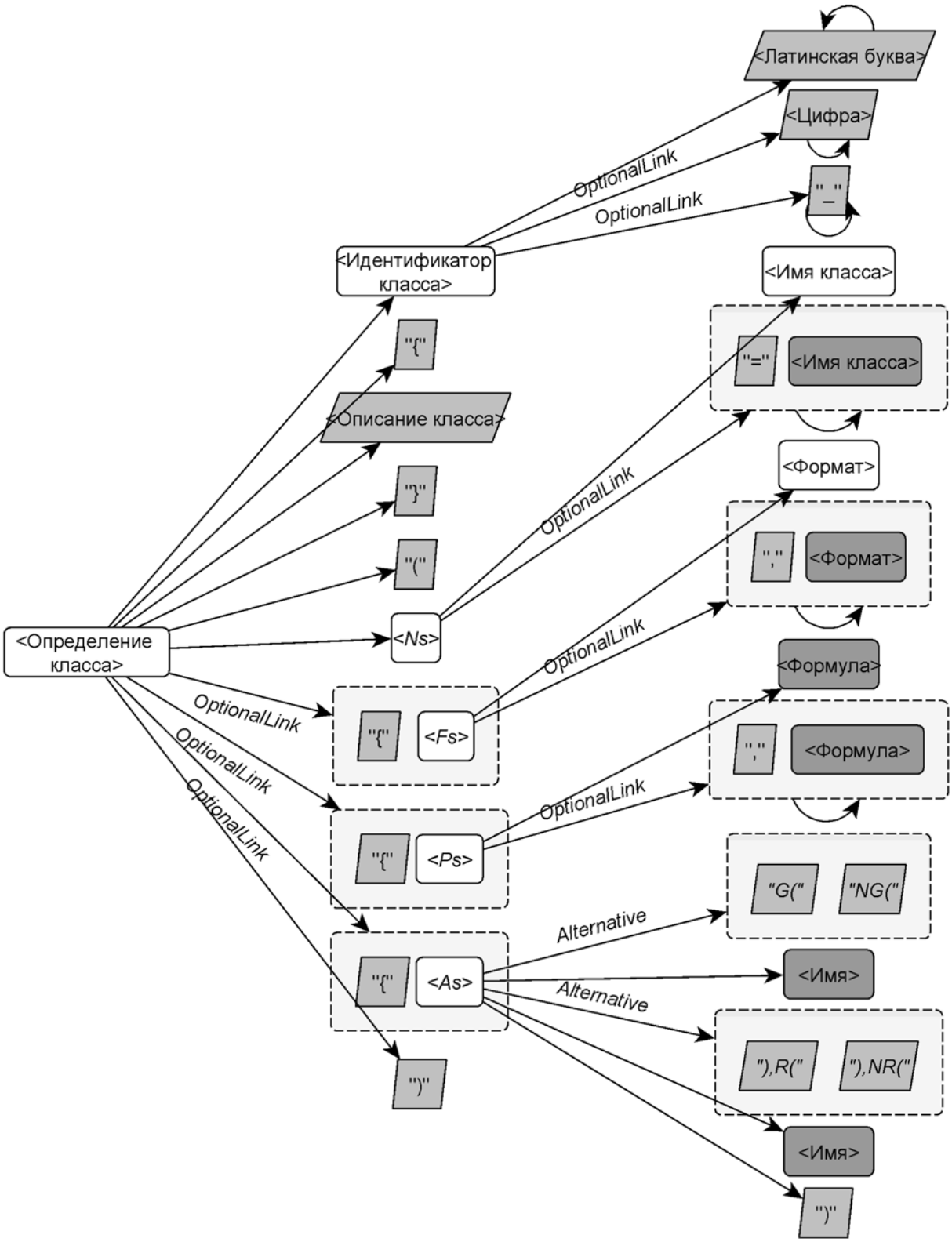

Рис. 3. Структура нетерминала <Определение класса>

<Число> = [<Знак>] <Цифра> {<Цифра>}.
<Латинская буква> = "A" | "B" | ... | "Z" | "a" | "b" | ... | "z".
<Греческая буква> = "A" | "B" | ... | "Ω" | "α" | "β" | ... | "ω".
<Буква> = <Латинская буква> | <Греческая буква>.
<Слово> = <Буква> {<Буква>}.
<Идентификатор класса> = <Латинская буква>
{<Латинская буква> | <Цифра> | "_"}.

Элемент <Описание класса> обозначает произвольную последовательность символов.

Области определений классов

Область имен определений классов

Именами классов являются символьные последовательности. Классу могут соответствовать несколько имен, разделяемых знаком "="

<*Ns*> = <Имя класса> {"=" <Имя класса>}.

Формальная структура нетерминала <Имя класса> приведена на рис. 4. Здесь <Имя класса>=(<Имя>{ "×" <Имя>})

|<Имя с параметром>|<Имя одноэлементного класса>.
<Имя> = <Слово> [<Слово> | "*" | <Число>] [<Слово> | <Число>]·

Основными вариантами задания имен являются: имя, имя с параметром и имя одноэлементного класса.

Конструкция <Имя с параметром> определяет семейство близких по структуре и свойствам классов:

<Имя с параметром> = <Имя> "(" <Имя>
{ "," <Имя> } ")".

Рассмотрим пример определения семейства классов вершин полных структурных представлений (ПСП) отдельных конфигураций из M. Если $z \in M$, то множество вершин ПСП этой конфигурации обозначается, как $D(z)$, и составляет бинарное дерево. Дерево $D(z)$ определяется с помощью вычислимого отображения $\varepsilon: M \to M \times M$. Такое отображение ставит в соответствие конфигурации z ее разложение (z_0, z_1). Дерево формируется, если продолжать разложение z до пустых конфигураций. Вершинами $D(z)$ являются двоичные наборы, определяющие пути из корня λ в такие

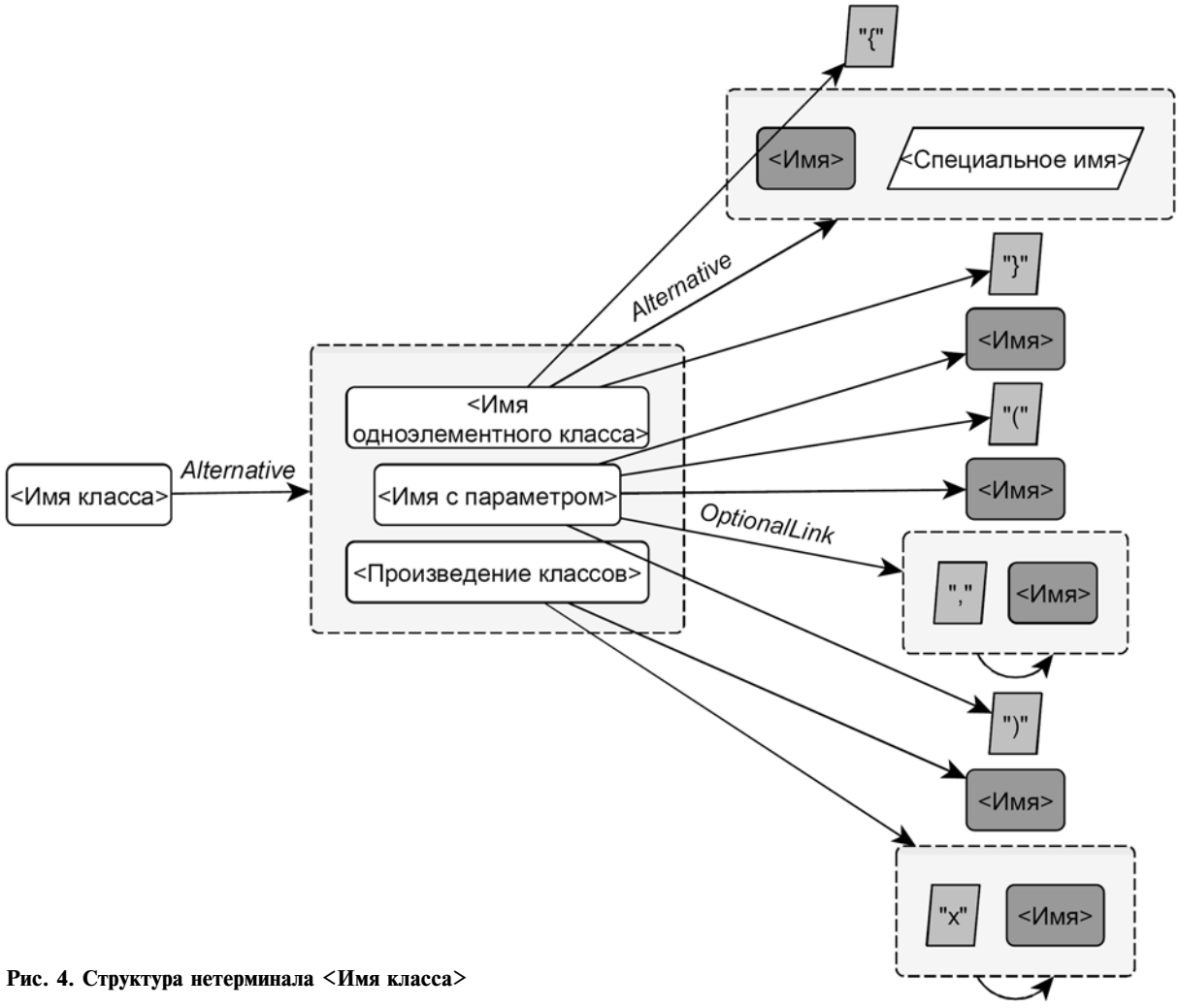

Рис. 4. Структура нетерминала <Имя класса>

вершины. Класс вершин деревьев обозначается как I. Висячие вершины $D(z)$ соответствуют конфигурациям, для которых $\varepsilon(z) = (\Lambda, \Lambda)$. Если $z \in M$ и $\beta \in D(z)$ — внутренняя вершина в $D(z)$, то $(z)_\beta$ — это конфигурация, представленная деревом с корнем β, а $\varepsilon((z)_\beta)$ — разложение $(z)_\beta$ на пару конфигураций, представляемых деревьями с корнями $\beta 0$ и $\beta 1$. Определение параметрического семейства классов вершин ПСП конфигураций имеет следующий вид.

Dt2. {Семейство классов вершин ПСП конфигураций}

$$(\{D(z)|z \in M\};\ \forall z \in M(D(z) \subseteq I);$$
$$\forall z \in M(D(z) = \{\alpha|\alpha = \lambda \vee \alpha =$$
$$= \beta\sigma\ \&\ \beta \in I\ \&\ \sigma \in \{0, 1\}\ \&\ \varepsilon((z)_\beta) \neq (\Lambda, \Lambda)\});$$
$$G(D(z)),\ R(D(z))).$$

Здесь Dt2 — уникальное имя описания семейства классов. Запись {Семейство классов вершин ПСП конфигураций} является комментарием. Область имен *Ns* составляет выражение $D(z)$, параметром которого является конфигурация. В области форматов указывается, что классы вершин являются частью I. Область свойств *Fs* содержит описание семейств двоичных наборов, составляющих классы $D(z)$, $z \in M$. Всякий класс $D(z)$ составляют пустой набор λ и наборы, соответствующие вершинам — потомкам внутренних вершин. Область *As* представлена именами алгоритмов $G(D(z))$ — перечисления класса $D(z)$ и распознавания элементов класса $R(D(z))$.

Конструкция <Имя одноэлементного класса> определяет класс, состоящий из единственного элемента.

<Имя одноэлементного класса> = "{" <Имя> "}" |
"{" <Специальное имя> "}".
<Специальное имя> = "$^{-}$" | "○" | "⊕" | "$^{-1}$".

Конструкция <Специальное имя> применяется для обозначения стандартных имен, используемых в предметной области.

Область форматов определений классов

Область форматов содержит экстенсивные описания компонентов модели, определяющие их структуру и состав. Она может содержать несколько описаний, разделенных запятыми. Структура *Fs* представлена на рис. 5.

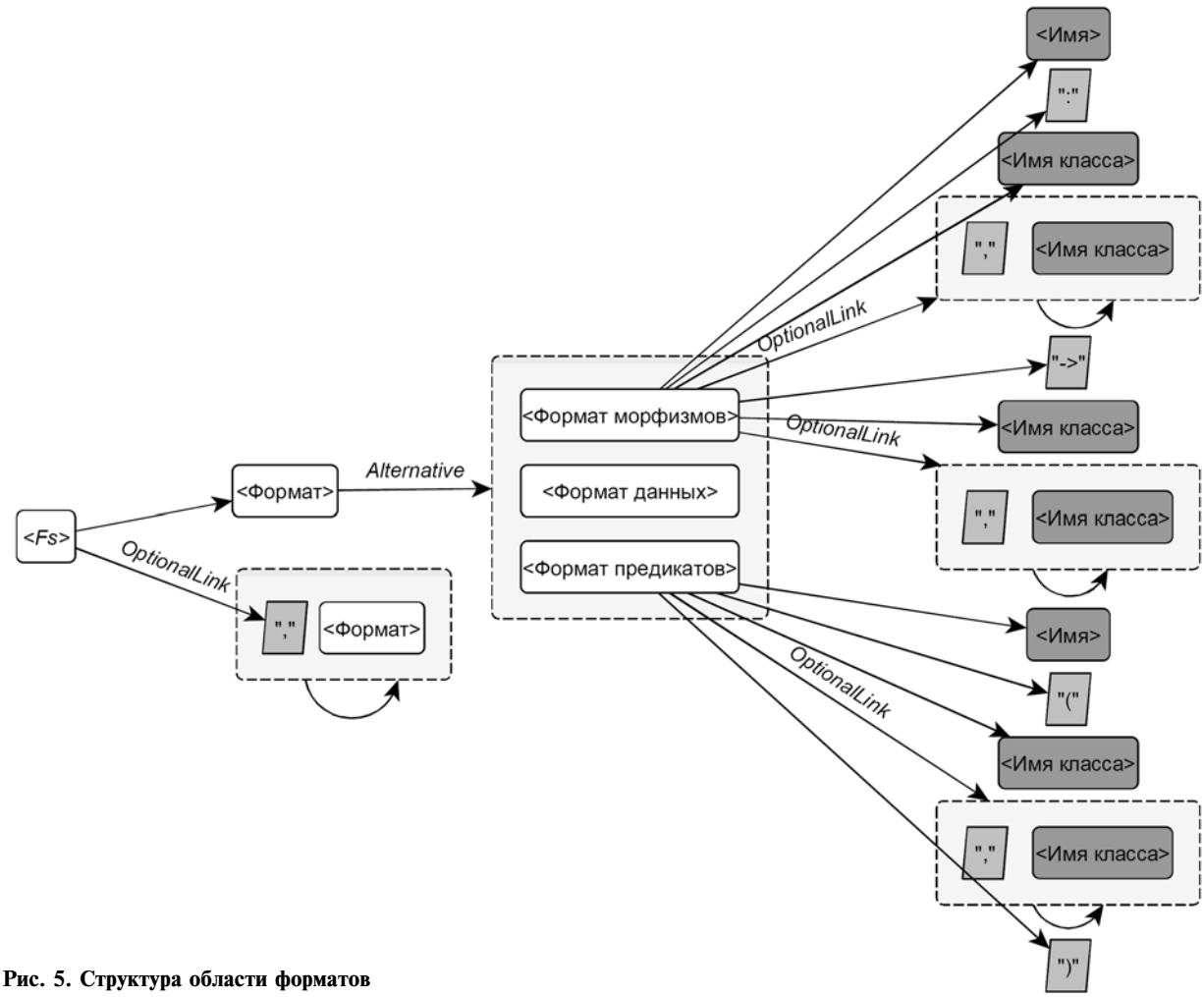

Рис. 5. Структура области форматов

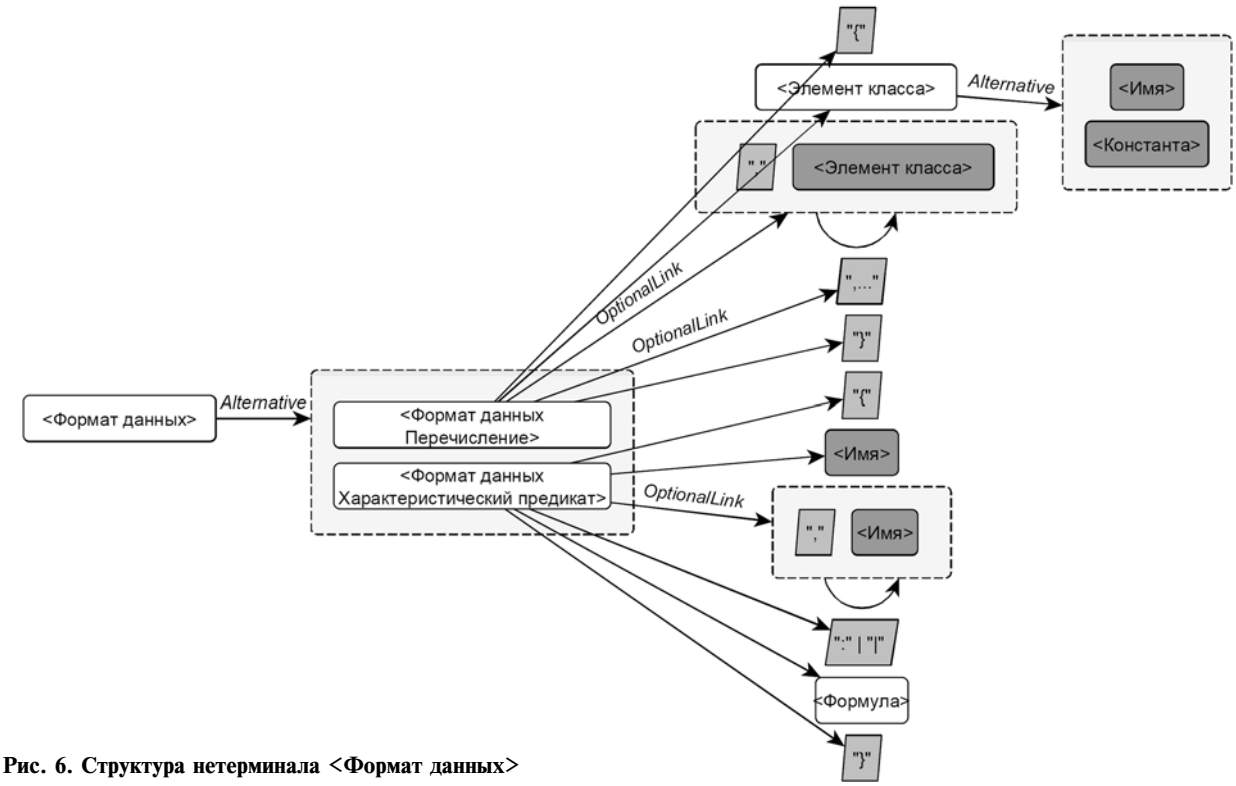

Рис. 6. Структура нетерминала <Формат данных>

На рис. 5 <*Fs*> = <Формат> { "," <Формат>).

<Формат> = <Формат данных> | <Формат морфизмов> | <Формат предикатов>.

Структура нетерминала <Формат данных> приведена на рис. 6.

В качестве основных форматов описаний классов данных предусмотрены схемы, основанные на перечислении элементов и характеристических свойствах элементов классов:

<Формат данных>=<Формат данных_ Перечисление>
| <Формат данных_Характеристический предикат>.
<Формат данных_Перечисление> =
"{" <Элемент класса> { ("," <Элемент класса>) | ", ..." } "}".

В последнем случае для определяемого класса *A* должны существовать алгоритмы генерации и распознавания элементов (*G*(*A*) и *R*(*A*)). Для обозначения элементов классов допускается использование конструкций <Имя>, т. е. имя переменной, значения которой генерируются алгоритмом *G*(*A*), и <Константа> — элемент предопределенного класса данных, а также ранее определенное имя, включая имя одноэлементного класса.

Кроме того, <Элемент класса> = <Имя> | <Константа>.

Определение формата, основанного на характеристических свойствах множеств, имеет следующий вид:

<Формат данных_Характеристический предикат> =
"{" ["("] <Имя> { "," < Имя > } [")"] (":" | "|")
<Формула> "}".

Приведем определение класса, использующее данный формат.

Dt3. {Класс неэлементарных конфигураций}

$$(M_1; \{z_i | z \in M \;\&\; \varepsilon(z_i) = (z', z'') \;\&\; (z' \neq \Lambda \vee z'' \neq \Lambda)\};$$
$$M_1 \subseteq M;\; G(M_1),\; R(M_1)).$$

Данный класс составляют конфигурации, разложения которых содержат непустую конфигурацию ($\varepsilon(z_i) = (z', z'') \;\&\; (z' \neq \Lambda \vee z'' \neq \Lambda)$).

Уточним правило составления области форматов классов морфизмов и предикатов.

< Формат морфизмов > = <Имя> ":"
<Имя класса> { "×" <Имя класса> } "→"
<Имя класса> { "×" <Имя класса> }.
< Формат предикатов > = <Имя> "(" <Имя класса>
{ "," <Имя класса> } ")".

Именование морфизмов отдельных классов возможно также с помощью составных имен, определяемых соотношением

<Составное имя> = <Имя класса >
{ "." <Имя подкласса> } "." <Имя морфизма>.

Область свойств определений классов

В область *Ps* помещают интенсивные описания свойств классов, задаваемые логическими формулами. Отдельные формулы разделяются знаком ", ", соответствующим логическому "И". В конструкции <Формула>, приведенной на рис. 7, использованы дополнитель-

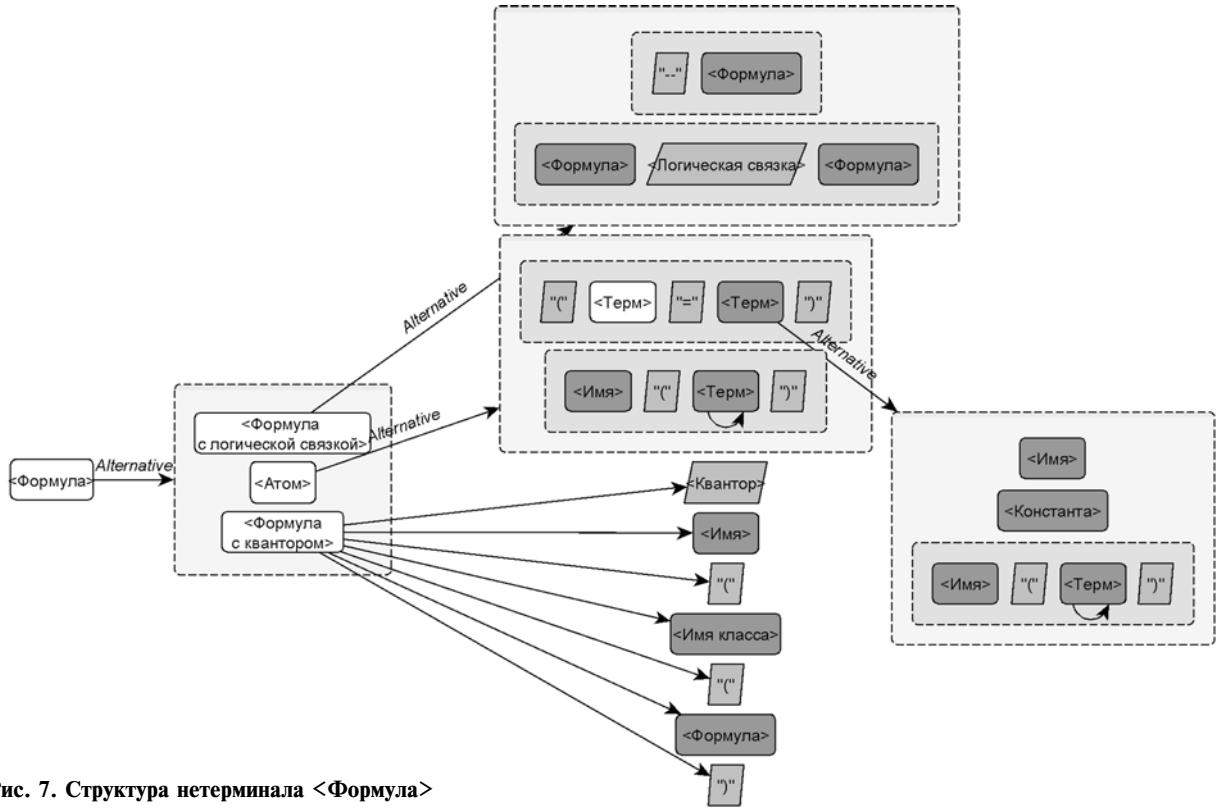

Рис. 7. Структура нетерминала <Формула>

ные элементы <Логическая связка> = "&" | "∨" | "→" и <Квантор> = "∀" | "∃" | "∃!".

Здесь <*Ps*> = <Формула > {"," < Формула > }.
<Терм> = <Имя> | <Константа> | <Имя> "(" {<Терм>} ")".
<Атом> = ("("<Терм>"="<Терм>")") | (<Имя> "(" {<Терм>} ")").
<Формула> = <Атом> | "(¬" <Формула> ")"
|"(" <Формула> <Логическая связка> <Формула> ")"
| <Квантор> <Имя> "∈" <Имя класса> "(" <Формула> ")".

Рассмотрим пример области свойств в определении одноэлементного класса морфизмов семантического связывания конфигураций (канонического связывания) [1]. Отображение ψ: $M \to R$ является морфизмом семантического связывания, если оно ставит в соответствие всякой неэлементарной конфигурации семантическое отношение, выполняющееся между парой конфигураций, составляющих ее разложение [3].

Df1. {Каноническое семантическое связывание}

$$(\{\psi\};\ \psi\colon M \to R;\ \forall z \in M(\varepsilon(z) = (z_1, z_2)\ \&$$
$$\&\ (z_1 \neq \Lambda \vee z_2 \neq \Lambda) \to \varepsilon(z) \in \psi(z)),$$
$$\forall r \in R \forall z_1, z_2 \in M((z_1, z_2) \in r \to$$
$$\to \exists!\ z \in M(\varepsilon(z) = (z_1, z_2)\ \&\ \psi(z) = r));\ G(\{\psi\})).$$

Здесь определяется одноэлементный класс $\{\psi\}$. Области определения и значений ψ уточняются в области форматов с помощью записи ψ: $M \to R$. Множество R образует специальный класс разрешимых отношений между парами конфигураций. Свойство $(z_1 \neq \Lambda \vee z_2 \neq \Lambda) \to \varepsilon(z) \in \psi(z)$ означает, что если пара (z_1, z_2) образует разложение некоторой неэлементарной конфигурации z, то (z_1, z_2) принадлежит отношению $\psi(z)$. Дополнительно требуется, чтобы для каждой пары конфигураций $z_1, z_2 \in M$ и отношения r, выполняющегося между этими конфигурациями, существовала единственная конфигурация z, представляющая z_1 и z_2, связанные отношением r [3].

Область алгоритмов определений классов

Область *As* содержит сведения об алгоритмах, которые связаны классами. К ним относятся алгоритмы генерации (перечисления) и распознавания элементов классов. Для рассматриваемых видов алгоритмов содержание области *As* определяется следующими правилами:

<*As*> = ("*G*" | "*NG*") "(" <Имя класса> "),"
("*R*" | "*NR*") "(" <Имя класса> ")".

As также может включать ссылки на другие алгоритмы и схемы их композиции и выбора.

Конструирование и моделирование цифровых пространств знаний

Приведенная в работе система конструктов языка описаний пространств знаний позволяет применять форматы расширенного языка TEX, адаптированного

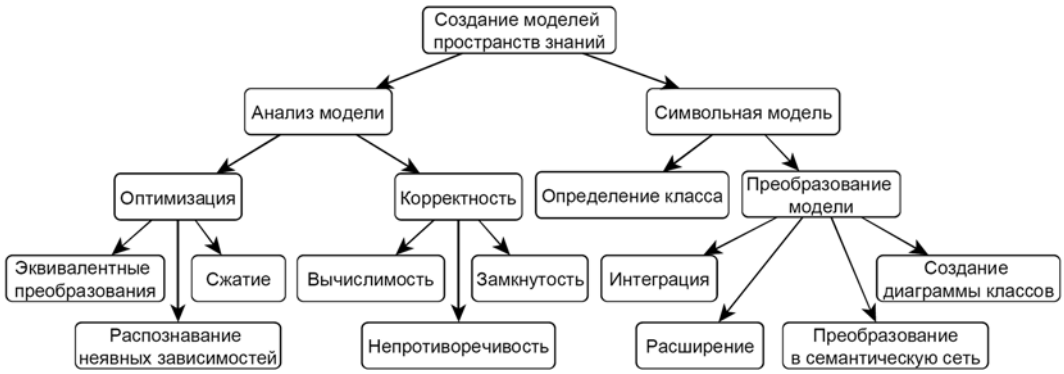

Рис. 8. Классы операций системы построения моделей пространств знаний

к особенностям определений классов и отношений между ними. Такие форматы удобны для конструирования и анализа логико-математических моделей, отражающих разнообразные атрибуты и свойства знаний. Естественным образом данных моделей, предназначенным для эффективного использования знаний в профессиональной деятельности, являются иерархические семантические сети конечной глубины [6]. Сети рассматриваемого вида составляют конечные множества вершин, связанных ориентированными дугами и размеченными семантическими отношениями. Вершины сетей являются элементарными или сложными (соответствующими семантическим сетям). Полное структурное представление пространства знаний в рассматриваемом формате составляет иерархия сетей. Каждая сложная вершина всякой сети связана с представляющей ее сетью, размещенной в следующем ярусе иерархии. Первый ярус такой сети составляет диаграмма классов. Элементарные вершины сети пространства знаний соответствуют таким фрагментам выражений (формул) в составе описаний классов, которые не являются фрагментами выражений (подформулами) других описаний классов, составляющих пространство знаний.

Инструментальная система построения моделей пространств знаний основана на правилах составления описаний классов, группирования классов и задания отношений между ними. Модели пространств знаний представляются как в символьном формате, так и в формате иерархических семантических сетей. Обработка описаний классов, составляющих пространства знаний, включает анализ синтаксической правильности, замкнутости, вычислимости, распознавания явных и неявных зависимостей классов и их элементов. Кроме того, выполняется построение и оптимизация семантической сети, анализ ее метрической и топологической структур. Фрагмент функциональной структуры инструментальной системы приведен на рис. 8.

Заключение

Приведенные в работе форматы описания пространств знаний позволяют задавать последние как математические системы. Применяемые для этого логико-математические выражения обеспечивают эффективное моделирование структурно-функциональных свойств систем знаний в произвольных областях. При этом достигаются полнота, целостность и возможность расширения моделей, управления процессами встраивания целостных фрагментов формализованных знаний в программные и информационные системы [3]. Унифицированный формат описаний разработан в целях абстрактного и прикладного моделирования целостных многообразий знаний. Он близок к алгебраическим системам и поэтому допускает применение для моделирования более широкого класса формальных систем. Последнее позволяет адаптировать и совместно использовать конструкции и инструменты моделей разных типов.

Знания, как один из способов отражения представлений о мире, обладают не только логико-математическими характеристиками. Содержательно полное моделирование знаний и процессов работы с ними использует разнообразные эмпирические, слабо формализуемые универсальные когнитологические, лингвистические, психологические и иные свойства. Они находят свое отражение и практическое применение в инвариантах пространств знаний.

Список литературы

1. **Васенин В. А., Лёвин В. Ю., Пучков Ф. М., Шапченко К. А.** К созданию защищенной наложенной сети нового поколения передачи данных // Программная инженерия. 2012. № 4. С. 2—9.

2. **Костенко К. И.** Операции унифицированной технологии построения цифровых пространств знаний // Информационные технологии. 2012. № 2. С. 8—13.

3. **Костенко К. И.** Компоненты и операции абстрактных пространств знаний // Материалы Всероссийской конференции ЗОНТ09. Новосибирск, 20—22 октября 2009. Т. 2. С. 36—40.

4. **Поспелов Д. А., Осипов Г. С.** Прикладная семиотика // Новости искусственного интеллекта. 1999 г. № 1. С. 9—35.

5. **ISO/IEC 14977: 1996.** Information technology — Syntactic metalanguage — Extended BNF.

6. **Кузнецов И. П.** Семантические представления. М.: Наука, 1986. 304 с.

УДК 004.31

П. Н. Бибило, д-р техн. наук, проф., зав. лаб., **Л. Д. Черемисинова**, д-р техн. наук, доц., глав. науч. сотр., **С. Н. Кардаш**, канд. техн. наук, стар. науч. сотр., **Н. А. Кириенко**, канд. техн. наук, доц., стар. науч. сотр., e-mail: kir@newman.bas-net.by, **В. И. Романов**, канд. техн. наук, доц., вед. науч. сотр., **Д. И. Черемисинов,** канд. техн. наук, доц., вед. науч. сотр., Объединенный институт проблем информатики Национальной академии наук Беларуси, г. Минск

Автоматизация логического синтеза КМОП-схем с пониженным энергопотреблением

Описывается структура и функциональные возможности созданного авторами программного комплекса энергосберегающего логического синтеза, предназначенного для автоматизации проектирования многоуровневых логических схем из библиотечных элементов заказных сверхбольших интегральных схем, выполненных по КМОП-технологии. За критерии оптимизации при проектировании приняты сложность КМОП-микросхем и их энергопотребление.

Ключевые слова: *автоматизация проектирования, заказные КМОП СБИС, синтез схем с пониженным энергопотреблением*

Введение

Прогресс в области микроэлектронных технологий приводит к постоянному повышению степени интеграции и тактовой частоты создаваемых с их помощью микросхем. Это, в свою очередь, позволяет создавать на одном кристалле более быстродействующие и функционально более сложные устройства. В первой половине 2000-х гг. сформировалось четкое представление, что одним из существенных препятствий на этом пути стала проблема, обусловленная сложностями снижения энергопотребления проектируемых устройств [1, 2] (для краткости изложения в статье использован термин "проблема энергопотребления"). Этому направлению исследований придается в последние годы все большее значение в связи с тем, что рассеивание энергии становится камнем преткновения при создании схем по субмикронным нормам проектирования.

Проектирование с учетом минимизации энергопотребления до настоящего времени остается в большей степени искусством. Такое положение, в частности, связано с отсутствием эффективных средств оценки влияния используемых в процессе проектирования эвристик на энергопотребление схемы, реализованной в дальнейшем на кристалле сверхбольших интегральных схем (СБИС). Решением проблемы энергопотребления при проектировании СБИС занимаются многие специалисты и фирмы, работающие в области автоматизации проектирования. К их числу относятся, например, Cadence Design Systems (лидер в области автоматизации проектирования), Apache Design, Atrenta, Magma Design Automation, Synopsys, Mentor Graphics и др. [3].

Попытка создания средств автоматизации проектирования с учетом энергопотребления предпринята и в Объединенном институте проблем информатики Национальной академии наук (ОИПИ НАН) Беларуси, где разработан комплекс программ "Энергосберегающий логический синтез" (ЭЛС) для проектирования логических схем из библиотечных КМОП-элементов. В данной работе описывается структура и функциональные возможности комплекса ЭЛС, который содержит: средства оптимизации проектируемых цифровых блоков заказных КМОП-микросхем на функциональном и структурном уровнях; средства верификации состояний проектируемых схем; средства оценки энергопотребления схем как в процессе их проектирования, так и спроектированных схем из библиотечных элементов. Представленный программный комплекс ЭЛС:

— позволяет по функциональному описанию проектируемого устройства на языке высокого уровня *VHDL* (*Very high speed integrated circuits Hardware Description Language*) или на языке *SF* (*Structural and Functional description language*) [4], который является внутренним языком комплекса, получить структурное описание логической схемы в библиотеке проектирования КМОП СБИС;

— реализует подход к синтезу, позволяющий минимизировать площадь кристалла КМОП СБИС и энергопотребление, измеряемое средним значением рассеиваемой схемой мощности;

— имеет интерактивные средства проектирования логических схем, верификации и оценки проектных решений;

— позволяет проводить оценку энергопотребления схем из библиотечных элементов на логическом и схемотехническом уровнях.

Рассматривается случай синтеза комбинационных схем, для которого:

- предполагается синхронная реализация схем;
- частота синхронизации и напряжение питания фиксированы;
- для оценки энергопотребления в процессе синтеза схем используется статистический метод, основанный на вероятностных характеристиках входных сигналов.

Оценка энергопотребления в рамках программного комплекса

В КМОП-технологии всю рассеиваемую схемой мощность можно разделить на статическую и динамическую составляющие. Статическая составляющая обусловлена наличием статических проводящих путей между шинами питания или токов утечки. Ее значение связано с топологией схемы на уровне транзисторов [1—3]. Учесть эту составляющую на этапе проектирования логической схемы не представляется возможным. Значительная часть рассеиваемой КМОП-схемой энергии (при технологических нормах производства интегральных схем 0,35...0,18 мкм) приходится на ее динамическую составляющую [3, 5], порождаемую нестационарным поведением узлов схемы. Эта энергия рассеивается только во время переходных процессов, когда сигналы на выходах элементов переключаются.

Компоненты СБИС, выполненные по КМОП-технологии, потребляют подавляющую часть необходимой для их функционирования энергии во время их переключения. Энергопотребление существенно зависит от переключательной активности элементов схемы. Переключательная активность элемента [2, 5, 6] определяется как математическое ожидание числа логических переходов (из 1 в 0 или из 0 в 1) значения сигнала на выходе элемента за один период синхронизации. Соответственно, переключательная активность элементов схемы существенно зависит от последовательности сигналов, подаваемых на входы схемы, т. е. от динамики ее функционирования, и может быть учтена на этапе логического синтеза.

Под оценкой энергопотребления далее понимается оценка среднего значения энергии, рассеиваемой схемой. Эта оценка существенно отличается от оценки мощности, максимально потребляемой в каком-то отдельном такте функционирования схемы.

В основе методов оценки переключательной активности, используемых в системе ЭЛС в процессе проектирования схемы, лежит подход, основанный на вероятностных характеристиках входных сигналов и функционально-структурных свойствах исследуемой схемы. Предполагается, что заданы сигнальные вероятности входных полюсов схемы. Сигнальная вероятность p_i i-го полюса схемы [6] определяется средней долей тактов, на которых сигнал на этом полюсе имеет единичное значение. Переключательная активность E_i i-го полюса схемы отражает частоту смены значений сигнала на этом полюсе, ее значение зависит от сигнальной вероятности p_i полюса и равно $E_i = 2p_i(1 - p_i)$ [2, 5].

Сигнальная вероятность p_i выходного полюса элемента зависит от вероятностных характеристик сигналов на входах элемента. В случае, когда сигналы на входах элемента не коррелируют в пространстве (отсутствует зависимость значений сигналов на разных полюсах друг от друга) и во времени (значение сигнала в любом такте синхронизации не зависит от его значений в предшествующих тактах), сигнальные вероятности элементов могут быть подсчитаны, исходя из таблиц истинности реализуемых ими функций [2]. Например, сигнальные вероятности простых элементов инвертор, И и ИЛИ с $n(e)$ входными полюсами равны соответственно:

$$p_e^{\neg} = 1 - p_1;\ p_e^{\wedge} = \prod_{i=1}^{n(e)} p_i;\ p_e^{\vee} = 1 - \prod_{i=1}^{n(e)} (1 - p_i).$$

Кроме вероятностных оценок энергопотребления, в комплексе предусмотрены средства, позволяющие оценить энергопотребление спроектированных комбинационных блоков заказных СБИС путем подсчета общего числа переключений транзисторов элементов схемы на заданных тестовых последовательностях входных наборов (тестах) с помощью быстродействующего *VHDL*-моделирования (либо *SF*-моделирования) [7]. Авторами разработаны средства получения *SPICE*-описаний логических схем и генерации различных тестов. Эти средства позволяют проводить оценку энергопотребления с помощью систем схемотехнического моделирования на основе *SPICE*-формата (*SPICE — Simulation Program with Integrated Circuit Emphasis* — программа, разработанная университетом Калифорнии в Беркли в начале 1970-х гг.).

Логический синтез с учетом энергопотребления

Синтез комбинационных многоуровневых сетей есть процесс преобразования исходной системы логических уравнений в результирующую систему уравнений. Каждое уравнение соответствует одному элементу технологической библиотеки. Каждый из элементов библиотеки характеризуется функцией (уравнением), которую он реализует, и его физическими характеристиками. В комплексе ЭЛС, как и в большинстве систем проектирования, процесс логического синтеза состоит из двух стадий: технологически независимая оптимизация и технологическое отображение [8].

Такой подход характерен для всех подобных систем автоматизированного проектирования. Первая стадия синтеза ориентирована на оптимизацию и декомпозицию логики реализации схемы, а вторая — на перевод схемы из технологически независимого базиса в технологический. Под технологически независимым базисом понимается некоторый функционально-полный набор простых вентилей. В практике проектирования широко используют следующие технологически независимые базисы: НЕ, И и ИЛИ; И-НЕ; ИЛИ-НЕ. Цель первого этапа заключается в минимизации сложности объектной сети, а именно многоуровневой схемы из вентилей (типа И, ИЛИ), измеряемой, как правило, числом полюсов вентилей (оценка по Квайну), и минимизации энергопотребления. Целью второго этапа является оптимальный перевод объектной схемы в технологический базис по критериям площади и энергопотребления.

Основной подход к решению задачи технологического отображения базируется на покрытии фрагментов объектной сети в технологически независимом базисе подсхемами. Такие подсхемы реализуют библиотечные элементы и представлены в том же вентильном базисе, что и покрываемая объектная сеть. Такой подход не предполагает кардинальную перестройку схемы, полученной на этапе технологически независимой оптимизации. Качество искомого покрытия существенно зависит от структуры объектной многоуровневой сети. По этой причине в разработанном комплексе большое внимание уделяется этапу технологически независимой оптимизации и декомпозиции реализуемого описания в объектную сеть.

Технологически независимая оптимизация в качестве первого этапа включает в себя этап минимизации функций реализуемых логических описаний в классе дизъюнктивных нормальных форм (ДНФ) [8]. На втором этапе минимизированная система ДНФ, которая представляется двухуровневой схемой, декомпозируется в объектную сеть из базовых элементов, обычно вентилей типа И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ. Основным методом решения задачи декомпозиции систем ДНФ, который используется во всех САПР, является алгебраическая декомпозиция. Подобная декомпозиция сводится к факторизации (поиску и выделению общих частей логических выражений — конъюнкций и дизъюнкций системы ДНФ как алгебраических выражений) и к вынесению за скобки переменных [9].

Цель этапа декомпозиции заключается в построении такого варианта представления схемы, который мог бы служить хорошей отправной точкой для этапа технологического отображения в базис библиотечных элементов, выполненных по КМОП-технологии. Исходя из принятых критериев оптимальности сети (сложность и энергопотребление), на обоих этапах технологически независимой оптимизации (минимизация и декомпозиция) целесообразно находить и выделять общие части логических выражений. Следовательно, необходимо минимизировать функции исходной системы совместно, а принимая во внимание специфику целевой библиотеки (реализовать можно как функции, так их инверсии), рационально проводить совместную минимизацию с учетом полярности функций [10]. В процессе такой минимизации для каждой из функций системы выбирается та форма ее реализации (функция или ее инверсия), которая имеет меньшую сложность и оценку энергопотребления соответствующей подсхемы. Минимизацию в классе ДНФ относят к оптимизации двухуровневых (И-ИЛИ) представлений систем функций.

Оптимизация многоуровневых представлений осуществляется как на уровне алгебраических скобочных представлений [11], так и на функциональном уровне на основе поиска *BDD*-представлений (*Binary Decision Diagram*) систем булевых функций [12]. Оптимизация *BDD* сводится к нахождению одинаковых коэффициентов разложения Шеннона функций системы (нахождению общих подфункций) в оптимизируемых представлениях систем функций. Заметим, что программы оптимизации могут обрабатывать как системы полностью определенных, так и системы частичных булевых функций, заданных либо наборами значений аргументов, либо в интервальной форме. Результатом оптимизации системы частичных функций является реализующая ее система полностью определенных функций.

Технология проектирования в рамках комплекса ЭЛС

Проектирование цифровых блоков СБИС в рамках комплекса представляет собой многоэтапный процесс изменения структурного или функционального описания схемы. Каждое из полученных описаний схемы задает новое состояние проекта и называется проектным решением. Процесс проектирования начинается с исходного описания цифрового блока на одном из входных языков проектирования и заканчивается представлением его схемы в технологическом базисе. Проектные решения могут быть получены как в автоматическом (с использованием программных средств синтеза и оптимизации), так и в полуавтоматическом (путем корректировки решения проектировщиком) режимах. Для того чтобы избежать распространения ошибок, допущенных на одном из ранних этапов проектирования, в комплекс включены средства верификации проектных решений на всех этапах проектирования [13, 14]. Верификация состоит в проверке, находятся полученные решения в отношении эквивалентности (если оба описания полностью определены) или реализации (если исходное не полностью определено).

Процесс проектирования с использованием программного комплекса ЭЛС включают следующие стадии (каждая из которых реализуется набором альтернативных проектных операций):

- разработка функционального описания проектируемой схемы;
- оптимизация функциональных описаний двухуровневых и многоуровневых схем с учетом сложности и энергопотребления;

Рис. 1. Синтез логических схем в рамках комплекса программ ЭЛС

- синтез и оптимизация схем в заданной библиотеке КМОП-элементов с учетом сложности и энергопотребления;
- верификация проектных решений на всех стадиях проектирования;
- генерация тестовых последовательностей и оценка энергопотребления на логическом и схемотехническом уровнях.

Общая структурная схема преобразования данных в экспериментальном программном комплексе ЭЛС автоматизированного проектирования логических схем из элементов КМОП-библиотеки с минимизацией энергопотребления показана на рис. 1.

Технология проектирования при использовании программного комплекса ЭЛС основывается на последовательном преобразовании описания проектируемой схемы, представленного на языке *SF* [4]. Язык *SF*, являясь также и внутренним языком системы, ориентирован на иерархические структурно-функциональные описания логических схем. Комбинационные блоки или элементы на языке *SF* задают либо в виде логических уравнений, либо в матричном виде (парой матриц, описывающих систему ДНФ булевых функций). Исходное функционально-структурное описание проектируемой схемы может быть представлено на языке *VHDL*.

Комплекс ЭЛС обеспечивает конвертацию описания схемы с *VHDL* на язык *SF* [15]. Обеспечивается и обратная конвертация полученного структурного описания схемы из элементов библиотеки КМОП СБИС в описание на языке *VHDL*. Конвертированным в *VHDL* может быть и любое промежуточное описание логической схемы, проектируемой в ЭЛС. Такое согласование данных позволяет использовать программный комплекс ЭЛС совместно с другими существующими средствами проектирования схем, например, с синтезатором логических схем LeonardoSpectrum [16]. Многочисленные эксперименты подтвердили эффективность такого подхода к проектированию. В этом случае предварительная оптимизация выполняется с помощью программ, отсутствующих в LeonardoSpectrum, а заключительный этап, связанный с покрытием оптимизированных представлений функциональными описаниями элементов целевой библиотеки, выполняет промышленный синтезатор LeonardoSpectrum.

Архитектура программного комплекса ЭЛС

Программная оболочка комплекса ЭЛС обладает средствами информационной и языковой поддержки процессов проектирования. Такие средства включают: справочную подсистему и выдачу экспресс-оценок результатов выполнения проектных операций; реализацию альтернативных технологических маршрутов проектирования и проектных операций. Программный комплекс ЭЛС состоит из четырех подсистем:

- формирования проекта;
- оптимизации проекта;
- верификации;
- оценки энергопотребления.

Первая подсистема обеспечивает поддержку создания, редактирования и преобразования форм исходного задания проекта. Вторая подсистема сопровождает выполнение проектных процедур оптимизации и синтеза. Третья подсистема осуществляет контроль преобразований описания проекта. Четвертая подсистема поддерживает механизмы генерации тестов и оценки энергопотребления на сгенерированных или заданных тестах.

Все данные о текущем состоянии проектируемой схемы образуют проект. Кроме *SF*-описания схемы в проекте определен ряд дополнительных данных, часть из которых представлена атрибутами и отражает некоторые свойства текущего описания схемы. К их числу относятся, например, текущий формат описа-

ния; параметры выполняемой проектной операции; имя последней выполненной проектной операции; возможные последующие действия; история выполнения проектных операций (например, выполнялась или нет минимизация схемы) и др.

В соответствии с назначением программного комплекса в качестве используемых входных данных для проектирования могут выступать:

— функциональное описание проектируемых схем комбинационной логики на языках *VHDL* и *SF*;

— функционально-структурное описание проектируемых схем комбинационной логики на языках *VHDL* и *SF*;

— распределение сигнальных вероятностей появления значения 1 на входах проектируемой схемы;

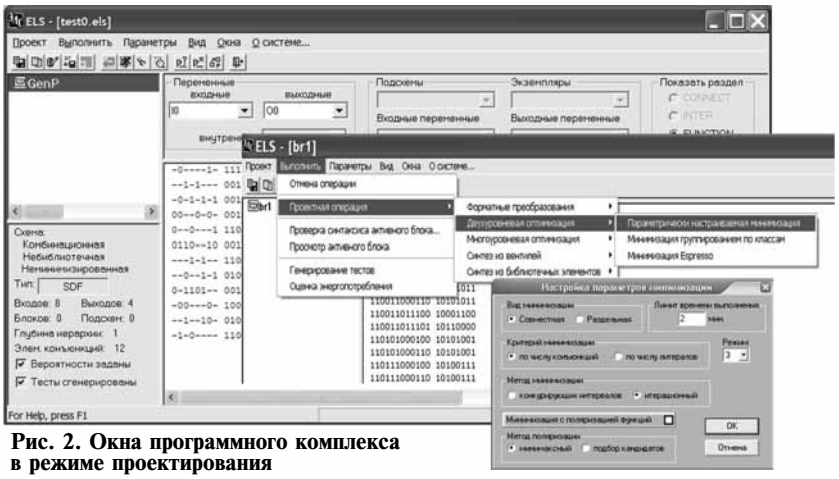

Рис. 2. Окна программного комплекса в режиме проектирования

— наборы тестовых последовательностей, которые используют для моделирования поведения проектируемой схемы на предмет построения оценок их энергопотребления;

— рабочие модели для проведения моделирования схем в системе *SPICE*.

Программная оболочка представляет собой набор инструментальных и сервисных средств для наблюдения за ходом проектирования и управления им. Она включает в себя совокупность следующих подсистем:

- развертывания сеанса;
- формирования или настройки проекта;
- импорта и экспорта *VHDL*-описаний;
- обслуживания используемых в проекте данных;
- организации выполнения проектных операций;
- верификации полученных проектных решений;
- оценки переключательной активности схемной реализации проектного решения на всех этапах оптимизации и синтеза логической схемы;
- оценки энергопотребления проектируемой схемы.

Весь процесс проектирования ориентирован на использование интерактивного, диалогового режима с поддержкой и подсказками системной части комплекса. Комплекс снабжен развитой справочной системой. Все программы комплекса ЭЛС написаны на языке C++ с использованием разработанных авторами структур данных и протестированы в рамках программы тестовых испытаний под управлением операционной системы WindowsXP.

Функциональные возможности программного комплекса

Программный комплекс ЭЛС включает в себя перечисленные ниже классы программ, поддерживающих полный цикл проектирования логических схем из библиотечных элементов. Эти программы обеспечивают синтез схем и их оптимизацию по площади и энергопотреблению на всех стадиях проектирования.

Двухуровневая оптимизация включает набор программ минимизации систем полностью и частично определенных булевых функций в классе ДНФ. Основным критерием оптимизации в данном случае является минимум интегральной оценки. Эта оценка учитывает сложность получаемых ДНФ и суммарную переключательную активность реализующих их двухуровневых схем.

Разработанный пакет программ двухуровневой оптимизации включает в себя программы, реализующие приближенные методы. Точные методы минимизации в комплекс включены не были. Причина в том, что для булевых функций реальной размерности получение минимальных ДНФ в приемлемое для практического применения время представляет большие сложности. Текущая версия программного комплекса ЭЛС включает в себя программы двухуровневой оптимизации [10] (рис. 2), реализующие следующие методы:

- параметрически настраиваемая минимизация;
- минимизация путем группирования по классам;
- минимизация Espresso.

Эти программы обеспечивают минимизацию при следующих потенциально возможных требованиях:

- требования к объекту оптимизации:
  - — одной полностью определенной булевой функции или системы таких функций,
  - — одной частично определенной булевой функции или системы таких функций;
- требования к методу решения:
  - ♦ итеративный метод минимизации [10],
  - ♦ метод конкурирующих интервалов [10, 17],
  - ♦ модифицированный метод минимизации Espresso [18];
  - ♦ метод группирования по классам [19];
- требования к способу учета совместности минимизации системы ДНФ:
  - ♦ раздельная минимизация функций системы ДНФ,
  - ♦ совместная минимизация функций системы ДНФ;
- требования к критерию оптимизации по энергопотреблению:
  - ♦ без учета энергопотребления,
  - ♦ с учетом энергопотребления;
- требования к учету дополнительных настроек, включающих:
  - ♦ получение парафазных реализаций ДНФ [17];

- минимизацию числа конъюнкций результирующей системы ДНФ;
- минимизацию числа литералов конъюнкций результирующей системы ДНФ;
- лимит времени минимизации;
- режим работы алгоритма (для метода конкурирующих интервалов).

Многоуровневая оптимизация включает набор программ построения и оптимизации многоуровневых представлений систем полностью определенных булевых функций. Многоуровневые представления задаются системами факторизованных форм, конъюнкции и дизъюнкции которых имеют ограниченные ранги [9]. Такой подход приближает соответствие многоуровневого представления ограничениям целевого библиотечного базиса. Текущая версия комплекса ЭЛС включает в себя программы, реализующие приближенные методы совместной и раздельной факторизации с учетом энергопотребления. Исходными данными для программ многоуровневой оптимизации являются:

- функциональное описание системы булевых функций, заданное на языке *SF*;
- сигнальные вероятности для входных полюсов схемы, если требуется учет энергопотребления;
- ограничение на ранги конъюнкций;
- ограничение на ранги дизъюнкций.

Синтез логических сетей из вентилей включает набор программ синтеза и оптимизации многоуровневых сетей из вентилей И, ИЛИ (с использованием инверторов на входах) с ограниченным числом входных полюсов. Основным критерием качества многоуровневой схемы является минимум интегральной оценки, которая учитывает сложность схемы (измеряется числом входных полюсов всех ее вентилей) и суммарную переключательную активность всех ее полюсов (включая входные, внутренние и выходные полюсы).

Текущая версия комплекса ЭЛС включает в себя программы синтеза логических сетей из двухвходовых вентилей; двухвходовых вентилей от частичных функций; многовходовых вентилей.

Первые две из этих программ реализуют методы синтеза многоуровневых логических схем из двухвходовых вентилей И и ИЛИ (с использованием инверторов) для полностью и частично определенных булевых функций [12]. Методы основаны на построении и оптимизации диаграммы двоичного выбора (*BDD*). Третья программа реализует метод синтеза многоуровневых логических схем из вентилей И и ИЛИ с ограниченным числом входов (эти числа определяются технологическим базисом). Метод основан на сочетании алгоритмов совместной и раздельной факторизации систем ДНФ булевых функций с учетом энергопотребления [11].

Исходными данными для алгоритмов многоуровневой оптимизации являются:

- функциональное описание системы булевых функций, заданное системой ДНФ в формате SDF или LOG языка *SF*;
- сигнальные вероятности для входных переменных, если требуется учет энергопотребления;
- ограничение на число входных полюсов вентилей И;
- ограничение на число входных полюсов вентилей ИЛИ.

Синтез из библиотечных элементов включает набор программ технологического отображения многоуровневых сетей из вентилей И, ИЛИ в базис элементов заданной библиотеки КМОП СБИС. Основным критерием качества многоуровневой схемы из библиотечных элементов является минимум интегральной оценки, которая учитывает сложность схемы (измеряется числом транзисторов всех элементов) и суммарную переключательную активность всех ее полюсов (входные, внутренние и выходные полюсы).

Текущая версия комплекса ЭЛС включает в себя программы синтеза схем из библиотечных элементов, которые реализуют следующие приближенные методы покрытия многоуровневых логических схем из вентилей И, ИЛИ:

- упрощенный метод технологического отображения схемы из вентилей в библиотечный базис;
- метод технологического отображения схемы из вентилей в библиотечный базис с оптимизацией [20];
- комбинированный метод технологического отображения системы полностью или частично определенных булевых функций в схему из библиотечных элементов.

Комбинированный метод синтеза схем из библиотечных элементов включает выполнение сразу нескольких операций, реализованных в известной системе SIS [21]: минимизация систем булевых функций, построение многоуровневой схемы из двухвходовых вентилей И-НЕ или ИЛИ-НЕ, покрытие этой схемы элементами заданной библиотеки.

Верификация состояний проекта включает набор программ, позволяющих осуществить верификацию заданной пары проектных решений [14]. Верификация работает для любых пар проектных решений одного и того же проекта (или разных проектов). Единственное ограничение: второе состояние не должно предшествовать первому (в цепочке выполняемых проектных операций), т. е. первый из сравниваемых объектов не может быть "более определенным", чем второй, если хотя бы одно из описаний содержит неопределенность поведения (например, в случае минимизации или схемной реализации систем частично определенных булевых функций).

Когда оба сравниваемых описания функционально полностью определены (например, в случае комбинационных схем или систем ДНФ), при верификации проверяется, имеет ли место эквивалентность этих описаний. Если хотя бы одно из описаний содержит неопределенность поведения, при верификации проверяется, имеет ли место отношение реализации между этими описаниями, т. е. реализуется ли первое описание вторым. В случае, когда первое описание не реализуется вторым, программный модуль верификации позволяет выявить причину нереализуемости, а именно определяет интервал (или набор) и функцию исходного описания или элемент схемы, которые являются причиной нарушения реализуемости.

Оценку энергопотребления выполняют с помощью программ генерации тестов и оценки уровня энергопотребления схемы по результатам анализа выполнения

заданного теста. Такой подход позволяет оценить эффективность проектного решения в процессе проектирования схемы после выполнения очередной операции оптимизации или синтеза, когда ее реализация на кристалле СБИС еще не получена. Оценка энергопотребления в рамках комплекса ЭЛС осуществляется путем:

— оценки переключательной активности схемы на основе подсчета сигнальных вероятностей для полюсов схемной реализации объекта проектирования, выполняемой автоматически после каждой проектной операции, изменяющей его состояние;

— подсчета числа переключений транзисторов для схемы из библиотечных элементов на основе логического (*VHDL* либо *SF*) моделирования для заданного теста и оценки значения среднего тока, потребляемого в одном такте [7];

— оценки энергопотребления схемы из библиотечных элементов на схемотехническом уровне [22] на основе прогона тестов и измерения потребляемого тока путем *SPICE*-моделирования в системе *Accusim* фирмы [23] Mentor Graphics.

Применение логического *VHDL/SF* моделирования для оценки энергопотребления позволяет на несколько порядков сократить время оценки при приемлемой (10...15 %) погрешности такой оценки. Тесты для моделирования схемы могут быть заданы в виде текстового файла определенного формата или сгенерированы автоматически. В автоматическом режиме генерируются тесты четырех типов: псевдослучайные тесты заданной длины с учетом сигнальных вероятностей входных полюсов; упорядоченные по возрастанию десятичного эквивалента последовательности из 2^n наборов булева пространства размерности n; упорядоченные по убыванию десятичного эквивалента последовательности из 2^n наборов булева пространства размерности n; последовательности 2^n ($2^n - 1$) упорядоченных пар всех наборов булева пространства [24]. Тесты строятся в двух форматах — для *VHDL*-моделирования и для *SPICE*-моделирования.

Заключение

Программный комплекс ЭЛС предназначен для проектирования схем комбинационной логики, имеющих сотни входных/выходных переменных и тысячи элементов. Исходное функциональное описание проектируемой логической схемы в переводе на эквивалентное представление в виде системы булевых функций в интервальной форме может иметь до нескольких десятков аргументов и функций, нескольких тысяч интервалов значений аргументов.

Синтез логических схем в ЭЛС осуществляется с учетом переключательной активности сигналов, что позволяет минимизировать энергопотребление проектируемых схем. Система ЭЛС имеет развитые средства технологически независимой оптимизации, что позволяет использовать ее при синтезе логических схем не только в базисе КМОП-элементов, но и в других технологических базисах.

Список литературы

1. **Рабаи Ж. М., Чандракасан А., Николич Б.** Цифровые интегральные схемы. Методология проектирования. М.: Вильямс, 2007. 912 с.
2. **Benini L., De Micheli G.** Logic Synthesis for Low Power // Logic Synthesis and Verification / eds. S. Hassoun, T. Sasao, R.K. Brayton. Boston, Dardrecht, London: Kluwer Academic Publishers. 2002. P. 197—223.
3. **Taking** a bite out of power: techniques for low-power-ASIC design. URL: http://www.edn.com/article/460106-Taking_a_bite_out_of_power_ techniques_for_low_power_ASIC_design.php.
4. **Бибило П. Н., Романов В. И.** Логическое проектирование дискретных устройств с использованием продукционно-фреймовой модели представления знаний. Минск: Беларуская навука, 2011. 279 с.
5. **Roy K., Prasad S. C.** Low Power CMOS VLSI Circuit Design. New York: John Wiley and Sons Inc., 2000. 376 p.
6. **Черемисинова Л. Д.** Оценка энергопотребления КМОП-схем на логическом уровне // Информационные технологии. 2010. № 8. С. 27—35.
7. **Бибило П. Н., Кириенко Н. А.** Оценка энергопотребления логических КМОП-схем по их переключательной активности // Микроэлектроника. 2012. Т. 41. № 1. С. 65—67.
8. **Брейтон Р. К., Хэчтел Г. Д., Санджованни-Винчентелли А. Л.** Синтез многоуровневых комбинационных логических схем // ТИИЭР. 1990. Т. 78. № 2. С. 38—83.
9. **Черемисинова Л. Д.** Синтез и оптимизация комбинационных структур СБИС. Минск: ОИПИ НАН Беларуси, 2005. 236 с.
10. **Черемисинов Д. И., Черемисинова Л. Д.** Минимизация двухуровневых КМОП-схем с учетом энергопотребления // Информационные технологии. 2011. № 5. С. 17—23.
11. **Черемисинова Л. Д., Кириенко Н. А.** Синтез многоуровневых логических схем с учетом энергопотребления // Информационные технологии. 2012. № 3. С. 8—14.
12. **Бибило П. Н., Леончик П. В.** Декомпозиция систем булевых функций, заданных диаграммами двоичного выбора // Известия РАН. Теория и системы управления. 2011. № 4. С. 86—101.
13. **Черемисинова Л. Д., Новиков Д. Я.** Формальная верификация описаний с функциональной неопределенностью на основе проверки выполнимости конъюнктивной нормальной формы // Автоматика и вычислительная техника. 2010. № 1. С. 5—16.
14. **Новиков Д. Я., Черемисинова Л. Д.** Программный комплекс для верификации комбинационных устройств в процессе логического проектирования // Пятый Белорусский космический конгресс: материалы конгресса. Минск, 25—27 октября 2011 г. Минск: ОИПИ НАН Беларуси, 2011. Т. 2. С. 289—293.
15. **Черемисинов Д. И.** Анализ и преобразование структурных описаний СБИС. Минск: Беларуская навука, 2006. 275 с.
16. **Бибило П. Н.** Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum. М.: СОЛОН-Пресс, 2005. 384 с.
17. **Торопов Н. Р.** Минимизация системы булевых функций с поляризацией их значений // Автоматизация проектирования дискретных систем: материалы IV междунар. конф. CAD DD'2001. Минск: Ин-т техн. кибернетики НАН Беларуси, 2001. С. 92—104.
18. **A short** introduction to Espresso. URL: http://www.uic.edu/classes/ece/ece465/06/tools/A%20short%20introduction%20to%20Espresso.pdf.
19. **Леончик П. В.** Минимизация систем булевых функций в классе дизъюнктивных нормальных форм // Информатика. 2006. № 1. С. 88—96.
20. **Черемисинова Л. Д.** Синтез комбинационных КМОП-схем с учетом энергосбережения // Информатика. 2010. № 4. С. 112—122.
21. **Sentovich E. M., Singh K. J., Lavagno L.** et al. SIS: A System for Sequential Circuit Synthesis. Technical Report. UCB/ERI, M92/41, ERL, Dept. of EECS, Berkeley, CA, May 1992.
22. **Бибило П. Н.** Оценка энергопотребления комбинационных блоков заказных КМОП СБИС на основе логического моделирования // Современная электроника. 2010. № 2. С. 54—59.
23. **Eldo** User's Manual, Software Version 6.6_1 Release 2005.3. Mentor Graphics Corporation, 2005. URL: http://www.engr.uky.edu/~elias/tutorials/Eldo/eldo_ur.pdf.
24. **Закревский А. Д.** Минимизация перебора ориентированных пар // Танаевские чтения: доклады Четвертой Междунар. науч. конф. Минск, 29—30 марта 2010 г. Минск: ОИПИ НАН Беларуси, 2010. С. 58—62.

УДК 004.932.2; 004.942

М. Г. Казаков, аспирант, Алтайский государственный технический университет им. И. И. Ползунова, г. Барнаул,
e-mail: mike.kazakov@gmail.com

Повышение качества извлечения 3D-геометрии методом семантического анализа изображений

Предложен метод улучшения качества извлечения 3D-геометрии объектов путем анализа входных изображений на семантическом уровне с использованием сегментирования изображений и корреляции полученных сегментов с помощью аффинных преобразований. Приведены описание шагов метода, визуализация этапов, значения выбранных констант и графики их влияния на поведение системы.

Ключевые слова: *компьютерное зрение, 3D-реконструкция, сегментирование изображений, аффинные преобразования, дескрипторы опорных точек*

Введение

Среди различных областей в сфере ИТ можно выделить компьютерное зрение, как одну из наиболее динамично развивающихся в последние два десятилетия. Компьютерное зрение — достаточно обширная дисциплина, которая в целом занимается анализом информации, зафиксированной различными растровыми датчиками. Высокие темпы развития обусловлены как стремительным ростом производительности вычислительных систем, так и разработкой сопутствующего математического аппарата. Если в 1980-х гг. компьютерному зрению были доступны лишь простейшие операции с информационной нагрузкой, поступающей от растровых изображений, то уже в 1990-х гг. стало возможно решение таких комплексных задач, как извлечение трехмерной геометрии объектов из набора их изображений. Эта задача получила название *Structure from Motion* (SfM), она имеет набор устойчивых методов [1], позволяющих получать информацию о геометрии объектов с некоторой степенью приближения. Специфика SfM заключается в достаточно больших допустимых значениях как изменений положений и углов поворота камер, так и изменений яркостных и контрастных свойств. Эта специфика дистанцирует SfM от задачи *Stereo Correspondence*, которая традиционно решается методом *Graph Cuts* [2]. Вместе с тем в настоящее время нельзя сказать, что эта задача решена в полном объеме. Те или иные комбинации разных подходов используют для решения различных, обладающих определенной спецификой задач и условий, которые позволяют добиться лучшего результата. Решение задачи SfM в полном объеме упирается в необходимость имитации интеллектуальных процессов, протекающих в живом организме.

Наиболее распространенный и традиционно используемый метод получения информации о геометрии объектов по их изображению представляет собой последовательность алгоритмов, которые можно описать следующим образом:

1) поиск устойчивых опорных точек, которые повторяются на изображениях объекта под другим ракурсом;

2) вычисление дескрипторов этих точек;

3) попарное сравнение дескрипторов для поиска соответствий в изображениях;

4) построение модели на основе эпиполярных ограничений, отбрасывание неверных соответствий и последующая триангуляция точек в пространстве;

5) добавление новых изображений объекта для уточнения геометрии, параллельное решение задачи глобальной оптимизации ошибки обратной проекции полученной структуры для каждого из добавленных видов.

Каждый из этих этапов представлен большим числом различных подходов со своими алгоритмами, особенностями и проблемными вопросами. В контексте настоящей статьи обратим внимание на следующие аспекты.

• Подавляющее большинство методов поиска и описания опорных точек (иногда это совмещенный алгоритм) оперируют с изображением в градациях серого. Это обстоятельство связано со спецификой соответствующего математического аппарата и моделей для обработки сигналов и с простотой их программной реализации. Исследования по поиску решений для обработки цветных изображений ведутся [3], однако на настоящее время предлагаются довольно громоздкие варианты, малопригодные для реального использования. В результате при решении задачи используется не вся доступная исходная информация (полученная с датчиков фотоаппаратов). В данной работе делается попытка использования цветного изображения наряду с оттенками серого.

• Качественный результат алгоритмов поиска соответствий дескрипторов в парах изображений может ухудшиться при увеличении объема входных данных. Для иллюстрации — при работе с изображениями размером 4000 × 4000 пикселей алгоритмы могут выдать худшие результаты, нежели при работе с изображениями размером 1000 × 1000 пикселей. Это связано с увеличением числа опорных точек и их дескрипторов и последующим увеличением неоднозначности при поиске соответствий дескрипторов. Этот вопрос можно решить введением более агрессивной фильтрации опорных точек (например, по минимальной контрастности или максимальной кучности), либо повышением требований критериев при поиске соответствий дескрипторов друг другу (это приводит к фильтрации, но на более позднем этапе), что так или иначе не позволяет использовать весь потенциально доступный спектр дескрипторов.

В качестве варианта решения указанных выше задач возможно сочетание двух подходов — классического (основанного на опорных точках и их дескрипторах) и квазисемантического анализа исходного цветного изображения. Используя методы контуризации и сегментирования изображений можно извлечь из изображения более высокоуровневую информацию, нежели просто множество дескрипторов опорных точек. Она может представлять собой, например, фрагменты изображения, полученные с помощью устойчивых алгоритмов сегментирования. Несмотря на то что задача сегментирования в общем случае также не решена, информация о сегментах может быть использована в целях снижения вычислительной сложности метода при росте размерности. В настоящей работе предложено использование сегментирования входных изображений для повышения качества работы SfM в целом.

Описание метода

Общая идея метода заключается в том, чтобы разбить исходные изображения на сегменты с использованием устойчивых алгоритмов для извлечения примерно одинаковых объектов. С некоторой долей приближения можно утверждать, что при различных ракурсах объектов изображения будут схожим образом сегментированы. При этом даже если один и тот же объект попадет в разные сегменты (или же будет сегментирован на различное число частей), на работе метода это не скажется негативно. После получения разбиений изображений на сегменты проводится их сопоставление путем подбора аффинного преобразования, которое могло бы трансформировать координаты опорных точек одного сегмента в координаты опорных точек другого. Такой подход имеет право на существование, так как при решении задач SfM ракурсы изображений отличаются незначительно — не более 30...40° наклона плоскости, на этой же предпосылке основываются алгоритмы извлечения дескрипторов опорных точек. В результате остаются только те опорные точки, которые при изменении положения камеры не приводят к появлению несопоставимых проекций. Метод работает либо с планарными объектами (сегментами), либо с такими изменениями камеры, которые не приводят к существенной модификации проекции объекта, в противном же случае эти опорные точки будут проигнорированы.

После того как выбраны соответствия сегментов и проведена верификация их правильности, опорные точки из этих сегментов поступают в последующие шаги SfM в качестве найденных соответствий изначальных изображений. При этом они являются верными с вероятностью близкой к единице, что значительно улучшает показатели дальнейших статистических алгоритмов. Кроме того, такой способ позволяет увеличить детализированность полученной в итоге геометрии, так как предоставляет большее число соответствий точек, нежели в случае с классическим подходом. Последовательность работы метода:

1. Сегментирование входного изображения.
2. Поиск опорных точек.
3. Извлечение дескрипторов.
4. Поиск соответствия дескрипторов.
5. Построение моделей аффинных преобразований.
6. Верификация моделей преобразований на согласованность.
7. Финальная верификация.

Этапы обработки в данной работе проиллюстрированы на одном изображении, что позволяет качественно оценить эффективность предлагаемой методики. Метод тестировался на изображениях с различной структурой и позволял получить схожие результаты. При тестировании на изображениях с разными качественными характеристиками, приведенные ниже константы не требовали изменений, либо были незначительно изменены.

Сегментирование изображений и извлечение дескрипторов

Для разбиения входных изображений на сегменты использован алгоритм *MeanShift*. Основная цель — получение какого-либо разбиения изображения на сегменты, содержащие объекты, чтобы имело смысл аффинное преобразование одних сегментов в другие. Размер сегментов при этом должен быть достаточен,

чтобы обеспечить возможность извлечения дескрипторов в количестве, позволяющем идентифицировать соответствие сегментов и их взаимное положение. На практике такому условию удовлетворяют сегменты размером 100 × 100 пикселов и более. На рис. 1 (см. третью сторону обложки) показаны исходное изображение и пример его разбиения. Для извлечения опорных точек использован алгоритм *Scale-invariant Feature Transform* (SIFT) [4], показавший себя одним из наиболее устойчивых к различным искажениям (из доступных). Ключевой особенностью этого метода является возможность работать с изображением в различных разрешениях. Он отличается инвариантностью к повороту изображения и масштабированию, существенному диапазону аффинных преобразований, наличию шума и изменению освещенности. Существуют другие методы поиска и извлечения дескрипторов этого класса (RIFT, G-RIF, SURF, PCA-SIFT и GLOH), которые могут незначительно превосходить SIFT по характеристикам в определенных ситуациях, однако SIFT наиболее удобен в использовании, так как является стандартом де-факто [5]. На рис. 2 (см. третью сторону обложки) изображен пример полученных дескрипторов опорных точек.

Попарное сравнение сегментов

После извлечения дескрипторов из всех сегментов проводится поиск соответствий опорных точек изображений. Так как на этом этапе каждый сегмент первого изображения теоретически может соответствовать каждому сегменту второго изображения, выполняется попарное сравнение дескрипторов для каждой возможной пары сегментов. Проводится $N \times M$ перекрестных сравнений множеств дескрипторов с поиском соответствий, где N и M — число сегментов в первом и втором изображениях соответственно. Так как дескрипторы SIFT представляют собой 128-мерные вектора и в каждом сегменте могут быть сотни опорных точек, а для сравнения используется мера евклидового пространства, подходы с полным перебором не представляются реализуемыми на практике. В данной реализации поиск оптимизируется с использованием алгоритма Nene-Nayar [6], который ориентирован на нахождение ближайших значений в пространствах с высокой размерностью. Он позволяет свести поиск ближайшего элемента во всем множестве к поиску среди элементов, находящихся в ε-окрестности, ограниченной гиперкубом. При этом вместо метода выбора соответствий, предложенного D. G. Lowe [4]: $L_1/L_2 < 0{,}8$, где L_1 и L_2 — это расстояния до ближайшего и второго ближайшего элементов соответственно (этот метод был предложен им для фильтрации неоднозначных соответствий), используется метод взаимного выбора элементов баз данных при перекрестном поиске ближайших элементов. Такой подход требует повторного прохождения базы данных (поиск ближайших элементов для каждого элемента первой базы и такой же

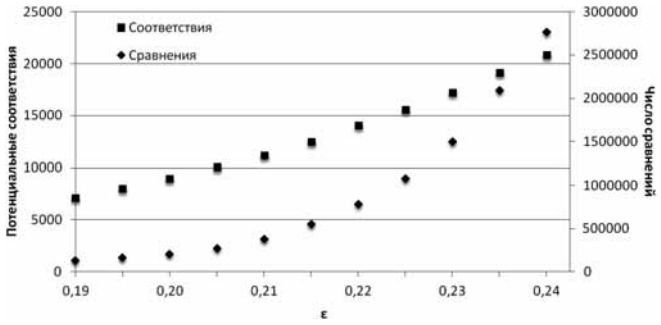

Рис. 3. Зависимость результатов поиска от значения ε

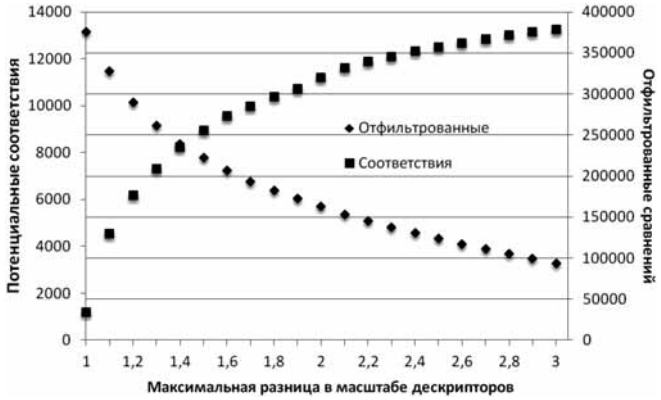

Рис. 4. Влияние ограничения масштаба на поиск

поиск для каждого элемента второй базы), однако позволяет получить большее число соответствий.

В качестве значения ε было использовано 0,21, как компромисс в балансе между полнотой поиска с одной стороны и производительностью системы с другой стороны. Эта зависимость показана на рис. 3. При этом следует понимать, что при увеличении числа соответствий, полученных из пары изображений, процент ложных срабатываний также возрастает.

Для дополнительной фильтрации соответствий, лишенных смысла, и для повышения производительности, вводится ограничение по масштабу, на котором была получена опорная точка и на котором был рассчитан дескриптор. Учитывая использование аффинной модели для последующего установления соответствий сегментов, было использовано ограничение разницы размера дескрипторов в 2 раза, т. е. те дескрипторы, размер которых отличается более чем в 2 раза, не рассматриваются как возможные соответствия. График поведения такой фильтрации представлен на рис. 4. Этот подход подразумевает, что настройки камеры не менялись, либо изменялись в ограниченном диапазоне, в моменты снятия разных ракурсов объектов. Если же такое условие удовлетворить невозможно, эта фильтрация должна быть отключена. Пример соответствий дескрипторов двух сегментов приведен на рис. 5 (см. третью сторону обложки).

Построение модели аффинного преобразования

На этом этапе осуществляется попытка поиска аффинного преобразования координат дескрипторов, имеющих соответствие для каждой пары сегментов первого и второго изображений. Такое преобразование должно перевести координаты дескрипторов сегмента первого изображения в координаты соответствующих дескрипторов сегмента второго изображения. Для этого используется статистическая модель RANSAC [7], суть которой заключается в том, чтобы N раз выбрать случайные базовые элементы, по которым будет построена максимально приближенная модель преобразования. Минимально необходимое число базовых элементов для поиска гомографии — четыре точки, но тесты показали гораздо более стабильные результаты при использовании пяти точек. Преобразование находится путем решения системы уравнений, приведенных к однородному виду, с использованием сингулярного разложения (*SVD*) [1]:

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{H}} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$SVD(\mathbf{A}^{\mathrm{T}}\mathbf{A}) = \mathbf{UDU}^{\mathrm{T}}.$$

Вектор (матрица, записанная в виде вектора) **H** будет равен столбцу **U**, соответствующему минимальному значению диагональной матрицы **D**, т. е. минимальному собственному значению. После того как построена матрица преобразования, все соответствия опорных точек проверяют на согласованность с ограничением модели: $|\mathbf{X}' - \mathbf{H}^*\mathbf{X}| < d$. Для этого вводится граничное значение погрешности преобразования координат $d = 0{,}03$, с учетом того, что координаты точек нормированы в рамках [−1...1]. Те соответствия, координаты которых после преобразования отличаются от целевых менее чем на максимальную погрешность, считаются удовлетворяющими модели (*inliners*), остальные считаются неверными (*outliners*). Влияние размера максимальной погрешности модели на общую суммарную погрешность показано на рис. 6. После всех

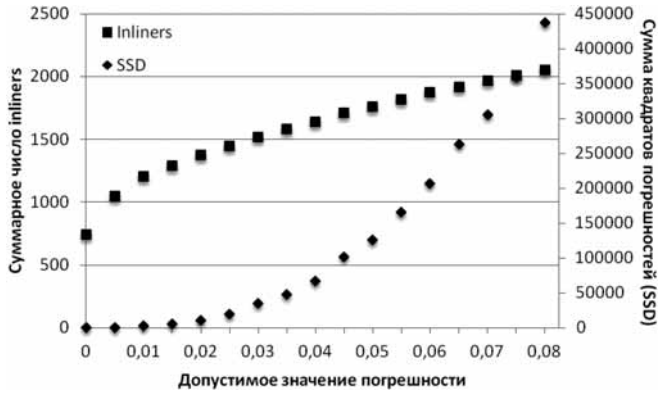

Рис. 6. Влияние максимальной погрешности

бросков кубиков выбирается тот вариант, который имел наибольшее число удовлетворяющих модели соответствий, и он принимается как потенциальное преобразование одного сегмента в другой. На рис. 7 (см. четвертую сторону обложки) дан пример удовлетворяющих модели соответствий дескрипторов.

Верификация соответствия сегментов

Самой нетривиальной задачей в предложенной системе является верификация того, что полученная модель аффинного преобразования имеет смысл, т. е. с большой долей вероятности опорные точки одного сегмента могут иметь возможное преобразование в опорные точки несоответствующего сегмента. Это связано с тем, что дескрипторы не могут однозначно идентифицировать место в изображении, а являются, по сути, хешированием в общем смысле этого слова. Несмотря на то что в базе метода RANSAC заложено то, что неверные соответствия не смогут между собой "договориться" об удовлетворяющей их модели, на практике такая ситуация может возникать и требует решения. На рис. 8 (см. четвертую сторону обложки) показан пример не имеющего смысла преобразования сегментов одного в другой, при этом связями показаны соответствия, которые смогли попасть в общую модель (*inliners*).

Автором была проведена экспериментальная проверка вариантов нивелирования этого эффекта и отсечения неверных соответствий без учета взаимного расположения опорных точек, ниже представлены основные из них:

- На этапе сравнения дескрипторов ввод более агрессивного ограничения на максимальное эвклидово расстояние между ними.
- Фильтрование модели по числу соответствий, удовлетворяющих преобразованию.
- Ввод ограничений на параметры аффинного преобразования, если наилучший вариант не попадает под эти критерии, считать соответствие не имеющим смысла.
- Ввод ограничения на суммарную погрешность при преобразовании попавших в модель опорных точек.

- Фильтрование на основе минимального количества *inliners*, или же на основе максимального значения $R = inliners/(inliners + outliners)$;
- Использование в качестве критерия среднего расстояния между точками, либо же их дисперсию.

Как показала практика, все подобные варианты не имеют удовлетворяющих характерик — они либо пропускают ложные соответствия, либо отсекают верные. Для решения этой задачи автором был разработан способ, базирующийся на преобразовании Hough [8]. Суть его заключается в том, чтобы закодировать взаимное расположение опорных точек в первом сегменте и затем раскодировать его, используя координаты второго сегмента. Дескрипторы SIFT (помимо непосредственно 128-мерного вектора) содержат такую информацию, как расположение (X, Y), его масштаб (S — scale) и угол поворота (α): $\mathbf{P} = \{X, Y, S, \alpha\}$.

За основу пересчета принята такая величина, как центр массы опорных точек, так как при невырожденных аффинных преобразованиях взаимное расположение центра масс и опорных точек останется тем же самым. Поэтому, если пересчитать каждую точку первого сегмента во взаимное положение относительно центра масс, и потом сделать обратный расчет для точек второго сегмента, можно получить некую меру согласованности. В идеальном случае, при отсутствии шумов, после обратного преобразования должна получиться координата центра масс второго сегмента (если соответствие первого и второго сегментов верно). В реальности, разумеется, имеет место дисперсия рассчитанных центров масс, но это не мешает использованию такой меры для оценки согласованности точек между собой.

Суть преобразования заключается в пересчете каждой опорной точки из четырех параметров в два:

$\mathbf{P} = \{X, Y, S, \alpha\} => \mathbf{P} = \{\theta, D/S\}$, где θ — угол между вектором дескриптора и вектором, соединяющим центр дескриптора и центр масс, а D/S — отношение расстояния между центром дескриптора и центром масс и масштабом дескриптора. Взаимное расположение дескрипторов и центров масс показано на рис. 9, при этом $\mathbf{P}' = \{X', Y', S', \alpha'\}$ — соответствующий дескриптор в другом изображении.

Для каждой точки второго сегмента проводится обратный расчет центра масс:

$$\mathbf{M}'' = \begin{bmatrix} X' + \cos(\theta + \alpha') + \dfrac{D}{S}S' \\ Y' + \sin(\theta + \alpha') + \dfrac{D}{S}S' \end{bmatrix}.$$

После того как получены координаты новых центров масс, можно оценить их скученность, тем самым определить согласованность взаимных положений опорных точек. Это можно делать множеством различных способов — как вводом общей меры дисперсии, так и оценкой каждой точки в отдельности. В рамках данного исследования наилучшим образом показал себя следующий вариант оценки каждой точки:

$D = \|\mathbf{M}' - \mathbf{M}''\|$, где $\mathbf{M}'$ — центр масс второго сегмента, рассчитанный прямым образом;

$D < F(S')$, где $F(S') = 2L \times S'$, L — сторона окна дескриптора на единичном приближении.

Таким образом, коррелируется погрешность обратного вычисления центра масс и масштаба дескриптора, для которого был проведен расчет, это позволяет добиться инвариантности относительно масштаба. При этом лучших результатов удалось достичь, когда данная проверка согласованности проводилась прямо в цикле RANSAC при составлении модели аффинного преобразования, и использовалась для фильтрации *inliners*/*outliners*, а не была следующим шагом после уже полученной модели преобразования. Взаимное расположение полученных центров масс $\mathbf{M}''$ в различных случаях показано на рис. 10 (см. четвертую сторону обложки).

Финальной фильтрацией служит проверка на минимальное число *inliners*, удовлетворяющих и модели аффинного преобразования и верификации согласованности, оно принято равным 7. Иллюстрация зависимости верных и ошибочных принятий решений показана на рис. 11.

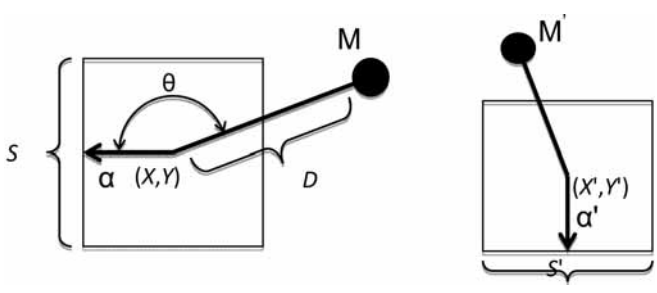

Рис. 9. Информация для преобразования дескрипторов:
M и **M′** — центры масс дескрипторов в первом и втором сегментах соответственно

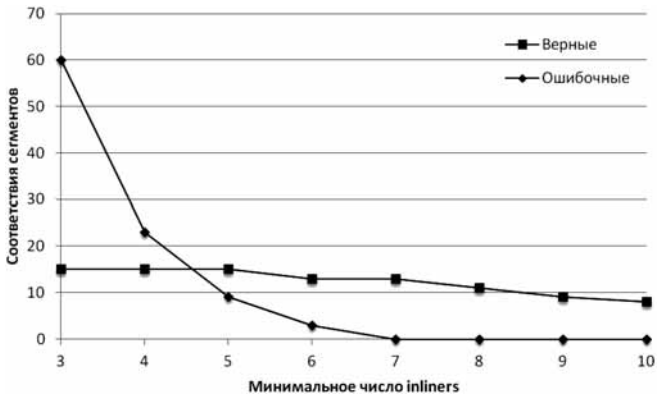

Рис. 11. Влияние минимального числа inliners на верность принятия решений

Заключение

Представленный метод позволяет получить большее число соответствий в изображениях при решении задачи SfM, нежели при поиске без декомпозиции. Это достигается снижением размерностей, в которых проводится извлечение опорных точек и сравнение дескрипторов, что позволяет уменьшать неоднозначность поиска, и как следствие, использовать те точки, которые в противном случае были бы отброшены как неоднозначные. Можно выделить следующие достигнутые результаты:

- увеличение числа соответствий между изображениями в высоком разрешении, различные тесты давали разницу от десятков до сотен процентов;
- увеличение процента соответствий, удовлетворяющих эпиполярным органичениям, в случае с соизмеримым числом опорных точек, что обусловлено дополнительной проверкой связности;
- улучшение детализированности 3D-геометрии, которая формируется на последующих шагах SfM.

Список литературы

1. **Hartley R., Zisserman A.** Multiple View Geometry in Computer Vision. Cambridge university press, 2004. 673 p.

2. **Szeliski R.** Computer Vision: Algorithms and Applications. Springer-Verlang New York, 2010. 832 p.

3. **Abdel-Hakim A. E., Faraq A. A.** CSIFT: A SIFT Descriptor with Color Invariant Characteristicts // IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2006. Vol. 2. P. 1978—1983.

4. **Lowe D. G.** Distinctive image features from scale-invariant keypoints // International Journal of Computer Vision. 2004. N 60 (2). P. 91—110.

5. **Mikolajczyk K., Schmid C.** A performance evaluation of local descriptors // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2005. N 27 (10). P. 1615—1630.

6. **Nene S. A., Nayar S. K.** A simple algorithm for nearest neighbor search in high dimensions // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1997. N 19 (9). P. 989—1003.

7. **Fischler M., Bolles R.** Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography // Communications of the ACM. 1981. N 24 (6). P. 381—395.

8. **Ballard D. H.** Generalizing the Hough transform to detect arbitrary patterns // Pattern Recognition. 1981. N 13 (2). P. 111—122.

ИНФОРМАЦИЯ

8—10 ноября 2013 г. в г. Львов пройдет Четырнадцатая Международная конференция в области обеспечения качества ПО
"Software Quality Assurance Days"

Конференция охватит широкий спектр профессиональных вопросов в области обеспечения качества, ключевыми из которых являются:

- методики и инструменты тестирования ПО;
- автоматизация тестирования ПО;
- подготовка, обучение и управление командами тестировщиков;
- процессы обеспечения качества в компании;
- управление тестированием и аутсорсинг;
- совершенствование процессов тестирования и инновации.

Предыдущая 13-я конференция проходила в Санкт-Петербурге, ее участниками стали более 600 профессионалов. Организатором традиционно выступает компания "Лаборатория тестирования". (http://www.sqalab.ru/)

Обращаем внимание, что 10 ноября пройдет дополнительный день SQA Days English Day, на котором будут представлены доклады на английском языке. Это отдельное событие в рамках конференции. Количество мест на этот день будет ограничено.

Сайт конференции: http://sqadays.com/index-news.sdf/sqadays/sqa_days14

CONTENTS

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4

Дизайнер *Т. Н. Погорелова*. Технический редактор *Е. М. Патрушева*. Корректор *Е. В. Комиссарова*

Сдано в набор 06.06.2013 г. Подписано в печать 22.07.2013 г. Формат 60×88 1/8. Заказ PI813
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
105120, г. Москва, ул. Нижняя Сыромятническая, д. 5/7, стр. 2, офис 2.