ISSN 2220-3397

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

ПрИн
8
2022
Том 13

XX НАЦИОНАЛЬНАЯ КОНФЕРЕНЦИЯ ПО ИСКУССТВЕННОМУ ИНТЕЛЛЕКТУ С МЕЖДУНАРОДНЫМ УЧАСТИЕМ (КИИ-2022)

Москва, с 21 по 23 декабря 2022 г.

Информационная поддержка **ООО «Лаборатория информационных технологий»** (Россия, Смоленск)

Конференция проводится **Российской ассоциацией искусственного интеллекта** (РАИИ) совместно с **Федеральным исследовательским центром «Информатика и управление» РАН**, **Национальным исследовательским университетом «МЭИ»** (НИУ «МЭИ»), **Московским физико-техническим институтом (национальным исследовательским университетом)** в НИУ «МЭИ».

В качестве пленарных докладчиков будут приглашены ведущие ученые в области искусственного интеллекта. В программе конференции предусматривается работа секций и лекции.

Сопредседатели конференции

Соколов И.А., акад. РАН, ФИЦ ИУ РАН, Москва
Рогалёв Н.Д., д.т.н., проф., НИУ «МЭИ», Москва

Основные направления конференции

- Инженерия знаний
- Интеллектуальный анализ данных
- Интеллектуальный анализ текстов и семантический Web
- Когнитивные и психологические исследования в искусственном интеллекте
- Моделирование рассуждений и неклассические логики
- Нечеткие модели и мягкие вычисления
- Интеллектуальные системы поддержки принятия решений и управления
- Многоагентные системы и искусственные сообщества
- Робототехнические системы
- Нейросетевые методы, нейроинформатика
- Эмоции и образы в искусственном интеллекте
- Компьютерное зрение
- Объясняемый искусственный интеллект в критических приложениях
- Инструментальные средства конструирования интеллектуальных систем
- Интеллектуальные технологии и прикладные интеллектуальные системы

Основные контакты по вопросам подачи докладов: Виноградов Д.В., krrguest@yandex.ru.
По общим вопросам: Борисов В.В., vbor67@mail.ru

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

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

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 подписных агентствах (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции (для юридических лиц).
Тел.: (499) 270-16-52.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в Российский индекс научного цитирования (РИНЦ) и Russian Science Citation Index (RSCI).
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.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 13
N 8
2022

Published since September 2010 **DOI 10.17587/issn.2220-3397** **ISSN 2220-3397**

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 519.714
DOI: 10.17587/prin.13.363-382

П. Н. Бибило, д-р техн. наук, проф., зав. лаб., bibilo@newman.bas-net.by,
Объединенный институт проблем информатики Национальной академии наук Беларуси, Минск

Аппаратная реализация преобразователей кодов, предназначенных для сокращения длины двоичных кодируемых слов

Рассмотрено аппаратное решение задачи проектирования кодовых преобразователей, предназначенных для сокращения длины слов из заданного набора кодируемых двоичных слов. Кодирование предполагает, что различные двоичные слова (наборы бит) будут закодированы различными двоичными кодами меньшей длины (разрядности). Предлагаемые способы решения данной задачи основаны на составлении и логической минимизации таких форм систем не полностью определенных булевых функций, как дизъюнктивные нормальные формы и бинарные диаграммы решений, называемые BDD-представлениями (BDD — Binary Decision Diagram). Минимизация составляемых функциональных описаний ориентирована на уменьшение аппаратной сложности комбинационных схем в базисе библиотечных элементов заказных СБИС либо программируемых элементов FPGA, реализующих преобразователи кода рассматриваемого класса.

Ключевые слова: *преобразователь кода, система булевых функций, дизъюнктивная нормальная форма (ДНФ), Binary Decision Diagram (BDD), разложение Шеннона, синтез логической схемы, VHDL, СБИС*

Введение

В цифровых вычислительных и управляющих системах широкое распространение получили преобразователи кодов — устройства, предназначенные для преобразования двоичного кода из одной формы в другую. Для представления информации используют разнообразные двоичные и двоично-десятичные коды чисел: прямой; обратный; дополнительный и их модификации [1]; унитарные коды; двоичные коды чисел по модулю [2] и т. д. Проектирование таких преобразователей кодов опирается на известные формы задания входной и выходной информации, т. е. кодируемых двоичных слов и кодов (кодирующих слов). Однако при проектировании цифровых систем могут возникать сложности передачи по шинам данных длинных двоичных слов, т. е. таких, разрядность которых превышает разрядность шины данных. Например, по 16-разрядной шине данных требуется передавать 18-разрядные или 17-разрядные слова. Конечно, каждое такое слово можно передать за два такта функционирования системы, однако такой подход снижает общее быстродействие всей системы. Одним из подходов для разрешения таких сложностей является разработка комбинационных схем, которые осуществляют преобразование длинных двоичных кодируемых слов в более короткие. При получении информации, переданной по шине данных, может потребоваться обратный преобразователь, осуществляющий преобразование переданного слова в исходную форму.

В настоящей работе рассмотрено решение задачи логического проектирования кодовых преобразователей, предназначенных для сокращения длины каждого слова из заданного набора кодируемых двоичных слов. Кодирование предполагает, что различные двоичные слова (наборы бит) одинаковой длины будут закодированы различными двоичными кодами меньшей длины (разрядности). Естественно, коды также должны иметь одинаковую длину. Предлагаемые способы решения задачи основаны на составлении и логической минимизации различных форм функционального описания проектируемых преобразователей кода, в качестве таких форм выступают дизъюнктивные нормальные формы (ДНФ) [3] и бинарные диаграммы решений, называемые BDD-представлениями [4]. Минимизация функциональных описаний ориентирована на уменьшение аппаратной сложности

преобразователей кода и существенным образом использует такие модели функционирования проектируемых цифровых устройств, как не полностью определенные двоичные (булевы) функции и многозначные функции, зависящие от булевых переменных. Уменьшение сложности однотактных комбинационных схем, реализующих функции преобразователей кодов, и является основной целью логического проектирования преобразователей кодов рассматриваемого класса.

1. Определения и постановка задачи

Булевыми называются двоичные (0, 1) функции $f(\boldsymbol{x}) = f(x_1, x_2, ..., x_n)$ двоичных (булевых) переменных $x_1, x_2, ..., x_n$. Пусть V^x — булево пространство, построенное над переменными булева вектора $\boldsymbol{x} = (x_1, ..., x_n)$. Элементами этого пространства являются n-компонентные наборы (векторы) $\boldsymbol{x}^*$ нулей и единиц. Булева функция, значения 0, 1 которой определены на всех элементах $\boldsymbol{x}^* \in V^x$, называется *полностью определенной*. Если на некоторых элементах булева пространства V^x значения функции не определены, то такая функция называется не полностью определенной, или *частичной*. Частичная булева функция принимает единичное значение на элементах $\boldsymbol{x}^*$ подмножества M_f^1 булева пространства V^x и нулевое значение на элементах подмножества M_f^0. На всех остальных элементах пространства V^x, образующих подмножество M_f^- пространства V^x, значения частичной функции не определены и имеют значение "–". Частичная функция f_1 *реализуется* частичной (либо полностью определенной) функцией f_2 (обозначается $f_1 \prec f_2$, $f_2 \succ f_1$), если $M_{f_1}^1 \subseteq M_{f_2}^1$, $M_{f_1}^0 \subseteq M_{f_2}^0$. Если функция f_2 *реализует* f_1 $(f_1 \prec f_2)$, то функцию f_2 называют *доопределением* функции f_1. Для пары f_1, f_2 полностью определенных булевых функций отношение реализации является отношением *равенства*. Под *векторной* булевой функцией $\boldsymbol{f}(\boldsymbol{x})$ будем понимать упорядоченную систему частичных булевых функций $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))$, значениями векторных функций на элементах $\boldsymbol{x}^*$ булева пространства являются m-компонентные троичные векторы $\boldsymbol{f}(\boldsymbol{x}^*)$.

Задача 1. Пусть задана булева (двоичная) матрица B, имеющая k различных строк b^i и n столбцов, при этом $k \leqslant 2^{n-1}$. Требуется закодировать строки b^i матрицы B различными минимальными по длине m булевыми векторами (строками) c^i, чтобы комбинационная логическая схема, осуществляющая преобразование строк b^i в строки c^i, имела возможно меньшую сложность.

Понятие сложности схем зависит от используемого технологического базиса и будет уточнено далее.

Пусть кодирующие комбинации c^i, соответствующие строкам матрицы B, образуют булеву матрицу C. Другими словами, для булевой матрицы B требуется получить булеву матрицу C, содержащую k различных строк и m столбцов, где $m = \lceil \log_2 k \rceil$, через $\lceil a \rceil$ обозначается ближайшее целое число, большее либо равное a.

Назовем задачу 1 задачей синтеза схемы кодового преобразователя (либо преобразователя кодов).

Обозначим столбцы булевой матрицы B через $x_1, x_2, ..., x_n$, столбцы матрицы C — через $c_1, c_2, ..., c_m$. Пара матриц (B, C) задает *частичную* векторную булеву функцию $\boldsymbol{c}(\boldsymbol{x}) = (c_1(\boldsymbol{x}), ..., c_m(\boldsymbol{x}))$, $\boldsymbol{x} = (x_1, x_2, ..., x_n)$, значениями данной векторной функции на двоичных наборах из матрицы B являются m-компонентные *двоичные* наборы (строки) из матрицы C. На наборах, принадлежащих множеству $V^x \backslash B$, значения векторной функции $\boldsymbol{c}(\boldsymbol{x})$ не определены — соответствующие *троичные векторы* значений компонентных функций состоят только из неопределенных значений "–".

Перейдем к обсуждению постановки задачи 1.

Во-первых, для решения задачи синтеза любой логической схемы (не только схемы кодового преобразователя) требуется указать базис синтеза (набор логических элементов), часто называемый также технологической библиотекой синтеза [5].

Во-вторых, логические схемы могут быть однотактными (срабатывать в течение одного такта функционирования дискретной системы) либо иметь конвейерную организацию и осуществлять требуемое преобразование набора значений входных сигналов в значения выходных сигналов за несколько тактов смены значений синхросигнала.

В-третьих, синтез схем осуществляется традиционно в два этапа: на первом этапе выполняется технологически независимая оптимизация, на втором — технологическое отображение (*technology mapping*), когда оптимизированные логические описания булевых функций покрываются функциональными описаниями библиотечных элементов [4, 5].

В-четвертых, будем считать, что на вход синтезированной логической схемы при ее функционировании могут поступать только входные наборы из матрицы B (т. е. наборы из множества $V^x \backslash B$ никогда не должны поступать на вход схемы), что позволит доопределять исходную частичную векторную функцию до полностью определенной произвольным образом.

Примем следующие *соглашения* (ограничения). Пусть библиотекой является библиотека логических элементов заказных КМОП СБИС, описанная в работе [6], системой синтеза логических схем — синтезатор LeonardoSpectrum [5], а программами технологически независимой оптимизации —

программы системы FLC2 логической оптимизации функциональных описаний систем булевых функций [7]. Синтез будет выполняться в целях получения однотактных (не конвейерных) реализаций схем. Синтезатор LeonardoSpectrum после синтеза схемы подсчитывает сложность (площадь) схем из библиотечных элементов как сумму площадей всех логических элементов схемы и выдает значение данного параметра под названием *Area* (площадь). Заметим, что задержка синтезированной схемы также вычисляется в виде значения параметра *Delay*.

Решение задачи 1 сводится к последовательному решению описанных далее взаимосвязанных задач 2—4. Критерием оптимизации при решении задач 2 и 3 является сложность функционального описания системы ДНФ либо сложность BDD-представления, задающих функции кодового преобразователя. Критерием оптимизации при решении задачи 4 является сложность комбинационной схемы кодового преобразователя. Для промышленных синтезаторов логических схем уменьшение сложности (либо задержки) результирующей логической схемы может достигаться установкой соответствующих управляющих опций синтезатора, в нашем случае LeonardoSpectrum.

Задача 2. Найти кодирующую матрицу C и, возможно, доопределение частичной векторной функции $\boldsymbol{c}(\boldsymbol{x}) = (c_1(\boldsymbol{x}), ..., c_m(\boldsymbol{x}))$ на наборах из множества $V^x \backslash B$, получить неоптимизированное представление функций $c_1(\boldsymbol{x}), ..., c_m(\boldsymbol{x})$ кодового преобразователя в виде ДНФ либо СДНФ (совершенной ДНФ).

Доопределение предполагает замену троичных векторов — значений функции $\boldsymbol{c}(\boldsymbol{x})$, состоящих только из неопределенных значений "–" полностью определенными (0, 1) двоичными векторами. В этом случае векторная функция $\boldsymbol{c}(\boldsymbol{x}) = (c_1(\boldsymbol{x}), ..., c_m(\boldsymbol{x}))$ является полностью определенной.

Задача 3. Выполнить логическую минимизацию частичной либо полностью определенной векторной функции $\boldsymbol{c}(\boldsymbol{x}) = (c_1(\boldsymbol{x}), ..., c_m(\boldsymbol{x}))$ и получить ее оптимизированное (минимизированное) алгебраическое представление.

Булевы векторы $c(b^i)$ значений функции $\boldsymbol{c}(\boldsymbol{x}) = (c_1(\boldsymbol{x}), ..., c_m(\boldsymbol{x}))$ на наборах b^i являются *кодами строк b^i матрицы B*. Для обратного преобразования в роли исходной матрицы B выступает матрица C. Пара матриц (C, B) задает систему булевых функций, которую надо реализовать комбинационной схемой обратного преобразования булевых строк матрицы C в булевы строки матрицы B. Частичные векторные булевы функции $\boldsymbol{c}(\boldsymbol{x}) = (c_1(\boldsymbol{x}), ..., c_m(\boldsymbol{x}))$ являются промежуточными математическими моделями при нахождении кодов строк матрицы B.

Решение задачи 3 будем выполнять описанными далее двумя программами системы FLC-2 [7].

Первая из них — программа Minim раздельной минимизации системы частичных либо полностью определенных функций в классе ДНФ, т. е. каждая из компонентных функций $c_1(\boldsymbol{x}), ..., c_m(\boldsymbol{x})$ будет минимизироваться независимо от других. Результат работы программы Minim — минимизированная система ДНФ, задающая полностью определенные функции кодового преобразователя.

Вторая программа — программа BDD_Builder совместной минимизации многоуровневых BDDI-представлений систем полностью определенных булевых функций [8]. Результат работы программы BDD_Builder — взаимосвязанные логические уравнения (формулы разложений Шеннона), задающие полностью определенные функции кодового преобразователя.

Результатом решения задачи 3 является минимизированное представление полностью определенной векторной функция $\boldsymbol{c}(\boldsymbol{x})$, являющееся исходными данными для решения задачи 4.

Задача 4. Синтезировать логическую схему в заданном базисе логических элементов по минимизированному представлению векторной функции $\boldsymbol{c}(\boldsymbol{x})$.

2. Способы решения задач 2, 3

Способ 1 (тривиальный). Закодируем строки матрицы B двоичными наборами, задающими их номера (нумерация начинается с нуля), выполним тривиальное (без комбинаторного поиска) доопределение системы частичных функций —заменим неопределенные значения системы функций $\boldsymbol{c}(\boldsymbol{x})$ на наборах из множества $V^x \backslash B$, нулевыми строками. Заметим, что номера строк (коды) можно задавать не по порядку, а ***случайным*** образом размещая их в кодирующей матрице C, однако это не меняет суть способа 1.

Пример 1. Пусть булева матрица B имеет 16 строк и 5 столбцов:

$$B = \begin{array}{c} x_1x_2x_3x_4x_5 \\ \begin{matrix} 0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,1 \\ 0\,1\,0\,1\,0 \\ 0\,1\,1\,0\,0 \\ 1\,0\,0\,0\,1 \\ 1\,0\,1\,1\,0 \\ 1\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1 \\ 0\,0\,1\,1\,0 \\ 0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,1 \\ 1\,0\,0\,1\,0 \\ 1\,0\,1\,0\,0 \\ 1\,1\,0\,0\,1 \\ 0\,0\,0\,1\,0 \\ 0\,0\,1\,0\,0 \end{matrix} \end{array}.$$

Таблица 1

Результаты логической оптимизации в способе 1 (пример 1)

Доопределение "нулевое", коды — номера строк (оптимизация не проведена)		Раздельно минимизированные ДНФ		BDDI-представление
$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	
0 0 0 0 0	0 0 0 0	0 0 − 0 1	0 0 0 1	c1=^x4*s0+x4*s1;
0 0 1 0 1	0 0 0 1	0 1 − 0 0	0 0 0 1	c2=^x4*s2+x4*s3;
0 1 0 1 0	0 0 1 0	1 0 − 1 0	0 0 0 1	c3=^x4*s4+x4*s5;
0 1 1 0 0	0 0 1 1	0 0 1 0 −	0 0 0 1	c4=^x4*s6+x4*s7;
1 0 0 0 1	0 1 0 0	1 1 0 0 0	0 0 1 0	s2=^x3*^s9+x3*s11;
1 0 1 1 0	0 1 0 1	0 0 0 0 1	0 0 1 0	s1=^x3*s11+x3*s13;
1 1 0 0 0	0 1 1 0	0 1 1 0 −	0 0 1 0	s3=^x3*s13+x3*s7;
0 0 0 0 1	0 1 1 1	0 − 0 1 0	0 0 1 0	s5=^x3*s14;
0 0 1 1 0	1 0 0 0	0 0 0 1 0	0 1 0 0	s0=^x3*s15+x3*s16;
0 1 0 0 0	1 0 0 1	− 0 0 0 1	0 1 0 0	s6=^x3*s19+x3*s20;
0 1 1 0 1	1 0 1 0	1 0 1 − 0	0 1 0 0	s4=^x3*s8+x3*s9;
1 0 0 1 0	1 0 1 1	1 1 0 0 −	0 1 0 0	s20=^x2*^x1+x2*s22;
1 0 1 0 0	1 1 1 0	0 1 0 0 0	1 0 0 0	s11=^x2*^x5;
1 1 0 0 1	1 1 0 1	0 1 1 0 1	1 0 0 0	s9=^x2*^x5+x2*^x1;
0 0 0 1 0	1 1 1 0	1 1 0 0 1	1 0 0 1	s14=^x2*^x5+x2*s22;
0 0 1 0 0	1 1 1 1	0 0 − 1 0	1 0 0 0	s16=^x2*^x5+x2*s26;
0 1 0 0 1	**0 0 0 0**	− 0 1 0 0	1 1 1 0	s13=^x2*s22;
0 1 1 1 0	**0 0 0 0**	− 0 0 1 0	1 0 1 0	s19=^x2*s26+x2*s30;
1 0 0 0 0	**0 0 0 0**			s8=^x2*s26+x2*s31;
1 0 1 0 1	**0 0 0 0**			s7=^x2*s31;
1 1 0 1 0	**0 0 0 0**			s15=x2*s30;
0 0 0 1 1	**0 0 0 0**			s22=^x1*^x5;
0 0 1 1 1	**0 0 0 0**			s30=^x1*^x5+x1*x5;
0 1 0 1 1	**0 0 0 0**			s26=^x1*x5;
0 1 1 1 1	**0 0 0 0**			s31=x1*^x5;
1 0 0 1 1	**0 0 0 0**			
1 0 1 1 1	**0 0 0 0**			
1 1 0 1 1	**0 0 0 0**			
1 1 1 0 0	**0 0 0 0**			
1 1 1 0 1	**0 0 0 0**			
1 1 1 1 0	**0 0 0 0**			
1 1 1 1 1	**0 0 0 0**			
Схема 1		**Схема 2**		**Схема 3**

В данном примере $m = \lceil \log_2 16 \rceil = 4$, доопределенные значения на наборах из множества $V^x \backslash B$ выделены в табл. 1 полужирным шрифтом. В табл. 1 в левой части приводится таблица истинности системы полностью определенных булевых функций, данной таблице соответствует система совершенных ДНФ булевых функций, каждой строке матрицы B соответствуют своя полная элементарная конъюнкция. Например, второй строке (0 0 1 0 1) соответствует полная элементарная конъюнкция $\bar{x}_1\bar{x}_2x_3\bar{x}_4x_5$, данная конъюнкция входит в СДНФ только одной компонентной функции c^4. В средней части табл. 1 находится функциональное описание, полученное в результате логической минимизации системы функций из левой части программой Minim. Например, матричная форма минимизированной ДНФ функции c_4 в алгебраической форме записывается в виде $c_4 = \bar{x}_1\bar{x}_2\bar{x}_4x_5 \vee \bar{x}_1x_2\bar{x}_4\bar{x}_5 \vee x_1\bar{x}_2x_4\bar{x}_5 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4$.

В правой части табл. 1 заданы формулы многоуровневого BDDI-описания, полученные программой BDD_Builder. Для упрощения текстовой записи логических уравнений использованы следующие обозначения: символ " + " обозначает дизъюнкцию, "*" - конъюнкцию, "^" - инверсию (отрицание) булевой переменной, которая идет после этого символа. Например, уравнение `c1 = = ^x4*s0 + x4*s1`; (табл. 1) в общепринятой записи имеет вид $c_1 = \bar{x}_4s_0 \vee x_4s_1$.

В нижней строке табл. 1 приводятся номера схем, которые будут синтезированы по соответствующим функциональным описаниям.

Если не проводить нулевое доопределение, приводящее к СДНФ, а рассматривать систему частичных функций, то можно улучшить решение путем раз-

дельной минимизации в классе ДНФ системы частичных функций. Однако на практике такая возможность использования модели частичных булевых функций при логической минимизации обычно игнорируется.

Способ 2. Замена двоичных строк матрицы *B* троичными строками, кодирование строк номерами. В способе 2 каждая двоичная строка матрицы *B* (полная элементарная конъюнкция) заменяется троичной строкой — элементарной конъюнкцией. Получение такой строки сводится к раздельной минимизации специально составленной (по матрице *B*) системы частичных функций в классе ДНФ. Для рассматриваемого примера исходная и результирующая системы функций заданы в табл. 2. По сути, минимизация сводится к расширению единственного набора из области единичных значений функции k_i до интервала за счет наборов из области неопределенных значений этой функции. Например, троичный вектор $k_2 = (-\ 0\ 1\ -\ 1)$ (минимизированная полностью определенная булева функция $k_2 = \bar{x}_2 x_3 x_5$) получается при доопределении неопределенных значений частичной функции k_2 единичными значениями на наборах (0 0 1 1 1), (1 0 1 0 1), (1 0 1 1 1).

Чтобы получить доопределение исходной частичной функции, получаемое в способе 2, доста-

Таблица 2

Расширение наборов до интервалов

Система частичных булевых функций		Минимизированная система полностью определенных булевых функций	
Наборы $x_1x_2x_3x_4x_5$	$k_1\ k_2\ k_3\ k_4 \ldots k_{14}\ k_{15}\ k_{16}$	Интервалы $x_1x_2x_3x_4x_5$	$k_1\ k_2\ k_3\ k_4 \ldots k_{14}\ k_{15}\ k_{16}$
0 0 0 0 0	1000000000000000	0 0 0 0 0	1000000000000000
0 0 1 0 1	0100000000000000	– 0 1 – 1	0100000000000000
0 1 0 1 0	0010000000000000	– 1 – 1 –	0010000000000000
0 1 1 0 0	0001000000000000	– 1 1 – 0	0001000000000000
1 0 0 0 1	0000100000000000	1 0 – – 1	0000100000000000
1 0 1 1 0	0000010000000000	1 – 1 1 –	0000010000000000
1 1 0 0 0	0000001000000000	1 1 – – 0	0000001000000000
0 0 0 0 1	0000000100000000	0 – 0 – 1	0000000100000000
0 0 1 1 0	0000000010000000	0 – 1 1 –	0000000010000000
0 1 0 0 0	0000000001000000	0 1 0 0 –	0000000001000000
0 1 1 0 1	0000000000100000	0 1 – – 1	0000000000100000
1 0 0 1 0	0000000000010000	1 – 0 1 –	0000000000010000
1 0 1 0 0	0000000000001000	1 – 1 0 –	0000000000001000
1 1 0 0 1	0000000000000100	1 1 – – 1	0000000000000100
0 0 0 1 0	0000000000000010	0 0 0 1 –	0000000000000010
0 0 1 0 0	0000000000000001	0 0 1 0 0	0000000000000001
0 1 0 0 1	----------		
0 1 1 1 0	----------		
1 0 0 0 0	----------		
1 0 1 0 1	----------		
1 1 0 1 0	----------		
0 0 0 1 1	----------		
0 0 1 1 1	----------		
0 1 0 1 1	----------		
0 1 1 1 1	----------		
1 0 0 1 1	----------		
1 0 1 1 1	----------		
1 1 0 1 1	----------		
1 1 1 0 0	----------		
1 1 1 0 1	----------		
1 1 1 1 0	----------		
1 1 1 1 1	----------		

Таблица 3

Результаты логической оптимизации в способе 2 (пример 1)

Доопределение по интервалам, коды — номера строк (оптимизация не проведена)		Раздельно минимизированные ДНФ		BDDI-представление
$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	
0 0 0 0 0	0 0 0 0	− 0 1 − 1	0 0 0 1	c1=^x4*s0+x4*s1;
− 0 1 − 1	0 0 0 1	− 1 1 − 0	0 0 0 1	c2=^x4*s2+x4*s3;
− 1 − 1 −	0 0 1 0	1 1 − − 1	0 0 0 1	c3=^x4*s4+x4*s5;
− 1 1 − 0	0 0 1 1	1 − − 1 −	0 0 0 1	c4=^x4*s6+x4*s7;
1 0 − − 1	0 1 0 0	0 0 1 0 −	0 0 0 1	s2=^x1*s11+x1*s12;
1 − 1 1 −	0 1 0 1	− 1 − 1 −	0 0 1 0	s4=^x1*s13+x1*s14;
1 1 − − 0	0 1 1 0	− − 0 1 −	0 0 1 0	s3=^x1*s15+x1*s12;
0 − 0 − 1	0 1 1 1	1 1 − − 0	0 0 1 0	s1=^x1*s17+x1*s18;
0 − 1 1 −	1 0 0 0	0 − 0 − 1	0 0 1 1	s6=^x1*s19+x1*s20;
0 1 0 0 −	1 0 0 1	0 − 1 0 0	0 0 1 0	s7=^x1*s8+x1;
0 1 − − 1	1 0 1 0	0 1 − − 1	0 0 1 0	s0=^x1*s9+x1*s10;
1 − 0 1 −	1 0 1 1	1 − 1 − −	0 1 0 0	s9=^x2*^s22+x2*s22;
1 − 1 0 −	1 1 0 0	− − 0 − 1	0 1 0 0	s5=^x2*^x3+x2;
1 1 − − 1	1 1 0 1	− 0 1 0 0	0 1 0 0	s18=^x2*^x3+x2*s22;
0 0 0 1 −	1 1 1 0	1 1 − − −	0 1 0 0	s15=^x2*^x3+x2*s37;
0 0 1 0 0	1 1 1 1	0 0 0 1 −	0 1 0 0	s13=^x2*s25+x2*s28;
		0 − 1 1 −	1 0 0 0	s11=^x2*s25+x2*s37;
		0 1 0 0 −	1 0 0 1	s12=^x2*s28+x2;
		1 − 0 1 −	1 0 0 0	s19=^x2*s28+x2*^s31;
		1 − 1 0 −	1 0 0 0	s20=^x2*s31+x2*s28;
		− 1 − − 1	1 0 0 0	s10=^x2*x3+x2*s28;
		0 0 − 1 −	1 0 0 0	s8=^x2*x5+x2*s25;
		0 0 1 − 0	1 0 0 0	s17=^x2+x2*s28;
				s14=x2*^x5;
				s37=^x3*x5;
				s28=^x3*x5+x3;
				s25=^x3*x5+x3*^x5;
				s22=^x3+x3*x5;
				s31=x3*x5;
Схема 4		**Схема 5**		**Схема 6**

точно построить таблицу истинности функций по системе ДНФ, приведенной в левой (либо средней) части табл. 3. Минимизация в классе ДНФ изменяет форму задания векторной функции, но не ее таблицу истинности.

Способ 3. Замена двоичных строк матрицы *B* троичными строками, кодирование строк матрицы *B* согласно эвристики 1. В способе 3 каждая двоичная строка матрицы *B* (полная элементарная конъюнкция) заменяется троичной строкой — элементарной конъюнкцией так же, как и в способе 2. Однако кодирование полученных троичных векторов выполняется с помощью ***эвристики 1****: троичные векторы, содержащие много определенных (0, 1) элементов, кодируются булевыми векторами, содержащими возможно меньшее число единичных элементов.*

Данная эвристика ориентирована на сокращение общего числа литералов в системе ДНФ, задающей векторную полностью определенную функцию. Составим множество строк (конъюнкций), содержащих пять определенных элементов (литерала):

$$M^5 = \{(0\ 0\ 0\ 0\ 0),\ (0\ 0\ 1\ 0\ 0)\}.$$

Аналогично:

$$M^4 = \{(0\ 1\ 0\ 0\ -),\ (0\ 0\ 0\ 1\ -)\},$$

$$M^3 = \{(-\ 0\ 1\ -\ 1),\ (-\ 1\ -\ 1\ -),\ (-\ 1\ 1\ -\ 0),\ (1\ 0\ -\ -\ 1),\ (1\ -\ 1\ 1\ -),\ (1\ 1\ -\ -\ 0),\ (0\ -\ 0\ -1),\ (0\ -\ 1\ 1\ -),\ (0\ 1\ -\ -\ 1),\ (1\ -\ 0\ 1\ -),\ (1\ -\ 1\ 0\ -),\ (1\ 1\ -\ -\ 1)\}.$$

$$M^2 = \{(-\ \ 1\ \ -\ \ 1-)\}.$$

В примере булева строка (0 0 0 0 0) кодируется комбинацией (0 0 0 0) (табл. 4), поэтому полная элементарная конъюнкция $\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5$ будет отсутствовать в ДНФ каждой из четырех компонентных функций c_1, c_2, c_3, c_4. Строка (0 0 1 0 0) кодируется вектором (0 0 1 0) с одной единичной компонентой, поэтому полная элементарная конъюнкция,

Таблица 4

Результаты логической оптимизации в способе 3 (пример 1)

Доопределение по интервалам, кодирование по эвристике 1		Раздельно минимизированные ДНФ		BDDI-представление
$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	
0 0 0 0 0	0 0 0 0	1 – 1 – –	0 0 0 1	c1=^x5*s0+x5*s1;
– 0 1 – 1	0 0 1 1	1 – – 1 –	0 0 0 1	c2=^x5*s2+x5*s3;
– 1 – 1 –	1 1 1 1	0 – – – 1	0 0 0 1	c3=^x5*s4+x5*s5;
– 1 1 – 0	0 1 1 0	– – 1 1 –	0 0 0 1	c4=^x5*s6+x5*s7;
1 0 – – 1	1 1 1 0	0 0 1 – –	0 0 1 0	s4=^x2*^s9+x2*s23;
1 – 1 1 –	1 1 0 1	– 1 1 – 0	0 1 1 0	s0=^x2*s10+x2*s8;
1 1 – – 0	1 1 0 0	1 0 – – 1	0 1 0 0	s3=^x2*s14+x2*x4;
0 – 0 – 1	1 0 1 1	1 1 – – 0	0 1 0 0	s6=^x2*s16+x2*s17;
0 – 1 1 –	0 1 1 1	– – – 1 –	0 1 0 0	s1=^x2*s9+x2;
0 1 0 0 –	1 0 0 0	– 1 – 1 –	1 0 1 1	s2=^x2*x4+x2*s19;
0 1 – – 1	1 0 0 1	1 – 1 1 –	1 0 0 0	s5=^x2+x2*s8;
1 – 0 1 –	0 1 0 1	1 – – – 1	1 0 1 0	s9=^x1*^x3+x1;
1 – 1 0 –	0 0 0 1	1 1 – – –	1 0 0 0	s16=^x1*s20+x1*s23;
1 1 – – 1	1 0 1 0	– – 0 – 1	1 0 1 0	s19=^x1*s23+x1;
0 0 0 1 –	0 1 0 0	– 1 – – 1	1 0 0 0	s8=^x1*s27+x1;
0 0 1 0 0	0 0 1 0	– 1 0 – –	1 0 0 0	s14=^x1*x4+x1;
				s17=^x1*x4+x1*s23;
				s7=^x1+x1*s23;
				s10=x1*s20;
				s23=^x3*x4+x3;
				s27=^x3+x3*x4;
				s20=x3*x4;
Схема 7		**Схема 8**		**Схема 9**

содержащая пять литералов, войдет в ДНФ только одной функции c_3. Строки из множества M^4 кодируются векторами (1 0 0 0), (0 1 0 0) соответственно. Затем кодируются строки из множества M^3, используя оставшийся вектор (0 0 0 1) с одной единичной компонентой и векторы с двумя и тремя единичными компонентами. Для единственного троичного вектора (– 1 – 1 –) из множества M^2 присваивается кодовая комбинация (1 1 1 1) с четырьмя единичными компонентами. Формульное задание системы ДНФ, полученной в способе 3, содержит 96 литералов и имеет вид

$$c_1 = x_2x_4 \vee x_1\overline{x}_2x_5 \vee x_1x_3x_4 \vee x_1x_2\overline{x}_5 \vee \vee\ \overline{x}_1\overline{x}_3x_5 \vee \overline{x}_1x_2\overline{x}_3\overline{x}_4 \vee \overline{x}_1x_2x_5 \vee x_1x_2x_5;$$

$$c_2 = x_2x_4 \vee x_2x_3\overline{x}_5 \vee x_1\overline{x}_2x_5 \vee x_1x_3x_4 \vee \vee\ x_1x_2\overline{x}_5 \vee \overline{x}_1x_3x_4 \vee x_1\overline{x}_3x_4 \vee \overline{x}_1\overline{x}_2\overline{x}_3x_4;$$

$$c_3 = \overline{x}_2x_3x_5 \vee x_2x_4 \vee x_2x_3\overline{x}_5 \vee x_1\overline{x}_2x_5 \vee \vee\ \overline{x}_1\overline{x}_3x_5 \vee \overline{x}_1x_3x_4 \vee x_1x_2x_5 \vee \overline{x}_1\overline{x}_2x_3\overline{x}_4\overline{x}_5;$$

$$c_4 = \overline{x}_2x_3x_5 \vee x_2x_4 \vee x_1x_3x_4 \vee \overline{x}_1\overline{x}_3x_5 \vee \vee\ \overline{x}_1x_3x_4 \vee \overline{x}_1x_2x_5 \vee x_1\overline{x}_3x_4 \vee x_1x_3\overline{x}_4.$$

После того как коды присвоены, можно выполнить минимизацию в классе ДНФ полученной системы полностью определенных булевых функций — решать задачу 3. В результате такой минимизации число литералов в системе ДНФ сокращается с 96 до 47. Матричная форма минимизированных ДНФ приведена в средней части табл. 4. При синтезе логических схем из библиотечных элементов уменьшение числа литералов в представлениях систем полностью определенных булевых функций приводит к менее сложным схемам [9]. Поэтому число литералов - это основной критерий минимизации алгебраических представлений функций на этапе технологически независимой оптимизации — первом этапе синтеза логических схем.

Способы 2 и 3 ориентируются на уменьшение сложности функционального описания кодового преобразователя в классе ДНФ на основе минимизации суммарного числа литералов в представлении функций в виде ДНФ. Кроме программы Minim может быть использована программа раздельной минимизации из системы ABC [10], для логической минимизации вместо программ раздельной минимизации могут быть применены мощные программы совместной минимизации, такие как ESPRESSO [3]. Описываемые далее способы 4 и 5 ориентируются на другой вид представления систем булевых функций — бинарные диаграммы решений (BDD).

Способ 4. Минимизация в классе BDD частичной многозначной функции, зависящей от булевых переменных. Доопределение BDD "вертикальное", кодирование строк матрицы *B* согласно эвристики 2. Данный способ состоит из описанных далее пяти этапов.

Этап 1. Составление по матрице B функционального описания не полностью определенной многозначной (k-значной) функции: каждой строке b^i матрицы B ставится в соответствие значение p^i многозначной переменной p, являющейся функцией $p(x_1, ..., x_n)$, зависящей от булевых переменных $x_1, ..., x_n$.

Этап 2. Построение такого графа BDD многозначной функции $p(x_1, ..., x_n)$, который имеет возможно меньшее число функциональных вершин. Листовые вершины минимизированного графа BDD соответствуют значениям k-значной функции, так как в результате минимизации числа вершин BDD каждое неопределенное значение "–" доопределяется некоторым значением p^i функции $p(x_1, ..., x_n)$. Соответствующее доопределение BDD, названное "вертикальным", будет рассмотрено на примере.

Заметим, что BDD, реализующие k-значные функции булевых аргументов, являются частным случаем k-значных функций, зависящих от k_i-значных аргументов x_i и представляющих эти функции диаграмм решений, рассмотренных в работе [11].

Этап 3. Кодирование значений k-значной функции булевыми кодами по ***эвристике 2****: соседние листовые вершины графа BDD кодируются соседними (различающимися значениями одной компоненты) m-компонентными булевыми векторами. Для оставшихся листовых вершин коды присваиваются произвольным образом.*

Этап 4. Получение системы ДНФ полностью определенных булевых функций по минимизированному графу BDD и кодированию листовых вершин.

Этап 5. Для полученной системы ДНФ выполняются программы логической минимизации.

Рассмотрим применение способа 4 для матрицы B из примера 1. В табл. 5 дана 16-значная функция $p = p(x_1, ..., x_5)$ — результат выполнения этапа 1. На этапе 2 рассмотрим только одну перестановку $<x_1, x_2, x_3, x_4, x_5>$ переменных, по которой построим BDD. Заметим, что для n переменных ($n = 5$) имеется $n!$ (факториал числа n) перестановок и задача поиска перестановки, обеспечивающей минимальное число функциональных вершин BDD, является сложной комбинаторной задачей, которой посвящено большое число работ для случая 2-знач-

Таблица 5

Частичная многозначная функция и кодирование ее значений (способ 4)

$x_1x_2x_3x_4x_5$	Многозначная функция p	Кодирующие переменные p_j^i	Значения кодирующих переменных, кодирование по эвристике 2
0 0 0 0 0	p^1	$p_1^1\ p_2^1\ p_3^1\ p_4^1$	1 1 0 1
0 0 1 0 1	p^2	$p_1^2\ p_2^2\ p_3^2\ p_4^2$	1 0 0 1
0 1 0 1 0	p^3	$p_1^3\ p_2^3\ p_3^3\ p_4^3$	0 1 1 0
0 1 1 0 0	p^4	$p_1^4\ p_2^4\ p_3^4\ p_4^4$	0 1 0 0
1 0 0 0 1	p^5	$p_1^5\ p_2^5\ p_3^5\ p_4^5$	0 0 1 0
1 0 1 1 0	p^6	$p_1^6\ p_2^6\ p_3^6\ p_4^6$	1 0 1 0
1 1 0 0 0	p^7	$p_1^7\ p_2^7\ p_3^7\ p_4^7$	0 0 0 0
0 0 0 0 1	p^8	$p_1^8\ p_2^8\ p_3^8\ p_4^8$	1 1 0 0
0 0 1 1 0	p^9	$p_1^9\ p_2^9\ p_3^9\ p_4^9$	1 1 1 0
0 1 0 0 0	p^{10}	$p_1^{10}\ p_2^{10}\ p_3^{10}\ p_4^{10}$	0 1 1 1
0 1 1 0 1	p^{11}	$p_1^{11}\ p_2^{11}\ p_3^{11}\ p_4^{11}$	0 1 0 1
1 0 0 1 0	p^{12}	$p_1^{12}\ p_2^{12}\ p_3^{12}\ p_4^{12}$	0 0 1 1
1 0 1 0 0	p^{13}	$p_1^{13}\ p_2^{13}\ p_3^{13}\ p_4^{13}$	1 1 1 1
1 1 0 0 1	p^{14}	$p_1^{14}\ p_2^{14}\ p_3^{14}\ p_4^{14}$	0 0 0 1
0 0 0 1 0	p^{15}	$p_1^{15}\ p_2^{15}\ p_3^{15}\ p_4^{15}$	1 0 1 1
0 0 1 0 0	p^{16}	$p_1^{16}\ p_2^{16}\ p_3^{16}\ p_4^{16}$	1 0 0 0
0 1 0 0 1	–	– – – –	
0 1 1 1 0	–	– – – –	
1 0 0 0 0	–	– – – –	
1 0 1 0 1	–	– – – –	
1 1 0 1 0	–	– – – –	
0 0 0 1 1	–	– – – –	
0 0 1 1 1	–	– – – –	
0 1 0 1 1	–	– – – –	
0 1 1 1 1	–	– – – –	
1 0 0 1 1	–	– – – –	
1 0 1 1 1	–	– – – –	
1 1 0 1 1	–	– – – –	
1 1 1 0 0	–	– – – –	
1 1 1 0 1	–	– – – –	
1 1 1 1 0	–	– – – –	
1 1 1 1 1	–	– – – –	

ных (булевых) функций. Краткий обзор таких работ представлен в [4, с. 39].

На этапе 2 требуется найти граф BDD функции с минимальным числом функциональных вершин. Граф BDD задает последовательные дизъюнктивные разложения функции $p(x_1, ..., x_n)$, в примере $n = 5$. Для булевых функций такие разложения называются разложениями Шеннона. По аналогии с булевыми функциями определим разложение Шеннона для k-значной функции. *Разложением Шеннона* полностью либо не полностью определенной k-значной функции $p = p(\boldsymbol{x}) = p(x_1, x_2, ..., x_n)$, зависящей от булевых переменных $x_1, x_2, ..., x_n$, по переменной x_i назовем представление

$$p = p(\boldsymbol{x}) = \overline{x}_i p(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n) \vee \vee x_i p(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n). \quad (1)$$

Многозначные функции $p_0 = p(x_1, ..., x_{i-1}, 0, x_{i+1}, ..., x_n)$, $p_1 = p(x_1, ..., x_{i-1}, 1, x_{i+1}, ..., x_n)$ в правой части (1) называются *остаточными функциями* либо *кофакторами* (*cofactors*, англ.) разложения по переменной x_i. Они получаются из функции p подстановкой вместо переменной x_i константы 0 и 1 соответственно. Каждый из кофакторов p_0 и p_1 может быть разложен по одной из переменных из множества $\{x_1, ..., x_{i-1}, x_{i+1}, ..., x_n\}$. Процесс разложения кофакторов заканчивается, когда все n переменных будут использованы для разложения, либо когда все кофакторы выродятся до констант $p^1, ..., p^k$. На каждом шаге разложения выполняется сравнение на равенство полученных кофакторов.

Кофакторы последовательных разложений исходной 16-значной функции $p(x_1, x_2, x_3, x_4, x_5)$ (табл. 5) по переменным x_1, x_2, x_3, x_4 приведены в табл. 6—9.

Таблица 6

Остаточные функции разложения по переменной x_1

$x_2x_3x_4x_5$	$x_1 = 0$	$x_1 = 1$
	p	
0 0 0 0	p^1	–
0 0 0 1	p^8	p^5
0 0 1 0	p^{15}	p^{12}
0 0 1 1	–	–
0 1 0 0	p^{16}	p^{13}
0 1 0 1	p^2	–
0 1 1 0	p^9	p^6
0 1 1 1	–	–
1 0 0 0	p^{10}	p^7
1 0 0 1	–	p^{14}
1 0 1 0	p^3	–
1 0 1 1	–	–
1 1 0 0	p^4	–
1 1 0 1	p^{11}	–
1 1 1 0	–	–
1 1 1 1	–	–

Таблица 7

Остаточные функции разложения по переменным x_1, x_2

$x_3\ x_4\ x_5$	$x_1 = 0$, $x_2 = 0$	$x_1 = 0$, $x_2 = 1$	$x_1 = 1$, $x_2 = 0$	$x_1 = 1$, $x_2 = 1$
	p			
0 0 0	p^1	p^{10}	–	p^7
0 0 1	p^8	–	p^5	p^{14}
0 1 0	p^{15}	p^3	p^{12}	–
0 1 1	–	–	–	–
1 0 0	p^{16}	p^4	p^{13}	–
1 0 1	p^2	p^{11}	–	–
1 1 0	p^9	–	p^6	–
1 1 1	–	–	–	–

Таблица 8

Остаточные функции разложения по переменным x_1, x_2, x_3

x_4x_5	$x_1 = 0$ $x_2 = 0$ $x_3 = 0$	$x_1 = 0$ $x_2 = 0$ $x_3 = 1$	$x_1 = 0$ $x_2 = 1$ $x_3 = 0$	$x_1 = 0$ $x_2 = 1$ $x_3 = 1$	$x_1 = 1$ $x_2 = 0$ $x_3 = 0$	$x_1 = 1$ $x_2 = 0$ $x_3 = 1$	$x_1 = 1$ $x_2 = 1$ $x_3 = 0$	$x_1 = 1$ $x_2 = 1$ $x_3 = 1$
	p							
0 0	p^1	p^{16}	p^{10}	p^4	–	p^{13}	p^7	–
0 1	p^8	p^2	–	p^{11}	p^5	–	p^{14}	–
1 0	p^{15}	p^9	p^3	–	p^{12}	p^6	–	–
1 1	–	–	–	–	–	–	–	–

Построим граф BDD функции $p(x_1, x_2, x_3, x_4, x_5)$ из табл. 5 для последовательности (перестановки) переменных $<x_1, x_2, x_3, x_4, x_5>$ (рис. 1). Вершины-кофакторы будем называть функциональными вершинами в BDD в отличие от вершин-переменных, по которым проводится разложение. Листовые вершины в графе BDD будут задавать определенные либо неопределенные "–" значения функции. Граф BDD, в котором не проведено сокращение функциональных вершин, показан на рис. 1.

Общий рисунок BDD является большим, поэтому разделен на три части. Одинаковыми функциональными вершинами (кофакторами) являются G^{22}, G^{28}, G^{29}, G^{30}. Эти вершины не представлены одной вершиной, чтобы показать, что одинаковые неопределенные кофакторы могут быть доопределены до различных полностью определенных кофакторов. Алгоритм "вертикального" доопределения BDD, реализующей частично определенную многозначную функцию, состоит в замене неопределенных значений листовых вершин определенными значениями соседних определенных листовых вершин. Соседними являются вершины пары кофакторов, получающихся в результате разложения Шеннона какого-либо кофактора. В примере кофактор G^{16} имеет две листовые вершины: p^{15} и "–".

Таблица 9

Остаточные функции разложения по переменным x_1, x_2, x_3, x_4

x_5	$x_1=0$ $x_2=0$ $x_3=0$ $x_4=0$	$x_1=0$ $x_2=0$ $x_3=0$ $x_4=1$	$x_1=0$ $x_2=0$ $x_3=1$ $x_4=0$	$x_1=0$ $x_2=0$ $x_3=1$ $x_4=1$	$x_1=0$ $x_2=1$ $x_3=0$ $x_4=0$	$x_1=0$ $x_2=1$ $x_3=0$ $x_4=1$	$x_1=0$ $x_2=1$ $x_3=1$ $x_4=0$	$x_1=0$ $x_2=1$ $x_3=1$ $x_4=1$	$x_1=1$ $x_2=0$ $x_3=0$ $x_4=0$	$x_1=1$ $x_2=0$ $x_3=0$ $x_4=1$	$x_1=1$ $x_2=0$ $x_3=1$ $x_4=0$	$x_1=1$ $x_2=0$ $x_3=1$ $x_4=1$	$x_1=1$ $x_2=1$ $x_3=0$ $x_4=0$	$x_1=1$ $x_2=0$ $x_3=1$ $x_4=1$	$x_1=1$ $x_2=1$ $x_3=1$ $x_4=0$	$x_1=1$ $x_2=1$ $x_3=1$ $x_4=1$
	p															
0 1	p^1 p^8	p^{15} –	p^{16} p^2	p^9 –	p^{10} –	p^3 –	p^4 p^{11}	– –	– p^5	p^{12} –	p^{13} –	p^6 –	p^7 p^{14}	– –	– –	– –

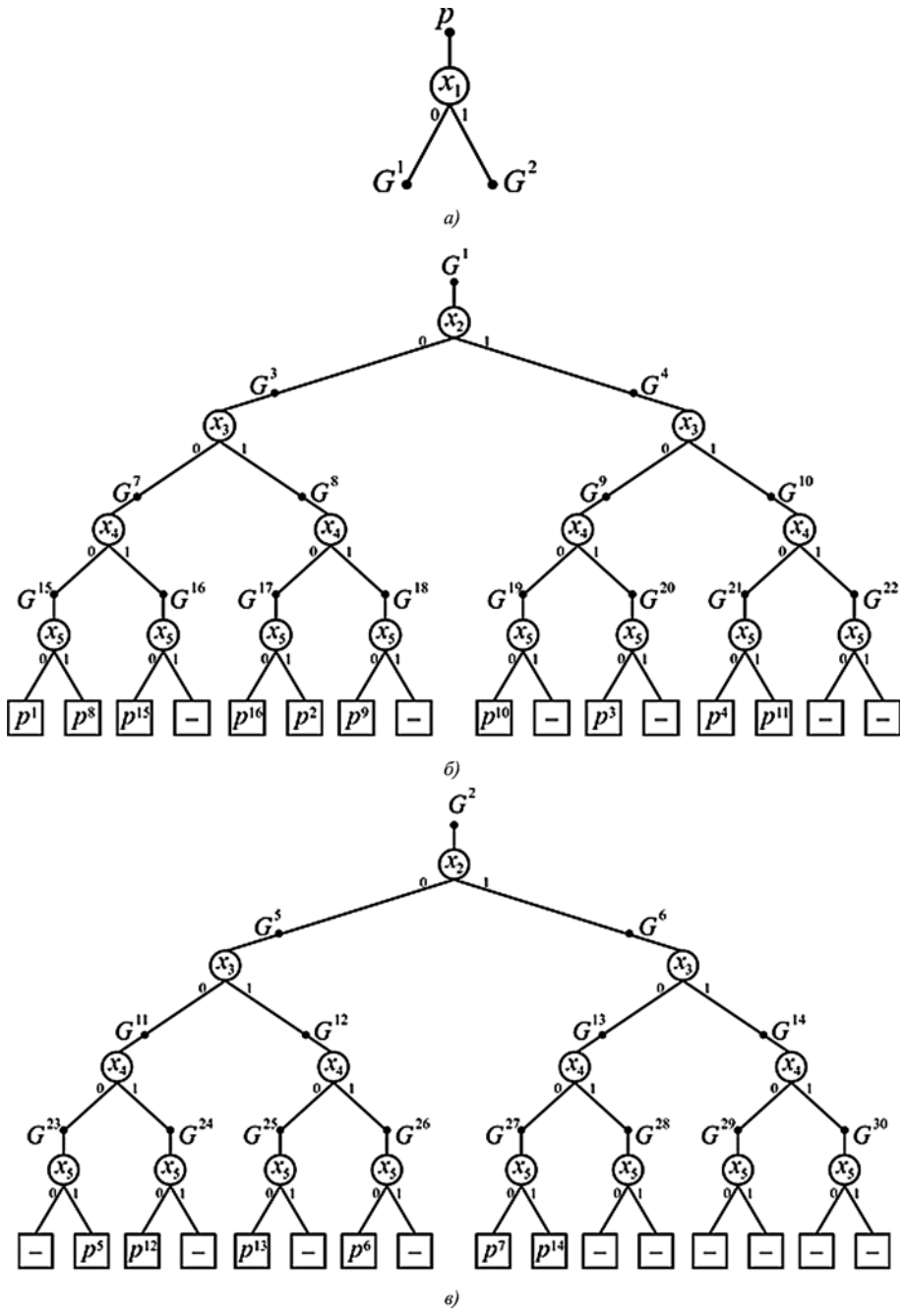


Рис. 1. Бинарная диаграмма решений частичной многозначной функции *p*:
а — разложение функции *p* по переменной x_1; *б* — разложение кофактора G^1; *в* — разложение кофактора G^2

Поэтому неопределенное значение заменяется значением p^{15} соседней листовой вершины. Для кофактора G^{19} соседнее неопределенное значение заменяется значением p^9. Аналогично для других кофакторов, которые имеют соседние листовые вершины, одна из которых является определенной p^i, другая — неопределенной "–". Для кофактора G^{10} соседними являются кофакторы G^{21}, G^{22}, поэтому кофактор G^{22} доопределяется до кофактора G^{21}, т. е. неопределенные листовые вершины кофактора G^{22} заменяются на p^4, p^{11}. Для кофактора G^6 в результате подобных пошаговых доопределений листовыми вершинами становятся p^7, p^{14}, т. е. неопределенные кофакторы G^{28}, G^{29}, G^{30} доопределяются до полностью определенного кофактора G^{28}.

Уравнения, описывающие доопределенный граф BDD (рис. 2), имеют следующий вид:

$$
\begin{gathered}
p = \overline{x}_1 G^1 \vee x_1 G^2; \\
G^1 = \overline{x}_2 G^3 \vee x_2 G^4;\ G^2 = \overline{x}_2 G^5 \vee x_2 G^6; \\
G^3 = \overline{x}_3 G^7 \vee x_3 G^8; \\
G^4 = \overline{x}_3 G^9 \vee x_3 G^{10};\ G^5 = \overline{x}_3 G^{11} \vee x_3 G^{12}; \\
G^6 = \overline{x}_5 p^7 \vee x_3 p^{14}; \\
G^7 = \overline{x}_4 G^{15} \vee x_4 p^{15};\ G^8 = \overline{x}_4 G^{17} \vee x_4 p^9; \\
G^9 = \overline{x}_4 p^{10} \vee x_4 p^3; \\
G^{10} = \overline{x}_5 p^4 \vee x_5 p^{11};\ G^{11} = \overline{x}_5 p^5 \vee x_5 p^{12}; \\
G^{12} = \overline{x}_4 p^{13} \vee x_4 p^6; \\
G^{15} = \overline{x}_5 p^1 \vee x_5 p^8;\ G^{17} = \overline{x}_5 p^{16} \vee x_5 p^2.
\end{gathered}
\qquad (2)
$$

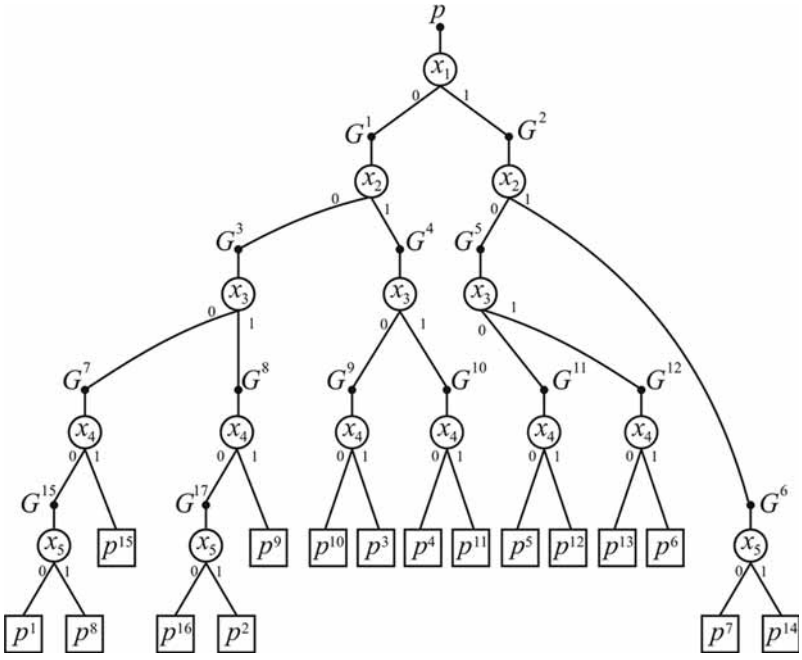


Рис. 2. Доопределенный граф BDD многозначной функции *p*

На этапе 3 осуществляется кодирование соседних листовых вершин соседними кодами. В примере соседним вершинам p^1, p^8 присваиваются соседние коды (1 1 0 1), (1 1 0 0), соответственно. *Соседние коды* (булевы векторы) различаются значениями в одном разряде, в данном случае в четвертом. Для других соседних пар: для пары p^{16}, p^2 кодами являются соседние векторы (1 0 0 0), (1 0 0 1); для пары p^4, p^{11} — соседние векторы (0 1 0 0), (0 1 0 1); для пары p^7, p^{14} — соседние векторы (0 0 0 0), (0 0 0 1). Булевы коды всех значений многозначной переменной p даны в правой части табл. 5.

На этапе 4 осуществляется замена каждого многозначного кофактора своей подсистемой булевых функций (табл. 10), зависящих от соответствующих множеств булевых переменных.

Заменим в графе BDD (см. рис. 2) многозначные значения булевыми значениями согласно табл. 10. Получим граф BDD (рис. 3), представляющий векторную булеву функцию $\boldsymbol{c}(\boldsymbol{x}) = (c_1(x), c_2(\boldsymbol{x}), c_3(\boldsymbol{x}), c_4(\boldsymbol{x}))$, $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$.

Таблица 10

Представление многозначных кофакторов в виде подсистем булевых функций

Многозначная функция (кофактор)	Подсистема булевых функций
p	$c_1\ c_2\ c_3\ c_4$
G^1	$g_1\ g_2\ g_3\ g_4$
G^2	$g_5\ g_6\ g_7\ g_8$
G^3	$g_9\ g_{10}\ g_{11}\ g_{12}$
G^4	$g_{13}\ g_{14}\ g_{15}\ g_{16}$
G^5	$g_{17}\ g_{18}\ g_{19}\ g_{20}$
G^6	$g_{21}\ g_{22}\ g_{23}\ g_{24}$
G^7	$g_{25}\ g_{26}\ g_{27}\ g_{28}$
G^8	$g_{29}\ g_{30}\ g_{31}\ g_{32}$
G^9	$g_{33}\ g_{34}\ g_{35}\ g_{36}$
G^{10}	$g_{37}\ g_{38}\ g_{39}\ g_{40}$
G^{11}	$g_{41}\ g_{42}\ g_{43}\ g_{44}$
G^{12}	$g_{45}\ g_{46}\ g_{47}\ g_{48}$
G^{15}	$g_{49}\ g_{50}\ g_{51}\ g_{52}$
G^{17}	$g_{53}\ g_{54}\ g_{55}\ g_{56}$

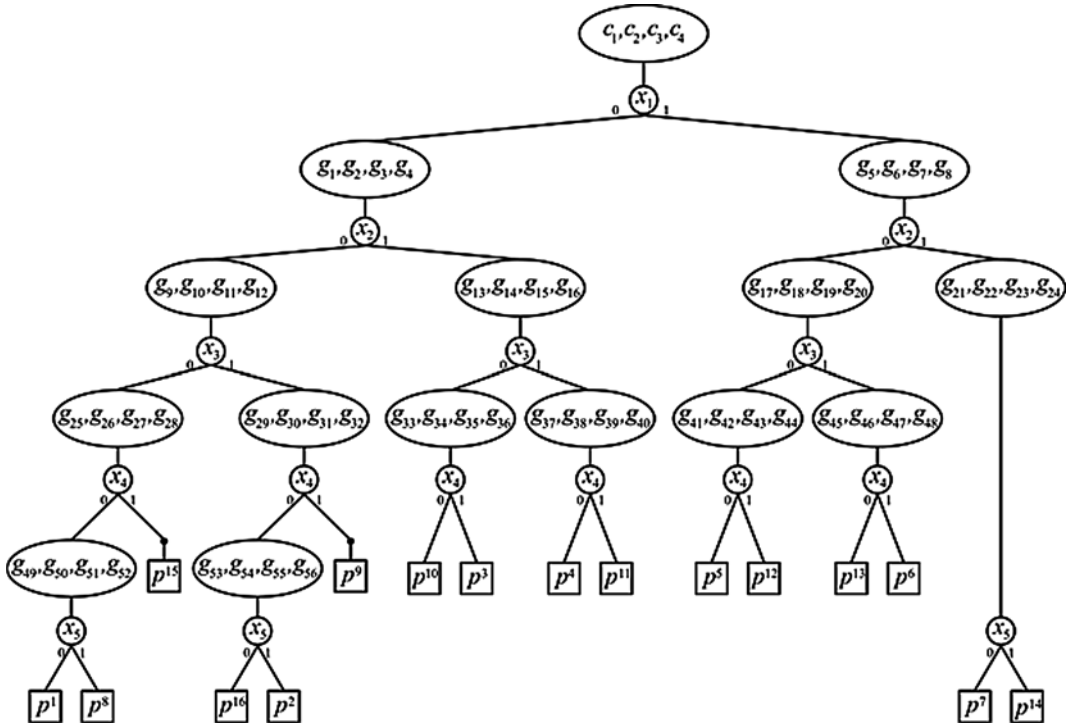


Рис. 3. BDD многозначной функции после замены многозначных кофакторов булевыми переменными

Уравнения (2) разложений Шеннона многозначной функции и ее многозначных кофакторов заменяются следующими уравнениями с булевыми переменными:

$$c_1 = \overline{x}_1 g_1 \vee x_1 g_5;\ c_2 = \overline{x}_1 g_2 \vee x_1 g_6;$$
$$c_3 = \overline{x}_1 g_3 \vee x_1 g_7;\ c_4 = \overline{x}_1 g_4 \vee x_1 g_8;$$
$$g_1 = \overline{x}_2 g_9 \vee x_2 g_{13};\ g_2 = \overline{x}_2 g_{10} \vee x_2 g_{14};$$
$$g_3 = \overline{x}_2 g_{11} \vee x_2 g_{15};\ g_4 = \overline{x}_2 g_{12} \vee x_2 g_{16};$$
$$g_5 = \overline{x}_2 g_{17} \vee x_2 g_{21};\ g_6 = \overline{x}_2 g_{18} \vee x_2 g_{22};$$
$$g_7 = \overline{x}_2 g_{19} \vee x_2 g_{23};\ g_8 = \overline{x}_2 g_{20} \vee x_2 g_{24};$$
$$g_9 = \overline{x}_3 g_{25} \vee x_3 g_{29};\ g_{10} = \overline{x}_3 g_{26} \vee x_3 g_{30};$$
$$g_{11} = \overline{x}_3 g_{27} \vee x_3 g_{31};\ g_{12} = \overline{x}_3 g_{28} \vee x_3 g_{32};$$
$$g_{13} = \overline{x}_3 g_{33} \vee x_3 g_{37};\ g_{14} = \overline{x}_3 g_{34} \vee x_3 g_{38};$$
$$g_{15} = \overline{x}_3 g_{35} \vee x_3 g_{39};\ g_{16} = \overline{x}_3 g_{36} \vee x_3 g_{40};$$
$$g_{17} = \overline{x}_3 g_{41} \vee x_3 g_{45};\ g_{18} = \overline{x}_3 g_{42} \vee x_3 g_{46};$$
$$g_{19} = \overline{x}_3 g_{43} \vee x_3 g_{47};\ g_{20} = \overline{x}_3 g_{44} \vee x_3 g_{48};$$
$$g_{21} = \overline{x}_5 p_1^7 \vee x_5 p_1^{14};\ g_{22} = \overline{x}_5 p_2^7 \vee x_5 p_2^{14};$$
$$g_{23} = \overline{x}_5 p_3^7 \vee x_5 p_3^{14};\ g_{24} = \overline{x}_5 p_5^7 \vee x_4 p_4^{14};$$
$$g_{25} = \overline{x}_4 g_{49} \vee x_4 p_1^{15};\ g_{26} = \overline{x}_4 g_{50} \vee x_4 p_2^{15};$$
$$g_{27} = \overline{x}_4 g_{51} \vee x_4 p_3^{15};\ g_{28} = \overline{x}_4 g_{52} \vee x_4 p_4^{15};$$
$$g_{29} = \overline{x}_4 g_{53} \vee x_4 p_1^{9};\ g_{30} = \overline{x}_4 g_{54} \vee x_4 p_2^{9};$$
$$g_{31} = \overline{x}_4 g_{55} \vee x_4 p_3^{9};\ g_{32} = \overline{x}_4 g_{56} \vee x_4 p_4^{9};$$
$$g_{33} = \overline{x}_4 p_1^{10} \vee x_4 p_1^{3};\ g_{34} = \overline{x}_4 p_2^{10} \vee x_4 p_2^{3};$$
$$g_{35} = \overline{x}_4 p_3^{10} \vee x_4 p_3^{3};\ g_{36} = \overline{x}_4 p_4^{10} \vee x_4 p_4^{3};$$
$$g_{37} = \overline{x}_5 p_1^{4} \vee x_5 p_1^{11};\ g_{38} = \overline{x}_5 p_2^{4} \vee x_5 p_2^{11};$$
$$g_{39} = \overline{x}_5 p_3^{4} \vee x_5 p_3^{11};\ g_{40} = \overline{x}_5 p_4^{4} \vee x_5 p_4^{11};$$
$$g_{41} = \overline{x}_4 p_1^{5} \vee x_4 p_1^{12};\ g_{42} = \overline{x}_4 p_2^{5} \vee x_4 p_2^{12};$$
$$g_{43} = \overline{x}_4 p_3^{5} \vee x_4 p_3^{12};\ g_{44} = \overline{x}_4 p_4^{5} \vee x_4 p_4^{12};$$
$$g_{45} = \overline{x}_4 p_1^{13} \vee x_4 p_1^{6};\ g_{46} = \overline{x}_4 p_2^{13} \vee x_4 p_2^{6};$$
$$g_{47} = \overline{x}_4 p_3^{13} \vee x_4 p_3^{6};\ g_{48} = \overline{x}_4 p_4^{13} \vee x_4 p_4^{6};$$
$$g_{49} = \overline{x}_5 p_1^{1} \vee x_5 p_1^{8};\ g_{50} = \overline{x}_5 p_2^{1} \vee x_5 p_2^{8};$$
$$g_{51} = \overline{x}_5 p_3^{1} \vee x_5 p_3^{8};\ g_{52} = \overline{x}_5 p_4^{1} \vee x_5 p_4^{8};$$
$$g_{53} = \overline{x}_5 p_1^{16} \vee x_5 p_1^{2};$$
$$g_{54} = \overline{x}_5 p_2^{16} \vee x_5 p_2^{2};\ g_{55} = \overline{x}_5 p_3^{16} \vee x_5 p_3^{2};$$
$$g_{56} = \overline{x}_5 p_4^{16} \vee x_5 p_4^{2}. \qquad (3)$$

Подстановка значений p_j^i из табл. 5 и элиминация (устранение) промежуточных переменных позволяют получить систему ДНФ полностью определенных функций, заданную в левой части табл. 11. Покажем, как упрощаются уравнения для кофакторов $g_{53}, g_{54}, g_{55}, g_{56}$ при подстановке значений $p^{16} = (1\ 0\ 0\ 0)$, $p^2 = (1\ 0\ 0\ 1)$ из табл. 5: $g_{53} = \overline{x}_5 1 \vee x_5 1 = 1$; $g_{54} = \overline{x}_5 0 \vee x_5 0 = 0$; $g_{55} = \overline{x}_5 0 \vee x_5 0 = 0$; $g_{56} = \overline{x}_5 0 \vee x_5 1 = x_5$. Присвоение соседних кодов ориентируется на возможно большее сокращение графа BDD и, соответственно,

Таблица 11

Результаты логической оптимизации в способе 4 (пример 1)

Доопределение по BDD, "вертикальное", кодирование по эвристике 2		Раздельно минимизированные ДНФ		BDDI-представление
$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	

0 0 0 0 0
0 0 1 0 1
0 1 0 1 –
0 1 1 – 0
1 0 0 0 –
1 0 1 1 –
1 1 0 0 0
0 0 0 0 1
0 0 1 1 –
0 1 0 0 –
0 1 1 – 1
1 0 0 1 –
1 0 1 0 –
1 1 – – 1
0 0 0 1 –
0 0 1 0 0
	1 1 0 1
	1 0 0 1
	0 1 1 0
	0 1 0 0
	0 0 1 0
	1 0 1 0
	0 0 0 0
	1 1 0 0
	1 1 1 0
	0 1 1 1
	0 1 0 1
	0 0 1 1
	1 1 1 1
	0 0 0 1
	1 0 1 1
	1 0 0 0
		– 0 0 1 –
		1 1 – – 1
		– 1 1 – 1
		– 1 – 0 1
		– – 1 0 1
		0 – 0 0 0
		– 0 – 1 –
		0 1 0 – –
		1 0 – – –
		1 0 1 0 –
		0 – 0 0 –
		0 – 1 1 –
		0 1 – – –
		0 0 – – –
		– 0 1 – –
			0 0 0 1
			0 0 0 1
			0 0 0 1
			0 0 0 1
			0 0 0 1
			0 0 0 1
			0 0 1 0
			0 0 1 0
			0 0 1 0
			0 1 0 1
			0 1 0 0
			0 1 0 0
			0 1 0 0
			1 0 0 0
			1 0 0 0
				c1=^x1*^x2+x1*s1;
				c2=^x1*s2+x1*s3;
				c3=^x1*s4+x1*^x2;
				c4=^x1*s6+x1*s7;
				s7=^x2*^s10+x2*x5;
				s2=^x2*s10+x2;
				s6=^x2*s15+x2*s16;
				s3=^x2*s9;
				s1=^x2*x3;
				s4=^x2*x4+x2*^x3;
				s10=^x3*^x4+x3*x4;
				s16=^x3*^x4+x3*x5;
				s15=^x3*s20+x3*^s20;
				s9=x3*^x4;
				s20=^x4*^x5+x4;

Схема 10		**Схема 11**		**Схема 12**

на сокращение уравнений, входящих в BDDI-представление.

Результаты логической оптимизации данной системы, полученные на этапе 5, даны в средней и правой частях табл. 11.

Способ 5. Минимизация в классе BDD частичной многозначной функции, зависящей от булевых переменных, доопределение BDD "горизонтальное", кодирование строк матрицы *B* согласно эвристики 2. В реализации способа 5 выполняются этапы 1 — 3 из способа 4. Затем по полученной системе ДНФ определяется код каждой строки b^i матрицы *B*. Для этого находится элементарная конъюнкция, которую *имплицирует* полная элементарная конъюнкция, соответствующая строке b^i. Например, строке (1 1 0 0 1) матрицы *B* соответствует полная элементарная конъюнкция $x_1x_2\bar{x}_3\bar{x}_4x_5$. Легко проверить, что выполняется единственная импликация $x_1x_2\bar{x}_3\bar{x}_4x_5 \to x_1x_2x_5$. Троичный вектор (1 1 – – 1), соответствующий $x_1x_2x_5$, имеет код (0 0 0 1). Следовательно, закодируем строку (1 1 0 0 1) кодом (0 0 0 1) (см. систему ДНФ в левой части табл. 11). Выполнив такую процедуру для каждой строки матрицы *B*, получим матричное описание частичной векторной функции в левой части табл. 12.

На этапе 4 способа 5 будем выполнять логическую оптимизацию тремя программами. Функциональное описание (схема 13) получается в результате построения BDD-описания полностью определенной функции с помощью программы из работы [4, с. 201]. Эта программа реализует алгоритм доопределения частичной BDD на основе "горизонтального" доопределения. Алгоритм доопределения сводится в работе [4, с. 64] к задаче раскраски вершин неориентированного графа в минимальное число красок (цветов) так, чтобы соседние вершины графа были раскрашены разными красками. Аналогичные алгоритмы могут быть использованы и при "горизонтальном" доопределении BDD неполностью определенной *k*-значной функции.

Результаты кодирования матрицы *B* рассмотренными способами 1—5 решения задач 2, 3 приведены в табл. 13.

3. Решение задачи 4

Функциональные описания схем 1—15 были получены программами логической оптимизации, имеющимися в системе FLC-2. Эти описания являются минимизированными ДНФ либо BDDI-описаниями на языке SF векторных полностью определенных функций. ДНФ представлялись матричными описаниями, BDDI-описания — логическими уравнениями в булевом базисе. Затем осуществлялся перевод оптимизированных SF-описаний в описания на языке VHDL [12]. Схемная реализация (синтез) полученных VHDL-описаний выполнялась с помощью синтезатора LeonardoSpectrum в библиотеке проектирования КМОП СБИС. Библиотека логических КМОП-элементов приведена в работе [6]. Для каждого описания схемы синтез осуществлялся с одни-

Таблица 12

Результаты логической оптимизации в способе 5 (пример 1)

Частичная векторная функция, кодирование по эвристике 2		BDD, реализующая частичную векторную функцию	Раздельно минимизированные ДНФ		BDDI-представление
$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$		$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	
0 0 0 0 0	1 1 0 1	c1=^x2*s18;	- 1 - 0 1	0 0 0 1	c1=^x3*s0+x3*^x2;
0 0 1 0 1	1 0 0 1	c2=^x4*s11+x4*s12;	0 - 1 0 1	0 0 0 1	c2=^x3*s2+x3*s3;
0 1 0 1 0	0 1 1 0	c3=^x4*s13+x4;	1 - 1 0 0	0 0 0 1	c3=^x3*s4+x3*s5;
0 1 1 0 0	0 1 0 0	c4=^x5*s7+x5*s8;	0 - 0 0 0	0 0 0 1	c4=^x3*s6+x3*s7;
1 0 0 0 1	0 0 1 0	s18=^x1+x1*x3;	- 0 0 1 0	0 0 0 1	s3=^x4*^s0+x4*s13;
1 0 1 1 0	1 0 1 0	s11=^x2*s20+x2*s18;	- - - 1 -	0 0 1 0	s2=^x4*^x1+x4*x2;
1 1 0 0 0	0 0 0 0	s12=^x2*s22+x2;	0 1 0 - -	0 0 1 0	s4=^x4*s11+x4;
0 0 0 0 1	1 1 0 0	s13=^x2*x1+x2*s20;	1 0 - - -	0 0 1 0	s7=^x4*s14;
0 0 1 1 0	1 1 1 0	s7=^x4*s20+x4*s16;	1 - 1 - -	0 0 1 0	s6=^x4*s15+x4*s16;
0 1 0 0 0	0 1 1 1	s8=^x4*s12;	- 1 - 1 -	0 1 0 0	s5=^x4*x1+x4;
0 1 1 0 1	0 1 0 1	s20=^x1*^x3+x1*x3;	1 - 1 0 -	0 1 0 0	s0=^x1*^x2;
1 0 0 1 0	0 0 1 1	s22=^x1*x3;	0 1 - - -	0 1 0 0	s15=^x1*s21+x1*s24;
1 0 1 0 0	1 1 1 1	s16=^x2*^x3;	0 - 1 1 -	0 1 0 0	s11=^x1*x2+x1*^x2;
1 1 0 0 1	0 0 0 1		0 - 0 0 -	0 1 0 0	s14=^x1*x5+x1*s21;
0 0 0 1 0	1 0 1 1		0 0 - - -	1 0 0 0	s13=^x1+x1*x2;
0 0 1 0 0	1 0 0 0		- 0 1 - -	1 0 0 0	s16=^x2*^x5;
0 1 0 0 1	- - - -				s21=^x2*^x5+x2;
0 1 1 1 0	- - - -				s24=x2*x5;
1 0 0 0 0	- - - -				
1 0 1 0 1	- - - -				
1 1 0 1 0	- - - -				
0 0 0 1 1	- - - -				
0 0 1 1 1	- - - -				
0 1 0 1 1	- - - -				
0 1 1 1 1	- - - -				
1 0 0 1 1	- - - -				
1 0 1 1 1	- - - -				
1 1 0 1 1	- - - -				
1 1 1 0 0	- - - -				
1 1 1 0 1	- - - -				
1 1 1 1 0	- - - -				
1 1 1 1 1	- - - -				
		Схема 13	**Схема 14**		**Схема 15**

Таблица 13

Коды матрицы *B* (пример 1)

Матрица B	Способы 1, 2	Способ 3	Способы 4, 5
$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	$c_1c_2c_3c_4$	$c_1c_2c_3c_4$
0 0 0 0 0	0 0 0 0	0 0 0 0	1 1 0 1
0 0 1 0 1	0 0 0 1	0 0 1 1	1 0 0 1
0 1 0 1 0	0 0 1 0	1 1 1 1	0 1 1 0
0 1 1 0 0	0 0 1 1	0 1 1 0	0 1 0 0
1 0 0 0 1	0 1 0 0	1 1 1 0	0 0 1 0
1 0 1 1 0	0 1 0 1	1 1 0 1	1 0 1 0
1 1 0 0 0	0 1 1 0	1 1 0 0	0 0 0 0
0 0 0 0 1	0 1 1 1	1 0 1 1	1 1 0 0
0 0 1 1 0	1 0 0 0	0 1 1 1	1 1 1 0
0 1 0 0 0	1 0 0 1	1 0 0 0	0 1 1 1
0 1 1 0 1	1 0 1 0	1 0 0 1	0 1 0 1
1 0 0 1 0	1 0 1 1	0 1 0 1	0 0 1 1
1 0 1 0 0	1 1 1 0	0 0 0 1	1 1 1 1
1 1 0 0 1	1 1 0 1	1 0 1 0	0 0 0 1
0 0 0 1 0	1 1 1 0	0 1 0 0	1 0 1 1
0 0 1 0 0	1 1 1 1	0 0 1 0	1 0 0 0

ми и теми же опциями управления синтезом. Для каждой полученной схемы подсчитывалась площадь схемы S_{ASIC} (в условных единицах площади логических элементов) и временная задержка τ (нс). Лучшие решения отмечены символом "*" (табл. 14). Схема 14 имеет наименьшую площадь, она получена в результате применения способа 5 путем оптимизации BDD частичных функций.

Результаты вычислительного эксперимента (см. табл. 14) показывают, что при одинаковом кодировании строк матрицы *B*, но при разных доопределениях частичной векторной функции $\boldsymbol{c}(\boldsymbol{x})$, сложности схем и их задержки могут иметь значительные различия.

В рамках исследования был проведен эксперимент, когда по матрицам *B*, *C* составлялись VHDL-описания систем частичных функций, которые отправлялось на синтез в LeonardoSpectrum. В этом случае технологически независимую оптимизацию

Таблица 14

Результаты синтеза логических схем (пример 1)

Способ решения задач 2 ,3	Вид логической минимизации	Сложность функционального описания			Логическая схема		Номер схемы
		Дизъюнкций	Конъюнкций	Литералов	S_{ASIC}	τ, нс	
1	Нет	29	132	165	14 274	3,89	1
	ДНФ	18	73	95	11 874	3,30	2
	BDDI	17	42	84	12 187	2,62	3
2	Нет	28	76	108	12 644	3,82	4
	ДНФ	21	50	75	10 178	3,41	5
	BDDI	26	49	104	13 269	2,92	6
3	Нет	28	64	96	7137	2,62	7
	ДНФ	17	26	47	7449	2,83	8
	BDDI	20	33	75	7466	2,20	9
4	Нет	28	103	135	10 490	2,61	10
	ДНФ	12	30	46	7226	2,85	11
	BDDI	12	25	52	7048	2,38	12
5	BDD	*9	*19	*41	6478	*1,79	13
	ДНФ	12	28	44	*5569	2,00	14
	BDDI	14	28	60	6930	3,83	15

выполнял синтезатор. Оказалось, что во всех случаях технологически независимая оптимизация системы FLC-2 приводила к лучшим результатам синтеза. Например, для примера 1 (способ 5) было составлено следующее VHDL-описание системы частичных функций:

```
Library IEEE;
use IEEE.STD _ LOGIC _ 1164.all;
entity Sposob _ 5 is
  port (x: in std _ logic _ vector (1 to 5);
        c: out std _ logic _ vector (1 to 4));
  end;
architecture BEH of Sposob _ 5 is
begin
c <= "1101" when x = "00000" else
     "1001" when x = "00101" else
     "0110" when x = "01010" else
     "0100" when x = "01100" else
     "0010" when x = "10001" else
     "1010" when x = "10110" else
     "0000" when x = "11000" else
     "1100" when x = "00001" else
     "1110" when x = "00110" else
     "0111" when x = "01000" else
     "0101" when x = "01101" else
     "0011" when x = "10010" else
     "1111" when x = "10100" else
     "0001" when x = "11001" else
     "1011" when x = "00010" else
     "1000" when x = "00100" else
     "----";
end BEH;
```

Результатом синтеза оказалась схема со следующими параметрами: $S_{ASIC} = 8002$, $\tau = 3{,}15$ нс, данная схема проигрывает всем вариантам способа 5 (схемам 13—15) по площади и схемам 13, 14 по задержке.

4. Синтез схем обратного преобразования

В табл. 15 приведены результаты синтеза схем для обратного преобразования. Лучшие результаты показал способ 5.

В данном примере лучшим способом по суммарной площади (5 569 + 5 988) схем прямого и обратного преобразования является способ 5, оптимизация в классе ДНФ, хотя доопределение было выбрано в целях минимизации многоуровневого BDD-представления функций.

5. Практический пример

Рассмотрим пример, когда матрица B имеет параметры $n = 17$, $k = 2^{16} = 65\ 536$.

Таблица 15

Результаты синтеза логических схем обратного преобразователя кода (пример 1)

Способ решения задач 2—4	Вид логической минимизации	Сложность функционального описания			Сложность схемы		Номер схемы
		Дизъюнкций	Конъюнкций	Литералов	S_{ASIC}	τ, нс	
1, 2	Нет	24	87	116	12 633	2,58	1, 4
	ДНФ	17	55	77	9804	2,51	2, 5
	BDDI	14	33	66	9893	2,45	3, 6
3	Нет	24	87	116	8934	2,43	7
	ДНФ	11	32	48	7488	2,89	8
	BDDI	11	28	56	7399	2,15	9
4, 5	Нет	24	87	116	7940	*1,98	10, 13
	ДНФ	11	32	*48	*5988	2,94	11, 14
	BDDI	*10	*24	51	6729	2,01	12, 15

Тогда $m = \lceil \log_2 k \rceil = 16$ и требуется построить кодовый преобразователь 17-разрядных двоичных слов в 16-разрядные. Оптимизация, такая же, как в способе 5 (схема 13), однако со случайным присвоением 16-разрядных кодов позволяет получить функциональное BDD-описание, содержащее 8553 двухвходовых конъюнкций, 4119 двухвходовых дизъюнкций и синтезировать схему, содержащую 43 818 транзисторов. Синтез был проведен в библиотеке проектирования системы CMOSLD [13]. Функциональное описание схемы для обратного преобразования (реализация системы СДНФ булевых функций заданной парой булевых матриц (C, B)) содержало 5473 двухвходовых конъюнкций, 2920 двухвходовых дизъюнкций, после BDDI-оптимизации и синтеза была получена логическая схема, содержащей 22 232 транзистора.

Схемная реализация того же примера кодового преобразователя для FPGA xc7k70tfbv676-1 семейства Kintex-7 [14] осуществлялась в системе автоматизированного проектирования Vivado [15], опции синтеза — Vivado Synthesis Default. Сложность схем оценивалась в числе программируемых элементов LUT-6, имеющих шесть входных переменных (LUT — Look-Up Table — таблица, реализующая логическую функцию). Сложность схемы кодового преобразователя составила 2014 LUT-6, сложность схемы обратного преобразования строк матрицы C в строки матрицы B составила 940 LUT-6.

Таким образом, построенный кодовый преобразователь позволил сократить вдвое число тактов для передачи информации по 16-разрядной шине, однако для этого потребовались соответствующие аппаратные затраты.

6. Дополнительные способы решения задач 2, 3

Рассмотренные примеры касались построения кодовых преобразователей, когда требовалось на единицу уменьшить длину кодовых слов $m = n - 1$, а число $k = 2^{n-1}$ и надо было использовать *все* 2^{n-1} различные кодирующие комбинации булевых векторов длины $m = n - 1$.

В случаях, когда k значительно меньше 2^{n-1}, появляется возможность выбора длины m кода и возможность выбора некоторого подмножества кодовых комбинаций, так как не все кодирующие комбинации заданной разрядности надо использовать. Указанные возможности позволяют также уменьшить сложность минимизированных функциональных представлений за счет того, что кодирование k булевых векторов может осуществляться не различными m-разрядными булевыми векторами, а попарно ортогональными *троичными* векторами выбранной длины m, где $\lceil \log_2 k \rceil \leqslant m \leqslant n - 1$.

Пример 2. Пусть матрицу B^2 образуют первые девять строк из уже рассмотренного примера матрицы B:

$$B^2 = \begin{matrix} x_1x_2x_3x_4x_5 \\ 0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,1 \\ 0\,1\,0\,1\,0 \\ 0\,1\,1\,0\,0 \\ 1\,0\,0\,0\,1 \\ 1\,0\,1\,1\,0 \\ 1\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1 \\ 0\,0\,1\,1\,0 \end{matrix}.$$

Тогда $m = \lceil \log_2 9 \rceil = 4$ и требуется использовать только девять кодирующих комбинаций из шестнадцати. Пусть кодирующие троичные векторы образуют троичную матрицу T.

Способ 7. Кодирование строк матрицы B попарно ортогональными m-разрядными троичными векторами. В способе 7 предполагается, что логической минимизации подвергается частичная векторная функция, так как на наборах, принадлежащих множеству $V^x \backslash B$, значения векторной функции не определены. Обозначим B^- через матрицу наборов, задающих множество $V^x \backslash B$ наборов. Пример кодирования строк матрицы B^2 троичными векторами приведен в табл. 16.

Таблица 16

Исходная частичная векторная функция в способе 7 (пример 2)

B^2	T
$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$
0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 1 0	0 0 0 0 – 0 0 1 – 1 1 1 – 0 1 0 – 1 0 0 – 1 1 0 – 0 1 1 1 0 0 0 – 1 0 1
B^-	– – – –

Данная частичная векторная функция может подвергаться логической минимизации программой Minim, результат минимизации — полностью определенная векторная функция — может быть обработан программой BDD_Builder.

Способ 4М (модифицированный). Минимизация в классе BDD частичной многозначной функции, зависящей от булевых переменных. Доопределение BDD "вертикальное", кодирование строк матрицы B соседними троичными векторами. При таком подходе доопределение функций, являющихся моделями функционирования кодового преобразователя, происходит в два этапа: на этапе 1 доопределяется многозначная функция, на этапе 2 — система частичных булевых функций. Кратко проиллюстрируем данный подход на примере матрицы B^2. Доопределенный граф BDD для 9-значной функции показан на рис. 4, где h_i — кофакторы разложения Шеннона компонентных функций c_1, c_2, c_3,

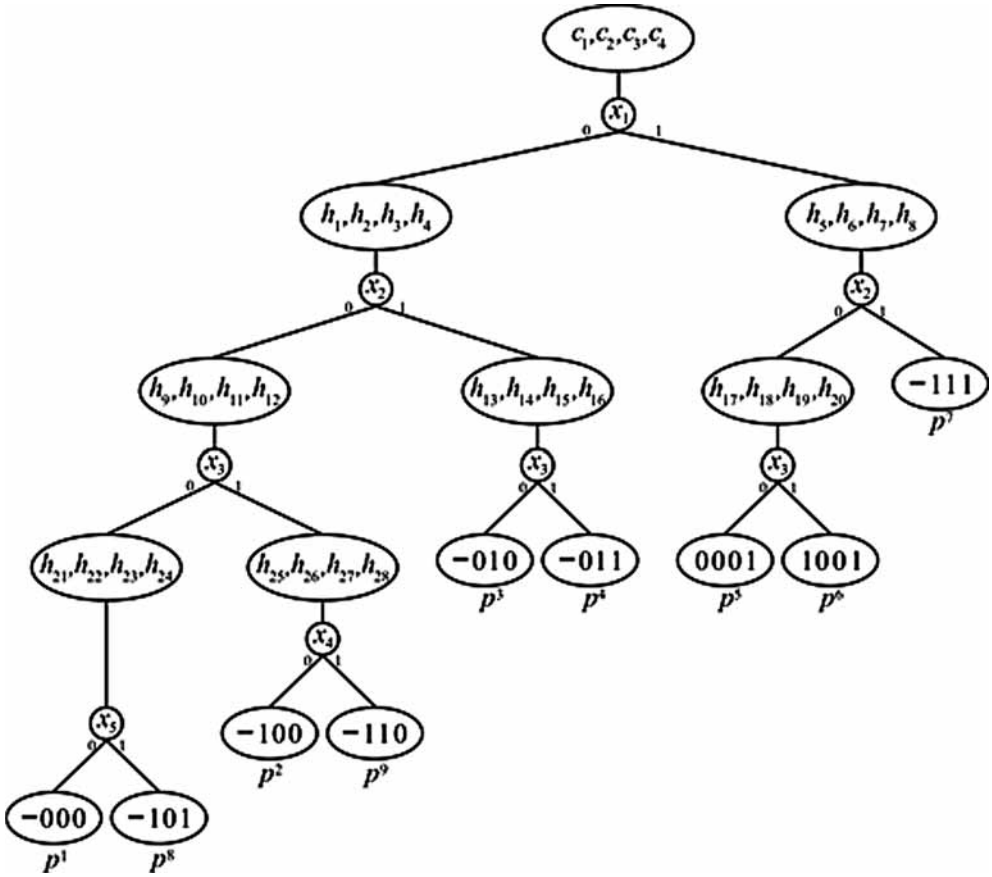


Рис. 4. Доопределенный граф BDD 9-значной функции и кодирование значений троичными векторами

Таблица 17

Частичная векторная функция в способе 4М, полученная кодированием троичными векторами (пример 2)

Компонентная функция	$x_1x_2x_3x_4x_5$	Область определения
c_1	1 0 0 – –	$M^0_{c_1}$
	1 0 1 – –	$M^1_{c_1}$
	0 0 0 – 0 0 0 0 – 1 0 0 1 1 – 0 1 0 – – 0 1 1 – – 1 1 – – –	$M^-_{c_1}$
c_2	0 0 0 – 0 0 1 0 – – 0 1 1 – – 1 0 0 – – 1 0 1 – –	$M^0_{c_2}$
	0 0 0 – 1 0 0 1 0 – 0 0 1 1 – 1 1 – – –	$M^1_{c_2}$
	Ø	$M^-_{c_2}$
c_3	0 0 0 – 0 0 0 0 – 1 0 0 1 0 – 1 0 0 – – 1 0 1 – –	$M^0_{c_3}$
	0 0 1 1 – 0 1 0 – – 0 1 1 – – 1 1 – – –	$M^1_{c_3}$
	Ø	$M^0_{c_3}$
c_4	0 0 0 – 0 0 0 1 0 – 0 0 1 1 – 0 1 0 – –	$M^0_{c_4}$
	0 0 0 – 1 0 1 1 – – 1 0 0 – – 1 0 1 – – 1 1 – – –	$M^1_{c_4}$
	Ø	$M^-_{c_4}$

c_4 (табл. 16). Практически все соседние листовые вершины закодированы соседними троичными векторами. Например, соседние листовые вершины p^2, p^9 получили соседние троичные коды (– 1 0 0), (– 1 1 0) соответственно. Теперь можно по графу BDD частичной векторной функции получить ее матричное задание (табл. 17), рассматривая пути из корневой вершины в листовые вершины каждой из компонентных функций c_1, c_2, c_3, c_4. Такой переход от BDD-представления к матричному представлению частичной функции подробно изложен в работе [4, с. 60].

Логическая оптимизация и полученное в ее результате доопределение частичной векторной функции приводят к доопределению кодов (троичных векторов) до булевых векторов (табл. 18). В табл. 18 в правой части дано минимизированное BDD-описание, содержащее 6 дизъюнкций, 12 конъюнкций и 27 литералов. Синтезированная логическая схема для кодирования строк матрицы B^2 имеет площадь S_{ASIC} = 4073 (условных единиц) и τ = 1,25 нс, что меньше, чем лучшие схемы 13, 14, предназначенные для кодирования строк матрицы B, содержащей 16 строк. Это и следовало ожидать, так как сложность схемы для перекодирования 9 векторов матрицы B^2 и должна быть меньше, чем сложность схемы для перекодирования 16 векторов, содержащихся в матрице B.

7. Методика логического проектирования кодовых преобразователей

Предложенная методика синтеза кодовых преобразователей включает:

- выбор вида описания, предназначенного для описания функций проектируемого преобразователя кодов; были рассмотрены два вида — ДНФ и BDD;
- выбор способа доопределения исходной системы частичных функций на множестве наборов

Таблица 18

Результаты логической оптимизации в способе 4М (пример 2)

Кодирование троичными векторами		Доопределение троичных векторов в результате оптимизации		BDD-представление
$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	$x_1x_2x_3x_4x_5$	$c_1c_2c_3c_4$	

0 0 0 0 0
0 0 1 0 1
0 1 0 1 0
0 1 1 0 0
1 0 0 0 1
1 0 1 1 0
1 1 0 0 0
0 0 0 0 1
0 0 1 1 0
	– 0 0 0
	– 1 0 0
	– 0 1 0
	– 0 1 1
	0 0 0 1
	1 0 0 1
	– 1 1 1
	– 1 0 1
	– 1 1 0
		0 0 0 0 0
		0 0 1 0 1
		0 1 0 1 0
		0 1 1 0 0
		1 0 0 0 1
		1 0 1 1 0
		1 1 0 0 0
		0 0 0 0 1
		0 0 1 1 0
			0 0 0 0
			0 1 0 0
			0 0 1 0
			0 0 1 1
			0 0 0 1
			1 0 0 1
			0 1 1 1
			0 1 0 1
			0 1 1 0
				c1=x1*s3;
				c2=^x1*s4+x1*s1;
				c3=^x1*s3+x1*s1;
				c4=^x1*s2+x1;
				s1=^x3*x2;
				s2=^x3*s4+x3*x2;
				s3=^x2*x4+x2;
				s4=^x2*s5;
				s5=^x4*x5+x4;

$V^x \backslash B$; для каждого вида минимизируемых функциональных описаний были предложены свои способы доопределения;

- способы (эвристики) кодирования наборов матрицы B;
- методы (программы) оптимизации полученной системы функций — раздельная минимизация в классе ДНФ, совместная минимизация в классе BDDI, совместная минимизация в классе BDD частичных функций.

Для различных исходных булевых матриц B и различных библиотек проектирования могут быть эффективными различные способы решения задач 2—4, однако практика проектирования показывает, что во многих случаях минимизация BDD и BDDI-представлений приводит к более простым комбинационным схемам, синтезируемым из библиотечных элементов [4, 8].

Для "обратных" кодовых преобразователей кодирование строк матрицы C уже имеется (это строки матрицы B), поэтому логическая оптимизация сводится к выбору такого доопределения частичной векторной функции, заданной парой матриц (C, B), для которого сложность функционального описания и, соответственно, логической схемы обратного преобразователя была бы по возможности меньшей.

Получение более эффективных схемных реализаций кодовых преобразователей рассматриваемого класса может включать комбинаторный перебор при решении задачи 2 доопределения BDD (перебор различных доопределений) и перебор кодирований строк матрицы B. Для решения задачи 3 могут быть использованы и другие методы оптимизации представлений систем частичных и полностью определенных функций, например, методы (и программы) совместной минимизации функций в классе ДНФ, минимизации BBDD-представлений (BBDD — *Biconditional BDD*) систем функций [16], а также методы дополнительной оптимизации полученных BDD-представлений, выполняемых с помощью замены формул разложения Шеннона более простыми формулами дизъюнктивного либо конъюнктивного разложений [17]. Аппаратная реализация может выполняться как для заказных СБИС, так и для FPGA.

Заключение

Предложена методика проектирования кодовых преобразователей, использующая различные способы составления функциональных описаний кодовых преобразователей в виде систем частичных булевых функций и различные методы логической минимизации таких систем функций. Предложенная методика проектирования кодовых преобразователей рассмотренного в статье класса может быть использована и при проектировании кодовых преобразователей других классов, если описания их функций сводятся к системам частичных булевых функций.

Список литературы

1. **Применение** интегральных микросхем в электронной вычислительной технике: Справочник / Под ред. Б. Н. Файзулаева, Б. В. Тарабрина. М.: Радио и связь, 1987. 384 с.
2. **Червяков Н. И., Сахнюк П. А., Шапошников А. В., Рядов С. А.** Модулярные параллельные вычислительные структуры нейропроцессорных систем. М.: Физматлит, 2003. 288 с.
3. **Brayton K. R., Hachtel G. D., McMullen C., Sangiovanni-Vincentelli A. L.** Logic Minimization Algorithm for VLSI SynthesIs. Boston, Kluwer Academic Publishers, 1984. 193 p.
4. **Бибило П. Н.** Применение диаграмм двоичного выбора при синтезе логических схем. Минск: Беларус. навука, 2014. 231 с.
5. **Бибило П. Н.** Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum. М.: СОЛОН-Пресс, 2005. 384 с.
6. **Авдеев Н. А., Бибило П. Н.** Эффективность логической оптимизации при синтезе комбинационных схем из библиотечных элементов // Микроэлектроника. 2015. Т. 44, № 5. С. 383—399. DOI: 10.7868/S0544126915050026.
7. **Бибило П. Н., Романов В. И.** Система логической оптимизации функционально-структурных описаний цифровых устройств на основе продукционно-фреймовой модели представления знаний // Проблемы разработки перспективных микро- и наноэлектронных систем. Сб. трудов / под общ. ред. акад. РАН А. Л. Стемпковского. М.: ИППМ РАН, 2020. № 4. С. 9—16.
8. **Бибило П. Н., Ланкевич Ю. Ю.** Использование полиномов Жегалкина при минимизации многоуровневых представлений систем булевых функций на основе разложения Шеннона // Программная инженерия. 2017. Том 8, № 8. С. 369—384. DOI: 10.17587/prin.8.369-384.
9. **Брейтон Р. К., Хэчтел Г. Д., Санджованни-Винчентелли А. Л.** Синтез многоуровневых комбинационных логических схем // ТИИЭР. 1990. Т. 78, № 2. С. 38—83.
10. **Mishchenko A.** An Introduction to Zero-Suppressed Binary Decision Diagrams. Berkeley Verification and Synthesis Research Center, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 2014. 15 p.
11. **Kam T., Villa T., Brayton R. K., Sangiovanni-Vincentelli A. L.** Multi-Valued Decision Diagrams for Logic Synthesis and Verification. Memorandum No. UCB/ERL M96/75, 1996. 39 p. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1996/ERL-96-75.pdf
12. **Ashenden P. J., Lewis J.** VHDL-2008. Just the New Stuff. Burlington, MA, USA. Morgan Kaufman Publishers, 2008. 909 p.
13. **Бибило П. Н., Авдеев Н. А., Кардаш С. Н.** и др. Система логического проектирования функциональных блоков заказных КМОП СБИС с пониженным энергопотреблением //Микроэлектроника. 2018. Т. 47, № 1. С. 72—88. DOI: 10.7868/S0544126918010076.
14. **Соловьев В. В.** Архитектуры ПЛИС фирмы Xilinx: FPGA и CPLD 7-й серии. М.: Горячая линия—Телеком, 2016. 392 с.
15. **Тарасов И. Е.** ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. М.: Горячая линия—Телеком, 2020. 538 с.
16. **Amaru L. G.** New Data Structures and Algorithms for Logic Synthesis and Verication. Springer, 2017. 156 p.
17. **Yang S., Ciesielski M.** BDS: a BDD-based logic optimization system // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2002. Vol. 21, No. 7. P. 866—876. DOI: 10.1109/TCAD.2002.1013899.

Hardware Implementation of Code Converters Designed to Reduce the Length of Binary Encoded Words

P. N. Bibilo, bibilo@newman.bas-net.by, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Minsk, 220012, Belarus

Corresponding author:

Petr N. Bibilo, Head of Laboratory, The United Institute of Informatics Problems of the National Academy of Sciences of Belarus, 220012, Minsk, Belarus
E-mail: petr.olibib@yandex.ru, bibilo@newman.bas-net.by

Received on June 09, 2022
Accepted on June 27, 2022

The problem of synthesis of combinational circuits of code converters designed to reduce the length of words from a given set of encoded binary words is considered. The encoding assumes that different binary words will be encoded by different binary codes of shorter length. Code converters of this type are designed to reduce the length of binary words transmitted in digital systems over data buses when the bit depth of the transmitted words exceeds the bit depth of the data bus. For example, 18-bit or 17-bit words need to be transmitted over a 16-bit data bus. Each such word can be transmitted in two cycles of operation of a digital system, however, this approach reduces the overall performance of the system. One of the approaches to solve such problems is the development of combinational circuits that convert long binary encoded words into shorter ones.

The proposed methods for solving the problem of synthesizing circuits of code converters are based on the compilation and logical minimization of such forms of systems of incompletely specified Boolean functions as disjunctive normal forms (DNF) and binary decision diagrams (BDD). Using BDD to minimize representations of k-valued functions that depend on Boolean variables is also proposed. Technologically independent logical minimization of functional descriptions of the designed code converters is proposed to be performed by programs for minimizing systems of Boolean functions in the DNF class and programs for joint minimization of BDD representations of systems of completely specified Boolean functions. Minimization of functional descriptions is aimed at reducing the hardware complexity of combinational circuits in the basis of library elements or programmable FPGA elements implementing code converters of the class in question.

Keywords: *code converter, system of Boolean functions, Disjunctive Normal Form, Binary Decision Diagram, Shannon expansion, digital logic synthesis, VHDL, VLSI*

For citation:

Bibilo P. N. Hardware Implementation of Code Converters Designed to Reduce the Length of Binary Encoded Words, *Programmnaya Ingeneria,* 2022, vol. 13, no. 8, pp. 363—382.

DOI: 10.17587/prin.13.363-382

References

1. ***Application** of integrated circuits in electronic computing*: Reference / Edited by B. N. Fayzulaev, B. V. Tarabrin. Moscow, Radio and Communications. 1987, 384 p. (in Russian).

2. **Chervyakov N. I., Sahnyuk P. A., SHaposhnikov A. V., Ryadnov S. A.** *Modular Parallel Computing Structures of Neuroprocessor Systems*, Moscow, Fizmatlit, 2003, 288 p. (in Russian).

3. **Brayton K. R., Hachtel G. D., McMullen C., Sangiovanni-Vincentelli A. L.** *Logic Minimization Algorithm for VLSI Synthesis,* Boston, Kluwer Academic Publishers, 1984, 193 p.

4. **Bibilo P. N.** *Application of Binary Decision Diagrams in the Synthesis of Logic Circuits,* Minsk, Belaruskaja navuka, 2014, 231 p. (in Russian).

5. **Bibilo P. N.** *Integrated Circuit Design Systems Based on the VHDL Language. StateCAD, ModelSim, LeonardoSpectrum,* Moscow, SOLON-Press, 2005, 384 p. (in Russian).

6. **Avdeev N. A., Bibilo P. N.** Logical optimization efficiency in the synthesis of combinational circuits, *Mikroelektronika,* 2015, vol. 44, no. 5, pp. 383—399. DOI: 10.7868/S0544126915050026 (in Russian).

7. **Bibilo P. N., Romanov V. I.** The system of logical optimization of functional structural descriptions of digital circuits based on production-frame knowledge representation model, *Problemy razrabotki perspektivnyh mikro- i nanoelektronnyh sistem*, 2020. Sb. trudov / pod obshch. red. akad. RAN A. L. Stempkovskogo. Moscow, IPPM RAN, 2020, no. 4, pp. 9—16.

8. **Bibilo P. N., Lankevich Yu. Yu.** The use of Zhegalkin polynomials in minimizing multilevel representations of systems of Boolean functions based on the Shannon expansion, *Programmnaya ingeneria*, 2017, vol. 8, no. 8, pp. 369—384. DOI: 10.17587/prin.8.369—384 (in Russian).

9. **Brayton R. K., Hachtel G. D., Sangiovanni-Vincentelli A. L.** Synthesis of multi-level combinational logic circuits, *Trudy Institute inzhenerov po jelektronike i radiotehnike*, 1990, vol. 78, no. 2, pp. 38—83 (in Russian).

10. **Mishchenko A.** An Introduction to Zero-Suppressed Binary Decision Diagrams. Berkeley Verification and Synthesis Research Center, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 2014, 15 p.

11. **Kam T., Villa T., Brayton R. K., Sangiovanni-Vincentelli A. L.** Multi-Valued Decision Diagrams for Logic Synthesis and Verification. Memorandum No. UCB/ERL M96/75, 1996. 39 p., available at: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1996/ERL-96-75.pdf

12. **Ashenden P. J., Lewis J.** *VHDL-2008. Just the New Stuff.* Burlington, MA, USA. Morgan Kaufman Publishers, 2008, 909 p.

13. **Bibilo P. N., Avdeev N. A., Kardash S. N.** et al. System of Logical Design of Functional Blocks of Custom CMOS VLSI with Reduced Power Consumption, *Mikroelektronika*, 2018, vol. 7, no. 1, pp. 72—88. DOI: 10.7868/S0544126918010076 (in Russian).

14. **Solovyov V. V.** *XILINX FPGA Architectures: FPGA and CPLD 7-Series*, Moscow, Goryachaya liniya Telekom, 2016, 392 p. (in Russian).

15. **Tarasov I. E.** *XILINX FPGA. Hardware Description Languages VHDL and Verilog, CAD, Design Techniques*, Moscow, Goryachaya liniya Telekom, 2020, 538 p. (in Russian).

16. **Amaru L. G.** *New Data Structures and Algorithms for Logic Synthesis and Verification*. Springer, 2017. 156 p.

17. **Yang S., Ciesielski M.** BDS: a BDD-based logic optimization system, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 2002, vol. 21, no. 7, pp. 866—876. DOI: 10.1109/TCAD.2002.1013899.

УДК 004.5
DOI: 10.17587/prin.13.383-393

Д. К. Левоневский, канд. техн. наук, зав. лабораторией, levonevskij.d@iias.spb.su,
А. И. Мотиенко, канд. техн. наук, ст. науч. сотр., anna.gunchenko@gmail.com,
Санкт-Петербургский Федеральный исследовательский центр Российской академии наук

Модели сценариев функционирования медицинской киберфизической системы в штатных и экстренных ситуациях

Представлен анализ существующих решений в области медицинских киберфизических систем, рассмотрены отдельные их компоненты и средства мониторинга состояния пациентов. Установлено, что существуют отдельные разработки в этой области, однако отсутствует комплексный подход к организации киберфизического окружения пациентов в больницах и других стационарных учреждениях.

В ходе исследования изучены некоторые штатные и экстренные сценарии функционирования медицинских киберфизических систем, механизмы которых обеспечивают взаимодействие пациентов с такими системами из дома, а также при их нахождении в больницах и различных других стационарных учреждениях. Выполнено моделирование таких сценариев взаимодействия на логическом уровне их описания. Для моделирования сценариев построены диаграммы активности, структуры данных описаны с помощью диаграмм классов. Представлена обобщенная архитектура медицинской киберфизической системы. Полученные решения можно использовать при проектировании и разработке медицинских киберфизических систем удаленного взаимодействия с пользователем и их составных частей, реализующих мониторинг состояния здоровья пациентов, сбор и визуализацию данных о здоровье, решение вспомогательных задач. Применение предложенных решений предоставит возможность медицинским специалистам более оперативно получать и оценивать информацию, что в свою очередь приведет к повышению качества оказываемых медицинских услуг.

Ключевые слова: *медицинская киберфизическая система, умное пространство, человеко-машинное взаимодействие, проектирование*

Введение

Современный мир все больше полагается на успешное сочетание цифровых и физических систем для выполнения сложных задач автоматизации и управления процессами во многих сферах деятельности человека. Разработка подобных киберфизических систем (КФС) и обеспечение их эффективности, надежности и безопасности в настоящее время являются важной областью исследований. К подобным системам можно отнести интеллектуальные сети, системы управления "умным" транспортом, интеллектуальное производство и сельское хозяйство, "умный" дом, "умные" здания и сообщества, а также интеллектуальное медицинское оборудование. Существуют масштабные и комплексные решения, называемые "умными" городами.

Высокие темпы развития КФС обеспечили будущее персонализированной медицины, которая является одним из приоритетных направлений в Стратегии развития медицинской науки в Российской Федерации на период до 2025 г. Медицинские киберфизические системы (МКФС) представляют собой важный для здравоохранения способ интеграции вычислительных и медицинских устройств. Медицинские киберфизические системы — это контекстно-зависимые, жизненно важные системы, главной задачей которых является безопасность пациентов. Деятельность таких систем требует строгих процессов проверки для гарантий их соответствия требованиям пользователя и точности, ориентированной на принятые спецификации. Они обеспечивают развитие персонализированной медицины в рамках направления 4П-медицины, которая построена на

принципах индивидуального подхода к здоровью человека и включает в себя следующие понятия:

1) персонализация — индивидуальный подход к каждому пациенту с учетом генетических, биохимических и физиологических особенностей человека;

2) предикция — создание вероятностного прогноза здоровья (выявление предрасположенности к развитию заболевания);

3) превентивность — предотвращение или снижение риска развития заболевания в будущем;

4) партисипативность — мотивированное участие пациента в профилактике возможных заболеваний и их лечении.

Медицинские киберфизические системы в развитых странах все чаще используются в больницах для обеспечения разных аспектов непрерывного высококачественного медицинского обслуживания. Подобные технологии позволяют выполнять мониторинг состояния здоровья человека и в экстренных случаях предупреждают медицинских работников о необходимости неотложной помощи при ведении пациента. Согласно работе [1], традиционные клинические сценарии реализуются в системах с обратной связью, в которых лица, осуществляющие уход, выполняют роль контроллеров, медицинские устройства действуют как датчики и исполнительные механизмы, а пациенты с точки зрения организации работы системы выполняют роль физических объектов. Как отдельный класс, МКФС модифицируют этот сценарий, вводя дополнительные вычислительные объекты, чтобы помочь лицу, осуществляющему уход, в поддержке принятия решений [2].

Комплексный подход к организации киберфизического окружения палат в больницах, клиниках, интернатах, в домах престарелых и иных стационарных учреждениях с учетом современных требований к организации пребывания пациентов в стационаре, а также на дому у пациента основан на использовании современных технологий в области человеко-машинного взаимодействия, интеллектуального видеонаблюдения, обработки гетерогенных данных, интеллектуальных пространств.

Целью исследования, результаты которого представлены в настоящей статье, является расширение функциональных возможностей МКФС в части интеграции поведения разнородных компонентов при появлении различных ситуаций, возникающих в процессе ухода за пациентом. Для достижения этой цели авторами выполняется разработка формальных моделей сценариев функционирования МКФС, основанных на возможных вариантах изменения физического состояния пользователей-пациентов и реагирования пользователей-специалистов. Поставленная цель предполагает проектирование технологических решений для реализации предлагаемого подхода в соответствии с парадигмами создания систем нового поколения, принятыми Европейским Консорциумом Периферийных Вычислений (EECC) в рамках "Industry 4.0" [3—5].

Обзор существующих решений в области медицинских киберфизических систем

В ходе исследования были рассмотрены отдельные аспекты МКФС — физиологические датчики, решения для "умных" пространств, и, в частности, "умных" палат, а также различные интегрированные решения для построения МКФС. Существующие системы мониторинга пациента предназначены, как правило, для анализа физического (физиологического) состояния различных групп населения, в первую очередь — для пожилых людей [6—12]. Так, в работе [6] предложена система определения факта падения человека на основе многомодальных беспроводных сенсорных сетей, включающих датчики температуры, движения и биосенсоры, видеокамеры, аудиодатчики и RFID-метки, которые позволяют идентифицировать объекты с помощью радиосигналов. Для повышения точности обнаружения падений и минимизации ложных срабатываний сенсоры используются совместно. Метки RFID применяют для отслеживания местоположения людей и активации видеокамер в помещениях, где последние находятся. Видеокамеры обеспечивают наблюдение и распознавание действий людей. Биосенсоры измеряют, например, артериальное давление или частоту сердечных сокращений, контролируя состояние здоровья. Датчики окружающей среды используют для предоставления контекстной информации об окружающей среде. Система позволяет осуществлять удаленный мониторинг ее пользователей посредством мобильных и локальных вычислительных сетей [7].

В работе [8] представлена концепция "умной" палаты с системой обнаружения падений, которая может подавать сигнал тревоги медицинским работникам при обнаружении падения. Она состоит из камеры глубины Kinect и алгоритма обнаружения падений на основе нейронной сети.

В работе [9] исследуются преимущества использования мобильной 3D-технологии визуализации для расширения мероприятий по оценке состояния окружающей среды, направленных на

преодоление внешних факторов риска падения в домашних условиях.

Беспроводная домашняя система мониторинга [10] включает в себя монитор состояния человека, носимый на запястье, и опционально пульсоксиметры или мониторы артериального давления, подключаемые по беспроводной сети. Такие средства позволяют собирать данные о деятельности и местоположении человека, распознавать падения, выдавать оповещения о паническом состоянии.

В работе [11] предложена система удаленного мониторинга здоровья пожилых людей на основе "умного дома", состоящая из "умной одежды", считывающей параметры сердцебиения и движения человека, и шлюза "умного дома", предназначенного для мультиплексирования и передачи данных на сервер здравоохранения. Предлагаемая система обладает хорошей масштабируемостью и простотой эксплуатации.

В России аналогом рассмотренных систем является автоматизированная система дистанционного мониторинга здоровья человека "Монитор здоровья" [12]. Такая система позволяет автоматически считывать данные с различных специализированных приборов анализа параметров здоровья человека. При необходимости пользователь может предоставить врачу доступ к хранящимся в системе измерениям.

К более сложным и интегрированным решениям относится решение, выполненное в рамках концепции "умной" палаты, которая предложена в работе [13]. Она позволит реализовать сетевой мониторинг состояния пациента с помощью физиологических датчиков, обеспечивающих мониторинг в реальном времени на медицинском планшете врача, а также поддерживающих двустороннюю связь с медперсоналом с помощью мобильной телемедицинской диагностической системы. Разработка медицинской информационной системы для управления "умной" палатой и наблюдения за пациентами в режиме реального времени и соответствующего программного обеспечения даст возможность врачу контролировать их состояние с помощью медицинских планшетов или стационарных компьютеров. Такая система может предоставить следующие возможности:

1) мониторинг состояния пациентов в режиме реального времени с помощью прикроватных мониторов или специальных датчиков, установленных в "умной" палате;

2) заказ еды или лекарств, прописанных врачом, и возможность оставлять запросы на другие медицинские услуги;

3) управление внутренней средой палаты с планшета пациента;

4) контроль и планирование терапии;

5) мобильная лабораторная диагностика с использованием мобильного телемедицинского диагностического комплекса (МТДК);

6) обратная связь с пациентом, телемедицинское обследование пациента с помощью медицинского персонала, использующего МДТК в больницах, клиниках, поликлиниках и дома;

7) подключение к wi-fi, ТВ.

Еще одним примером интегрированной системы является "Электронная система больницы Bundang для полного ухода", реализованная в клинике Сеульского национального университета (Южная Корея) Bundang Hospital, которая является признанным лидером среди "умных" больниц. Реализованная в этой больнице цифровая система BESTCare представляет собой интегрированную систему, включающую:

— электронную медицинскую карту;

— системы компьютерного заказа назначений и рекомендаций предупреждающего характера (когда врачебные назначения несут в себе определенные риски) для врачей и медсестер;

— систему поддержки клинических решений и управление движением лекарственных средств;

— хранилище клинических данных;

— обмен медицинской информацией и аварийное восстановление [14].

Программный продукт Умная палата SMART ROOM от компании "Аналитика М" позволяет управлять комплексом действий, которые могут происходить в палате пациента, а именно: усилить персональный контроль за пациентом; снизить риски возникновения форс-мажорных ситуаций и при этом снизить нагрузку на медицинский персонал. С помощью "умной" панели управления — планшета, как пациент, так и медицинский персонал могут управлять следующими функциями в палате:

— светом, жалюзи, телевидением, климат-контролем;

— кнопкой вызова в режиме "форс-мажор";

— обратной громкой связью с постом медсестры;

— контрольным пунктом вызова в ванной комнате, туалете;

— движением кровати;

— сохранением в памяти истории вызовов и др.

Благодаря специальному фитнес-браслету пациента медицинский специалист может дистанционно проводить мониторинг его состояния и незамедлительно реагировать при резких изменениях

показателей. Специальные кнопки вызова "вода", "обезболивание", "острая боль" и т. д. позволяют медицинскому персоналу быстро реагировать на конкретный вызов и быть готовыми к конкретному запросу пациента [15].

Искусственный интеллект позволил совершить качественный скачок качественный скачок в медицинских технологиях. Его применение позволяет решать сложные задачи с множеством приложений в областях с огромным объемом данных [16]. Интеллектуальные медицинские технологии, основанные на искусственном интеллекте, позволяют использовать концепцию 4П-медицины, которая требует применения современных диагностических технологий и использования основанных на их результатах алгоритмов принятия решений врачами в их деятельности.

Таким образом, ряд вопросов, связанных с построением МКФС, является предметом исследования многих ведущих научных коллективов. Для этого существует много технологий, идей и подходов, разработаны концептуальные модели и архитектуры для МКФС различного назначения [17]. При этом часть существующих решений имеют закрытые реализации, как следствие, не находит достаточного отражения вопрос моделирования сценариев функционирования таких интегрированных МКФС. Таким образом, при наличии отдельных разработок в этой области на настоящее время отсутствует целостный, комплексный, отраженный в научных публикациях подход к организации киберфизического окружения пациентов в больницах, клиниках, интернатах, домах престарелых, других стационарных учреждениях и иных местах нахождения пациентов. Такой подход призван учитывать современные требования к организации ухода за пациентом и поддерживать интеграцию способов и систем мониторинга здоровья пациентов, отслеживать экстренные ситуации, аналитику медицинских данных и поддержку вспомогательных обслуживающих функций.

Сценарии функционирования медицинской киберфизической системы в штатных и экстренных ситуациях

Необходимым условием функционирования МКФС является спецификация ее поведения. Поведение системы может быть задано с помощью сценариев ее функционирования, которые задаются формальными моделями. В данной работе используются модели в нотации UML 2.0. Возможные сценарии функционирования информационной системы сбора и визуализации данных МКФС можно разделить на следующие категории:

— сценарии системы для пациентов при нахождении на дому;

— сценарии системы для пациентов при их нахождении в больницах, клиниках, интернатах, домах престарелых и иных стационарных учреждениях.

Каждая из представленных выше категорий предполагает функционирование в штатной и экстренной ситуациях. Сценарии функционирования можно описать с помощью диаграмм последовательности, в которых объектами являются компоненты МКФС и акторы: в простейшем случае — пациент и врач, в более сложных сценариях могут присутствовать и другие акторы, например, медсестры, технический персонал.

На рис. 1 представлена общая диаграмма последовательностей действий для предварительного взаимодействия пациента и врача с учетом использования датчиков физиологических показателей организма пациента и системы многомодального ввода информации для него. Взаимодействие предполагает регулярный предварительный сбор данных о состоянии пациента в автоматическом (с помощью датчиков), полуавтоматическом (с помощью средств многомодального ввода информации в клиентском приложении пациента) и ручном (с помощью медицинского персонала) режимах. Эти данные поступают на сервер, систематизируются и визуализируются в клиентском приложении врача, что позволяет ему более оперативно оценить состояние пациента и принять решение о том, насколько срочно необходима очная консультация, и о необходимости неотложной помощи (вызова бригады).

К экстренным сценариям относится возникновение неотложных состояний у пациента, под которыми понимается совокупность симптомов (клинических признаков), требующих оказания неотложной первой медицинской помощи, либо госпитализации пострадавшего или пациента [18]. В случае экстренного сценария, когда пациент находится дома, необходим вызов бригады скорой помощи, в стационаре — вызов специалиста непосредственно в палату пациента. Различают следующие состояния, требующие оказания скорой помощи в *экстренной* форме:

— нарушения сознания, представляющие угрозу жизни;

— нарушения дыхания, представляющие угрозу жизни;

— нарушения системы кровообращения, представляющие угрозу жизни;

Рис. 1. Диаграмма последовательности для предварительного взаимодействия пациента и врача

— психические расстройства, сопровождающиеся действиями пациента, представляющими непосредственную опасность для него или других лиц;

— внезапный болевой синдром, представляющий угрозу жизни;

— внезапные нарушения функции какого-либо органа или системы органов, представляющие угрозу жизни;

— травмы любой этиологии, представляющие угрозу жизни;

— термические и химические ожоги, представляющие угрозу жизни;

— внезапные кровотечения, представляющие угрозу жизни;

— роды, угроза прерывания беременности;

— дежурство при угрозе возникновения чрезвычайной ситуации, оказание скорой медицинской помощи и медицинская эвакуация при ликвидации медико-санитарных последствий чрезвычайной ситуации.

Поводом для вызова скорой медицинской помощи в *неотложной* форме являются:

— внезапные острые заболевания (состояния) без явных признаков угрозы жизни, требующие срочного медицинского вмешательства;

— внезапные обострения хронических заболеваний без явных признаков угрозы жизни, требующие срочного медицинского вмешательства.

Под явными признаками угрозы жизни понимаются выраженные проявления заболевания (состояния), которые могут привести к смерти пациента. Перечисленные в табл. 1 состояния могут не угрожать жизни явно, однако требуют оказания помощи в целях предотвращения значительного и долгосрочного воздействия на физическое или психическое здоровье человека, оказавшегося в таком состоянии [19].

Необходимые действия со стороны сервера можно представить диаграммой активности на рис. 2. Эта диаграмма также отражает процесс контроля показателей на сервере, что позволяет выявить экстренные сценарии развития ситуации, например, сильное проявление аллергической реакции, такое как ангионевротический отек (отек Квинке), которое часто требует неотложной помощи. Его основная опасность состоит в быстром сужении просвета носовых ходов и гортани отекшими тканями и вследствие этого — резкое затруднение дыхания. При несвоевременном обращении может привести к летальному исходу.

Штатный сценарий функционирования системы при нахождении пациента дома предполагает ведение дневника пациента с регулярными записями о его состоянии. В системе могут быть настроены напоминания о внесении информации о текущем состоянии, а также предусматривается запись к врачу на ближайшее доступное время. При появлении новых симптомов, пациент в любое время может внести их в дневник. При этом возможна связь системы с "умной" колонкой с голосовым помощником, который сможет вносить записи о состоянии пользователя в дневник пациента, периодически (период можно настроить)

Таблица 1

Классификация неотложных состояний

Раздел	Подраздел
Неотложная хирургия	Раны
	Кровотечения
	Травмы и повреждения
	Шок
	Ожоги
	Отморожения
Неотложные состояния в клинике глазных болезней	—
Неотложные состояния в клинике болезней уха, горла и носа	—
Неотложные состояния в урологии	—
Неотложные состояния в клинике внутренних болезней	Острые аллергические реакции
	Неотложные состояния в клинике инфекционных болезней
	Неотложные состояния в кардиологии
	Неотложные состояния в пульмонологии
	Неотложные состояния в токсикологии и радиационной медицине
	Неотложные состояния в клинике эндокринных болезней
Неотложные состояния в акушерстве и гинекологии	—
Неотложные неврологические и нейрохирургические состояния	—
Неотложные состояния в психиатрии	—
Неотложные состояния в клинике детских болезней	—

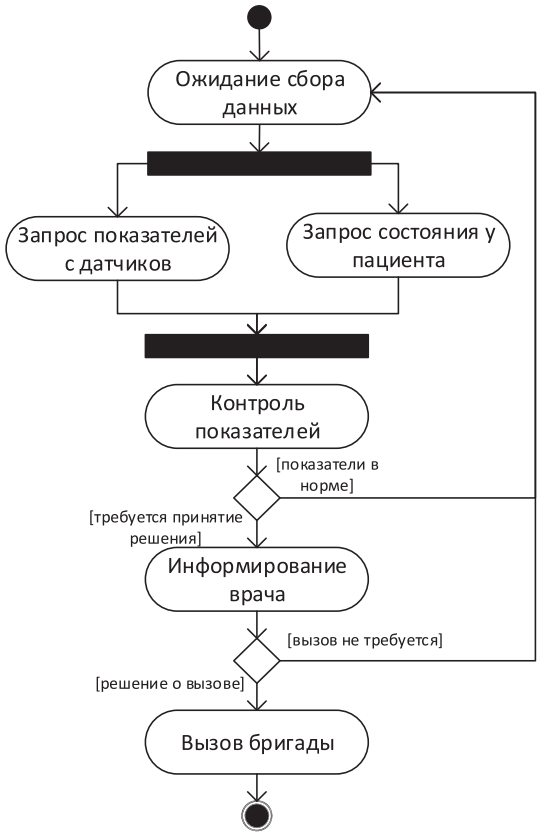


Рис. 2. Диаграмма активности для сервера при предварительном взаимодействии пациента и врача

уточнять информацию об изменениях в состоянии, напоминать о необходимости записи к врачу или приеме лекарственных средств и т. п. Помимо симптомов в систему возможен ввод всех действий, которые совершает пациент, чтобы улучшить или стабилизировать свое состояние, а именно, прием лекарственных и нелекарственных средств, различные манипуляции. При значительном ухудшении состояния предусмотрена возможность вызова бригады скорой помощи и передачи всех записей о состоянии.

Штатные сценарии включают действия МКФС в стационаре, выполняемые по указанию пациента (например, регулирование комфортного нахождения в палате — свет, температура, проветривание, вызов персонала т. п.). Если в каких-то ситуациях сценарии конфликтуют, то выполняется сценарий по указаниям врача. Например, проветривание помещения по просьбе пациента может выполняться только в условиях отсутствия запрета на проветривание со стороны врача. Диаграмма последовательности для такого случая изображена на рис. 3. Здесь врач с помощью своего клиентского приложения управляет разрешениями для системы климат-контроля, а пациент может запросить те или иные действия у МКФС, которые выполняются при условии, если они разрешены. Соот-

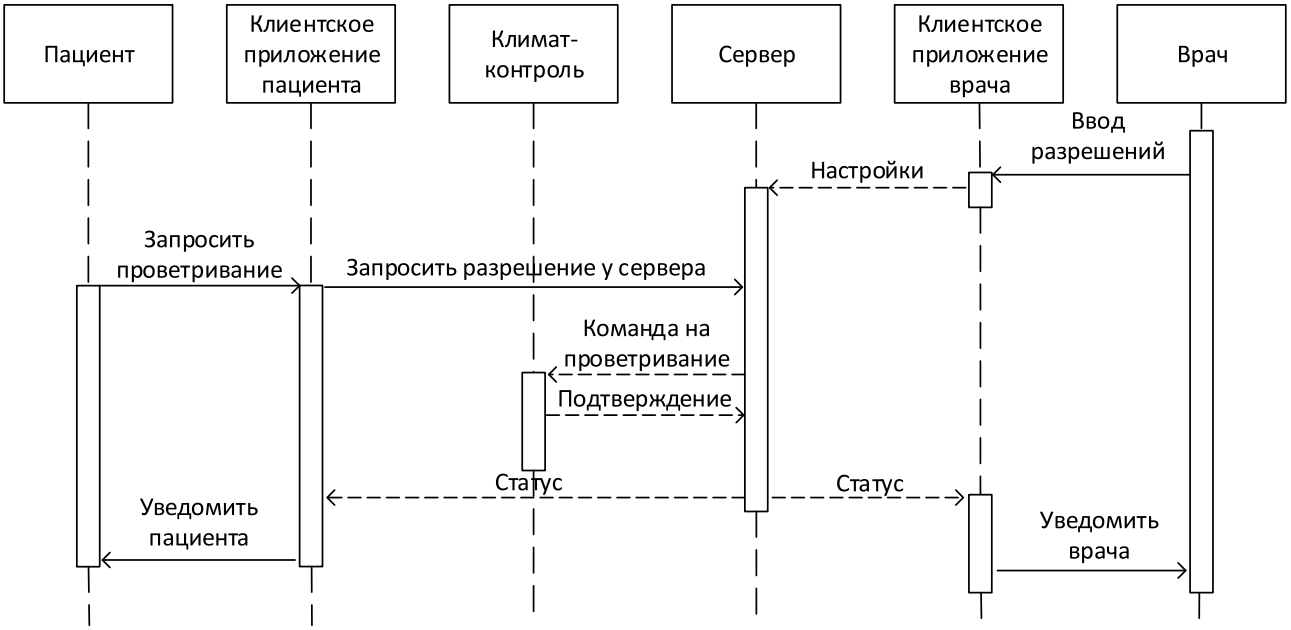


Рис. 3. Диаграмма последовательности для управления проветриванием помещения

ветствующая диаграмма активности изображена на рис. 4.

Таким образом, для некоторых сценариев функционирования МКФС были построены формальные модели в виде диаграмм последовательностей

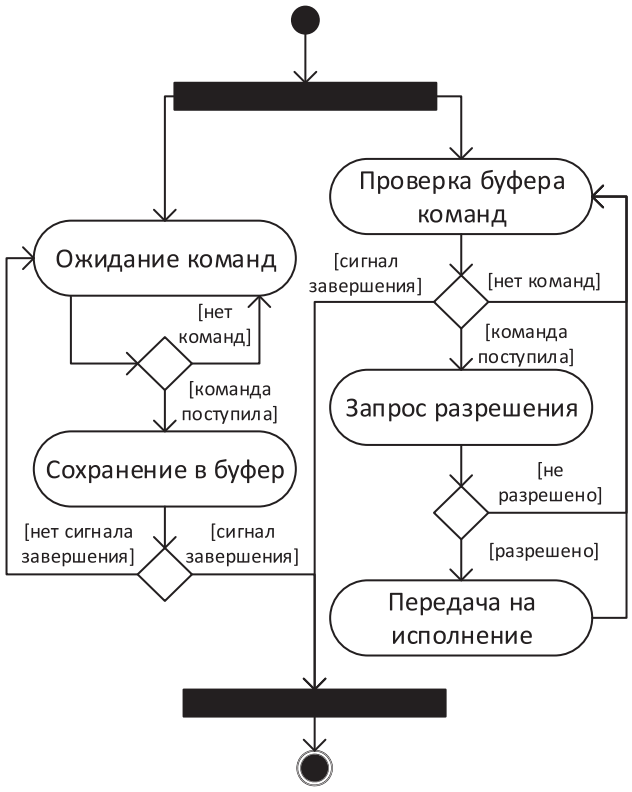


Рис. 4. Диаграмма активности для обработки команд климат-контроля

и активностей. Такие модели, а также аналогичные модели других сценариев и ситуаций могут быть использованы при проектировании составных частей интегрированных решений по уходу за пациентами. В частности, для разработки программного обеспечения строятся диаграммы классов, а для проектирования систем — диаграммы компонентов, которые могут включать известные и предлагаемые решения. Так, в работе [20] предложен ряд решений для подсистемы сбора и визуализации медицинских данных для оптимизации процесса ведения пациента и приведена клиент-серверная архитектура, в рамках которой могут быть реализованы эта и подобные подсистемы. Однако более широкий взгляд на МФКС предполагает не только сбор и визуализацию медицинской информации, но и наличие контура управления. Так, объектно-ориентированная модель в форме диаграммы классов для реализации процесса управления модулями климат-контроля представлена на рис. 5.

Диаграмма описывает множество компонентов МКФС, характеризующихся параметрами, которые пользователи могут регулировать. При этом пользователи, обладающие необходимой ролью (медицинские специалисты), могут устанавливать политики, ограничивающие возможность регулирования параметров для той или иной группы пользователей. В этой диаграмме не выполняется детализация представления множества состояний компонентов МКФС, так как представление этого множества носит технический характер и обе-

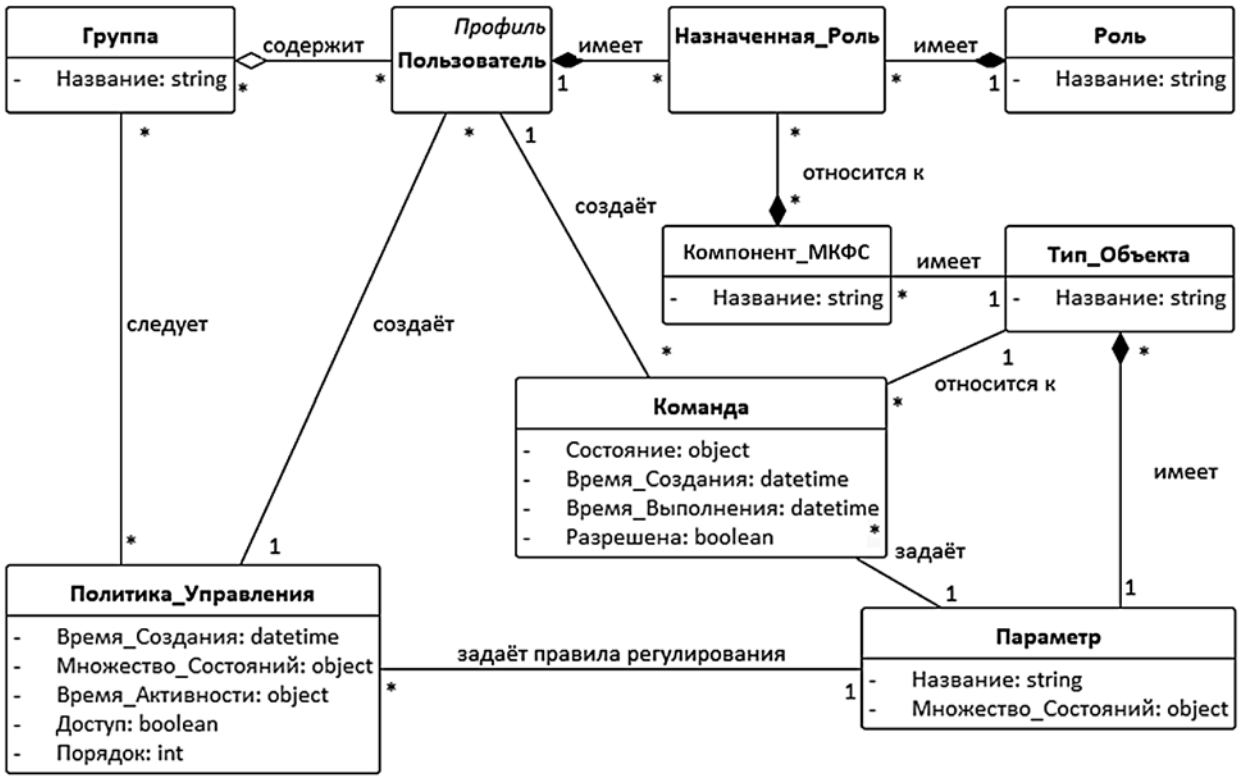


Рис. 5. Диаграмма классов для управления модулями климат-контроля

спечивается стандартными типами и структурами данных (целые числа, интервалы, массивы).

Подобные диаграммы могут использоваться для разработки программного обеспечения и позволяют автоматически создавать каркасы программных модулей в системах автоматизированного проектирования.

В частности, на основе диаграммы классов на рис. 5 в приложении Enterprise Architect были сгенерированы прототипы программных модулей на языке Java. Фрагмент листинга приведен на рис. 6.

Указанные программные компоненты могут использоваться в рамках общей архитектуры МКФС, представленной на рис. 7.

Представленная на рис. 7 архитектура предусматривает разделение компонентов системы на две подгруппы — клиенсткую и серверную. Клиентская часть включает в себя конечные устройства доступа — компьютеры, планшеты, датчики, физические компоненты, которые связываются с серверной частью через API и веб-интерфейс. Серверная часть отвечает за хранение и обработку данных, реализацию описанных сценариев, аналитику и визуализацию. Таким образом, компоненты системы в полной мере обладают функциональными возможностями, которые необходимы для обеспечения взаимодействия пациентов с МКФС.

Время реализации сценария от момента отправки команды до получения обратной связи имеет вид

```
package System;

/**
 * @author Dmitri
 * @version 1.0
 * @created 14-май-2022 12:05:32
 */
public class ManagementPolicy {
    private Date createTime;
    private Object states;
    private Object activityCondition;
    private boolean access;
    private int order;
    public User m_User;
    public Parameter m_Parameter;
    public Group m_Group;

    public ManagementPolicy(){
        // TO DO Construct
    }

    public void finalize() throws Throwable {
        // TO DO Finalize
    }
}
```

Рис. 6. Фрагмент листинга класса, описывающего политику управления

$$t = t_{ph} + t_{\inf} + t_{conn}, \quad (1)$$

где t_{ph} — время реализации физических процессов; t_{inf} — время реализации информационных процессов; t_{conn} — время, затраченное на взаимодействие

Таблица 2

Задержки при выполнении запросов в МКФС

Технология	Среднее значение задержки, мс	Среднее квадратическое отклонение, мс
Задержка сети t_{conn} (Ethernet)	2,8	1,1
Задержка сети t_{conn} (wi-fi)	4,8	1,4
Выполнение типового запроса t_{inf}	227	58,9

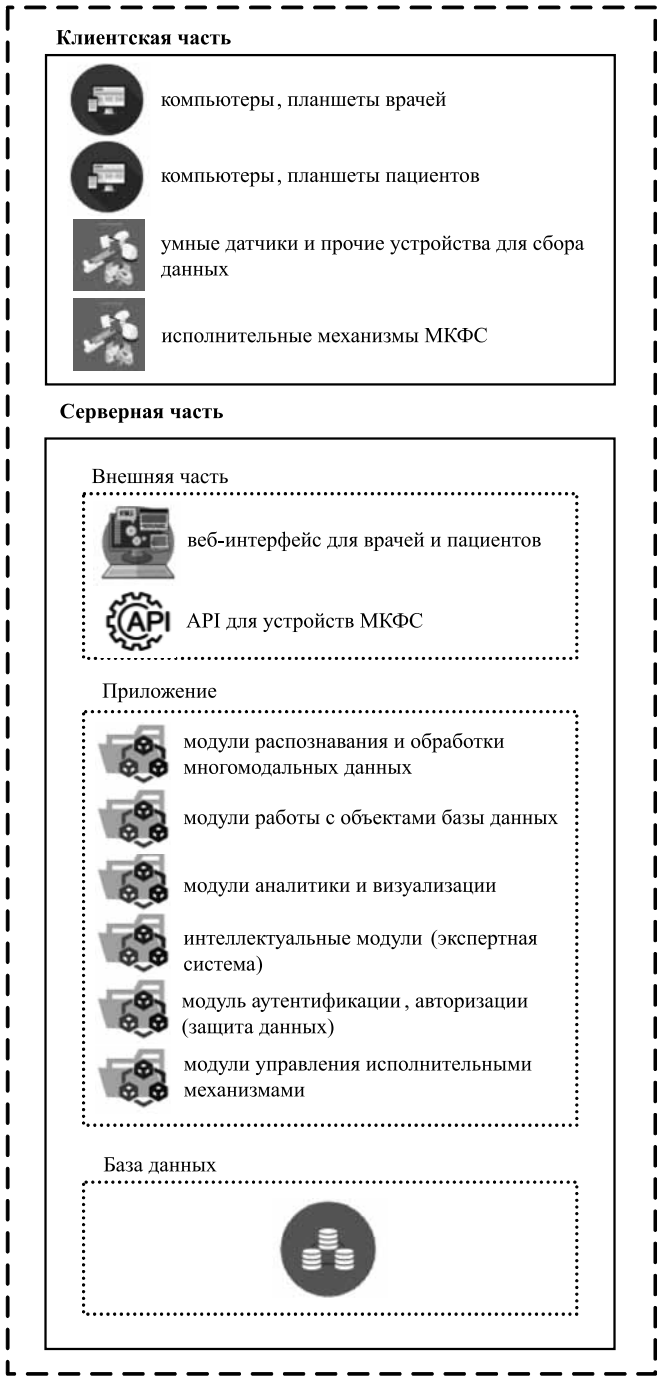


Рис. 7. Архитектура медицинской киберфизической системы

между компонентами, которое определяется задержками в передаче данных по сети. Указанное время не должно превышать некоторых граничных значений, которые определяются для экстренных ситуаций специальными требованиями, для штатных — инженерными нормативами, которые касаются удобства использования систем и времени реакции пользовательских интерфейсов.

К примеру, сценарий обращения к модулю климат-контроля от выдачи команды до получения подтверждения занимает шесть шагов. Реализация физического процесса в этом сценарии происходит параллельно с другими процессами, поэтому слагаемое t_{ph} в формуле (1) можно не учитывать. Измеренные задержки при нахождении компонентов в локальной сети приведены в табл. 2.

Таким образом, рассмотренное взаимодействие реализуется за приемлемое время. Кроме того, процесс в лучшем случае удовлетворяет критерию восприятия взаимодействия пользователем как непрерывного процесса (задержки до 1 с), в худшем случае — критерию, при котором пользователь может сохранять концентрацию внимания на процессе взаимодействия (задержки до 10 с) [21].

Заключение

В ходе исследования были изучены некоторые штатные и экстренные сценарии функционирования МКФС, было выполнено их моделирование в UML на логическом уровне. Для моделирования процессов построены диаграммы активности, структуры данных описаны с помощью диаграмм классов. Приведена обобщенная архитектура МКФС.

Полученные решения возможно использовать при проектировании и разработке МКФС и их составных частей, реализующих мониторинг состояния здоровья пациентов, взаимодействие с ними, сбор и визуализацию данных о здоровье для медицинского персонала. Включение этих компонентов в контур принятия врачебных решений позволит специалистам более оперативно получать и оценивать информацию, что в свою очередь приведет к повышению качества оказываемых медицинских услуг. Использование МКФС также позволяет упростить выполнение широкого круга вспомогательных задач средствами "умной" палаты, в том числе — задачи управления климатом, оснащением палаты, оперативной связи с персоналом, доставки, уборки и т. п.

Дальнейшие направления работы могут включать в себя создание общего классификатора сценариев функционирования МКФС, разработку

моделей и архитектур программного обеспечения для таких систем не только на концептуальном и логическом, но и на физическом уровне, и оценку их эффективности.

Список литературы

1. **Lee I., Sokolsky O., Chen S.** et al. Challenges and research directions in medical cyber—physical systems // Proceedings of the IEEE. 2011. Vol. 100, No. 1. P. 75—90. DOI: 10.1109/JPROC.2011.2165270.
2. **Lee E. A.** The past, present and future of cyber-physical systems: A focus on models // Sensors. 2015. Vol. 15, No. 3. P. 4837—4869. DOI: 10.3390/s150304837.
3. **Klitou D., Conrads J., Rasmussen M.** et al. Digital Transformation Monitor Germany: Industrie 4.0. European Commission, European Union. 2017. URL: https://ati.ec.europa.eu/sites/default/files/2020-06/DTM_Industrie%204.0_DE.pdf
4. **Schwab K.** The fourth industrial revolution. NY.: Currency Books, 2017. 192 p.
5. **European** Edge Computing Consortium (EECC). URL: https://ecconsortium.eu/
6. **Alemdar H. Ö., Yavuz G. R., Özen M. O.** et al. Multimodal fall detection within the WeCare framework // Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks. 2010. P. 436—437.
7. **Саитов И. А., Мотиенко А. И., Астапов С. С., Басов О. О.** Синтез топологической структуры распределенной терминальной системы для аудиомониторинга пользователей локальных информационных пространств // Информатика и автоматизация. 2019. Т. 18, № 6. С. 1357—1380. DOI: 10.15622/sp.2019.18.6.1357-1380.
8. **Su M. C., Liao J. W., Wang P. C., Wang C. H.** A smart ward with a fall detection system // 2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe). 2017. P. 1—4.
9. **Hamm J. J.** Technology-assisted healthcare: exploring the use of mobile 3D visualisation technology to augment home-based fall prevention assessments. Ph. D. Thesis. Brunel University London, 2018. 316 p.
10. **Papadopoulos A., Crump C., Wilson B.** Comprehensive home monitoring system for the elderly // Wireless Health 2010. 2010. P. 214—215. DOI: 10.1145/1921081.1921118.
11. **Guan K., Shao M., Wu S.** A remote health monitoring system for the elderly based on smart home gateway // Journal of healthcare engineering. 2017. Vol. 2017. 10 p. DOI: 10.1155/2017/5843504.
12. **Шалковский А. Г., Купцов С. М., Берсенева Е. А.** Актуальные вопросы создания автоматизированной системы дистанционного мониторинга здоровья человека // Врач и информационные технологии. 2016. № 1. С. 67—69.
13. **Orsaeva A. T., Tamrieva, L. A., Mischvelov A. E.** et al. Development of a Prototype of a "Smart Ward" as an Element of a Digital Polyclinic // Pharmacophore. 2020. Vol. 11, No. 1. P. 142—146.
14. **Кобринский Б. А.** "Умная" больница как инструмент цифровой медицины // Информационные технологии и вычислительные системы. 2018. № 4. С. 3—14. DOI: 10.14357/20718632180401.
15. **Компания** "АНАЛИТИКА М" предлагает управление и интеграцию комплексных решений для эффективной работы Вашего ЛПУ. URL: http://www.analitika-m.ru/services/
16. **Peng Y., Zhang Y., Wang L.** Guest editorial: Artificial intelligence in biomedical engineering and informatics: An introduction and review // Artificial intelligence in medicine. 2010. Vol. 48, No. 2—3. P. 71—73. DOI: 10.1016/j.artmed.2009.07.007.
17. **Chen F., Tang Y., Wang C.** et al. Medical Cyber-Physical Systems: A Solution to Smart Health and the State of the Art // IEEE Transactions on Computational Social Systems. 2021. P. 1—21.
18. **Руководство** по скорой медицинской помощи / Под ред. С. Ф. Багненко, А. Л. Верткина, А. Г. Мирошниченко, М. Ш. Хубутия. М.: ГЭОТАР-Медиа, 2012. 783 с.
19. **Смагин А. Ю., Белых Т. Н., Белоусова Т. Н., Девяткина Н. П.** Первая и неотложная медицинская помощь: методическое пособие, 2-е изд., Омск: БУ ДПО ОО ЦПК РЗ, 2018. 84 с.
20. **Левоневский Д. К., Мотиенко А. И.** Информационная система сбора и визуализации медицинских данных для оптимизации процесса ведения пациента // Информатизация и связь. 2021. № 7. С. 21—29. DOI: 10.34219/2078-8320-2021-12-7-21-29.
21. **Henty S.** UI Response Times. URL: https://medium.com/@slhenty/ui-response-times-acec744f3157.

Scenario Models for the Functioning of a Medical Cyber-Physical System in Normal and Emergency Situations

D. K. Levonevskiy, levonevskij.d@iias.spb.su, **A. I. Motienko,** anna.gunchenko@gmail.com,
St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS),
St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, St. Petersburg, 199178, Russian Federation

Corresponding author:

Dmitrii K. Levonevskiy, PhD, Head of Laboratory,
St. Petersburg Federal Research Center of the Russian Academy of Sciences, Saint Petersburg, 199178, Russian Federation
E-mail: levonevskij.d@iias.spb.su

Received on May 05, 2022
Accepted on June 27, 2022

The existing solutions in the field of medical cyber-physical systems, considers their individual components and patient monitoring systems are studied in the paper. It has been established that there are separate developments in this area, but there is no comprehensive approach to organizing the cyber-physical environment of patients in hospitals and various other inpatient institutions.

In the course of the study, some regular and emergency scenarios for the functioning of medical cyber-physical systems have been studied. Its are implemented when patients interact with the system while at home, as well as while in hospitals and various other inpatient institutions. These scenarios have been simulated at the logical level. To model processes, activity diagrams are constructed, data structures are described using class diagrams. The generalized architecture of the medical cyber-physical system is given. The obtained solutions can be used in the design and development of medical cyber-physical systems and their components that monitor the health status of patients, interact with them, collect and visualize health data, and solve auxiliary problems. The application of the proposed solutions will enable medical specialists to receive and evaluate information more quickly, which, in turn, will lead to an increase in the quality of medical services provided.

Keywords: *medical cyber-physical system, smart space, human-machine interaction, decomposition, software design*

For citation:

Levonevskiy D. K., Motienko A. I. Scenario Models for the Functioning of a Medical Cyber-Physical System in Normal and Emergency Situations, *Programmnaya Ingeneria,* 2022, vol. 13, no. 8, pp. 383—393.

DOI: 10.17587/prin.13.383-393

References

1. **Lee I., Sokolsky O., Chen S.** et al. Challenges and research directions in medical cyber—physical systems, *Proceedings of the IEEE*, 2011, vol. 100, no. 1, pp. 75—90. DOI: 10.1109/JPROC.2011.2165270.

2. **Lee E. A.** The past, present and future of cyber-physical systems: A focus on models, *Sensors*, 2015, vol. 15, no. 3, pp. 4837—4869. DOI: 10.3390/s150304837.

3. **Klitou D., Conrads J., Rasmussen M.** et al. Digital Transformation Monitor Germany: Industrie 4.0, European Commission, *European Union*, 2017, available at: https://ati.ec.europa.eu/sites/default/files/2020-06/DTM_Industrie%204.0_DE.pdf

4. **Schwab K.** *The fourth industrial revolution, Currency,* 2017, 192 p.

5. **European** Edge Computing Consortium (EECC), available at: https://ecconsortium.eu/

6. **Alemdar H. Ö., Yavuz G. R., Özen M. O.** et al. Multi-modal fall detection within the WeCare framework, *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2010, pp. 436—437.

7. **Saitov I., Motienko A., Astapov S., Basov O.** Synthesis of the Topological Structure of Distributed Terminal System for Audio Monitoring of Users of Local Information Spaces, *SPIIRAS Proceedings*, 2019, vol. 18, no. 6, pp. 1357—1380. DOI: 10.15622/sp.2019.18.6.1357-1380.

8. **Su M. C., Liao J. W., Wang P. C., Wang C. H.** A smart ward with a fall detection system, *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, 2017, pp. 1—4.

9. **Hamm J. J.** Technology-assisted healthcare: exploring the use of mobile 3D visualisation technology to augment home-based fall prevention assessments, Ph. D. Thesis, Brunel University London, 2018. 316 p.

10. **Papadopoulos A., Crump C., Wilson B.** Comprehensive home monitoring system for the elderly, *Wireless Health 2010*, 2010, pp. 214-215. DOI: 10.1145/1921081.1921118.

11. **Guan K., Shao M., Wu S.** A remote health monitoring system for the elderly based on smart home gateway, *Journal of healthcare engineering*, 2017, vol. 2017, 10 p. DOI: 10.1155/2017/5843504.

12. **Shalkovsky A. G., Kuptsov S. M., Berseneva E. A.** Topical issues of person health remote monitoring automated system creation, *Vrach i informacionnye tehnologii*, 2016, vol. 1, pp. 67—69 (in Russian).

13. **Orsaeva A. T., Tamrieva L. A., Mischvelov A. E.** et al. Development of a Prototype of a "Smart Ward" as an Element of a Digital Polyclinic, *Pharmacophore*, 2020, vol. 11, no. 1, pp. 142—146.

14. **Kobrinskii B. A.** Smart hospital as a tool of digital medicine, *Informacionnye tehnologii i vychislitel'nye sistemy*, 2018, vol. 4, pp. 3—14. DOI: 10.14357/20718632180401 (in Russian).

15. **The company** "ANALITIKA M" offers management and integration of complex solutions for the efficient operation of your healthcare facility, available at: http://www.analitika-m.ru/services/ (in Russian).

16. **Peng Y., Zhang Y., Wang L.** Guest editorial: Artificial intelligence in biomedical engineering and informatics: An introduction and review, *Artificial intelligence in medicine*, 2010, vol. 48, no. 2—3, pp. 71—73. DOI: 10.1016/j.artmed.2009.07.007.

17. **Chen F., Tang Y., Wang C.** et al. Medical Cyber-Physical Systems: A Solution to Smart Health and the State of the Art, *IEEE Transactions on Computational Social Systems*, 2021, pp. 1—21.

18. **Emergency** Medical Guide / Edited by S. F. Bagnenko, A. L. Vertkina, A. G. Miroshnichenko, M. Sh. Hubutiya, Moscow, GJeOTAR-Media, 2012, 783 p (in Russian).

19. **Smagin A. Yu., Belyh T. N., Belousova T. N., Devyatkina N. P.** *First and emergency medical care: tutorial*, 2nd ed. Omsk, BU DPO OO CPK RZ, 2018, 84 p. (in Russian).

20. **Levonevskiy D. K., Motienko A. I.** Information system for collection and visualization of medical data for optimizing the patient management process, *Informatization and communication,* 2021, vol. 7, pp. 21—29. DOI: 10.34219/2078-8320-2021-12-7-21-29 (in Russian).

21. **Henty S.** UI Response Times, available at: https://medium.com/@slhenty/ui-response-times-acec744f3157

УДК 004.056, 004.043
DOI: 10.17587/prin.13.394-403

Д. А. Капустин, канд. техн. наук, доц., kap-kapchik@mail.ru,
В. В. Швыров, канд. физ.-мат. наук, доц., slsh@i.ua,
Т. И. Шулика, ассистент, shulika-tatyana@mail.ru,
Луганский государственный педагогический университет

Статический анализ корпуса исходных кодов Python-приложений

Одним из популярных методов анализа кода программного обеспечения является метод статического анализа. Такой метод позволяет не только проверять код на соответствие спецификации языка, но и находить в коде потенциальные уязвимости. В работе выполнен статический анализ корпуса листингов приложений на Python с открытыми исходными кодами. С использованием библиотеки Bandit найдены статистические показатели различных категорий потенциальных уязвимостей, построена рейтинговая таблица уязвимостей, найденных в исследуемом наборе данных. Проведен качественный анализ угроз на основе данных каталога CWE.

Ключевые слова: *анализ безопасности, большие данные, статический анализ, уязвимость, угроза, bandit, CWE, OWASP, linters, source code analysis, python dataset*

Введение

Вопросы написания безопасного кода, в частности, отсутствия уязвимостей и ошибок в приложениях являются актуальной темой исследования в последние годы. Это связано с растущими вызовами и требованиями эффективной и надежной работы приложений в условиях агрессивной конкуренции между разработчиками с одной стороны и растущего числа хакерских атак с другой стороны [1, 2].

Одним из эффективных методов анализа программного кода является статический анализ кода, который получил свое развитие в работах Кузо [3] и Аллена [4]. Одним из первых приложений, которое выполняло проверку кода языка С с помощью статического анализа, было приложение Lint [5]. Впоследствии класс программных средств, осуществляющих статический анализ, стали называть линтерами.

Методы статического анализа программного кода активно развивались в последнее десятилетие. Об этом свидетельствует значительное число публикаций и обзорных работ по данному направлению [6—11]. Существенный прогресс в развитии и использовании данных методов связан с возросшими вычислительными способностями современных систем, а также необходимостью поиска потенциальных угроз и уязвимостей в программном обеспечении (ПО). Современные подходы в области анализа безопасности веб-приложений представлены в проекте OWASP (*Open Web Application Security Project*) [12].

Следует отметить, что кроме статического анализа используются также методы динамического анализа и методы формальной верификации программ. Динамический метод анализа предполагает мониторинг выполнения программы преимущественно на этапе тестирования, на некотором наборе входных данных. Основная идея фаззинга состоит в том, чтобы "выявить ошибку до тех пор пока она не стала уязвимостью". Выделяют фаззинг-анализаторы, которые используют специально сгенерированные наборы входных данных для тестирования приложения, например, путем мутационного или генеративного фаззинг-тестирования. В качестве примера можно привести утилиту Burp Suite, которую можно использовать для тестирования веб-приложений методом фаззинга. Каждый из методов анализа программного кода имеет свои преимущества и недостатки, в связи с этим на практике желательно использование различных методов.

Значительную роль в развитии методов статического анализа для исследования проблем безопасности приложений сыграли каталоги уязвимостей и дефектов ПО. Одними из наиболее известных таких каталогов являются полученные в рамках

проектов CWE (*Common Weakness Enumeration*) и CVE (*Common Vulnerabilities and Exposures*) [13, 14]. Последний, по состоянию на начало мая 2022 г., насчитывает более 175 000 описанных уязвимостей. С помощью списков CWE и CVE составлен рейтинг OWASP Top 10 [15]. Данный рейтинг приведен в табл. 1.

В связи со стремительным ростом популярности языка Python [16] в различных проектах с открытым исходным кодом и упомянутыми ранее каталогами CVE и CWE, возникает естественный вопрос построения рейтинга уязвимостей и дефектов аналогичного OWASP Top 10 для Python-приложений.

Подобные исследования проводились в ряде работ зарубежных авторов, например, в направлении анализа безопасности кода во фреймворке Django [17], в контексте анализа динамической типизации в Python [18]. В работе [19] проводился анализ уязвимостей в Python-пакетах для разработки веб-приложений. Комплексный и системный анализ безопасности пакетов Python Package Index (PyPI) [20] выполнен в исследовании [21]. Для проверки кода пакетов в контексте настоящей статьи авторы используют статический анализатор Bandit [22]. В отличие от аналогичных работ, в статье представлен рейтинг потенциальных уязвимостей согласно каталогу CWE для проектов с открытым исходным кодом на языке Python, кроме того, полученные результаты сопоставляются с данными рейтинга OWASP Top 10, также выполнен частотный анализ жестко вписанных в программный код паролей.

Целью работы, результаты которой представлены в настоящей статье, являются анализ корпуса листингов Python-приложений с открытыми исходными кодами на предмет наличия потенциальных уязвимостей с использованием статического анализатора и составление рейтинга наиболее распространенных угроз на основании найденных потенциальных уязвимостей.

Таким образом, исследование должно дать ответ на перечисленные далее вопросы.

В1. Какие категории потенциальных уязвимостей по каталогу CWE наиболее распространены в проектах с открытым исходным кодом на Python?

В2. Какими категориями тестов (проверок) библиотеки Bandit найдено больше всего уязвимостей?

В3. Существует ли корреляция полученных статистических данных с данными рейтинга OWASP?

Для достижения поставленной цели исследования необходимо решить следующие задачи:

- выполнить предварительную обработку набора данных для анализа (выбор корпуса листингов);
- оценить возможности и параметры модулей проверки выбранного статического анализатора;
- разработать скрипт для автоматизации обработки результатов (построить выборку по заданным критериям);
- выполнить анализ результатов с использованием статистических методов.

Данные для анализа

Несмотря на рост использования методов машинного обучения и технологий анализа больших данных в сфере программной инженерии и информационной безопасности, поиск подходящего для анализа набора данных остается актуальной

Таблица 1

Рейтинг OWASP Top 10

№	Код категории	Категория уязвимостей
1	A01:2021	Нарушение контроля доступа (*Broken Access Control*)
2	A02:2021	Неудачное Использование Криптографии (*Cryptographic Failures*)
3	A03:2021	Внедрение кода (*Injection*)
4	A04:2021	Небезопасный дизайн (*Insecure Design*)
5	A05:2021	Неправильная конфигурация (*Security Misconfiguration*), категория A04:2017 Внедрение внешних XML сущностей — на данный момент входил в категорию A05:2021
6	A06:2021	Уязвимые и устаревшие компоненты (*Vulnerable and Outdated Components*)
7	A07:2021	Ошибки идентификации и аутентификации (*Identification and Authentication Failures*)
8	A08:2021	Нарушение целостности данных и ПО (*Software and Data Integrity Failures*)
9	A09:2021	Журнал безопасности и сбои мониторинга (*Security Logging and Monitoring Failures*)
10	A10:2021	Подделка запросов со стороны сервера (*Server-Side Request Forgery*)

Таблица 2

Общие характеристики набора данных

Характеристика	Значение
Число файлов	150 000
Размер на диске, Гбайт	1,05
Средний размер файла, Кбайт	7,39

задачей. Проведенный анализ каталогов с наборами данных [23] показал, что для языка Python для решения поставленных задач подходит датасет PY150, который содержит 150 000 файлов листингов на языке Python. Данный корпус листингов был разработан в рамках проекта SRILAB [24]. Корпус исходных кодов был сформирован путем сбора данных с GitHub-проектов на датасет Python в открытом доступе, с последующим удалением веток версий и дубликатов. В табл. 2 приведены общие характеристики корпуса листингов.

Статический анализатор Bandit

На данный момент существует широкий спектр как универсальных статических анализаторов, например, Infowatch Appercut [25], АК-ВС 2 [26], так и специализированных линтеров для языка Python, таких как Pylint [27], MyPY [28]. Хорошая документация и большой спектр модулей для тестирования обусловили выбор ранее упомянутого статического анализатора Bandit.

Утилита выполняет проверку кода в указанной директории с помощью множества отдельных тестов, которые представлены как плагины. При необходимости возможна разработка пользовательских плагинов для других проверок. В табл. 3 представлены категории тестов утилиты Bandit. Число тестов варьируется в различных категориях. Общее число детекторов — 67.

Таблица 3

Категории плагинов для поиска уязвимостей

Код теста	Описание категории
B1xx	Детекторы общих уязвимостей, в том числе детектор жестко вписанных паролей в коде
B2xx	Детектор режима отладки веб-приложения
B3xx	Детекторы вызова различных функций, которые могут привести к проблемам безопасности
B4xx	Детекторы небезопасного импорта
B5xx	Детекторы проблем, связанных с сетевыми и криптографическими протоколами
B6xx	Детекторы инъекций
B7xx	Детекторы межсайтового скриптинга

После выполнения проверки формируется отчет в выбранном формате (csv, json, html, text, xml, yaml). Результирующий отчет содержит ряд показателей для каждой найденной ошибки. В частности, это может быть: степень доверия (*Confidence* — низкая, средняя, высокая); код теста, с помощью которого обнаружена потенциальная уязвимость; степень угрозы (*Severity* — низкая, средняя, высокая); краткое текстовое описание ошибки; детальная информация о коде с ошибкой и ее расположении. На рис. 1 представлен пример фрагмента отчета в формате .json. Степень угрозы определяется непосредственно детектором библиотеки Bandit на основании данных каталога CWE.

Следует отметить, что параметр степени доверия может быть полезен при оценке количества ложно отрицательных срабатываний и ложно положительных случаев [29].

Анализ данных

После подготовки и загрузки корпуса исходных кодов был выполнен его анализ утилитой Bandit в стандартной конфигурации с последующим вы-


```
{
  "code": "10 import stat\n11 from subprocess import Popen, PIPE\n12 from riak.util import deep_merge\n",
  "col_offset": 0,
  "filename": "w:\\py\\astexample\\ex\\test_server22311.py",
  "issue_confidence": "HIGH",
  "issue_cwe": {
    "id": 78,
    "link": "https://cwe.mitre.org/data/definitions/78.html"
  },
  "issue_severity": "LOW",
  "issue_text": "Consider possible security implications associated with the subprocess module.",
  "line_number": 11,
  "line_range": [
    11
  ],
  "more_info": "https://bandit.readthedocs.io/en/1.7.4/blacklists/blacklist_imports.html#b404-import-subprocess",
  "test_id": "B404",
  "test_name": "blacklist"
}
```


Рис. 1. Пример фрагмента отчета об ошибке

водом отчета в файл json. В табл. 4—6 приведены общие метрики анализа по различным показателям. В таблицах в колонке "Значение, %" указана доля в процентах от общего числа ошибок.

Полученные характеристики показывают относительно невысокое число критически важных уязвимостей в исследуемом наборе данных.

В табл. 7 представлена детальная информация по категориям ошибок с указанием их кода по каталогу CWE. Данные получены путем анализа файла отчета. Значения сгруппированы по ключам "issue_cwe", "id" с последующим подсчетом числа ошибок. Вспомогательные вычисления проводились средствами Python.

Таблица 4

Общие данные проверки корпуса

Характеристика	Значение
Обработанных файлов	132 217
Ошибок обработки (ошибка построения AST — абстрактного синтаксического дерева)	17 783
Обработанных строк	18 364 306
Общее число потенциальных уязвимостей (предупреждений)	178 328

Таблица 5

Показатели по уровню доверия

Характеристика	Найдено потенциальных уязвимостей	Значение, %
Низкий уровень доверия	1370	0,77
Средний уровень доверия	10435	5,85
Высокий уровень доверия	166 523	93,38

Таблица 6

Показатели по уровню опасности

Характеристика	Найдено потенциальных уязвимостей	Значение, %
Низкий уровень опасности	161 396	90,51
Средний уровень опасности	14 469	8,11
Высокий уровень опасности	2463	1,38

Полученные результаты дают ответ на вопрос В1. Детальный анализ показал, что на первом месте оказались показатели уязвимостей, которые связаны с использованием оператора assert(), который, в свою очередь, определяется детектором B101 (134 699 значения). В табл. 7 в последних двух колонках также приведены показатели без учета значений детектора B101. Эти данные визуально представлены в диаграмме на рис. 2.

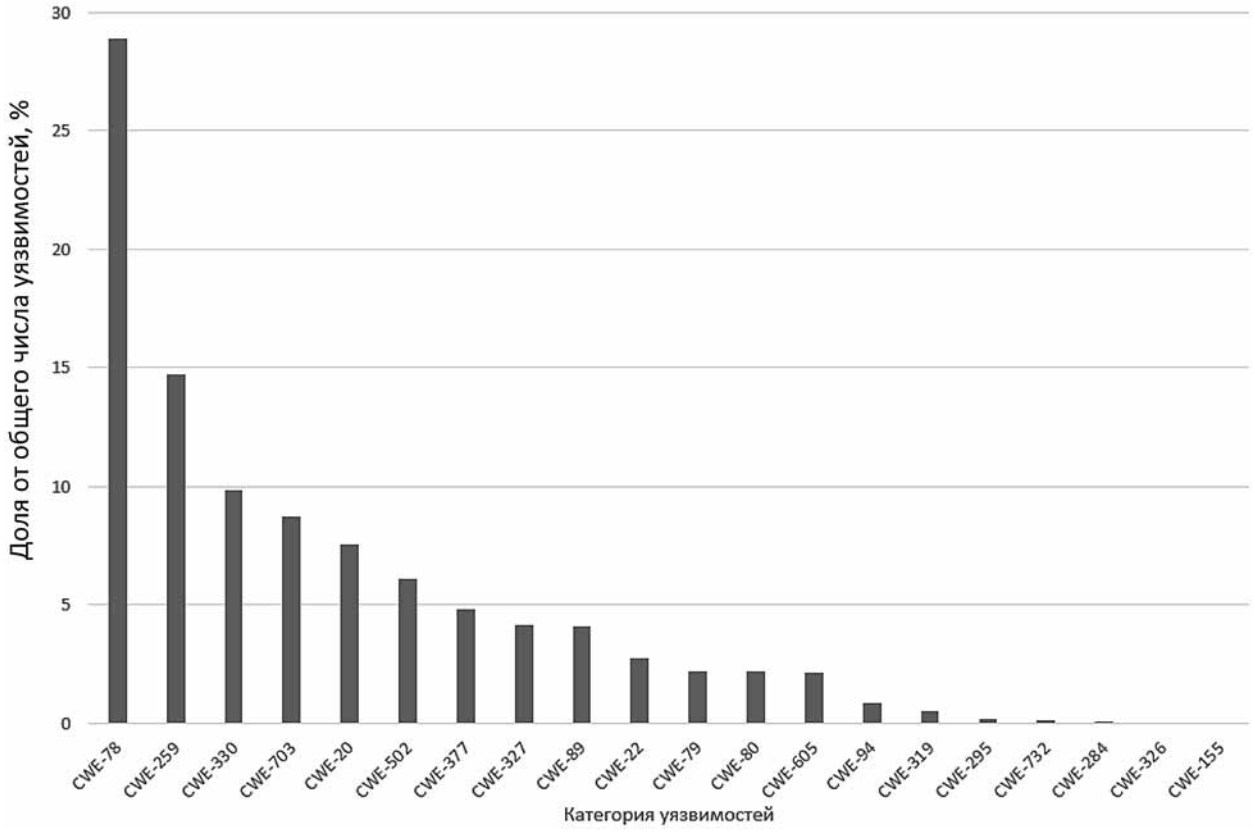


Рис. 2. Распределение частот найденных потенциальных уязвимостей без учета значений детектора B101

Таблица 7

Показатели по найденным кодам уязвимостей

Категория потенциальной уязвимости	Код	Абсолютная частота	Доля от общего числа уязвимостей, %	Абсолютная частота без B101	Доля от общего числа уязвимостей, % (без B101)
Некорректная обработка исключений	CWE-703	138 508	77,670	3809	8,73
Некорректная фильтрация специальных элементов в командах ОС (Командные инъекции)	CWE-78	12 608	7,070	12 608	28,90
Жестко закодированный пароль	CWE-259	6417	3,598	6417	14,71
Недостаточное качество случайных значений	CWE-330	4291	2,406	4291	9,84
Неправильная проверка ввода	CWE-20	3309	1,856	3309	7,58
Десериализация непроверенных данных	CWE-502	2665	1,494	2665	6,11
Небезопасный временный файл	CWE-377	2101	1,178	2101	4,82
Использование скомпрометированных или слабых криптографических алгоритмов	CWE-327	1812	1,016	1812	4,15
SQL-инъекции	CWE-89	1781	0,999	1781	4,08
Неправильное ограничение пути к каталогу с ограниченным доступом ("Обход пути")	CWE-22	1198	0,672	1198	2,75
Неправильная нейтрализация ввода во время создания веб-страницы ("Межсайтовый скриптинг")	CWE-79	953	0,534	953	2,18
Неправильная нейтрализация HTML-тегов, связанных со сценариями, на веб-странице (базовый XSS)	CWE-80	953	0,534	953	2,18
Несколько привязок к одному и тому же порту	CWE-605	936	0,525	936	2,15
Неправильный контроль над генерацией кода ("Внедрение кода")	CWE-94	385	0,216	385	0,88
Передача конфиденциальной информации открытым текстом	CWE-319	235	0,132	235	0,54
Неправильная проверка сертификата	CWE-295	78	0,044	78	0,18
Неверное назначение разрешений для критического ресурса	CWE-732	56	0,031	56	0,13
Неправильный контроль доступа	CWE-284	29	0,016	29	0,07
Недостаточная сила шифрования	CWE-326	9	0,005	9	0,02
Неправильная нейтрализация подстановочных знаков или совпадающих символов	CWE-155	4	0,002	4	0,01

Напомним, что сериализацией называется процесс преобразования информации из приложения в бинарный формат, проблемы десериализации (процесс обратный сериализации) непроверенных данных зачастую связаны с наличием встроенных методов для вывода данных на диск и с возможностью удаленного выполнения кода. На практике, проблемы могут возникать в связи с использованием уязвимостей в библиотеках Pickle и Os.system, которые могут приводить к возможности просмотра содержимого файлов или выполнению других инструкций в процессе десериализации. Детальное описание эксплуатации уязвимостей библиотеки Pickle можно найти в обзоре [30]

Таким образом, по результатам анализа лидируют потенциальные уязвимости, связанные с вопросами фильтрации данных, которые могут приводить к выполнению командных инъекций (CWE-78). В частности, они связаны с использованием модулей os и subprocess. Кроме того, широко распространены проблемные вопросы с жестко вписанными паролями в программном коде, что соответствует классу уязвимостей CWE-259, общее

число таких вопросов составило 6417. В случае если пароль жестко вписан в программный код, такой пароль будет идентичен для каждой установки, не может быть изменен администратором без ручной правки программного кода. Потенциально жестко вписанные пароли могут привести к массированным атакам на различные организации, которые используют приложение. Одним из путей решения данного проблемного вопроса является использование защищенных конфигурационных файлов, либо требование к пользователям ввода нового пароля после использования "первого логина".

Рассмотренные проблемные вопросы фильтрации данных относятся к критически значимым угрозам с высокой опасностью, показатель числа таких ошибок занимает лидирующую позицию среди показателей уязвимостей с высокой степенью угрозы (табл. 8). Поскольку в отчете об ошибках статического анализатора Bandit присутствуют фрагменты кода, которые содержат найденные строки с жестко вписанными паролями, то на основании анализа частот таких строк можно получить представление о наиболее часто встречаемых паролях в программном коде. Эти данные приведены в табл. 9.

Интересно заметить, что пароль "123456" встречается почти в три раза чаще, чем "12345", что свидетельствует о психологическом его восприятии разработчиками как более "надежного".

Группировка по числу найденных потенциальных уязвимостей по категориям проверочных тестов позволяет найти суммарное число значений в каждой из категорий и дает ответ на вопрос B2 (табл. 10).

Данные даже без учета теста B101 показывают, что наибольший процент выявленных проблем-

Таблица 8

Распределение числа уязвимостей по степеням угроз

Уязвимость	Код	Степень угрозы		
		Низкая	Средняя	Высокая
Некорректная обработка исключений	CWE-703	138 508	—	—
Некорректная фильтрация специальных элементов в командах ОС (Командные инъекции)	CWE-78	9290	2015	1303
Жестко закодированный пароль	CWE-259	6417	—	—
Недостаточное качество случайных значений	CWE-330	4291	—	—
Неправильная проверка ввода	CWE-20	1468	1724	117
Десериализация непроверенных данных	CWE-502	1337	1328	—
Небезопасный временный файл	CWE-377	—	2101	—
Использование скомпрометированных или слабых криптографических алгоритмов	CWE-327	85	1376	351
SQL-инъекции	CWE-89	—	1781	—
Неправильное ограничение пути к каталогу с ограниченным доступом ("обход пути")	CWE-22	—	1198	—
Неправильная нейтрализация ввода во время создания веб-страницы ("межсайтовый скриптинг")	CWE-79	—	953	—
Неправильная нейтрализация HTML-тегов, связанных со сценариями, на веб-странице (базовый XSS)	CWE-80	—	953	—
Несколько привязок к одному и тому же порту	CWE-605	—	936	—
Неправильный контроль над генерацией кода ("внедрение кода")	CWE-94	—	—	385
Передача конфиденциальной информации открытым текстом	CWE-319	—	52	183
Неправильная проверка сертификата	CWE-295	—	11	67
Неверное назначение разрешений для критического ресурса	CWE-732	—	32	24
Неправильный контроль доступа	CWE-284	—	—	29
Недостаточная сила шифрования	CWE-326	—	9	—
Неправильная нейтрализация подстановочных знаков или совпадающих символов	CWE-155	—	—	4

Таблица 9

Наиболее встречаемые пароли

№	Пароль	Абсолютная частота	Доля от общего числа паролей, %	№	Пароль	Абсолютная частота	Доля от общего числа паролей, %
1	Пароль отсутствует (пустой)	612	9,54	11	normaluser	64	1,00
2	password	525	8,18	12	123	59	0,92
3	secret	228	3,55	13	POST	55	0,86
4	pass	132	2,06	14	bar	52	0,81
5	test	130	2,03	15	foo	49	0,76
6	testpass	119	1,85	16	token	42	0,65
7	admin	108	1,68	17	user	40	0,62
8	123456	99	1,54	18	dummy	39	0,61
9	foobar	73	1,14	19	moo	39	0,61
10	cat	65	1,01	20	12345	31	0,48

ных вопросов связан с категорией тестов B1xx. Это обусловлено большим числом жестко вписанных паролей в коде. На втором месте находятся вопросы, связанные с использованием небезопасных функций. На третьем месте группа детекторов возможных инъекций в программном коде.

Заметим, что согласно рейтингу OWASP Top 10, проблемные вопросы использования инъекций также находятся на третьей позиции (A03:2021). В то же время вопросы использования устаревших или ненадежных компонентов (A06:2021) находятся на шестой строчке по рейтингу OWASP. Эта категория может быть связана с блоком импортируемых библиотек, за который отвечают детекторы группы B4xx, которые, исходя из данных табл. 10, находятся на четвертой позиции.

Сравнивая лидирующие позиции рейтингов A01:2021 и B1xx, можно отметить, что связь также присутствует, поскольку вписанные пароли в коде могут привести к нарушениям контроля доступа. Полученные числовые результаты свидетельствуют о наличии тесных связей между рассмотренными рейтингами и дают ответ на вопрос В3, поставленный в начале работы.

Среди российских ресурсов, которые могут быть использованы в качестве источника данных уязвимостей для анализа безопасности, можно отметить банк данных угроз безопасности информации [31], однако он также опирается на базу данных каталога CWE. Так, по результатам, представленным на данном ресурсе (рис. 3), наибольшее распространение получила уязвимость CWE-119, связанная с ошибками переполнения буфера, которая преимущественно связана с языками C и C++. Описание уязвимостей, связанных с ошибками переполнения буфера для языка Python, можно найти в отчете [32].

Распределение на рис. 3 носит более общий характер. Тем не менее, можно заметить, что также широко распространены уязвимости, обусловленные инъекциями и некорректной обработкой входных данных.

Таблица 10

Показатели по категориям тестов

Категория	Всего значений	Доля от общего числа по категории, %	Всего без значений детектора B101	Доля от общего числа по категории, %
B1xx	148 557	83,31	13 858	31,76
B2xx	143	0,08	143	0,33
B3xx	11 683	6,55	11 683	26,78
B4xx	5927	3,32	5927	13,59
B5xx	751	0,42	751	1,72
B6xx	10 072	5,65	10 072	23,09
B7xx	1195	0,67	1195	2,74

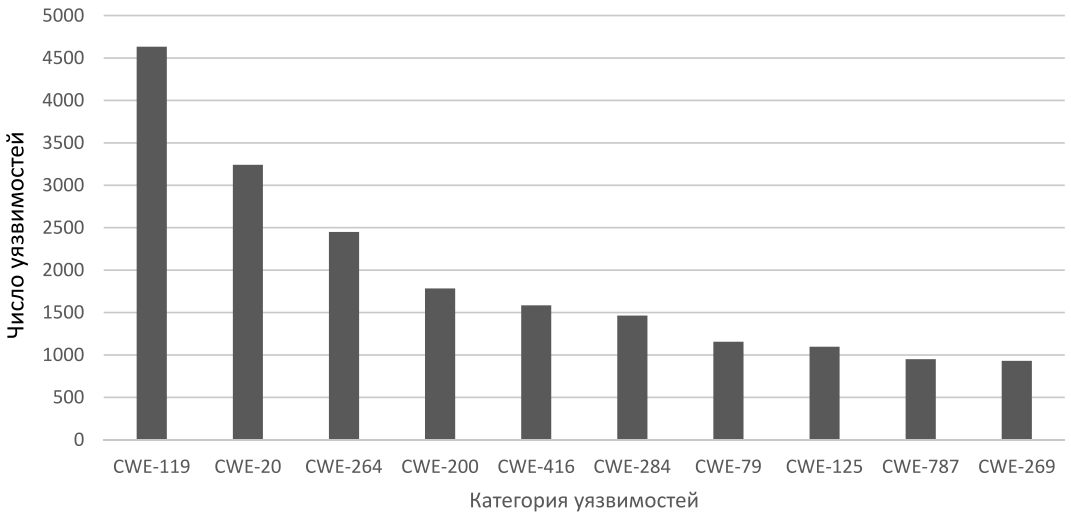


Рис. 3. Распределение по типам ошибок

Заключение

Представленные в статье результаты анализа свидетельствуют о том, что проблемные вопросы, связанные с написанием безопасного кода для приложений на Python, остаются открытыми. Многие библиотеки с открытым исходным кодом содержат достаточно важные уязвимости, множественные ошибки фильтрации данных, используют жестко вписанные пароли либо устаревшие версии криптографических библиотек.

Одним из путей решения задачи повышения качества программного кода и его безопасности может быть использование методов статического анализа и библиотек для поиска ошибок еще на этапе разработки. Эффективность таких анализаторов достаточно высока даже для больших проектов. В то же время, остается актуальным вопрос с ложно положительными срабатываниями модулей тестирования. Кроме того, необходимо дальнейшее развитие российских средств линтеринга и статического анализа, которые будут опираться на национальный каталог уязвимостей.

Отметим, что перспективным направлением исследования в области информационной безопасности является использование комбинированных методов. В частности, это относится к методам машинного обучения, статическим, латентно-семантическим методом и методам динамического анализа для проверки программного кода. Разработки в данных направлениях сопряжены с рядом трудностей, которые обусловлены отсутствием в открытом доступе качественных больших наборов данных для обучения.

Список литературы

1. **О техническом** регулировании. Федеральный закон Российской Федерации от 27 дек. 2002 г. № 184-ФЗ, ред. от 28.11.2018. URL: http://www.consultant.ru/document/ cons_doc_LAW_40241/

2. **О сертификации** средств защиты информации. Постановление Правительства Российской Федерации от 26 июня 1995 г. № 608, ред. от 21.04.2010. URL: http://www.consultant.ru/document/cons_doc_LAW_7054/

3. **Cousot P.** Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints // Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages. 1977. P. 238—252.

4. **Allen F. E.** Control flow analysis // ACM SIGPLAN Notices. 1970. Vol. 5, Is. 7. P. 1—19.

5. **Johnson S. C.** Lint, C Program Checker // COMP. SCI. TECH. REP. 1978. P. 78—90.

6. **Beller M., Bholanath R., McIntosh S., Zaidman A.** Analyzing the State of Static Analysis: A Large-Scale Evaluation in Open Source Software // In Proceedings of the 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER 2016), Suita, IEEE, 2016. P. 470—481. DOI: 10.1109/SANER.2016.105.

7. **Chess B., McGraw G.** Static Analysis for Security // IEEE Security & Privacy, 2004. Vol. 2, Is. 6. P. 76—79. DOI: 10.1109/MSP.2004.111.

8. **Fromherz A.**, **Ouadjaout A., Mine A.** Static Value Analysis of Python Programs by Abstract Interpretation // In Proceedings of the 10th NASA Formal Methods International Symposium (NFM 2018), Springer, Newport News, 2018. P. 185—202. DOI: 10.1007/978-3-319-77935-5_14.

9. **Oyetoyan T. D., Milosheska B., Grini M., Cruzes D. S.** Myths and Facts about Static Application Security Testing Tools: An Action Research at Telenor Digital // In Proceedings of the 19th Conference on Agile Processes in Software Engineering and Extreme Programming (XP 2018), Porto, Springer, 2018. P. 86—103. DOI:10.1007/978-3-319-91602-6_6.

10. **Vassallo C.**, **Panichella S., Palomba F.** et al. How Developers Engage with Static Analysis Tools in Different Contexts // Empirical Software Engineering. 2020. Vol. 25, Is. 2. P. 1419—1457. DOI: 10.1007/s10664-019-09750-5.

11. **Smith J., Johnson B., Murphy-Hill E.** et al. How Developers Diagnose Potential Security Vulnerabilities with a Static Analysis

Tool // IEEE Transactions on Software Engineering, 2019. Vol. 45. Is. 9. P. 877— 897. DOI: 10.1109/TSE.2018.2810116.
12. **OWASP** Web Security Testing Guide. URL: https://github.com/OWASP/wstg
13. **Common** Weakness Enumeration. URL: https://cwe.mitre.org/about/index.html
14. **CVE**. URL: https://cve.mitre.org/
15. **OWASP** Top 10 — 2021. URL: https://owasp.org/Top10/
16. **TIOBE** Index for March 2022. URL: https://www.tiobe.com/tiobe-index/
17. **Django** Software Foundation. Security in Django. URL: https://docs.djangoproject.com/en/3.0/topics/security/
18. **Xia X., He X., Yan Y.** et al. An Empirical Study of Dynamic Types for Python Projects // In Proceedings of the 8th International Conference on Software Analysis, Testing, and Evolution (SATE 2018), Springer, 2018. P. 85—100.
19. **Ruohonen J.** An Empirical Analysis of Vulnerabilities in Python Packages for Web Applications // In Proceedings of the 9th International Workshop on Empirical Software Engineering in Practice (IWESEP 2018). Nara, IEEE, 2018. P. 25—30.
20. **The** Python Package Index (PyPI) is a repository of software for the Python programming language. URL: https://pypi.org/
21. **Ruohonen J., Hjerpee K., Rindell K.** A Large-Scale Security-Oriented Static Analysis of Python Packages in PyPI // Proceedings of the 18th Annual International Conference on Privacy, Security and Trust (PST 2021), Auckland (online), IEEE, 2021. P. 1—10.
22. **Welcome** to the Bandit documentation! — Bandit documentation. URL: https://bandit.readthedocs.io/en/latest/
23. **A Collection** of Datasets for Big Code AnalysIs. URL: https://github.com/CUHK-ARISE/ml4code-dataset
24. **Secure,** Reliable, and Intelligent Systems Lab | SRI Group Website. URL: https://www.sri.inf.ethz.ch/
25. **Infowatch** Appercut**.** URL: https://www.infowatch.ru/products/appercut
26. **АК-ВС** 2. URL: https://npo-echelon.ru/production/65/4243
27. **pylint** 2.14.5 URL: https://pypi.org/project/pylint/
28. **Welcome** to mypy documentation! — Mypy 0.942 documentation. URL: https://mypy.readthedocs.io/en/stable/#
29. **Edmundson A., Holtkamp B., Rivera E.** et al. An Empirical Study on the Effectiveness of Security Code Review //In Proceedings of the 5th International Symposium on Engineering Secure Software and Systems (ESSoS 2013), Paris, Springer, 2013. P. 197— 212. DOI: 10.1007/978-3-642-36563-8_14.
30. **Hamann D.** Exploiting Python pickles. URL: https://davidhamann.de/2020/04/05/exploiting-python-pickle/
31. **Банк** данных угроз безопасности информации. URL: https://bdu.fstec.ru/vul
32. **Python**: List of security vulnerabilities. URL: https://www.cvedetails.com/vulnerability-list/vendor_id-10210/product_id-18230/opov-1/Python-Python.html

Static Analysis of the Source Code of Python Applications

D. A. Kapustin, kap-kapchik@mail.ru, **V. V. Shvyrov**, slsh@i.ua, **T. I. Shulika** shulika-tatyana@mail.ru, Lugansk State Pedagogical University, Lugansk, 91011, Lugansk People's Republic

Corresponding author:

Denis A. Kapustin, Associate Professor,
Lugansk State Pedagogical University, Lugansk, 91011, Lugansk People's Republic
E-mail: kap-kapchik@mail.ru

Received on July 08, 2022
Accepted on July 21, 2022

One of the popular methods of software code analysis is the static analysis method. This method allows not only to check the code for compliance with the language specification, but also to find potential vulnerabilities. The work performs a static analysis of a corpus of open source Python application. Using the Bandit library, statistical indicators of various categories of potential vulnerabilities are found, a rating table of vulnerabilities found in the studied data set is built. A qualitative analysis of threats is carried out according to their danger based on the CWE catalog data.

The purpose of this work is to analyze a corpus of open source Python listings for potential vulnerabilities using a static analyzer and rank threats based on the potential vulnerabilities found.

Thus, the study should answer the following questions:

Q1. What categories of potential vulnerabilities in the CWE catalog are most common in Python open source projects?

Q2. What categories of tests (checks) of the Bandit library found the most vulnerabilities?

Q3. Is there a correlation between the obtained statistical data and the OWASP rating data?

Keywords: *bandit, big data, CWE, linters, OWASP, python dataset, security analysis, static analysis, source code analysis, threat, vulnerability*

For citation:

Kapustin D. A., Shvyrov V. V., Shulika T. I. Static Analysis of the Source Code of Python Applications, *Programmnaya Ingeneria,* 2022, vol. 13, no. 8, pp. 394—403.

DOI: 10.17587/prin.13.394-403

References

1. **O tekhnicheskom** regulirovanii. Federal'nyj zakon Rossijskoj Federacii ot 27 dek. 2002 g. № 184-FZ, red. ot 28.11.2018, available at: http://www.consultant.ru/document/ cons_doc_LAW_40241/ (in Russian).
2. **O sertifikacii** sredstv zashchity informacii. Postanovlenie Pravitel'stva Rossijskoj Federacii ot 26 iyunya 1995 g. № 608, red. ot 21.04.2010, available at: http://www.consultant.ru/document/cons_doc_LAW_7054/ (in Russian).
3. **Cousot P.** Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints, *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages,* 1977, pp. 238—252.
4. **Allen F. E.** Control flow analysis, *ACM SIGPLAN Notices*, 1970, Vol. 5, Is. 7, July, pp. 1—19.
5. **Johnson S. C.** Lint, C Program Checker, *COMP. SCI. TECH. REP,* 1978, pp. 78—90.
6. **Beller M., Bholanath R., McIntosh S., Zaidman A.** Analyzing the State of Static Analysis: A Large-Scale Evaluation in Open Source Software, *In Proceedings of the 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER 2016)*, Suita, IEEE, 2016, pp. 470—481. DOI: 10.1109/SANER.2016/105.
7. **Chess B., McGraw G.** Static Analysis for Security, *IEEE Security & Privacy*, 2004, vol. 2, is. 6, pp. 76—79. DOI: 10.1109/MSP.2004.111.
8. **Fromherz A.**, **Ouadjaout A., Mine A.** Static Value Analysis of Python Programs by Abstract Interpretation, *In Proceedings of the 10th NASA Formal Methods International Symposium (NFM 2018),* Springer, Newport News, 2018, pp. 185—202. DOI: 10.1007/978-3-319-77935-5_14.
9. **Oyetoyan T. D., Milosheska B., Grini M., Cruzes D. S.** Myths and Facts About Static Application Security Testing Tools: An Action Research at Telenor Digital, *In Proceedings of the 19th Conference on Agile Processes in Software Engineering and Extreme Programming (XP 2018)*, Porto, Springer, 2018, pp. 86—103, DOI: 10.1007/978-3-319-91602-6_6.
10. **Vassallo C., Panichella S., Palomba F.** et al. How Developers Engage with Static Analysis Tools in Different Contexts, *Empirical Software Engineering*, 2020. Vol. 25, is. 2, P. 1419—1457. DOI: 10.1007/s10664-019-09750-5.
11. **Smith J., Johnson B., Murphy-Hill E.** et al. How Developers Diagnose Potential Security Vulnerabilities with a Static Analysis Tool, *IEEE Transactions on Software Engineering*, 2019, vol. 45, is. 9, pp. 877— 897. DOI: 10.1109/TSE.2018.2810116.
12. **OWASP** Web Security Testing Guide, available at: https://github.com/OWASP/wstg
13. **Common** Weakness Enumeration, available at: https://cwe.mitre.org/about/index.html
14. **CVE,** available at: https://cve.mitre.org/
15. **OWASP** Top 10 — 2021, available at: https://owasp.org/Top10/
16. **TIOBE** Index for March 2022, available at: https://www.tiobe.com/tiobe-index/
17. **Django** Software Foundation. Security in Django. available at: https://docs.djangoproject.com/en/3.0/topics/security/
18. **Xia X., He X., Yan Y.** et al. An Empirical Study of Dynamic Types for Python Projects, *In Proceedings of the 8th International Conference on Software Analysis, Testing, and Evolution (SATE 2018)*, Springer, 2018, pp. 85—100.
19. **Ruohonen J.** An Empirical Analysis of Vulnerabilities in Python Packages for Web Applications, *In Proceedings of the 9th International Workshop on Empirical Software Engineering in Practice (IWESEP 2018)*, Nara, IEEE, 2018, pp. 25—30.
20. **The Python** Package Index (PyPI) is a repository of software for the Python programming language, available at: https://pypi.org/
21. **Ruohonen J., Hjerpee K., Rindell K.** A Large-Scale Security-Oriented Static Analysis of Python Packages in PyPI, *Proceedings of the 18th Annual International Conference on Privacy, Security and Trust (PST 2021), Auckland (online)*, IEEE, 2021, pp. 1—10.
22. **Welcome** to the Bandit documentation! — Bandit documentation, available at: https://bandit.readthedocs.io/en/latest/
23. **A Collection** of Datasets for Big Code Analysis, available at: https://github.com/CUHK-ARISE/ml4code-dataset
24. **Secure,** Reliable, and Intelligent Systems Lab | SRI Group Website, available at: https://www.sri.inf.ethz.ch/
25. **Infowatch** Appercut, available at: https://www.infowatch.ru/products/appercut (in Russian).
26. **AK-VS 2**, available at: https://npo-echelon.ru/production/65/4243 (in Russian).
27. **pylint** 2.14.5, available at: https://pypi.org/project/pylint/
28. **Welcome** to mypy documentation! — Mypy 0.942 documentation, available at: https://mypy.readthedocs.io/en/stable/#
29. **Edmundson A.**, **Holtkamp B., Rivera E.** et al. An Empirical Study on the Effectiveness of Security Code Review, *In Proceedings of the 5th International Symposium on Engineering Secure Software and Systems (ESSoS 2013)*, Paris, Springer, 2013, pp. 197— 212. DOI: 10.1007/978-3-642-36563-8_14.
30. **Hamann D.** Exploiting Python pickles, available at: https://davidhamann.de/2020/04/05/exploiting-python-pickle/
31. **Bank** dannyh ugroz bezopasnosti informacii, available at: https://bdu.fstec.ru/vul (in Russian).
32. **Python:** List of security vulnerabilities, available at: https://www.cvedetails.com/vulnerability-list/vendor_id-10210/product_id-18230/opov-1/Python-Python.html

ИНФОРМАЦИЯ

УДК 004.82
DOI: 10.17587/prin.13.404-412

А. В. Лукоянычев, аспирант, dizzystyle@yandex.com,
Новосибирский государственный технический университет

Интегрированная система обучения жестовому языку

Рассмотрена разработанная автором интегрированная система обучения жестовому языку, ориентированная на широкий круг пользователей. В системе используются нотационная запись Димскис для реализации отображения жестового языка и межплатформенная среда разработки Unity 3D для управления анимированным персонажем. Предложен подход для описания движений 3D-персонажа на основе метаязыка. Рассмотрена методика создания справочника жестового языка, приведены преимущества такого подхода. Интегрированная система, кроме полноценного инструментария для создания справочника, представляет возможность проверки знаний пользователя для мобильных и стационарных устройств. Рассматриваются этапы реализация данного режима. Приведены результаты экспериментального исследования.

Ключевые слова: *жестовый язык, нотация Димскис, Unity 3D, метаязык управления аватаром, локализация жеста, идентификация жеста*

Введение

Обучение жестовому языку необходимо людям, контактирующим со слабослышащими. Создание обучающих жестовому языку систем актуально и для решения задачи повышения уровня автоматизации и роботизации всех сфер деятельности человека. В настоящее время уделяется значительное внимание технологиям организации эффективного и более естественного человеко-машинного взаимодействия. С помощью жестов рук можно передавать управляющие команды интеллектуальной информационной системе на расстоянии, в зашумленной обстановке, в экстремальных условиях, когда речевые команды затруднительны.

Классические пользовательские интерфейсы уступают место естественным для человека интерфейсам. Интеллектуальные информационные системы начинают применять в медицине, образовании, социальной сфере, военном деле, а также при управлении робототехническими комплексами [1—4].

В документе "Стратегия развития отрасли информационных технологий в Российской Федерации на 2014—2020 годы и на перспективу до 2025 года" выделены приоритетные направления государственной политики по исследованиям и разработкам в области информационных технологий. Один из пунктов этого документа формулирует следующее направление: "Новые человеко-машинные интерфейсы, включая новые методы использования жестов, ... а также новые программные средства и устройства, повышающие социальную адаптацию людей с ограниченными возможностями". В соответствии с этим положением необходима разработка технологий и средств автоматизированного обучения языку жестов глухих людей, а также людей, использующих жесты для эффективного взаимодействия со слабослышащими, для организации межчеловеческого и человеко-машинного взаимодействия.

Несмотря на большое число работ в данном направлении, вопросам демонстрации и обучения жестовому языку (ЖЯ) в настоящее время уделяется недостаточное внимание. На настоящее время не существует действующей, полностью автоматизированной универсальной расширяемой системы обучения и проверки знаний ЖЯ.

Создание таких систем связано с рядом проблемных вопросов: сложность использования нотационных записей для представления ЖЯ; ограниченный круг разработчиков систем, что не позволяет сторонним заинтересованным лицам пополнять справочник ЖЯ; ориентация систем на специальное оборудование для описания и распознавания жестов; стремление решать общие проблемы, связанные с сурдопереводом, даже в рамках обучающей системы. В результате возможности таких систем обычно ограничиваются небольшим набором используемых жестов или имеются существенные ограничения на минимальную сложность оборудования, что сужает круг пользователей этих систем [5, 6].

Нотации жестового языка

Анализ возможности применения компьютеров для отображения ЖЯ показывает, что наиболее перспективным является использование 3D-анимации для демонстрации жестов.

Основой любого ЖЯ является жестовая транскрипция (нотационная запись), на основе которой составляется справочник ЖЯ. Существует несколько систем нотаций, различных по возможностям и принципам представления жеста в записи.

В настоящее время международными исследователями ЖЯ используются следующие системы: нотация Стокоу; система транскрипции Berkeley; система транскрипции ЖЯ HamNoSys; система нотации SignWriting. Кроме перечисленных систем существуют и другие менее известные системы записи ASL, к числу которых относятся Sign Script, Si5s, ASL-phabet. У каждой из них есть свои преимущества и недостатки [6—8]. Для учета особенностей русского жестового языка (РЖЯ) разработана система нотации Л. Димскис [9].

Нотации Димскис содержат около 150 нотационных записей. Конфигурация пальцев (форма руки) представляется в виде иконографического изображения и содержит менее 40 вариантов. Место исполнения жеста описывается 50 нотационными записями, а характеристик локализации жеста — менее 80.

Иконографическое представление конфигурации пальцев на современном уровне развития 3D-графических редакторов позволяет достаточно легко создавать такие изображения на компьютере. Такое изображение формы руки уменьшает число символов и повышает наглядность при описании жеста. Ограниченный интуитивно понятный базовый набор символов в нотации облегчает создание описания транскрипционных записей даже сложных движений пальцев и рук анимированного персонажа. В нотации Димскис отсутствуют взаимодействия с окружающей обстановкой и описания не мануальных действий (глаза, рот). Это обстоятельство свидетельствует о том, что система нотации Димскис, как и любая другая нотационная компьютерная система, должна совершенствоваться и развиваться. Так, в системе SignWriting на настоящее время насчитывается более 38 тыс. знаков [5]. Массовое использование нотации Димскис в РФ потенциально расширяет круг пользователей и составителей справочника и уроков [10]. Применение нотации Димскис для создания открытой интегрированной системы обучения ЖЯ на текущий момент является приемлемым обоснованным решением.

Для управления анимированным 3D-персонажем (аватаром) существует несколько графических пакетов (игровых движков). Наиболее универсальными и развитыми являются Unity 3D, Cry ENGINE и UDK [11]. Сравнение их характеристик показало, что наиболее подходящим для решения поставленных задач является пакет Unity 3D. Он позволяет создавать приложения под несколько операционных систем для персональных компьютеров и мобильных устройств. Возможно использование этого приложения в виде интернет-сервиса.

Методика создания мультимедийного справочника жестового языка

В нотации Димскис все слова и дактилемы начинаются с пятисимвольного описания положения руки и пальцев. Эти символы описывают: характер исполнения жеста (одноручный, двуручный); положение пальцев; направление ладони; направление пальцев; место расположения жеста. Затем следуют дополнительные символы, с помощью которых отображается динамика выполнения жеста.

Анализ анатомического строения руки человека показывает, что контроль состояния движения руки можно осуществлять посредством управления 19 подвижными элементами для одной руки.

Для удобства работы с Unity 3D разработаны метаязык (скриптовый язык) и программное обеспечение, которое позволяет переводить скриптовый текстовый формат в анимационное движение в Unity 3D. Создание метаязыка позволило упростить управление процессом отображения движений аватара. Это обстоятельство позволяет наглядно и оперативно корректировать файл. При этом сокращается объем файлов клипов, что особенно актуально для мобильной реализации.

Нотация Димскис не является полностью компьютерно-ориентированной, поэтому в рамках метаязыка расширена нотационная запись Димскис. Введены дополнительные и служебные символы, управляющие слова, специальные координаты [12]. Разработаны специальные управляющие символы и правила для синхронизации процессов, позволяющие реализовать движение нескольких элементов за различное время (с разной скоростью), создавать сложные движения, осуществлять возврат в начальное состояние, отменять перемещения отдельных подвижных элементов за счет использования "фиктивного" движения, изменять только отдельно взятые координаты руки аватара. Данная модификация нотации Димскис позволяет формировать набор элементарных анимаций

(клипов), которые составляют основу визуализированных нотаций, и создавать эффект непрерывности демонстрации жестов, реализовывать любое желаемое движение. Правила, разработанные для синхронизации процессов, отличаются наглядностью представления, уменьшением сложности составления описания и производительностью по сравнению со стандартными методами анимации. Структура разработанной интегрированной системы обучения РЖЯ представлена на рис. 1.

Инструментарий интегрированной системы ориентирован на разработчиков и на пользователей и, соответственно, имеет два режима: "Разработчик" и "Пользователь".

Технологический инструментарий для разработчика включает все необходимое программное обеспечение для создания анимационного файла и демонстрации жеста, для соединения созданного клипа с нотацией Димскис. На основе созданных нотаций разрабатываются мультимедийный справочник по РЖЯ и уроки по обучению РЖЯ. Редакторы справочника и уроков в режиме "Разработчик" доступны в сети Интернет и могут использоваться сторонними специалистами в области ЖЯ. Разработанная методика создания справочника и уроков РЖЯ позволяет оперативно проверять и настраивать сценарии демонстрации слов, высказываний и уроков на их основе.

Просмотр библиотеки дактилем и слов справочника и разработанных уроков возможен сторонними пользователями как через сеть Интернет, так и на персональных и мобильных устройствах при загрузке модулей режима "Пользователь".

Основным модулем для разработки анимационных файлов (клипов) является "Редактор клипов". В редакторе доступны для отображения все созданные анимационные файлы. Разработчик имеет возможность визуально создавать клипы и сценарии любой сложности за счет полноценного инструментария, используя всю иерархию библиотеки от простейших анимаций до составных (рис. 2, *а*). В результате создается файл клипа на метаязыке, который можно редактировать либо во встроенном, либо в любом текстовом редакторе (см. рис. 2, *б*, *в*).

Следующим шагом является связывание созданных клипов с нотациями Димскис. "Редактор знаков нотации" позволяет создать словесное описание выбранного анимированного символа и указать его расположение в библиотеке знаков нотации. На рис. 3 представлен пример связки анимационного клипа с элементом нотации Димскис.

На основе полученных анимированных знаков в "Редакторе справочника" в соответствии с нотацией Димскис составляются описания дактилем и слов, сценарии реализации которых вносятся в справочник (рис. 4). Сформированные записи в справочнике РЖЯ в дальнейшем могут использоваться для составления новых слов и уроков.

Справочник РЖЯ включает в себя свыше 3000 слов, поэтому редакторы справочника и уроков в режиме "Разработчик" могут использоваться сторонними специалистами в области ЖЯ, а интегрированная система доступна через сеть Интернет. Такой подход позволяет привлечь заинтересованных лиц к наполнению справочника и уроков. Для изучения РЖЯ в режиме "Пользователь" возможна загрузка на персональные компьютеры и мобильные устройства модулей просмотра библиотеки дактилем и слов, а также созданных уроков.

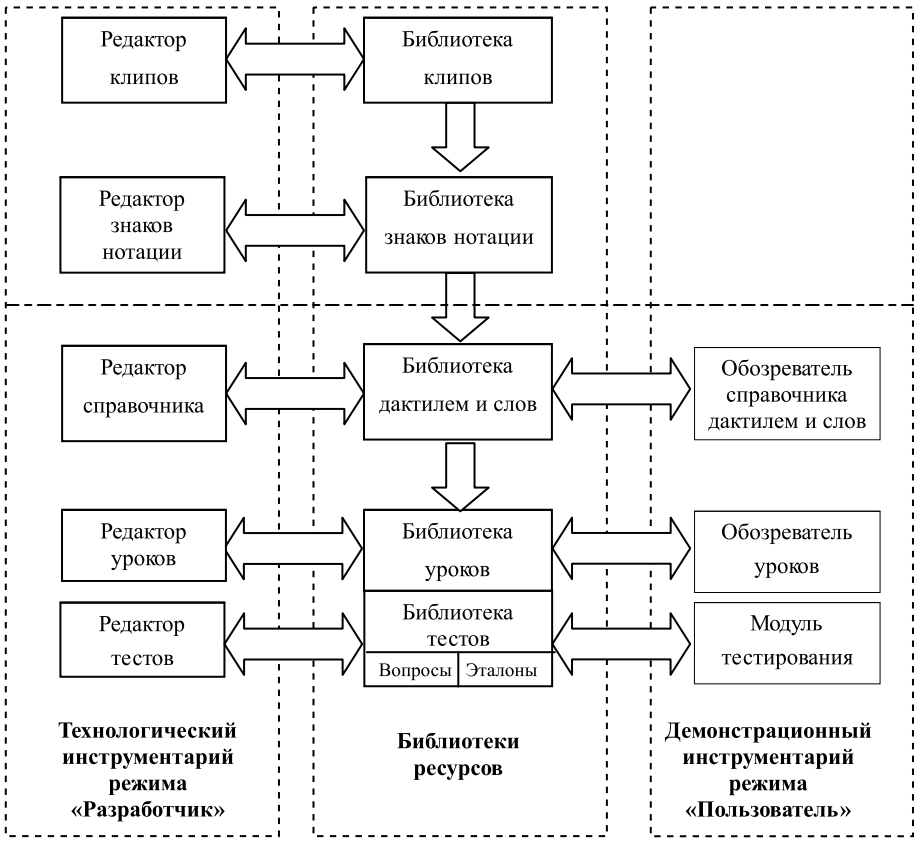


Рис. 1. Структура интегрированной системы обучения РЖЯ

```
{
  "Description":   "Клип 18 ",                заголовок (название)
  "PlayMode": 0,                               (режим проигрывания)
  "curves": [                                  (кривые анимаций)
   {
  "TargetName" :"RightHandMiddle2",            (название сустава)
     "modelRotationNodes": [                   (описание поворота)
       {
         "angle": {
           "x": 10.832117080688477,            (углы поворота по x)
           "y": 0.2659042179584503,            (углы поворота по y)
           "z": 1.4758174419403077             (углы поворота по z)
                 },
         "time": 0.3                           (время выполнения)
       }
                                  ]
   },
   {                                           (описание
     …….                                       следующего сустава )
```

а) *б*) *в*)

Рис. 2. Редактор клипов (*а*), анимационный файл на метаязыке (*б*) и комментарий (*в*)

Режим тестирования

Разработанная автором интегрированная система обучения ЖЯ имеет два режима: демонстрация дактилем, слов и высказываний ЖЯ (справочник и уроки); проверка знаний (тестовый режим). Для реализации тестового режима в системе используются следующие модули: редактор тестов; библиотека тестов и эталонов; модуль тестирования. Тестовый режим имеет два варианта проверки знаний. Первый вариант: аватар демонстрирует жест (слово, дактилему, высказывание), а пользователь должен выбрать из списка правильный ответ (рис. 5). Второй вариант: система просит показать жест, соответствующий дактилеме или слову, и пользователь должен показать его самостоятельно. Второй вариант тестирования опирается на распознавание жеста пользователя.

В редакторе тестов для первого варианта проверки знаний создается библиотека вопросов, которая может содержать дактилемы, слова и высказывания. Редактор доступен через сеть Интернет и сторонним разработчикам.

Для второго из перечисленных выше вариантов тестирования в библиотеку вопросов загружаются эталоны с соответствующими атрибутами, которые получены при демонстрации жестов РЖЯ аватаром.

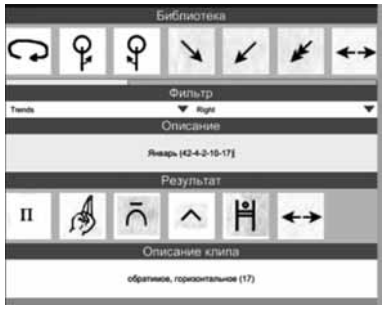
К разрабатываемой системе специальных требований не предъявляется. Для распознавания жеста используется стандартная веб-камера, поэтому второй вариант тестирования реализуется только на стационарных устройствах с неподвижной веб-камерой.

Распознавание жеста складывается из двух этапов: обнаружение руки на изображении и ее локализация (поиск области интересов); непосредственно распознавание и классификация жеста (сравнение с эталонными образцами по выделенным характерным признакам жеста).

```
{
  "Description": "на уровне груди с
касанием справа (15)",
  "IconForClip": "Where\\15.png",
  "ModelAnimaitionTextFile":
"Where\\15WhereRightHand.animation
",
  "ClipType": 4,
   "HandType": 1
}
```

а) *б*)

Рис. 3. Редактор знаков нотации (*а*) и описание символа Димскис на метаязыке (*б*)




```
{
   "Name": "Январь",
   "Description": "Январь (42-4-2-10-17)",
   "modelClips": [
     "Service\\RightHand.clip",
     "Fingers\\42fingersRightHand.clip",
     "Palm\\4PalmRightHand.clip",

"FingersMerge\\2FingersMergeRightHand.clip",
     "Where\\10WhereRightHand.clip",
     "Trend\\17fingersRightHand.clip"
   ]
}
```

Рис. 4. Пример нотационной записи слова "Январь" и анимационного сценария

Рис. 5. Режим тестирования с выбором ответа

Требований к окружающей обстановке и специальных требований к освещенности не предусматривается. По этой причине сложно использовать метод выделения жеста по цвету кожи, не усложняя систему. Выполнение тестового вопроса занимает несколько секунд. В течение этого промежутка времени освещенность и окружающую обстановку можно считать неизменными. По этой причине наиболее эффективным методом предварительной обработки изображения является удаление фона. Также в течение этого времени подвижных объектов в кадре, кроме руки, не будет. Таким образом, для обнаружения жеста используется метод детектирования движения.

При переходе в режим тестирования включается веб-камера. В режиме тестирования обучающегося система просит показать определенный жест. С этого момента (или по нажатию клавиши) начинается видеозахват изображения. Система фиксирует первый кадр, который используется для регистрации фона. После этого включается детектор движения. Если соседние кадры отличаются больше, чем на определенное значение T, то фиксируется начало движения. Детектор переключается в режим фиксации окончания движения. Если соседние кадры не отличаются больше, чем на T, то обучающийся показывает ключевой жест. Зафиксированный ключевой кадр записывается. Детектор переходит в начальное состояние для определения нового движения, т. е. переключается на фиксацию следующего ключевого кадра. Для статических жестов фиксируется один ключевой кадр, а для динамических — несколько. На демонстрацию жеста отводится несколько секунд или окончание жеста фиксируется нажатием кнопки. По завершению демонстрации жеста записывается финальный кадр. Он используется для устранения остатков фона. Это связано с тем, что пользователь во время демонстрации может двигаться, и, следовательно, первый и последний кадры будут несколько различаться.

По окончании записи начинается этап локализации жеста. Из всех зафиксированных кадров вычитается первый (фон). Из ключевых кадров вычитается последний кадр, над которым предварительно проведена операция "размытия" в целях надежного устранения остатков фона. В результате остаются ладонь, рука и остатки фона. Наибольшую площадь с наибольшей яркостью занимает кисть и, возможно, часть руки, что зависит от одежды пользователя.

Для удаления контрастных областей строится гистограмма в градациях серого цвета и используется модифицированный метод Оцу [13]. Для удаления мелких остатков шума применяется метод фильтрации по яркости и связности. Изображение разбивается на N частей по осям X и Y, в каждой части вычисляется число ярких точек. Проверяется связность соседних частей. Такой анализ позволяет исключить разрыв контура изображения руки на этапе локализации. Удаляются области, в которых таких точек меньше Z (Z — пороговое значение) и которые являются не связанными или плохо связанными (пороговое значение — S). Локализованное изображение области интересов заключается в прямоугольник, и кадр обрезается до его размеров. При этом для динамических жестов фиксируются координаты прямоугольника на исходном кадре. Алгоритм локализации жеста устойчиво работает при слабом освещении как окружающей обстановки (фона), так и рабочего места пользователя. На этом этап локализации жеста заканчивается, дальнейшие действия связаны со следующим этапом — с идентификацией жеста, сравнением локализованного изображения руки обучающегося с подобным изображением аватара.

В ЖЯ основную информацию о жесте несет конфигурация пальцев, поэтому идентификация жеста проводится в два этапа. На первом этапе анализируется кисть целиком. Если на первом этапе принять решение затруднительно, то выполняется второй этап, на котором анализируются только пальцы.

Из локализованного изображения руки выделяется кисть. При разрешении фиксируемого изображения 640×480 пикселей кисть занимает не

более 200 пикселей по вертикали или горизонтали в зависимости от демонстрируемого жеста. Это позволяет еще уменьшить размер исследуемой области и при этом удалить малоинформативную часть руки. Изображение выделенной кисти пользователя переводится в градации серого, и строятся две проекции по яркости на осях X и Y. При этом проводится дополнительная фильтрация за счет отсечения не связанных или слабо связанных областей проекций. Такими могут являться часть лица пользователя. Применение метода проекций для сравнения кисти пользователя и аватара позволяет определить качество изображения. Проводится сравнение проекций кисти и эталона, хранящегося в библиотеке тестов. Для этого проекции приводятся к одному масштабу (рис. 6).

Вычисляется корреляция изображений по проекциям. Полученные значения приводятся к интервалу [0,1] и определяется вероятность совпадения жестов по осям X и Y (табл. 1).

Полученные результаты подобия проекций жеста и эталона обрабатываются статистическими методами для принятия решения, которые применяются на нескольких этапах идентификации.

Для принятия решения о правильности показанного жеста используется правило Байеса на основе апостериорной вероятности:

$$P(A) = P(B_1)P(A|B_1) + P(B_2)P(A|B_2), \qquad (1)$$

где $P(A)$ — полная вероятность события A (совпадение жестов); $P(B_i)$ — априорная вероятность события B_i (совпадение по осям X и Y); $P(A|B_i)$ — апостериорная вероятность (результаты сравнения).

Априорные вероятности являются независимыми и имеют нормальное распределение, поэтому равны 0,5:

$$P(A) = 0{,}5\times0{,}818 + 0{,}5\times0{,}9757 = 0{,}8969.$$

При локализованном изображении жеста и заявленном эталоне S система идентификации должна выбрать одну из следующих гипотез:

H_S — жест определяется как эталон S;

$H_{\bar{S}}$ — жест не определяется как эталон S.

Решение о выборе между двумя гипотезами основано на отношении правдоподобия:

$$W(X) = \frac{p(X \mid H_S)}{p(X \mid H_{\bar{S}})} = \frac{P(A)}{1 - P(A)} = \begin{cases} > s, \text{ совпадает с эталоном} \\ \leqslant s, \text{ не совпадает с эталоном} \end{cases}. \qquad (2)$$

Величина s вводится как порог принятия или отклонения гипотезы.

$$W(X) = 0{,}89689/0{,}1031 = 8{,}699.$$

Если имеются сомнения в правильности идентификации и принять окончательное решение затруднительно, то проводится второй этап.

Локализованное изображение жеста обрезается наполовину по высоте при вертикальном жесте и наполовину по ширине при горизонтальном жесте. Также обрезается эталон (извлекается из библиотеки эталонов). Отсчет ведется со стороны пальцев (сверху и справа). По полученным изображениям строятся проекции по вертикали и горизонтали в одинаковом масштабе (рис. 7).

Вычисляется совпадение изображений по проекциям (табл. 2).

Вычисляется вероятность совпадения изображения с эталоном и вычисляется отношение правдоподобия по формулам (1) и (2):

$$P(A) = 0{,}8190287960, \; W(X) = 4{,}5257409928.$$

Таблица 1

Результаты сравнения проекций жеста и эталона для дактилемы *H*

Проекции	Приведенные значения
Вертикальная по оси X	0,8181264923
По оси Y только пальцы	0,9756716953

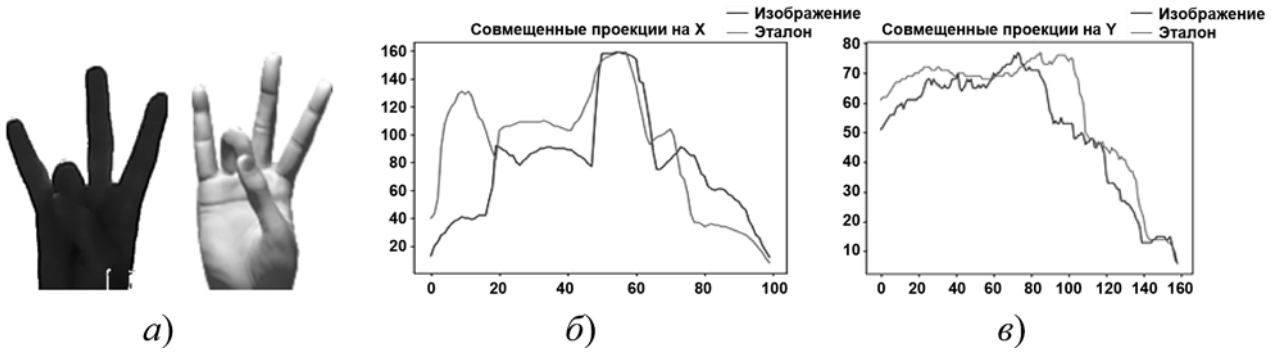
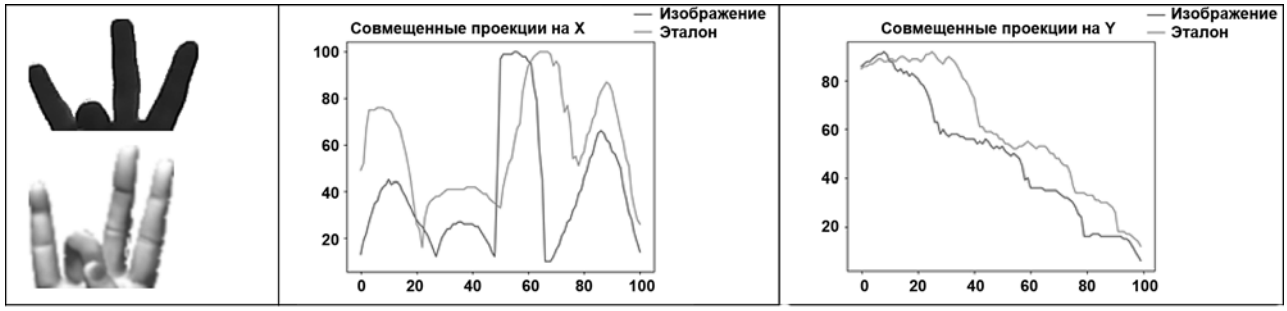


Рис. 6. Изображения кисти пользователя и аватара (*а*), их совмещенные проекции по оси X (*б*) и по оси Y (*в*) для дактилемы *H*



а) *б)* *в)*

Рис. 7. Изображения пальцев пользователя и аватара, их совмещенные проекции:
а — изображения; *б* — вертикальная проекция; *в* — горизонтальная проекция

На этом этапе принимается окончательное решение с учетом результатов предыдущего этапа.

Рассматривались все возможные варианты локализованного жеста: изображения разного качества, неправильно показанные жесты, вертикальные и горизонтальные жесты, жесты с различным наклоном. Определены границы принятия решения. Значение порога принятия решения зависит от этапов идентификации. В результате проведения экспериментов определены пороговые значения.

Алгоритм принятия решения определяется по следующим действиям и может состоять из двух этапов.

На этапе 1:

1) если значение отношения правдоподобия $W(X) <= 3$ ($s_1 = 3$), то жест не распознан по причине плохого качества;

2) если значение отношения правдоподобия $W(X) > 6$ ($s_2 = 6$), то имеем отличное качество и жест идентифицируется как правильный;

3) если значение отношения правдоподобия $3 < W(X) \leqslant 6$ (находится в интервале между значениями 3 и 6), то продолжаем исследование; переход на 2 этап.

На этапе 2:

1) если значение отношения правдоподобия $W(X) \leqslant 3$ ($s_3 = 3$), то жест идентифицируется как неверный;

2) если значение отношения правдоподобия $W(X) > 3$, то жест идентифицируется как правильный.

Для динамического жеста между ключевыми кадрами определяется тренд перемещения кисти. Сравниваются координаты центров кисти соседних ключевых кадров, вычисляется тренд по горизонтали и вертикали (по координатам X и Y), который сравнивается с эталонным трендом (с ограничениями и неравенствами эталона).

Неравенства формулируются как отношения "больше" или "меньше" в зависимости от движения вниз или вверх:

Таблица 2

Показатели подобия

Проекции	Приведенные значения
Вертикальная по оси X	0,6694093655
Горизонтальная по оси Y	0,9686482266

$$(X\text{к} - X\text{н}) > e_\text{г} \text{ или } (X\text{к} - X\text{н}) < -e_\text{г}$$
$$\text{и } (Y\text{к} - Y\text{н}) > e_\text{в} \text{ или } (Y\text{к} - Y\text{н}) < -e_\text{в},$$

где Xн, Yн — координаты центра кисти первого ключевого кадра; Xк, Yк — координаты центра кисти второго ключевого кадра; $e_\text{г}$, $e_\text{в}$ — допустимые отклонения по горизонтали и вертикали.

В результате проведения экспериментов определены пороговые значения $e_\text{г} = 20$ и $e_\text{в} = 15$ пикселей.

Ограничения используются при повороте кисти, при изменении конфигурации пальцев, при строго горизонтальном или вертикальном движении (кисть остается на месте по соответствующим координатам):

$$|X\text{к} - X\text{н}| < e_\text{г} \text{ и/или } |Y\text{к} - Y\text{н}| < e_\text{в}.$$

Пример распознавания динамической дактилемы "Щ" показан на рис. 8 и 9.

Вычисляются координаты центра кисти в первом ($x_1 = 228$, $y_1 = 229$) и втором ($x_2 = 238$, $y_2 = 278$) ключевых кадрах.

Вычисляется перемещение:

$$(x_2 - x_1) = |238 - 228| = 10 < e_\text{г},$$
$$(y_2 - y_1) = (278 - 229) = 49 > e_\text{в}.$$

Рис. 8. Структура нотации дактилемы "Щ"

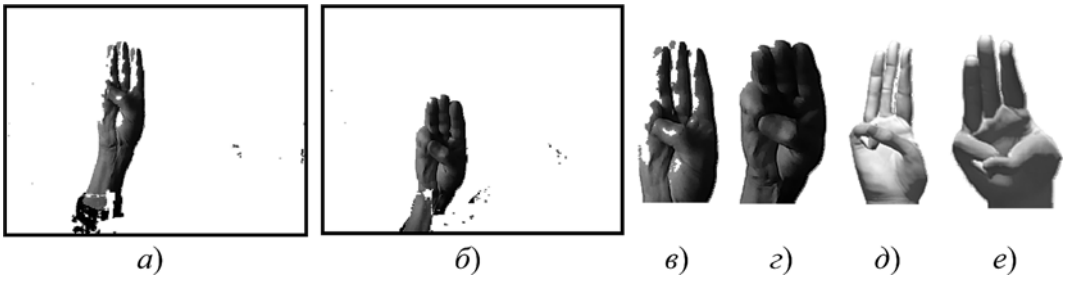


Рис. 9. Первый (*а*) и второй (*б*) ключевые кадры после удаления фона на полном растре, локализованные жесты (*в*) и (*г*) и соответствующие эталоны (*д*) и (*е*)

Определяется тип движения: "перемещение кисти вертикально вниз". Проводится сравнение проекций жестов и эталонов, вычисляются вероятности совпадения изображений с эталонами и определяются отношения правдоподобия ($W_1(X)$ = 10,6073739989, $W_2(X)$ = 6,5354590156): "оба ключевых кадра определены как совпадающие с эталонами". Итоговый результат: "Жест идентифицируется как правильный".

Для анализа эффективности распознавания жестов в режиме тестирования в интегрированной системе обучения ЖЯ были проведены экспериментальные исследования разработанных средств. Предложенный метод идентификации жеста показывает результат правильного распознавания 95,2 %.

Заключение

Разработанная автором интегрированная система обучения ЖЯ обладает полным функционалом для создания справочника и уроков ЖЯ. Предложенный метаязык для описания движения подвижных элементов аватара обладает наглядностью, удобством редактирования и позволяет сократить объем файлов более чем на 300 % по сравнению с анимационными файлами Unity 3D.

Система работает в реальном времени и доступна как через сеть Интернет, так и на персональных устройствах. Методика наполнения справочника позволяет сторонним пользователям легко дополнять справочник. Разработаны средства для тестирования знаний пользователя на мобильных устройствах и на стационарных устройствах с неподвижной веб-камерой.

Автор выражает благодарность научному руководителю работы д-ру техн. наук, проф. кафедры АСУ НГТУ Грифу Михаилу Геннадьевичу.

Список литературы

1. **O'hara K., Gonzalez G., Sellen A. l.** et al. Touchless interaction in surgery // Communications of the ACM. 2014. Vol. 57, No. 1. P. 70—77. DOI: 10.1145/2541883.2541899.

2. **Hartmann F., Schlaefer A.** Feasibility of touch-less control of operating room lights // International Journal of Computer Assisted Radiology and Surgery. 2012. Vol. 8. P. 259—268. DOI: 10.1007/s11548-012-0778-2.

3. **Hongyi Liu, Lihui Wang** Gesture recognition for human-robot collaboration // A review International Journal of Industrial Ergonomics Elsevier. 2018. Vol. 68. P. 355—367. DOI:10.1016/J.ERGON.2017.02.004.

4. **Hartmann F., Schlaefer A.** Feasibility of touch-less control of operating room lights // International Journal of Computer Assisted Radiology and Surgery. 2012. Vol. 8, No. 2. P. 259—268. DOI: 10.1007/s11548-012-0778-2.

5. **Кагиров И. А., Рюмин Д. А., Аксенов А. А., Карпов А. А.** Мультимедийная база данных жестов русского жестового языка в трехмерном формате // Вопросы языкознания. 2020. № 1. С. 104—123. DOI: 10.31857/S0373658X0008302-1.

6. **Карпов А. А., Кагиров И. А.** Формализация лексикона системы компьютерного синтеза языка жестов // Труды СПИИРАН. 2011. Вып. 1 (16). С. 123—140. DOI: 10.15622/sp.16.4

7. **Гриф М. Г., Лукоянычев А. В.** 3D-анимация русского жестового языка на основе нотации Димскис // Программная инженерия. 2017. Т. 8, № 7. С. 310—318. DOI: 10.17587/prin.8.310-318.

8. **Мясоедова М. А. Мясоедова З. П.** Жестовые нотации и их сравнительный анализ // Современные информационные технологии и ИТ-образование. 2018. [S.l.]. Т. 14. № 1. С. 183—192. DOI: 10.25559/SITITO.14.201801.183-192.

9. **Димскис Л. С.** Изучаем жестовый язык: учебное пособие. М.: Академия, 2002. 128 с.

10. **Речицкая Е. Г.** Дактильная и жестовая речь как средства коммуникации лиц с нарушением слуха: учебно-методическое пособие: в 2 частях. М.: Московский педагогический государственный университет (МПГУ), 2016. 144 с.

11. **Buttussi F., Chittaro L., Coppo M.** Using Web3D technologies for visualization and search of signs in an international sign language dictionary. ACM Press, 2007. 61 p.

12. **Гриф М. Г., Лукоянычев А. В.** Программный комплекс для обучения русскому жестовому языку на основе Unity3D // Программная инженерия. 2018. Т. 9, № 8. С. 375—384. DOI: 10.17587/prin.9.375-384.

13. **Otsu N.** A Threshold Selection Method from Gray-Level Histograms // IEEE Transactions on Systems, Man, and Cybernetics. 1979. Vol. 9, No. 1. P. 62—66.

Integrated Learning System of Sign Language

A. V. Lukoyanychev, dizzystyle@yandex.com, Novosibirsk state technical University, Novosibirsk, 630073, Russian Federation

Corresponding author:

Alexey V. Lukoyanychev, Undergraduate,
Novosibirsk state technical University, Novosibirsk, 630073, Russian Federation
E-mail: dizzystyle@yandex.com

Received on June 10, 2022
Accepted on June 27, 2022

The developed integrated system of sign language teaching, aimed at a wide range of users, is considered. The Dimskis notation for sign language writing implementation and the Unity 3D cross-platform development to control an animated 3D character are used in this system. The approach for the describing of 3D character movements based on the meta-language is proposed. The Dimskis notation has been expanded, within the framework of the meta-language. Additional and service characters, control words, special coordinates were introduced. Special control symbols and rules for processes synchronization have been developed. It is allows one to implement the movement of several elements at different times, to create complex movements, returning them to the initial state, to cancel the transferences of individual moving elements by using the "fictitious" movement, to change only individual coordinates of the avatar's hand. The technique of creating the sign language reference book is considered. The advantages of this approach are analyzed. In addition to this full-fledged functional toolkit for creating the reference book the integrated system has the ability to test the user's knowledge. The mode gesture demonstration — answer selection is proposer for mobile devices. For stationary devices with a fixed web camera the user has to show the gesture proposed by the system. The following stages of this mode implementation are considered: image capture, gesture localization, gesture identification, decision making. The gesture localization is based on the double background subtraction and histogram method. The gesture identification is based on the projection method and is carried out in two stages. At the first stage the entire hand is analyzed. If it is difficult to make a decision at this stage, then the second stage is carried cut, where only fingers are analyzed. The trend of arm movement between key frames is additionally determined for dynamic gestures. The features of each of the stages are shown. The algorithm for making a decision is given. The results of an experimental investigation are presented.

Keywords: *sign language, Dimskis notation, Unity 3D, avatar control meta-language, gesture localization, gesture identification*

For citation:

Lukoyanychev A. V. Integrated Learning System of Sign Language, *Programmnaya Ingeneria,* 2022, vol. 13, no. 8, pp. 404—412.

DOI: 10.17587/prin.13.404-412

References

1. **O'hara K., Gonzalez G., Sellen A.** et al. Touchless interaction in surgery, *Communications of the ACM,* 2014, vol. 57, no. 1, pp. 70—77. DOI: 10.1145/2541883.2541899.

2. **Hartmann F., Schlaefer A.** Feasibility of touch-less control of operating room lights, *International Journal of Computer Assisted Radiology and Surgery*, 2012, vol. 8, pp. 259—268. DOI: 10.1007/s11548-012-0778-2.

3. **Hongyi Liu, Lihui Wang** Gesture recognition for human-robot collaboration *A review International Journal of Industrial Ergonomics Elsevier*, 2018, vol. 68, pp 355-367. DOI:10.1016/J.ERGON.2017.02.004.

4. **Hartmann F., Schlaefer A.** Feasibility of touch-less control of operating room lights, *International Journal of Computer Assisted Radiology and Surgery*, 2012, vol. 8, no. 2, pp 259—268. DOI: 10.1007/s11548-012-0778-2.

5. **Kagirov I. A., Ryumin D. A., Aksenov A. A., Karpov A. A.** Multimedia database of Russian sign language gestures in three-dimensional format, *Voprosy yazykoznaniya*, 2020, no. 1, pp. 104—123. DOI: 10.31857/S0373658X0008302-1 (in Russian).

6. **Karpov A. A., Kagirov I. A.** The formalization of the lexicon of computer synthesis of sign language system, *Trudy SPIIRAN*, 2011, iss. 1 (16), pp. 123—140. DOI: 10.15622/sp.16.4.

7. **Grif M. G., Lukoyanychev A.V.** 3D - animation of Russian sign language on the basis of Dimskis notation, *Programmnaya Ingeneria,* 2017, vol. 8, no. 7, pp. 310-318. DOI: 10.17587/prin.8.310-318. (in Russian).

8. **Myasoedova M. A., Myasoedova Z. P.** Sign notations and their comparative analysis, *Sovremennye informacionnye tekhnologii i IT-obrazovanie*, 2018, [S.l.], vol. 14, no. 1, pp. 183—192. DOI: 10.25559/SITITO.14.201801.183-192 (in Russian).

9. **Dimskis L. S.** *The study of sign language: study guide*, Moscow, Academy, 2002, 128 p. (in Russian).

10. **Rechitskaya E. G.** *Dactyl and sign language as a means of communication of persons with hearing impairment: educational and methodical manual*: in 2 parts, Moscow, Moscow Pedagogical State University (MPSU), 2016, 144 p. (in Russian).

11. **Buttussi F., Chittaro L., Coppo M.** *Using Web3D technologies for visualization and search of signs in an international sign language dictionary*, ACM Press, 2007, 61 p.

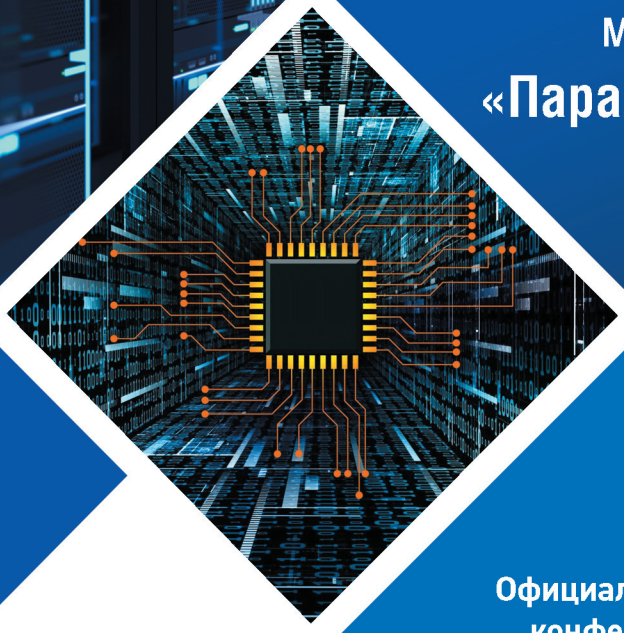
12. **Grif M. G., Lukoyanychev A. V.** The program complex for training russian sign language based on Unity3D, *Programmnaya Ingeneria*, 2018, vol. 9, no. 8, pp. 375—384. DOI: 10.17587/prin.9.375-384 (in Russian).

13. **Otsu N.** A Threshold Selection Method from Gray-Level Histograms, *IEEE Transactions on Systems, Man, and Cybernetics*, 1979, vol. 9, no. 1, pp. 62—66.

ООО "Издательство "Новые технологии". 107076, Москва, ул. Матросская Тишина, д. 23, стр. 2
Технический редактор *Е. М. Патрушева*. Корректор *А. В. Чугунова*.

Сдано в набор 21.07.2022 г. Подписано в печать 24.08.2022 г. Формат 60×88 1/8. Заказ PI821
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**



Международная научная конференция

«Параллельные вычислительные технологии (ПаВТ) 2023»

28–30 марта 2023 г.,
Санкт-Петербург, Университет ИТМО

Официальный сайт конференции:

http://agora.guru.ru/pavt2023/

Параллельные вычислительные технологии (ПаВТ) 2023» – международная научная конференция, 17-я в серии ежегодных конференций, посвященных развитию и применению параллельных вычислительных технологий и машинного обучения в различных областях науки и техники. **Главная цель конференции –** предоставить возможность для представления и обсуждения результатов, полученных ведущими научными группами в использовании суперкомпьютерных и нейросетевых технологий для решения практических задач.

Организаторы конференции:

- Министерство науки и высшего образования РФ;
- Суперкомпьютерный консорциум университетов России.

Тематика конференции покрывает все аспекты применения облачных, суперкомпьютерных и нейросетевых технологий в науке и технике, включая приложения, аппаратное и программное обеспечение, специализированные модели, языки, библиотеки и пакеты.

В первый день работы конференции будет объявлена **38-я редакция списка Top50** самых мощных компьютеров СНГ.

Во все дни работы конференции будет действовать **суперкомпьютерная выставка,** на которой ведущие производители аппаратного и программного обеспечения представят свои новейшие разработки в области высокопроизводительных вычислений.

ВАЖНЫЕ ДАТЫ

Представление аннотации:
1 декабря 2022 г.

Представление статьи:
15 декабря 2022 г.

Уведомление о включении в программу конференции:
1 февраля 2023 г.

Регистрация заявок авторов принятых докладов:
15 февраля 2023 г.

Представление окончательного варианта статьи:
15 февраля 2023 г.

Завершение приема заявок на участие в суперкомпьютерной выставке:
1 марта 2023 г.

Завершение приема заявок на участие без доклада:
21 марта 2023 г.

Издательство «НОВЫЕ ТЕХНОЛОГИИ»

выпускает *научно-технические журналы*

Теоретический и прикладной научно-технический журнал

ПРОГРАММНАЯ ИНЖЕНЕРИЯ

В журнале освещаются состояние и тенденции развития основных направлений индустрии программного обеспечения, связанных с проектированием, конструированием, архитектурой, обеспечением качества и сопровождением жизненного цикла программного обеспечения, а также рассматриваются достижения в области создания и эксплуатации прикладных программно-информационных систем во всех областях человеческой деятельности.

Подписной индекс по Объединенному каталогу «Пресса России» – 22765

Ежемесячный теоретический и прикладной научно-технический журнал

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

В журнале освещаются современное состояние, тенденции и перспективы развития основных направлений в области разработки, производства и применения информационных технологий.

Подписной индекс по Объединенному каталогу «Пресса России» – 72656

Междисциплинарный теоретический и прикладной научно-технический журнал

НАНО- и МИКРОСИСТЕМНАЯ ТЕХНИКА

В журнале освещаются современное состояние, тенденции и перспективы развития нано- и микросистемной техники, рассматриваются вопрос разработки и внедрения нанои микросистем в различные области науки, технологии и производства.

Подписной индекс по Объединенному каталогу «Пресса России» – 79493

Ежемесячный теоретический и прикладной научно-технический журнал

МЕХАТРОНИКА, АВТОМАТИЗАЦИЯ, УПРАВЛЕНИЕ

В журнале освещаются достижения в области мехатроники, интегрирующей механику, электронику, автоматику и информатику в целях совершенствования технологий производства и создания техники новых поколений. Рассматриваются актуальные проблемы теории и практики автоматического и автоматизированного управления техническими объектами и технологическими процессами в промышленности, энергетике и на транспорте.

Подписной индекс по Объединенному каталогу «Пресса России» – 79492

Научно-практический и учебно-методический журнал

БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

В журнале освещаются достижения и перспективы в области исследований, обеспечения и совершенствования защиты человека от всех видов опасностей производственной и природной среды, их контроля, мониторинга, предотвращения, ликвидации последствий аварий и катастроф, образования в сфере безопасности жизнедеятельности.

Подписной индекс по Объединенному каталогу «Пресса России» – 79963

Адрес редакции журналов для авторов и подписчиков:
107076, Москва, ул. Матросская Тишина, д. 23, стр. 2, оф. 45. Издательство "НОВЫЕ ТЕХНОЛОГИИ".
Тел.: (499) 270-16-52. E-mail: antonov@novtex.ru

ISSN 2220-3397. Программная инженерия. 2022. Том 13. № 8. 361 – 412. Подписной индекс по Объединенному каталогу "Пресса России" – 22765