ВЕСТНИК
НОВОСИБИРСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

Научный журнал
Основан в ноябре 1999 года

Серия: Информационные технологии 2021. Том 19, № 4

СОДЕРЖАНИЕ

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2021. Том 19, № 4
Vestnik NSU. Series: Information Technologies, 2021, vol. 19, no. 4

VESTNIK
NOVOSIBIRSK STATE UNIVERSITY

Scientific Journal
Since 1999, November
In Russian

Series: Information Technologies 2021. Volume 19, № 4

CONTENTS

ISSN 1818-7900 (Print). ISSN 2410-0420 (Online)
Вестник НГУ. Серия: Информационные технологии. 2021. Том 19, № 4
Vestnik NSU. Series: Information Technologies, 2021, vol. 19, no. 4

Editor in Chief M. M. Lavrentiev

Vice-Editor A. V. Avdeev

Executive Secretary D. P. Iksanova

Editorial Board of the Series

I. V. Bychkov, professor, academician (Irkutsk), *B. M. Glinsky*, professor (Novosibirsk)
A. N. Gorban, professor (Lester, GB), *E. P. Gordov*, professor (Tomsk)
B. S. Dobronets, professor (Krasnoyarsk), *A. M. Elizarov*, professor (Kazan)
G. N. Erokhin, professor (Kaliningrad), *A. I. Kamyshnikov*, professor (Khanty-Mansijsk)
G. P. Karev, professor (Maryland, USA), *N. A. Kolchanov*, professor, academician (Novosibirsk)
M. M. Lavrentjev, professor (Novosibirsk), *V. E. Malyshkin*, professor (Novosibirsk)
N. N. Mirenkov, professor (Aizu, Japan), *N. M. Oskorbin*, professor (Barnaul)
D. E. Palchunov, professor (Novosibirsk), *T. Pizansky*, professor (Ljubljana, Slovenia)
V. P. Potapov, professor (Kemerovo), *O. I. Potaturkin*, professor (Novosibirsk)
V. A. Serebryakov, professor (Moscow), *A. V. Starchenko*, professor (Tomsk)
S. I. Smagin, professor, corresponding member of RAS (Khabarovsk)
D. A. Tusupov, professor (Astana, Kazakhstan)
V. V. Shajdurov, professor, corresponding member of RAS (Krasnoyarsk)
Yu. I. Shokin, professor, academician (Novosibirsk)

The journal is published quarterly in Russian since 1999
by Novosibirsk State University Press

The address for correspondence

Faculty of Information Technologies, Novosibirsk State University

1 Pirogov Street, Novosibirsk, 630090, Russia

Tel. +7 (383) 363 42 46

E-mail address: inftech@vestnik.nsu.ru

On-line version: http://elibrary.ru

Научная статья

УДК 004.4’2, 004.49, 681.5, 004.056.53
DOI 10.25205/1818-7900-2021-19-4-5-15

Методы защиты команд управления в инструментальной среде для автоматизированных систем управления технологическими процессами

Анатолий Иванович Благодарный [1]
Борис Николаевич Пищик [2]

[1, 2] Федеральный исследовательский центр информационных и вычислительных технологий
Новосибирск, Россия

[2] Новосибирский государственный университет
Новосибирск, Россия

[1] ablagodarnyj@mail.ru, https://orcid.org/0000-0001-8668-1094
[2] pishchik.boris@ya.ru, https://orcid.org/0000-0001-9083-4807

Аннотация

Рассматриваются методы, обеспечивающие выполнение только корректных и легальных команд управления в автоматизированных системах управления технологическими процессами (АСУ ТП). Защита целостности программного обеспечения АСУ ТП на всех узлах технологической сети, включая защиту от несанкционированной подмены, осуществляется проверкой контрольных параметров программного обеспечения во время запуска. Защита команд управления от искажений в процессе передачи по сети реализуется двухкратным кодированием на разных уровнях системы. Защита от подачи на технологическое оборудование несанкционированных команд управления осуществляется посредством распознавания этих команд в подсистеме нижнего уровня и сравнением их текущих характеристик с контрольными.

Определенный вклад в безопасность работы АСУ ТП вносит использование безопасной среды исполнения – ОС CentOS 7.

Ключевые слова

АСУ ТП, SCADA, методы защиты, команды управления

Для цитирования

Благодарный А. И., Пищик Б. Н. Методы защиты команд управления в инструментальной среде для автоматизированных систем управления технологическими процессами // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 4. С. 5–15. DOI 10.25205/1818-7900-2021-19-4-5-15

Methods for Protecting Control Commands in the Instrumental Environment for Automated Process Control Systems

Anatoly I. Blagodarny [1], **Boris N. Pishchik** [2]

[1, 2] Federal Research Center for Information and Computing Technologies
Novosibirsk, Russian Federation

[2] Novosibirsk State University
Novosibirsk, Russian Federation

[1] ablagodarnyj@mail.ru, https://orcid.org/0000-0001-8668-1094
[2] pishchik.boris@ya.ru, https://orcid.org/0000-0001-9083-4807

Abstract

The paper discusses methods that ensure the execution of only correct and legal control commands in the process control system. Protection of the integrity of the APCS software at all nodes of the industrial network, including protec-

© Благодарный А. И., Пищик Б. Н., 2021

tion against unauthorized substitution, is carried out by checking the control parameters of the software during their startup. Protection of control commands from distortions during transmission over the network is realized by double coding at different levels of the system. Protection against the submission of unauthorized control commands to technological equipment is carried out by recognizing these commands in the lower-level subsystem and comparing their current characteristics with the control ones.

A certain contribution to the safety of the process control system is made by the use of a secure execution environment – OS CentOS 7.

Keywords

APCS, SCADA, protection methods, control commands

For citation

Blagodarnyy A. I., Pishchik B. N. Methods for Protecting Control Commands in the Instrumental Environment for Automated Process Control Systems. *Vestnik NSU. Series: Information Technologies*, 2021, vol. 19, no. 4, p. 5–15. (in Russ.) DOI 10.25205/1818-7900-2021-19-4-5-15

Введение

Для создания автоматизированных систем управления технологическими процессами (АСУ ТП), как правило, используется некоторая инструментальная среда (ИС), именуемая в различных источниках как ICS, SCADA или PLC.

Рассматриваемая в данной статье инструментальная среда является развитием и обобщением SCADA-системы «Блакарт», использованной при создании нескольких конкретных АСУ ТП, внедренных на предприятиях транспорта нефти [1; 2]. «Блакарт» имеет реализации для операционных систем реального времени «QNX 4.25» [3] и «QNX 6.5 Нейтрино» [4]. На ее основе были реализованы АСУ ТП на предприятиях угледобывающей промышленности [5–9], которые показали свои высокие эксплуатационные качества [10].

Безопасность АСУ ТП как проблема обсуждается в работах многих авторов [11–15]. Проблема многоаспектная и, безусловно, актуальная и в настоящее время.

В ФИЦ ИВТ на основе «Блакарт» ведется разработка ИС в операционной системе с открытым исходным кодом CentOS 7. Цель проекта – разработка ИС, обеспечивающей *безопасное управление* в создаваемых на их основе АСУ ТП. Не умаляя важности других проблем, в данном проекте особое внимание уделяется процессу выполнения команд управления, поскольку обеспечение корректности работы АСУ ТП в значительной степени связано с возможностью контроля исполнения команд управления с целью не допустить их несанкционированного исполнения, инициированного кибератаками, которые, в свою очередь, начинаются с поиска и использования уязвимостей как самой системы, так и среды ее исполнения.

1. Среда исполнения

Для обеспечения киберустойчивости прикладного программного обеспечения АСУ ТП инструментальная среда реализуется на операционной системе CentOS 7. Эта операционная система (ОС) выбрана как безопасная среда с открытым исходным кодом для исполнения программного обеспечения АСУ ТП. Под безопасностью здесь понимается отсутствие уязвимостей, которые могут быть использованы злоумышленником для атаки на АСУ ТП.

Исследование банка данных угроз безопасности информации ФСТЭК [1] и открытых баз данных уязвимостей NIST [2] показывает, что уязвимости в компонентах CentOS 7, необходимых для функционирования АСУ ТП (ядро системы, различные пакеты сервиса rsyslog, библиотеки OpenSSL, технологии WPA и сетевого обмена файлами (Samba)), которые могут быть использованы для удаленного доступа и исполнения кода, устранены. Эти уязвимости

[1] Банк данных угроз безопасности информации. ФСТЭК России, ФАУ «ГНИИИ ПТЗИ ФСТЭК России». URL: https://bdu.fstec.ru/.

[2] National vulnerability database. URL: https://nvd.nist.gov/.

(более 180) были обнаружены в период 2014–2015 гг., и с тех пор о новых уязвимостях в этих компонентах сообщений не было.

2. Структура инструментальной среды

Разрабатываемая ИС позволяет реализовать АСУ ТП для широкого спектра технологических процессов и состоит из двух подсистем: подсистемы верхнего уровня и подсистемы нижнего уровня. Подсистема верхнего уровня реализует функции автоматизированного рабочего места (АРМ) оператора и сервера архивных данных, подсистема нижнего уровня – доступ к контролируемому технологическому оборудованию.

Обмен данными между указанными выше подсистемами осуществляется по каналам связи локальной вычислительной сети (технологическая сеть), объединяющей только АРМы и технологическое оборудование. Сеть состоит из узлов, каждый из которых является вычислительным устройством, на котором запущена или подсистема верхнего уровня, или (и) подсистема нижнего уровня. В технологической сети особое место занимает один выделенный узел, который называется узлом системного администратора. Связь технологической сети с другими сетями предприятий осуществляется только через выделенные сетевые адаптеры шлюзового компьютера с контролем используемых сетевых сервисов, например, при помощи односторонних сетевых адаптеров.

Подсистема верхнего уровня состоит из набора программных модулей: модуль графического интерфейса оператора, модуль базы данных, модуль синхронизации времени, модуль контроля состояния технологической сети, модуль контроля текущего состояния программного обеспечения на узлах технологической сети, модуль контроля целостности программного обеспечения на узлах технологической сети, а также буферные модули обмена данными.

Подсистема нижнего уровня включает в себя набор драйверов обслуживания плат ввода / вывода, драйверов связи с логическими контроллерами и буферные модули обмена данными. Общая структура среды представлена на рисунке, где светлые линии показывают пути передачи данных от датчиков к компьютерам, а темные линии – пути передачи команд управления от компьютеров к драйверам и исполнительным механизмам. Серые линии указывают на каналы синхронизации времени в узлах.

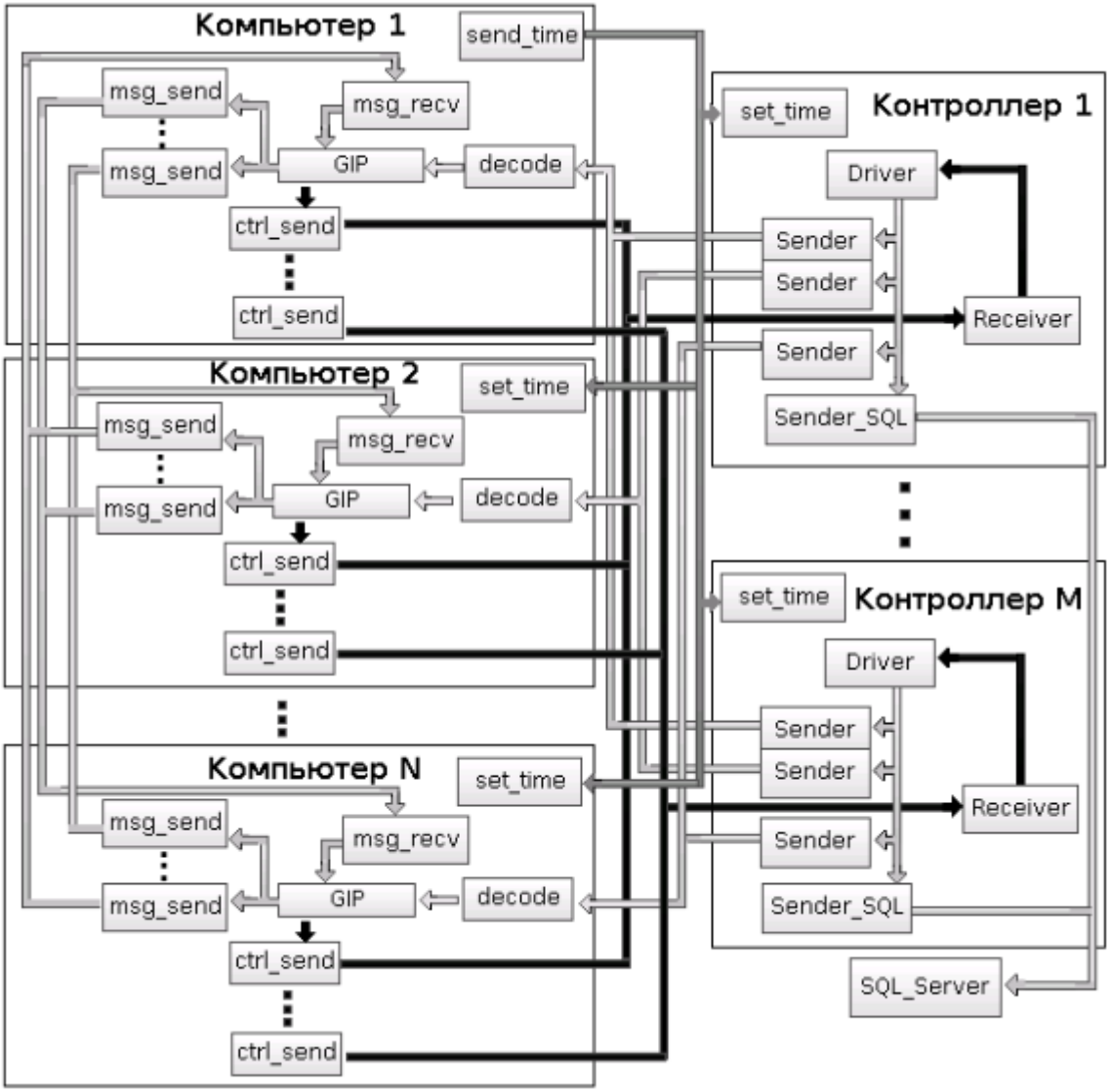
На физическом уровне в технологической сети используется канальный протокол технологии Ethernet. Обмен данными между программными модулями, находящимися на различных узлах технологической сети, осуществляется только по протоколу нижнего уровня UDP стека протоколов TCP/IP.

Основными достоинствами указанного протокола являются максимальная помехоустойчивость, максимальное быстродействие, максимально низкий сетевой трафик, а также наличие в пакетах данных всей идентифицирующей взаимодействующих абонентов информации. Для адресации абонентов в пределах технологической сети используются локальные IP-адреса.

К недостаткам протокола UDP относится необходимость подтверждения обмена данными, поскольку надежность доставки данных, а также отсутствие искажений в переданных данных не гарантируются. Впрочем, несмотря на декларируемый метод гарантированной доставки данных при использовании транспортного протокола TCP, как правило, для всего разрабатываемого прикладного программного обеспечения все равно используется или механизм подтверждения обмена данными, или контроль последовательности передаваемых пакетов данных, поскольку всегда существует возможность несанкционированного разрыва установленного соединения между взаимодействующими абонентами.

Обмен данными между программными модулями, находящимися на одном узле технологической сети, осуществляется только через *буферы данных в разделяемой оперативной памяти*. Указанные буферы памяти открываются только на чтение для всех программных

модулей, исключая владельца буфера данных (того, кто создал буфер данных). Для владельца доступны как операция чтения, так и операция записи.



Общая структура инструментальной среды
General structure of the instrumental environment

Создание конкретного проекта АСУ ТП в инструментальной среде включает в себя только разработку графического интерфейса оператора и написание текстовых конфигурационных файлов подсистемы верхнего уровня, подсистемы нижнего уровня и конфигурационного файла технологической сети (средства конфигурирования).

Конфигурационные файлы подсистемы верхнего уровня (КФВУ) определяют 17-значные символьные коды сигналов состояния и управления технологическими объектами, а также методы обработки и методы отображения сигналов, полученных от контролируемого технологического оборудования сигналов, на графическом интерфейсе оператора.

Конфигурационные файлы подсистемы нижнего уровня (КФНУ) задают кодировку сигналов состояния и управления, представляющую собой пятерку неотрицательных коротких (16-разрядных) целых чисел и определяют последовательность распределения сигналов состояния и управления по датчикам и исполнительным реле системы сопряжения с технологическим оборудованием.

Конфигурационный файл технологической сети определяет список допустимых номеров узлов технологической сети, присвоенные узлам локальные IP-адреса и прочую информацию, идентифицирующую узлы.

3. Методы защиты команд управления

Методы защиты команд управления технологическим оборудованием в АСУ ТП можно разделить на три основных направления:

- методы защиты целостности прикладного программного обеспечения;
- методы защиты команд управления от ошибок передачи данных по технологическим сетям;
- методы защиты от подачи на контролируемое технологическое оборудование несанкционированных команд управления.

3.1. Защита целостности прикладного программного обеспечения

Прикладное программное обеспечение (ППО) АСУ ТП имеется на каждом узле сети. Однако особым узлом технологической сети является узел системного администратора. На узле системного администратора хранится образцовое программное обеспечение инструментальной среды, часть модулей которого тиражируется на другие узлы и используется в конкретной АСУ ТП в качестве прикладного программного обеспечения на верхнем и нижнем уровнях. Таким образом, особого внимания требует защита узла системного администратора
от несанкционированного внешнего доступа и / или нарушения целостности программного обеспечения на нем. Здесь применяются криптографические методы аутентификации и авторизации, а также административные регламенты допуска к работе с системой.

Методы защиты целостности ППО АСУ ТП на всех узлах технологической сети включают защиту от несанкционированной подмены любого программного модуля или средств конфигурирования как в процессе их функционирования в оперативной памяти, так и при подмене соответствующих файлов на долговременном устройстве хранения.

При запуске ППО каждого узла производится контроль целостности всех файлов, содержащих соответствующие программные модули. Такой контроль осуществляется сравнением с образцовыми файлами ППО, находящимися на узле системного администратора. Метод сравнения включает в себя контроль времени и даты формирования файлов, контроль их размеров и проверку контрольных сумм программных модулей. В случае обнаружения нарушения целостности файлов ППО модуль контроля целостности прекращает запуск ППО узла и формирует аварийный сигнал с указанием номера узла и имени поврежденного программного модуля, который передается на все АРМ оператора технологической сети.

ППО ИС функционирует на узлах технологической сети в виде заданного набора процессов. Каждый процесс в каждый момент времени характеризуется своим состоянием. Из всех возможных состояний процессов аварийными состояниями (иначе называемые сбоем ППО) являются только два: состояние «убит (закрыт)» и состояние «зомби (закрыт, но информация о процессе еще существует в ядре операционной системы)». При обнаружении указанных аварийных состояний каких-либо процессов АСУ ТП программный модуль контроля состояния ППО автоматически формирует аварийный сигнал с указанием названия аварийного программного модуля, его состояния и номера узла. Далее сформированный ава-

рийный сигнал немедленно пересылается на все АРМ оператора. Таким образом, всякая попытка подмены любого программного модуля в процессе его функционирования будет немедленно замечена системой контроля.

Контроль целостности файлов средств конфигурирования при запуске ППО какого-либо узла технологической сети полностью аналогичен контролю целостности файлов программных модулей и осуществляется модулем контроля целостности ППО.

Средства конфигурирования при запуске ППО загружаются в конфигурационные таблицы в разделяемой оперативной памяти, открытой только на чтение. Поскольку доступ к конфигурационным таблицам на запись запрещен, то для подмены конфигурационных таблиц в разделяемой памяти в процессе функционирования ППО необходимо предварительно удалить ранее созданные таблицы и только затем создать новые конфигурационные таблицы. Но в результате подобной операции доступ к вновь созданным таблицам со стороны функционирующих модулей АСУ ТП будет утерян (меняется их адрес в разделяемой оперативной памяти). Это обстоятельство приводит к сбою ППО и формированию программой контроля состояния ППО соответствующих аварийных сигналов с передачей на все АРМ оператора.

3.2. Защита от ошибок передачи команд управления

Протокол нижнего уровня «UDP» контролирует только управляющую информацию в передаваемых дейтаграммах. Контроль пользовательских данных должен осуществляться программным обеспечением АСУ ТП.

Все коды сигналов управления, имеют 2 кодировки: в подсистеме верхнего уровня и в подсистеме нижнего уровня. Все эти коды перечислены в КФВУ и КФНУ.

Передача команд управления из подсистемы верхнего уровня в подсистему нижнего уровня осуществляется управляющими пакетами.

В состав управляющего пакета входят:

- поле кода сигнала управления в кодировке подсистемы верхнего уровня (17-значный символьный код суммарной длиной 136 бит) из КФВУ;
- поля кода сигнала управления в кодировке подсистемы нижнего уровня (пятерка коротких целых неотрицательных чисел суммарной длиной 80 бит) из КФНУ;
- поля авторизации (пароль и идентификатор) оператора и некоторые другие поля.

Далее управляющий пакет передается в подсистему нижнего уровня, где соответствующий драйвер принимает управляющий пакет и проверяет наличие принятого кода сигнала управления в КФВУ. Если такой сигнал существует в КФВУ, то из КФНУ извлекается соответствующий ему код. Далее принятый код сигнала управления и код сигнала из КФНУ сравниваются, и если коды совпали, то сигнал передается на исполнение.

Если сигнал управления с кодом в кодировке подсистемы верхнего уровня не существует или обнаружено расхождение в сравниваемых кодах в кодировке подсистемы нижнего уровня, то производится повторный запрос сигнала управления и далее, при повторной ошибке, драйвер формирует сигнал о невозможности управления, который передается на все АРМ оператора.

Отметим, что при таком подходе вероятность того, что *искаженный* при передаче сигнал управления окажется *допустимым*, очень мала.

Предположим, что общее число команд управления в конфигурационных таблицах равно 1000. А общее число возможных кодов сигналов управления в кодировке подсистемы верхнего уровня (общая длина кода 136 бит) равно 2^{136} или $\sim 10^{45}$. Тогда вероятность того, что искаженный при передаче сигнал управления в кодировке подсистемы верхнего уровня окажется допустимым, равна примерно $1000/10^{45}$, или числу с 42 нулям после запятой. Искаженному сигналу управления может соответствовать только один допустимый сигнал управления в кодировке подсистемы нижнего уровня (общая длина кода 80 бит). Вероятность его синхронного искажения равна $1/2^{80}$, или примерно числу с 26 нулями после запятой.

Общая вероятность незамеченного подсистемой нижнего уровня искажения при передаче сигнала управления является произведением вышеуказанных вероятностей и равна примерно числу с 68 нулям после запятой, т. е. *практически нулевой*.

3.3. Защита от подачи на контролируемое технологическое оборудование несанкционированных команд управления

3.3.1. Распознавание несанкционированных команд управления в подсистеме нижнего уровня

Структура технологической сети имеет ключевое значение для безопасного управления, определяется на этапе ее проектирования и записывается в конфигурационный файл технологической сети (КФТС). Этот файл содержит следующую информацию для каждого узла сети: номер узла, имя узла, тип подсистемы, IP-адрес, флаг базового узла.

Параметр «имя узла» содержит имя установленной операционной системы на узле с десятичным номером – «номер узла». Параметр «тип подсистемы» определяет тип подсистемы (верхний или нижний уровень), запущенной на данном узле. Параметр «флаг базового узла» определяет указанный узел как узел системного администратора с образцовым системным временем.

Если на каком-либо узле технологической сети одновременно запущены и подсистема верхнего уровня, и подсистема нижнего уровня, то такой узел указывается дважды, но с различными значениями номера узла.

Номера портов обмена данными рассчитываются в соответствии с типом подсистемы (верхний или нижний уровень) и номером узла по определенному алгоритму и являются строго фиксированными для каждого узла технологической сети.

Все команды управления технологическим оборудованием, поступающие на вход подсистемы нижнего уровня, *формируются исполняющей системой стека протоколов TCP/IP*, являются дейтаграммами в формате протокола UDP и несут в себе полную информацию о взаимодействующих абонентах, в частности IP-адреса и номера портов обмена данными. Порты обмена данными протокола UDP являются числовыми идентификаторами, однозначно определяющими программные модули, участвующие в обмене.

После получения команды управления подсистема нижнего уровня, используя КФТС, по полученному IP-адресу определяет номер узла, с которого была подана команда. Далее рассчитывается номер порта, с которого должна была быть подана команда.

Команда управления считается несанкционированной если:

- команда является неавторизованной, или код авторизации не является разрешенным системным администратором;
- полученный IP-адрес не найден в конфигурационной таблице технологической сети, или узел с указанным IP-адресом не является узлом с подсистемой верхнего уровня в той же конфигурационной таблице;
- рассчитанный номер порта не совпадает с полученным номером порта в дейтаграмме команды.

Несанкционированная команда управления игнорируется, и драйвер подсистемы нижнего уровня формирует аварийный сигнал, который передается на все АРМ оператора.

3.3.2. Защита от внедрения в дисциплину обмена несанкционированных команд управления

Возможны только два варианта внедрения в алгоритм передачи команд управления с верхнего на нижний уровень системы:

- от внешнего по отношению к технологической сети абонента;

- с помощью программного модуля, внедренного на какой-либо узел технологической сети.

В первом варианте (внедрение внешнего абонента) IP-адрес внешнего по отношению к технологической сети абонента является *посторонним*, не содержится в конфигурационном файле технологической сети, и команда, посланная внешним абонентом, будет отвергнута драйвером подсистемы нижнего уровня с передачей на все АРМ оператора сигнала о попытке постороннего вторжения.

Во втором варианте (команда от внедренного модуля) также возможны только два способа попытки внедрения в дисциплину обмена несанкционированных команд управления: запись команды в буфер команд в разделяемой оперативной памяти в подсистеме верхнего или подсистеме нижнего уровня и непосредственная передача команды в подсистему нижнего уровня.

При первом способе запись несанкционированной команды в буфер команд в разделяемой оперативной памяти будет отвергнута операционной системой, так как буфер команд создан с атрибутом только для чтения.

При втором способе IP-адрес переданной команды может быть вполне легальным, с точки зрения конфигурационного файла технологической сети. Но номера портов обмена данными определяются только номерами узлов технологической сети и жестко закреплены за всеми программными модулями ППО узлов технологической сети. Исходя из вышесказанного, номер порта переданной в подсистему нижнего уровня команды от постороннего программного модуля не будет являться легальным, и команда будет отвергнута драйвером подсистемы нижнего уровня с передачей на все АРМ оператора сигнала о попытке постороннего вторжения.

Заключение

Изложенные выше методы используются в инструментальной системе, на базе которой реализованы промышленные АСУ ТП в нефтегазовой отрасли и угольной промышленности. Отказоустойчивость и киберустойчивость этих АСУ ТП проверена многолетним опытом эксплуатации.

В настоящей статье не рассматриваются вопросы защиты системного программного обеспечения. Основное внимание уделяется защите прикладного программного обеспечения инструментальной среды в части защиты контролируемого технологического оборудования от подачи на него неверных или несанкционированных команд управления, которые несут в себе наибольшую опасность.

Однако представляется очевидным тот факт, что описанная методика защиты команд управления технологическим оборудованием вполне применима и к защите обмена самыми произвольными данными по технологическим сетям.

Список литературы

1. **Золотухин Е., Михальцов Э., Старшинов А., Стратула В., Чейдо Г.** Модернизация АСУ ТП магистральных нефтепроводов // Современные технологии автоматизации. 1997. № 4. С. 18–26.
2. **Благодарный А., Зензин А., Михальцов Э., Петков А., Чейдо Г.** Программируемая информационно-управляющая система – инструмент создания АСУ ТП магистральных нефтепроводов // IT-решения в нефтегазовой отрасли. М.: ИД «Нефть и капитал», 2002. С. 51–58.
3. **Благодарный А. И., Каратышева Л. С.** Универсальная SCADA-системы «Блакарт» под управлением операционной системы QNX // Проблемы информатики. 2009. № 3. С. 62–67.

4. **Благодарный А. И.** Программный инструментарий для построения систем автоматизированного управления в среде отечественной операционной системы // Проблемы информатики. 2018. № 2. С. 41–51.
5. **Благодарный А. И., Гаркуша В. В., Цыба А. М., Чейдо Г. П., Шевченко Д. О., Яковлев В. В.** Автоматизированное управление электроснабжением угольной шахты // Энергетическая безопасность России. Новые подходы к развитию угольной промышленности: Сб. тр. XV Междунар. науч.-практ. конф. / Под ред. В. И. Клишина, З. Р. Исмагилова, В. Ю. Блюменштейна, С. И. Протасова, Г. П. Дубинина. Кемерово: Ин-т угля СО РАН, 2013. С. 264–266.
6. **Благодарный А. И., Гаркуша В. В., Цыба А. М., Шевченко Д. О., Яковлев В. В., Чейдо Г. П.** Автоматизированная система управления наземным и подземным электроснабжением угольной шахты // Вычислительные технологии. 2013. Т. 18. С. 113–116.
7. **Благодарный А. И., Гусев О. З., Журавлев С. С., Зензин А. С., Золотухин Е. П., Каратышева Л. С.** Автоматизированная система наблюдения, оповещения и поиска персонала при авариях в шахтах // Горная промышленность. 2009. № 1. С. 34–38.
8. **Благодарный А. И., Гусев О. З., Журавлев С. С., Золотухин Е. П., Каратышева Л. С., Колодей В. В., Михальцов Э. Г., Чейдо Г. П., Шакиров Р. А., Шакиров С. Р.** Автоматизированная система контроля и управления ленточными конвейерами на угольных шахтах // Горная промышленность. 2008. № 5 (81). С. 38–44.
9. **Чейдо Г. П., Благодарный А. И., Собстель Г. М., Сабуров В. С., Шакиров С. Р.** Возможности диагностики состояния системы электроснабжения горнодобывающего предприятия по ансамблям измерений // Фундаментальные и прикладные исследования, разработка и применение высоких технологий в промышленности и экономике: Сб. ст. XIII Междунар. науч.-практ. конф. СПб., 2012. С. 309–311.
10. **Благодарный А. И.** Особенности и опыт применения надежной SCADA-системы «Блакарт // Индустриальные информационные системы: Сб. тез. докл. Всерос. конф. Новосибирск, 2013. С. 10.
11. **Голушко С. К., Пищик Б. Н.** Функциональность и безопасность АСУ ТП // Фундаментальная информатика, информационные технологии системы управления: реалии и перспективы. FIITM-2014: Материалы Междунар. науч.-практ. конф. Красноярск, 2014. С. 85–91.
12. **Воронцов А.** Автоматизированные системы управления технологическими процессами // Вопросы безопасности: Информ. бюлл. компании «Инфосистемы Джет». Информационная безопасность промышленных объектов. URL: http://www.jetinfo.ru/jetinfo_arhiv/informatsionnaya-bezopasnost-promyshlennykh-obektov/2011/?nid=77f3dbdaa8dfb77077c0888a712a3e1a
13. **Грицай Г., Тиморин А., Гольцев Ю., Ильин Р., Гордейчик С., Карпин А.** Безопасность промышленных систем в цифрах v2.1*. URL: http://www.ptsecurity.ru/ download/SCADA_analytics_russian.pdf
14. **Пищик Б. Н.** Безопасность АСУ ТП // Вычислительные технологии. 2013. Т. 18. Спец. вып. Труды Всероссийской конференции «Индустриальные информационные системы-2013».
15. **Мальнев А.** Защита АСУ ТП: от теории к практике // Информационная безопасность. 2012. № 6. С. 24–26.

References

1. **Zolotukhin E., Mikhaltsov E., Starshinov A., Stratula V., Cheido G.** Modernization of automated control systems of trunk oil pipelines. *Modern automation technologies*, 1997, no. 4, pp. 18–26. (in Russ.)

2. **Blagodarny A. I, Zenzin A., Mikhaltsov E., Petkov A., Cheido G.** A programmable information and control system is a tool for creating automated control systems for trunk oil pipelines. In: IT solutions in the oil and gas industry. Moscow, Oil and Capital Publ., 2002, pp. 51–58. (in Russ.)
3. **Blagodarny A. I., Karatysheva L. S.** Universal SCADA system "Blackart" running the QNX operating system. *Problems of Computer Science*, 2009, no. 3, pp. 62–67. (in Russ.)
4. **Blagodarny A. I.** Software tools for building automated control systems in the domestic operating system environment. *Problems of Computer Science*, 2018, no. 2, pp. 41–51. (in Russ.)
5. **Blagodarny A. I., Garkusha V. V., Tsyba A. M., Cheido G. P., Shevchenko D. O., Yakovlev V. V.** Automated control of coal mine power supply. In: Energy Security of Russia. New approaches to the development of the coal industry. Proc. of the XV International Scientific and Practical Conference. Eds. V. I. Klishin, Z. R. Ismagilov, V. Yu. Blumenstein, S. I. Protasov, G. P. Dubinin. Kemerovo: Institute of Coal SB RAS, 2013, pp. 264–266. (in Russ.)
6. **Blagodarny A. I., Garkusha V. V., Tsyba A. M., Shevchenko D. O., Yakovlev V. V., Cheido G. P.** Automated control system for ground and underground power supply of a coal mine. *Computing Technologies*, 2013, vol. 18, pp. 113–116. (in Russ.)
7. **Blagodarny A. I., Gusev O. Z., Zhuravlev S. S., Zenzin A. S., Zolotukhin E. P., Karatysheva L. S.** Automated system of monitoring, notification and personnel search in case of accidents in mines. *Mining Industry*, 2009, no. 1, pp. 34–38. (in Russ.)
8. **Blagodarny A. I., Gusev O. Z., Zhuravlev S. S., Zolotukhin E. P., Karatysheva L. S., Kolodey V. V., Mikhaltsov E. G., Cheido G. P., Shakirov R. A., Shakirov S. R.** Automated control system for belt conveyors at coal mines. *Mining Industry*, 2008, no. 5 (81), pp. 38–44. (in Russ.)
9. **Cheido G. P., Blagodarny A. I., Sobstel G. M., Saburov V. S., Shakirov S. R.** Possibilities of diagnostics of the state of the power supply system of a mining enterprise by ensembles of measurements. In: Fundamental and applied research, development and application of high technologies in industry and economics. Collection of articles of the XIII International Scientific and Practical Conference. St. Petersburg, 2012, pp. 309–311. (in Russ.)
10. **Blagodarny A. I.** Features and experience of using a reliable SCADA system “Blackart”. In: Industrial Information Systems. Collection of abstracts. Novosibirsk, 2013, p. 10. (in Russ.)
11. **Golushko S. K., Pishchik B. N.** Functionality and safety process control system. In: Fundamental science, information technology management system: realities and prospects. FIITM-2014. Proc. of the International scientific-practical conference. Krasnoyarsk, SFU Press, 2014, pp. 85–91. (in Russ.)
12. **Vorontsov A.** Automated process control systems. In: Safety issues. Information bulletin of the company “Jet Infosystems”. Information security of industrial facilities. (in Russ.) URL: http://www.jetinfo.ru/jetinfo_arhiv/informatsionnaya-bezopasnost-promyshlennykh-obektov/ 2011 /?nid=77f3dbdaa8dfb77077c0888a712a3e1a
13. **Gritsai G., Timorin A., Goltsev Yu., Ilyin R., Gordeychik S., Karpin A.** Safety of industrial systems in numbers v2.1*. (in Russ.) URL: http://www.ptsecurity.ru/download/SCADA_ analytics_russian.pdf
14. **Pishchik B. N.** Safety of the automated control system. *Computing Technologies*, 2013, vol. 18. Special issue: Proc. of the All-Russian Conference “Industrial Information Systems-2013”. (in Russ.)
15. **Malnev A.** Protection of automated control systems: from theory to practice. *Information Security*, 2012, no. 6, pp. 24–26. (in Russ.)

Информация об авторах

Анатолий Иванович Благодарный, научный сотрудник
Борис Николаевич Пищик, кандидат технических наук, старший научный сотрудник, доцент
SPIN 4297-3046
Author ID 13080

Information about the Authors

Anatoly I. Blagodarny, Researcher
Boris N. Pishchik, Candidate of Technical Sciences, Senior Researcher, Associate Professor
SPIN 4297-3046
Author ID 13080

Статья поступила в редакцию 11.09.2021;
одобрена после рецензирования 01.12.2021; *принята к публикации 01.12.2021*
The article was submitted 11.09.2021;
approved after reviewing 01.12.2021; *accepted for publication 01.12.2021*

Научная статья

УДК 004.43
DOI 10.25205/1818-7900-2021-19-4-16-35

Абстрактная машина языка программирования учебного назначения СИНХРО

Лидия Васильевна Городняя

Новосибирский государственный университет
Новосибирск, Россия

Институт систем информатики им. А. П. Ершова
Сибирского отделения Российской академии наук
Новосибирск, Россия

lidvas@gmail.com, https://orcid.org/0000-0002-4639-9032

Аннотация

Статья посвящена ряду решений по организации работы с памятью в учебных языках и системах программирования, нацеленных на обучение подготовке многопоточных программ над общей памятью. Рассмотрение выполнено на материале учебного языка программирования СИНХРО, что позволило анализировать варианты таких решений без ограничений, характерных для традиционных производственных инструментов и устройства стандартных систем программирования. В статье дано определение абстрактной машины и расширение ее системы команд, позволяющее определять поведение программы как распределенной системы из ряда потоков, взаимодействующих в терминах доступа к значениям переменных, расположенных в общей памяти. Описано устройство общей памяти и механизмы доступа к ней отдельных процессов, представляющих собой последовательности выполнения команд, часть из которых являются запросами к общей памяти. В центре внимания удобство отладки небольших программ, используемых для ознакомления с проблемами параллелизма в процессе обучения, когда темп понимания проблем обучаемыми важнее достижения эффективности и производительности учебных программ. Решение проблем отладки программ полезно при изучении методов программирования, а также при исследовании истории языков программирования, сравнения парадигм программирования, потенциала используемых схем и моделей, оценки уровня новизны создаваемых языков программирования, создания методики измерения разных характеристик программ на моделях и выборе критериев практичности создаваемых программ.

При расширении системы команд абстрактной машины учтены принципы функционального программирования как популярной парадигмы на этапе подготовки прототипов и моделей многопоточных программ. Из этих принципов выполнен вывод следствий, позволяющих успешно выбрать элементарные команды, поддерживающие работу с памятью в стиле неизменяемости данных, обратимости действий и транзакций при обработке данных. Для функционального программирования, как и для учебных задач параллельного программирования, умение обеспечить правильность и полноту решений важнее эффективности и производительности полученных программ. Это путь к созданию надежного и безопасного программного обеспечения.

Ключевые слова

абстрактная машина, система команд, методы определения языков программирования, функциональное программирование, неизменяемость данных, восстановление данных, освобождение памяти

Для цитирования

Городняя Л. В. Абстрактная машина языка программирования учебного назначения СИНХРО // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 4. С. 16–35. DOI 10.25205/1818-7900-2021-19-4-16-35

© Городняя Л. В., 2021

Abstract Machine of the Programming Language for Educational Purposes SYNCHRO

Lydia V. Gorodnyaya

Novosibirsk State University
Novosibirsk, Russian Federation

A. P. Ershov Institute of Informatics Systems
of the Siberian Branch of the Russian Academy of Sciences
Novosibirsk, Russian Federation

lidvas@gmail.com, https://orcid.org/0000-0002-4639-9032

Abstract

The article is devoted to a number of solutions for organizing work with memory in educational languages and programming systems aimed at teaching the preparation of multi-threaded programs over shared memory. The consideration was carried out within the framework of the SYNCHRO programming language for educational purposes, which made it possible to analyze variants of such solutions without the limitations typical for traditional manufacturing tools and devices of standard programming systems. The article gives a clear formulation of an abstract machine and a diagram of a command system that allows you to define the behavior of a program as a distributed system from a number of threads interacting in terms of access to the values of variables located in shared memory. The device of shared memory and mechanisms of access to it by individual processes, which are a sequence of command execution, some of which are requests to shared memory, are described. The focus is on the convenience of debugging small programs used to introduce concurrency problems in the learning process, where the pace of learners' understanding of the problems is more important than achieving program efficiency and performance. The solution to this problem is useful when studying programming methods, as well as studying the history of programming languages, comparing programming paradigms, the potential of the schemes and models used, assessing the level of novelty of the programming languages being created, creating a technique for measuring different characteristics of programs on models and choosing criteria for the practicality of the created programs. When choosing a system of commands for an abstract machine, the principles of functional programming were taken into account as a popular paradigm at the stage of preparing prototypes and models of multi-threaded programs. From these principles, the conclusion is drawn of the consequences that allow you to successfully select elementary instructions that support working with memory in the style of data immutability and their transactional processing. For educational tasks in programming, the ability to ensure the correctness and completeness of solutions is more important than the efficiency and productivity of the received programs. This is the path to building reliable and secure software.

Keywords

abstract machine, instruction set, methods for defining programming languages, functional programming, data immutability, freeing memory

For citation

Gorodnyaya L. V. Abstract Machine of the Programming Language for Educational Purposes SYNCHRO. *Vestnik NSU. Series: Information Technologies*, 2021, vol. 19, no. 4, p. 16–35. (in Russ.) DOI 10.25205/1818-7900-2021-19-4-16-35

Введение

Со времен появления Венской методики определения языков программирования (ЯП) [1; 2] операционная семантика языка программирования обычно базируется на определении абстрактной машины (АМ), допускающей не слишком трудоемкую реализацию на широком классе конкретных машин. Практика создания ЯП накопила уже заметное количество АМ [3–5]. В случае многопроцессорных конфигураций АМ, естественно, становится конструкцией из абстрактных процессоров – абстрактным многопроцессорным комплексом (АПК). С середины 1970-х гг. популярным стал формализм SECD-машины, предложенный П. Лэндиным при создании аппаратной реализации языка Lisp [6], подробно описанный в книге П. Хэндерсона [4]. В конце 1970-х гг. в рамках проекта языка БАРС В. Е. Котов предложил структурировать определение ЯП на четыре основных подъязыка, независимо представляющих изобразительные средства для выражений, структур данных, управления процессами и дисциплины памяти [5]. Именно подъязык дисциплины памяти вызвал заметные трудно-

сти, возможно, связанные с тем, что в те времена было мало опыта работы с многопроцессорными конфигурациями и распределенными системами [7; 8]. Практика реализации систем программирования и компиляторов сосредоточилась на задачах оптимизации распределения памяти, а вопросы разнообразия дисциплины работы с памятью, с ее неоднородностью и внешними устройствами, слабо учитываемые в ЯП, остались без особого развития [9]. С середины 1990-х гг. в сфере параллельных вычислений обрели популярность системы функционального программирования, принципы которого, такие как гибкость ограничений и неизменяемость данных, показали удобство для подготовки многопоточных программ, устроенных из независимых протоков [10]. Проблемы многопоточных программ с взаимодействующими потоками ждут своего решения.

Проблема понимания многопоточности

На уровне первичного элементарного программирования мало заметна принципиальная разница между хранением данных в локальной, общей оперативной или внешней памяти. Особенности работы с общей и внешней памятью много детальнее изучены в практике организации реляционных баз данных и применения языка запросов к базам данных типа SQL, использующих транзакции [11].

Нормальное завершение многопоточной программы происходит при исчерпании комплекта пассивных потоков, готовых к выполнению, и плановом завершении всех активных потоков. Кроме того, общее завершение работы может происходить преждевременно, при невозможности завершить работу каких-либо потоков из-за взаимоблокировки или исчерпания ресурсов, таких как память. В последнем случае обычно подключается дополнительная память, или привлекаются механизмы ее освобождения. Часть таких проблем алгоритмически не разрешима, поэтому в задачи обучения входит предвидеть опасности и научаться отлаживать программы при их обнаружении. Для простоты учебной модели здесь не рассматривается учет разнообразия категорий систем команд отдельных процессоров и видов используемой памяти с различной дисциплиной функционирования. Механизмы освобождения памяти обычно требуют приостановки основных вычислений, что снижает общую производительность программы. Это особенно заметно при необходимости освобождения общей памяти, требующей приостановки всего многопроцессорного комплекса.

Наполнение многопоточной программы может развиваться независимо от схем управления вычислениями в отдельных потоках, а схемы можно реорганизовывать без дополнительной отладки наполнения. Они играют роль макетов или моделей программ и работают, подобно макросам (открытая подстановка), но с контролем соответствия параметров объявленным синтаксическим типам фрагментов. Кроме того, техника отладки программ может быть обогащена возможностью привлечения протоколов ранее выполненных вычислений и приведения программ к нормальным формам, удобным для сведения к базовым / стандартным моделям параллельных вычислений. В таких случаях более точное описание абстрактного комплекса требует некоторого пересмотра системы команд абстрактной машины для отдельных процессоров – переход к модели абстрактного комплекса.

Абстрактный многопроцессорный комплекс предназначен для определения механизмов выполнения многопоточных программ на многопроцессорных комплексах. АПК обладает конкретной **с**истемой команд, расширенной средствами работы с общей памятью и внешними устройствами. При функционировании комплекса число процессоров может изменяться, что влечет появление динамических запросов к памяти, не заметных при статическом анализе и компиляции программы. Кроме собственно процессоров к выполнению программы можно привлекать дополнительные устройства, которые рассматриваются как специальные процессоры без заранее определенной системы команд, способные выполнять некоторые действия по командам процессора. Системе команд следует поддерживать обработку данных, их размещение и реорганизацию в общей памяти программы или в локальной памяти

процесса, управление ходом выполнения процессов, включая взаимодействие процессоров и устройств, и резервирование данных для их защиты от случайных изменений, подобно средствам операционных систем [7].

Каждый активный процессор выполняет один процесс, порождаемый одной последовательностью команд. Предполагается, что программа – результат компиляции многопоточной программы с предельным распараллеливанием. Программа выполняется в определенном контексте – общей обстановке, данные из которой доступны отдельным процессорам, выполняющим программу. Обстановка, кроме локальной памяти, содержит описания глобальных, возможно изменяемых, общих данных, доступных во всей программе. Возможно, что по умолчанию, имеется один процессор, на котором происходит выполнение основной программы. Такой процессор может не отличаться от остальных по устройству и работать без приоритета. Каждый процессор работает по шагам, соответствующим выполнению одной команды процессора, после каждого шага происходит переход к общему правилу управления многопроцессорной программой. Шаги разных процессоров не синхронизованы, но могут происходить одновременно. Общее правило управления может включать параметр порядка перебора команд процессоров, допускающий и учет приоритетов, и правил тасования, и временной шкалы, и случайного выбора очередного процессора. Если используется временная шкала, то предполагается, что существует дополнительный процессор времени, непрерывно вырабатывающий очередную метку времени, которую могут учитывать остальные процессоры как данное из общей памяти. Для работы общей памяти может существовать отдельный процессор общей памяти.

При сравнении императивного и функционального подходов к программированию П. Лендин (P. J. Landin) предложил специальную абстрактную машину SECD, удобную для спецификации машинно-зависимых аспектов семантики языка Lisp. Подробное описание этой машины можно найти в книге П. Хендерсона по функциональному программированию, где предложен и вариант расширения АМ для поддержки параллельных вычислений. Рассматриваемые здесь процессоры устроены, примерно как абстрактная машина SECD П. Лендина, с добавлением набора команд работы с памятью и внешними устройствами, как у Н. Вирта в Пи-коде [4; 12], команд организации взаимодействия процессоров уровня UNIX [7] и транзакциями, характерными для баз данных [11], восстановления данных, как в системах редактирования [13], с учетом JVM [14] и машины MMIX Д. Кнута [1]. Дальнейшие идеи определения системы команд для многопроцессорных конфигураций сложились при анализе парадигм программирования, включая особенности функционального программирования [9; 15; 16], а также работ по дисциплине программирования [17] и языкам Forth, mpC и СИНХРО [18–20].

Расширение абстрактной машины

Машина SECD работает над четырьмя регистрами: стек для промежуточных результатов, контекст для размещения именованных значений, управляющая вычислениями программа, резервная память (Stack, Environment, Control_list, Dump). Регистры приспособлены для хранения выражений в форме символов или списков. Состояние машины полностью определяется содержимым этих регистров. Поэтому функционирование машины можно описать достаточно точно как переходы изменения содержимого регистров при выполнении команд, что выражается следующим образом:

s e c d → s' e' c' d' – переход от старого состояния к новому.

Вводим расширенно обозначение на случай работы с общей памятью.

M(SECD)+ – многопроцессорный комплекс над общей памятью.

Обозначение **M(SECD)+** символизирует, что **M** – общая память для всех процессоров, **(Процессор)+** – что хотя бы одни процессор обязателен, общее число процессоров произвольно, и не исключено изменение их числа в динамике.

M – общая память, соответствует принципу неизменяемости данных, состоит из регистров произвольного доступа, хранящих счетчик числа потоков, использующих эти регистры, имя переменной, ее текущее значение и протокол произошедших изменений, возможно, устроенный как вектор или список.

Система команд в таком случае представляет собой реализацию базовой семантики языка многопоточных программ, дополненную рядом системных действий по передаче параметров и защите областей действия, подразумеваемых языком, но не имеющих четкого синтаксического представления. Такое определение может быть машинно-независимым и переносимым. Абстрактный многопроцессорный комплекс языка СИНХРО как примера языка многопоточных программ различает следующие категории команд:

1) засылка значений из общей памяти и разных локальных регистров АПК в стек S;

2) вычисления над безымянными операндами из стека S при обработке выражений;

3) пересылка значений из стека S в общую память M или регистры АПК с учетом локализации участков процесса;

4) организация ветвлений, точнее – обходов, что удобно для кодирования программ с параллелизмом и эффективно для архитектур с разрядами управления выполнимостью команд методом прошивания 0/1 в коде команды; (If... Then... без Else);

5) организация циклов, вызовов процедур или функций, определения которых не сложнее одного комплекта потоков, с сохранением контекста в локальной обстановке E и / или в дампе D с возможностью восстановления;

6) организация комплекта процессов с передачей их результатов в общую память M и возможностью обмена сообщениями или ожидания событий при синхронизации.

Суммарно комплект команд включает в себя следующие действия (выделены добавленные к SECD команды):

LD – ввод данного из локального контекста в стек;

LDC – ввод константы из программы в стек;

LDF – ввод определения функции в стек;

LDM – ввод данного из общей памяти в стек;

COMPSUB – компиляция «на лету» фрагмента программы, заданного в виде текста;

AP – применение функции, определение которой уже в стеке;

RTN – возврат из определения функции к вызвавшей ее программе;

RAP – применение **рекурсивной** функции;

DUM – резервирование памяти для хранения **поколений** аргументов рекурсивной функции или цикла;

SEL – **обход** участка или ветвление в зависимости от активного (верхнего) значения стека;

JOIN – переход к общей точке после обхода;

CRCL – цикл – повтор;

BR – принудительное завершение выполнения цикла, комплекта, потока или функции;

PAIR – CONS – формирование **узла** по двум верхним значениям стека для списков;

HEAD – CAR – первый элемент узла из активного значения стека;

TAIL – CDR – остаток узла без первого элемента активного значения стека;

ARR – формирование **вектора** из заданного числа элементов, расположенных в стеке;

ELM – выбор элемента из вектора по заданному индексу;

ATOM – **неделимость** (атомарность) верхнего элемента стека;

NUMBER – проверка на число;

LIST – проверка на список;

ARRAY – проверка на вектор;

TEXT – проверка на текст или строку;

EQ – **равенство** двух верхних значений стека с учетом произвольного значения;
STOP – **останов;**
SET – **запись** в общую память комплекса из стека;
LET – сохранение локальных значений в **контексте;**
MLL – пересылка головного элемента из списка в список головного элемента;
MLV – пересылка из списка головного элемента в указанный элемент вектора;
MVL – пересылка из указанного элемента вектора в головной элемент списка;
MVV – пересылка из указанного элемента вектора в указанный элемент другого вектора;
CHNG – обмен данными в общей памяти комплекса;
DELETE – удаление верхнего элемента стека;
WRT – вывод данных из стека в процесс внешнего периферийного устройства с сигналом успеха-провала;
RD – ввод данных от процесса внешнего периферийного устройства в стек с сигналом успеха-провала;
ANY – размещение в стеке выравнивающего данного, если данное с устройства не введено;
KIT – **комплекс** из процессоров или процессов выполнения действий-директив, мощность комплекса известна;
ROW – **процесс для** выполнения ряда действий-директив на одном процессоре, длина ряда известна в каждый момент;
REZ – размещение заданного числа результатов процесса в свой стек;
WAIT – **ожидание** приостановки указанного процесса (завершения или сообщения);
SEND – передача **сообщения** указанному процессу;
NEXT – ожидание завершения предыдущего действия;
INC – **увеличение** числа на 1;
DEC – **уменьшение** числа на 1;
SUB – **вычитание** из верхнего элемента стека;
ADD – **прибавление** к верхнему элементу стека;
MUL – **произведение** двух чисел, первое в стеке;
DIV – **частное** от деления верхнего элемента стека на второй элемент;
LT – выяснение**,** верхнее значение в стеке **меньше** ли, чем второе;
ERR – реакция на ошибку, аппаратную или программируемую, если в коде программы она встроена.

Для начала абстрактная машина считается идеальной, программу для нее компилятор строит без учета возможных аварийных ситуаций, считает, что все данные на устройстве ввода расположены в соответствии с запросами программы, именно те, что нужны. Состояния стеков к моменту выполнения команд сформированы вполне корректно. Регистры общей памяти подчинены принципу неизменяемости данных. Новые данные размещаются на верху стека или в первом элементе списка, а к прежним значениям, хранимым в общей памяти, можно вернуться при необходимости, например при отладке посмотреть историю изменения данных. При выполнении обратимых действий допускается и обратимость воздействий на общую память. Могут быть и другие категории команд, реализуемые как не консервативное расширение абстрактного комплекса ради эффективности или решения других задач.

Принцип неизменяемости данных

Для языка СИНХРО предлагается усилить абстрактную машину добавлением регистра, используемого при определении механизма работы с общей памятью для семейства процессоров в рамках принципа неизменяемости данных, характерного для ФП. Таким образом можно от машины SECD перейти к абстрактному комплексу вида M[SECD]+, где явно отмечена многопроцессорность и существование общей памяти. Такая работа с памятью должна

быть поддержана в транзакционном стиле, подобно обработке записей в базах данных, т. е. каждое воздействие на память или выполняется полностью, или восстанавливается состояние до начала выполнения не завершившейся команды. Кроме того, каждый элемент памяти в любой момент времени обрабатывается только в одном процессе программы, т. е. подобно захвату-освобождению файла или устройства. Хранение данных в общей памяти сопровождается протоколом изменений, чтобы при отладке можно было видеть, какой процесс внес изменения, и при необходимости вернуть утраченное значение.

Таким образом, абстрактный комплекс определяется над функционирующими как стеки или списки неограниченного размера пятью регистрами: M – общая память, S – стек для окончательных и промежуточных результатов, E – контекст для размещения локальных именованных значений, C – управляющая вычислениями программа, D – резервная память (Memory, Stack, Environment, Control_list, Dump). В определенном смысле состояние общей памяти M можно рассматривать как непременное дополнение к формальному результату работы программы. Ниже приведено более детальное определение работы команд как переходов от одного состояния машины к другому:

m s e c d → m' s' e' c' d' – переход от старого состояния к новому,

где

m – Memory = общая память, выглядит как внешнее устройство, доступное всем процессам;

s – Stack = стек для окончательных и промежуточных результатов;

e – Environment = контекст для размещения локальных значений переменных;

c – Control_list = управляющая вычислениями программа;

d – Dump = резервная память для обеспечения надежности вычислений.

Стек устроен традиционно по схеме «первый пришел, последний ушел». Размер стека не ограничен. Каждая команда абстрактной машины «знает» число используемых при ее работе элементов стека S и их форматы, элементы она удаляет из стека S и вместо них размещает единственный выработанный результат для обычных команд или ряд результатов для многопроцессорных команд, размещая число процессоров в верхнем элементе стека. Всегда известно число текущих результатов в стеке S, которые можно явно свернуть в единственный результат специальной, редуцирующей командой. Такие команды выясняет компилятор и размещает в регистре C для абстрактной машины.

Фактически исполняются команды по очереди, последовательно, начиная с первых в регистре C управляющей программы, хотя формально может считаться, что они могут исполняться и в произвольном порядке. Отдельная машина прекращает работу при выполнении команды «останов» – STOP, которая формально характеризуется отсутствием изменений в состоянии машины:

m s e (**STOP**) d → m s e (STOP) d

Следуя П. Хендерсону, для четкого отделения обрабатываемых элементов АПМ от остальной части списка будем использовать следующие обозначения:

(x . L) – это значит, что первый, головной элемент списка – x, а остальные, хвостовые, находятся в списке L;

(x y . L) – первый элемент списка – x, второй элемент списка – y, остальные находятся в списке L и т. д.

Теперь мы можем методично описать эффекты всех перечисленных выше команд.

m s e (**LDC** q . c)d → m (q . s) e c d	% константы из программы в стек
m (a b . s) e (**PAIR** . c)d → m ((a . b) . s) e c d	% узел структуры из стека
m ((a . b) . s) e (**HEAD** . c) → m (a . s) e c d	% головной элемент списка из стека

m ((a . b) . s) e (**TAIL** . c) → m (b . s) e c d % остальные элементы списка без головного

ARR – формирование вектора из заданного числа элементов, предварительно размещенных в стеке.

m (a b . s) e (**ARR** 2 . c)d → m (@[a, b] . s) e c d % вектор, содержащий 2 элемента

@ – **адрес** данного

ELM – выбор элемента из вектора по заданному индексу.

m (i @X . s) e (**ELM** . c) → m (X[i] . s) e c d % элемент вектора

m (2 @[a, b] . s) e (ELM . c) → m (b . s) e c d % 2-й элемент вектора

Для предиката оговариваем, в каких случаях вырабатывается значение «истина»:

m ((a . b) . s) e (**ATOM** . c) d → m (NIL . s) e c d % ложь

m (a . s) e (ATOM . c) d → m (a . s) e c d % истина

m (Nil . s) e (ATOM . c) d → m (T . s) e c d % истина

Для составных, неатомарных значений – NIL, т. е. ложь, истина «T» – для атомов:

m (a a . s) e (**EQ** . c) d → m (T . s) e c d % два верхних элемента совпали

m (a b . s) e (**EQ** . c) d → m (NIL . s) e c d % два верхних элемента различны

Истина «T» – для совпадающих указателей, для несовпадающих – NIL, т. е. ложь.

Для доступа к значениям, расположенным в **контексте**, можно определить специальную функцию N-th, выделяющую из списка элемент с заданным номером N в предположении, что длина списка превосходит заданный адрес:

m s e (**LD** n . c) d → m (x . s) e c d

где x – это значение (N-th n e), т. е. значение по указанному номеру в контексте.

При реализации обхода управляющая программа соответствует шаблону:

(... SEL (... JOIN))

Необязательный участок размещен в подсписке с завершителем JOIN, после следует общая часть вычислений. Для большей надежности на время выполнения обхода общая часть сохраняется в дампе – **резервной памяти**, а по завершении **восстанавливается** в регистре управляющей программы:

m (NIL . s) e (**SEL** cc . c) d → m s e c d % обход участка

m (T . s) e (**SEL** cc. c) d → m s e cc (c . d) % выполняется участок

m s e (**JOIN**) (c . d) → m s e c d % восстановление при выходе из участка

Получается нечто вроде оператора If ... Then .. без Else, что позволяет делать исполнимый код без ветвлений и тем самым уменьшать потребность в динамическом порождении новых процессов.

Организация вызова функций или процедур требует защиты контекста E от локальных изменений, происходящих при интерпретации тела определения. Для этого при вводе определения функции в стек S создается специальная структура, содержащая код определения функции и **копию текущего состояния контекста E.** До этого в стеке должны быть размещены фактические параметры. Завершается функция командой RTN, восстанавливающей регистры, заранее сохраненные в регистре D, и размещающей в стеке S полученный при выполнении тела функции результат:

m s e (**LDF** f . c) d → m ((f . e) . s) e c d % замыкание функции в стек
m ((f . ef) vf . s) e (**AP** . c) d → m NIL (vf . ef) f (s e c . d) % вызов функции,
% ранее размещенной в стеке, и пополнение контекста значениями связанных переменных
m (x) e (**RTN**) (s e c . d) → m (x . s) e c d % выход их функции

где f – тело определения,
ef – контекст в момент вызова функции,
vf – фактические параметры для вызова функции,
x – результат функции.

Рекурсивные вызовы функций дополнительно требуют резервирования памяти для уникальных ссылок на разные поколения фактических аргументов, что достигается путем размещения фиктивного объекта «**ω**», который структуро-разрушающая, деструктивная функция rplaca, в порядке исключения, замещает на реальные данные:

m s e (**DUM** . c) d → m s (**ω** . e) c d

% **ω** – элемент памяти для размещения очередного поколения фактических параметров,
m ((f . ef) vf . s) e (**RAP** . c) d → m NIL (rplaca vf ef) f (s e c . d)
% размещение очередного поколения параметров рекурсивной функции

Получается нечто вроде многоголового списка. На каждом витке рекурсии порождается узел, становящийся головой хвоста списка локальных параметров E, а прежний список запоминается в дампе D. При выходе из рекурсии восстанавливаются прежние состояния локального контекста E.

Для M(SECD)+, как и для SECD, реализационное замыкание ядра включает в себя структуроразрушающие, деструктивные функции **rplaca** и **rplacd**, размещающие свой результат непосредственно в памяти второго аргумента (табл. 1). Это требует соответствующих ветвей в определениях синтаксиса и интерпретатора, что можно рассматривать как шаг раскрутки СП.

Таблица 1

Расширение системы команд леструктивными операциями

Table 1

Expansion of the command system by destructive operations

	Новая конструкция	Значение	Примечание
Семантика	(RPLACA x (y . z))	(x . z)	**Новое** значение размещается в **старой** памяти, замещая прежнее значение
	(RPLACD x (y . z))	(y . x)	

Система команд АПМ может быть расширена простым дополнением правил для новых команд. Так, можно ее механизм распространить на другие типы данных, например на целые числа (табл. 2).

Контроль типов данных на уровне системы команд отсутствует. Считается, что компилятор создал программу после проверки соответствия типов данных, и при исполнении кода она уже не нужна.

Таблица 2

Расширение абстракной машины на работу с числами

Table 2

Extension of an abstract machine to work with numbers

m (a b . s) e (**ADD** . c) d	→ m ((a + b) . s) e c d
m (a b . s) e (**SUB** . c) d	→ m ((a – b) . s) e c d
m (a b . s) e (**MUL** . c) d	→ m ((a * b) . s) e c d
m (a b . s) e (**DIV** . c) d	→ m ((a / b) . s) e c d
m (a . s) e (**INC** . c) d	→ m ((a + 1) . s) e c d
m (a . s) e (**DEC** . c) d	→ m ((a – 1) . s) e c d
m (a . s) e (**NUMBER** . c) d	→ m (t . s) e c d % число или нет
m (a b . s) e (**LT** . c) d	→ m (t . s) e c d % верхнее **меньше**, чем второе

Примечание: t – **истинностное** значение NIL или T.

Общая память

Для производственного применения ЯП требуется расширение семантики ЯП средствами обмена данными с общей памятью и периферийными устройствами, позволяющими вводить программы и видеть результаты ее выполнения. В SECD и то, и другое происходит чудом. Машина SECD уже достаточна для обеспечения полноты вычислений по Тьюрингу, а для соответствия архитектуре Фон Неймана нужна определенность средств работы с общей памятью и ввода-вывода данных, что дает машина M(SECD)+.

Подобные средства Н. Вирт включил в описание Пи-кода для языков Pascal и Oberon, команды которой можно рассматривать как дополнение и расширение SECD. Разница между SECD и M(SECD)+ в основном сводится к операциям над данными. Кроме того, введены дополнительные команды по непосредственной работе с общей памятью для реализации присваиваний, передаче управления для реализации итераторов и обмена с внешними устройствами. Обе абстрактные машины, как SECD, так и M(SECD)+, содержат кроме образа базовых средств ЯП дополнительные команды, обеспечивающие пересылки данных между регистрами, реализационные средства, поддерживающие эффективность реализации ЯП (rplaca, метки и др.) и профилактику некорректного нарушения непрерывности действий при обмене данными.

Используем дополнительные обозначения:

[x] – содержимое памяти по адресу x;

e[n] – содержимое n-го элемента контекста;

C(Pr) – номер процесса

@c – адрес позиции «c» в программе или векторе;

_ – Произвольное значение (_ подчерк).

Регистр M является списком списков, или стеков, каждый элемент которого начинается с имени глобальной переменной, вслед за которым расположено ее текущее значение, а дальше следует протокол изменения значений, выглядящий как последовательность идентификаторов процесса с последующим установленным этим процессом значением.

Если M содержит элемент X вида M = [. . . (**X xt** At xt A1 x1 A2 x2 . . .)], то после изменения процессом Ak значения переменной X на xk этот элемент приобретет вид

M' = [. . . (**X xk Ak xk** At xt A1 x1 A2 x2 . . .)].

Возникнет побочный эффект присваивания, допускающий при необходимости восстановление прежних значений. Такая реализация позволяет поддержать транзакции и обратимость обработки памяти.

Ввод данного X из общей памяти в стек. В памяти имеется элемент вида (**X xt .** Px), где

X – имя **глобальной** переменной,

xt – текущее значение переменной,

Px – протокол изменения значений переменной.

m=[. . . (**X xt .** Px)] s e (**LDM** X . c) d → m (xt . s) e c d

Сохранение значения из стека в глобальной памяти;

m=[. . . (**X x0 .** Px) . . .] (xt . s) e (**SET** X . c) d → m'==[. . . (**X xt Ct xt .** Px)] s e c d

m=[. . .] (xt . s) e (**SET** X . c) d → m'==[. . . (**X xt Ct xt**)] s e c d % новая переменная,

где Ct – номер текущего процесса, задавшего изменение значения глобальной переменной. Имена данных в общей памяти все различны.

LET – сохранение локальных значений после однократных присваиваний в локальном контексте;

m (ssa . s) e (**LET** x . c) d → m s (e[x]=ssa) c m d

В стеке список результатов однократных присваиваний. К верхнему элементу контекста прицепляется этот список в хвост, вслед за аргументами. Предполагается, что имена формальных параметров и локальных данных различны.

m (ssa . s) (et . e) (**LET** . c) d → m (s) ((**APPEND** et ssa) . e) c d

APPEND – сцепление структур

ssa – участок однократных присваиваний локальным переменным, имена которых уникальны, отличаются от имен формальных параметров. Для глобальных переменных такую работу выполняет SET.

Пересылки и обмен данными нужны для профилактики возникновения временных интервалов между парами взаимосвязанных присваиваний в общей памяти. Без такой профилактики могут возникать так называемая «фантомная» память или доступ к формально уже удаленным данным, что иногда обнаруживается при использовании JVM, стандарт на которую был утвержден более 20-ти лет назад.

MLL – пересылка из списка в список головного элемента

MLV– пересылка из списка головного элемента al в указанный элемент i вектора v

MVL– пересылка из указанного элемента i вектора v в головной элемент списка al

MVV – пересылка из указанного элемента вектора в указанный элемент другого вектора

m ((al . s) e (**MLL** bl . c) d) → m ((PAIR (HEAD bl) (TAIL al)) (PAIR (HEAD al) (TAIL bl)) . s) e c d)

В стеке 2 результата: пара из головы второго списка с хвостом первого и пара из головы первого списка с хвостом второго.

m ((i v . s) e (**MLV** al . c) d) → m (@(v [i] = (HEAD al)) (TAIL al) . s) e c d)

В стеке 2 результата: адрес вектора, в котором i-й элемент заменен на головной элемент списка al, и хвост списка al.

m ((i v . s) e (**MVL** al . c) d) → m ((PAIR v [i] al) @v . s) e c d)

В стеке 2 результата: удлинненный список и адрес вектора. *Здесь вектор не изменился, но возможна реализация, в которой элемент будет объявлен неопределенным.*

m ((i1 v1 . s) e (**MVV** i2 v2 . c) d) → m (@(v1 [i1] = v2 [i2]) @(v2 [i2] = v1 [i1]) . s) e c d)

В стеке 2 результата: адреса векторов, элементы которых изменены.

m ((a . s) e (**DEL** . c) d) → m (s e c d)
CHNG – обмен данными в глобальной памяти

m=[. . . (**X xt** . Px) . . . (Y **yt** . Py) . . .] (Y . s) e (**CHNG** X . c) d)
→ m'=[. . . (**X yt Ct xt** . Px) . . . (Y **xt Ct yt** . Py) . . .] s e c d)
В стеке имя одной переменной, в программе – другой.

Внешние устройства

В результате АПМ способна выполнять вычисления несколько более широкого класса, чем задано базовой семантикой данного ЯП. Это приводит к идее определения реализационного замыкания ЯП в соответствии с фактическими возможностями АМ, включая работу с внешними устройствами. Такое замыкание выполняет роль ядра СП.

m ((**x0 .** s) e (WRT **OUT** . c) d) → m ((+ **x0 .** s) e c d) – OK
m ((**x0 .** s) e (WRT **OUT** . c) d) → m ((- **x0 .** s) e c d) – ESC

OUT' = (x0 . OUT) – на устройстве OUT появляется выведенное данное. В стеке S оно остается, чтобы вывод можно было вставлять в любой позиции выражений программы.

Вывод значения из стека S в процесс над внешним устройством OUT.

Реально при работе с устройствами через конечное время вырабатывается сигнал, говорящий или об успешном выполнении действия, или о причине отказа в его завершении. Здесь для простоты команды определены как абстрактное понимание идеального выполнения действий, без аварийных ситуаций. Выработка сигнала об отказе внешнего устройства приводит к запоминанию в протоколе отказов информации об ущербно выработанных результатах, что можно учесть при отладке, и выполнению остаточной программы в стиле смешанных вычислений.

m (s e (RD **IN** . c) d) → m ((+ **xt (IN) .** s) e c d) – OK
m (s e (RD **IN** . c) d) → m ((- ESC **(IN) .** s) e c d) – ESC

Ввод данного из процесса над устройством IN в стек S.

В зависимости от специфики устройства введенное данное может исчезнуть из него или сохраниться.

(xt . IN) → IN
(xt . IN) → (xt . IN)

Считается, что на внешнем устройстве расположено именно то данное, которое требуется по программе из стека управления «C».

m ((ESC . s) e (**ANY** . c) d) → m ((**xany .** s) e c d)
xany – выравнивающее данное, если был **ESC** – безуспешный ввод.

Параллелизм

Для поддержки параллельных вычислений требуется еще несколько команд по организации комплексов (неупорядоченных комплексов процессов **KIT**) и процессов (последовательности действий – **ROW**), передаче их результатов, локализации воздействий на транзакционную общую память и явную обработку ошибок.

AMn – обозначение отдельного процессора АМ с номером n. Каждый АМ знает свой номер.

AM0 = m (2 . s) e (**KIT** c1 c2 . c) d → AM0 = m s e c d
| AM1 = @m s e c1 d | AM2 = @m s e c2 d

Число процессов в комплексе заранее известно, по их числу порождаются независимые автоматы для асинхронного выполнения процессов: AM1, AM2

Общая память для порожденных АМ доступна через ссылку=**адрес @m .**

AM0 = m (2 . s) e (**ROW** c1 c2 . c) d → AM0 = m (1 . s) e (**ROW** c2 . c) d
| AM1 = @m s e c1 . c) d

Число элементов процесса заранее известно, по их числу последовательно порождаются независимые автоматы для последовательного запуска элементов процессов: AM1, AM2

AM0 = m (**1** . s) e (**ROW** c2 . c) d → AM0 = m s e c d
| AM2 = @m s e (c2 . c) d

Остался 1 элемент – выход из рекурсии порождения процессов.

NEXT – поддержка строго императивного запуска элементов процесса. Без этого второй элемент процесса работает, не дожидаясь завершения предшественника.
NEXT = WAIT (CNP-1) – ожидание завершения заданного процесса.

REZ – РЕЗУЛЬТАТ – запись в стек числа результатов комплекса или процесса

AM0 = m s e c d % c2 уже запущен, только **ждет** c1
| AM1 = @m (a b s) e (**REZ** 2 ass . c) d % получены результаты c1
| AM2 = @m s e (**WAIT** (AMP-1) c2 . c) d % c2 дождался завершения c1
→ AM0 = m (2 b a . s) e (ass . c) d
%AM0 забирает результаты c1 и может выполнить их свертку
| AM2 = @m s e c2 d % c2 становитя исполнимым фрагментом,

где
AMn – номер текущего процесса,
ass – представление функции, выполняющей свертку указанного числа элементов в стеке в одно данное. Например: ***список, структура, сумма, произведение, макс, мин, последний*** и т. п.

Забрали в основной стек результаты выполненного процесса, и можно запустить следующий:

AM0 = m s e (**REZ** 2 . c) d | AM1 = @m (a b s) e (**REZ** 2 . c) d
→ AM0 = m (2 b a . s) e c d

% забрали в основной стек результаты выполненного процесса AM1, и его можно отключить.
% передача результата синхронизована с готовностью их приема.

SEND – передача **сообщения** указанному процессу

AMk = @m (message . s) e (**SEND** AMn ck . c) d % подготовлено сообщение для AMn
| AMn = @m s e (**WAIT** AMk cn . c) d % AMn дожидается сообщения от AMk
→ AMk = m s e (ck . c) d %AMk отправил сообщение для AMn
| AMn = @m (message . s) e cn d % c2 сообщение получено

Реализация подобна рандеву в языке ADA – обмен происходит между активными процессами, и каждый знает, что и кому он передает или от кого сообщение получает.

WAIT – ожидание события, сообщения или завершения заданного процесса.

Реализация таких команд существенно зависит от распределения памяти и независимости ее частей, что приводит к необходимости контроля границ памяти при индексировании векторов и доступе по указателю. Информация для такого контроля во многих ЯП представляется в форме предписания типа данных (ТД) переменным. Таким образом, реализационное замыкание ЯП над M[SECD]+ включает в себя представление ТД для всех переменных, в том числе параметры процедур и их расширение для удобства лаконичных представлений, синтаксического конструирования.

Считаем, что при компиляции создана программа для АМК, пригодная для безаварийного выполнения при условии бесперебойного функционирования всех устройств. Все внешние данные соответствуют требованиям программы, и команды устройств срабатывают неделимо.

Дальнейшее расширение ЯП может быть сведено к подключению ТД и присоединению вычислительных средств методом раскрутки или пошаговой разработки. Цели раскрутки:

- снижение трудоемкости начального этапа программирования;
- увеличение потенциала СП, т. е. числа типовых задач, решение которых обладает приемлемой для практики трудоемкостью.

Такая идеальная конфигурация работает как бы на бесконечной общей памяти, не решает проблемы ее исчерпания и необходимости освобождения от ставших ненужными данных. Проблема освобождения памяти на отдельных локальных потоках имеет стандартные, не слишком обременительные решения. Для общей памяти ее решение резко снижает производительность из-за приостановки всех процессов. Не исключено, что освобождение общей памяти потребует решений, подобных опробованным в практике операционных систем при решении вопросов управления распределенными системами с совмещением работы независимых программ. Каждая отдельная программа для совмещения их работы сопровождалась паспортом, в котором представлялась информация о требуемом объеме памяти, ожидаемом времени счета и внешних устройствах и других атрибутах.

Паспорт переменной

Рассмотрим для примера определение абстрактного комплекса, допускающее управление доступом к общей памяти с использованием подобных паспортов, хранящих вектор необходимых регистров общей памяти. Каждый локальный процесс при «сборке мусора» формирует новое состояние такого вектора, доступного процессору внешней памяти. Конфигурация комплекса расширяется выделением отдельного процессора для работы с общей памятью. Остальные процессоры вместо полного доступа к общей памяти обладают лишь вектором V для доступа к определенным регистрам.

v s e c d → v' s' e' c' d' – переход от старого состояния к новому,

где

v – вектор доступа к отдельным регистрам общей памяти, доступный процессам;
s – Stack = стек для окончательных и промежуточных результатов;
e – Environment = контекст для размещения локальных значений переменных;
c – Control_list = управляющая вычислениями программа;
d – Dump = резервная память для обеспечения надежности вычислений.

Если команды процессора общей памяти можно особо не пересматривать, локальные процессоры несколько изменяют формат взаимодействия с общей памятью. Она уже не принадлежит им равноправно, а доступна через вектор указателей на регистры общей памяти. Это можно видеть на примере команды MLL – пересылка из списка в список головного элемента, доступные через указатели @al и @b.

V ((@al . s) e (**MLL** @bl . c) d) → V’ ((PAIR (HEAD @bl) (TAIL @al)) (PAIR (HEAD @al) (TAIL@ bl)) . s) e c d)

В стеке разместится 2 результата: пара из головы второго списка с хвостом первого и пара из головы первого списка с хвостом второго.

При освобождении общей памяти появляется возможность частичного освобождения памяти, не задействованной в объединении векторов от локальных процессоров, без приостановки всех процессоров. В частности, при отказе отдельного процессора процессор общей памяти может по шкале глобальных переменных выполнить уменьшение счетчиков доступа к этим переменным, влияющих на решения по освобождению или восстановлению памяти.

Восстановление состояний общей памяти

Несколько проще выглядит модель комплекса со специальным процессором общей памяти, работающим как пассивный процесс. Такой АПК состоит из ряда обычных процессоров и одного процессора общей памяти, с которым могут взаимодействовать обычные процессоры. Между собой процессоры взаимодействуют только через общую память. Каждый процессор выполняет одну последовательность команд, среди которых встречаются команды запросов к процессору общей памяти. Это **активные команды**, выполняемые по ходу процесса без особенностей. Команды запроса к общей памяти **имеют двойников** среди команд процессора общей памяти. Это **пассивные команды**, типа ленивых вычислений, возбуждаемые при выполнении запросов из активных процессов. Пассивные команды процессора общей памяти структурированы в ряд очередей, каждая из которых соответствует обычному процессору и содержит двойников запросов в том же порядке, что и в активном процессе. Пары «запрос и его двойник» выполняются неразрывно в стиле рандеву языка Ada. Механизм рандеву обеспечивает исключение случайного вмешательства сторонних процессов. При этом происходит обмен данными между локальной памятью активного процесса и общей памятью (табл. 3). В случае одновременного поступления запросов от разных процессоров они упорядочиваются случайным образом и выполняются по одному. При отладке фактический порядок можно установить по протоколам изменения значений переменных, сопровождаемых номерами процессоров. Кроме того, процессор общей памяти может обладать небольшим набором своих специальных команд, выполняемых по запросам из активных процессов или отдельного процесса управления отладкой многопроцессорной программой.

Команды обычных процессоров, на которых выполняются активные процессы:

GIVE – запрос регистра для переменной с номером H;
SAFE – запрос на хранение данного в переменной по регистру H;
READ – запрос на чтение данного из переменной по регистру H;
UNDO – **з**апрос на получение предыдущего данного из переменной по адресу H;
FREE – объявление, что не нужна больше переменная по регистру H.

Команды-двойники процессора общей памяти, пассивно ждущего запросов от активных процессов или отладчика:

TAKE – выбор регистра для переменной с указанным номером H, счетчик кратности доступа к нему его надо увеличить на 1;

WRITE – размещение данного в регистре H. Если данное – указатель на вектор или список, то они копируются полностью в общую память. В протокол вносится копия указателя на данного и номер активного процесса;

VAR – чтение данного из регистра H для передачи обычному процессору – состояние регистра H не меняется;

UNDO – чтение предыдущего данного из регистра H – состояние регистра H не меняется;

FREE – освобождение памяти, доступной через регистр H, – счетчик кратности доступа уменьшается на 1.

Таблица 3

Дуплеты – парные команды – срабатывают только неразрывно вместе

Table 3

Duplets – paired commands – work only inextricably together

Запросы из активных процессоров	Двойники в процессоре общей памяти	Неразделимая пара
GIVE	**TAKE**	**Дай-Бери**
SAFE	**WRITE**	**Храни-Пишу**
READ	**VAR**	**Читаю-Доступ**
UNDO	**UNDO**	**UNDO-UNDO**
FREE	**FREE**	**Свободен-Свободен**

Сохранение значения из стека в глобальной памяти:

T – **номер текущего процесса**, задавшего запрос на использование или изменение значения глобальной переменной, совпадающий с номером очереди ленивых партнеров-двойников для него;

H – имя переменной для данного в общей памяти. Все имена различны, они известны на всех процессорах, определены при компиляции;

@H – адрес текущего значения переменной H в регистре, хранящем и протокол изменения значений – историю изменения значений;

Ц – **кратность доступа** к переменной в текущий момент;

[@H …] – шкала свободных регистров, содержащая адрес @H.

Дай-Бери

% на стеке локального процесса номер переменной замещается на адрес ее значения
% имя переменной заносится в паспорт процесса
V= [...] (H . s) e (**GIVE** . c) d → V= [H …] (@H . s) e c d
% @H адрес переменной в общей памяти исчезает из шкалы свободных регистров
m=[[@H …] …] s e (**TAKE** . c) d → m'=[[…] … (1 H)] s e c d % новая переменная
m=[[@H …] … (Ц H . Px) …] s e (**TAKE** . c) d → m'=[[…] … ((Ц+1) H . Px)] s e c d
% переменная уже была

Храни-Пишу

% переменная уже выделена, xt – данное для хранения в переменной H
% T – номер потока, известный в очереди с номером, совпадающим с его номером
% значение переменной остается на стеке после записи его в регистр переменной
% присваивание возможно лишь переменным из паспорта процесса
V= [H …] (xt . s) e (**SAFE** H . c) d → V= [H …] (xt . s) e c d
m=[… (Ц H . Px) …] (H . s) e (**WRITE** . c) d → m'= [… (Ц H **xt T xt** . Px)] s e c d
m=[… (Ц H)] (H . s) e (**WRITE** . c) d → m'= [… (Ц H **xt T xt**)] s e c d
% чистая переменная, в ней еще не было значений

Читаю-Доступ

% имя переменной на стеке замещается значением переменной из регистра в общей памяти

% чтение возможно лишь из переменных из паспорта процесса

V= [H …] (H . s) e (**READ** . c) d → V= [H …] (xт . s) e c d

m=[… (Ц H xт T xт . Px) …] (H . s) e (**VAR** . c) d → m s e c d % память не изменилась

UNDO-UNDO

% на стеке появляется предыдущее значение из регистра в общей памяти

% восстановление возможно лишь переменным из паспорта процесса

V= [H …] (H . s) e (**UNDO** X . c) d → V= [H …] (xтт . S) e c d

m=[… (Ц H xт T xт Тт xтт . Px) …] (H . s) e (**UNDO** X . c) d → m s e c d % память не изменилась

Свободен-Свободен

% имя переменной исчезает из стека, а в общей памяти уменьшается счетчик доступа

% удалить можно лишь переменную из паспорта процесса, она оттуда исчезает

V= [H …] (H . s) e (**FREE** . c) d → V= [...] s e c d

m=[… (Ц H xт T xт . Px) …] (H . s) e (**FREE** . c) d → m'==[… ((Ц-1) H xt T xt . Px)] s e c d

% если счетчик должен стать нулем, то сразу адрес регистра объявляется свободным

m=[… (1 H xт T xт . Px) …] (H . s) e (**FREE** . c) d → m'==[[@H …] …] s e c d

% важно, что это происходит неразрывно, без опасности вмешательства других команд

% адрес удаленной переменной размещается в шкале свободных регистров общей памяти

Специальные команды могут выполняться время от времени, можно их запускать после каждой команды «**FREE**». Если нашли счетчик ноль, то возбуждается команда «**Отключить**», и соответствующий адрес переходит в свободные адреса. Такой запрос может произойти при освобождении памяти при сборке мусора в активном процессе. Явного запроса в активном процессе нет, а при анализе выясняется, что некоторые переменные больше не нужны.

m=[… (0 H . Px) …] s e c d → m'==[… (0 H . Px)] s e (**Отключить** . c) d

m=[[…] … (0 H . Px) …] s e (**Отключить** . c) d → m'==[[@H …] …] s e c d

Таким образом, процессор общей памяти может функционировать как элемент распределенной системы, включающийся по мере поступления запросов из активных процессов и не требующий приостановки всех процессов на время решения проблемы освобождения памяти. Техника такой работы характерна для работы с памятью во многих системах программирования. Здесь она дополнена хранением протоколов изменения значений.

При рассмотрении традиционного варианта с восстановлением общей памяти по методике «Stop&Copy» оказалось, что он более сложен и не решает проблему приостановки активных процессоров на время освобождения общей памяти. Представленный здесь метод состоит из ряда процессоров со своей локальной памятью и процессора общей памяти. Локальные процессоры работаю автономно, по мере необходимости обращаясь к общей памяти и время от времени освобождая локальную память. В каждом процессоре хранится шкала регистров используемой общей памяти, которая может то расширяться, то сужаться по мере инструкций из программы или сужаться в результате сборки мусора. Общая память состоит из двух частей – регистров прямого доступа и кучи для хранения протоколов, а также структур данных и старых значений после присваивания на случай необходимости восстановления состояний памяти. При необходимости можно восстанавливать отдельные старые значения. Время от времени при исчерпании той или иной области общей памяти или просто по графику запускается алгоритм, похожий на «Stop&Copy» в системах функционального программирования.

Заключение

Представленное построение показывает, что на этапе ознакомления с проблемами параллелизма можно строить и отлаживать небольшие многопоточные программы взаимодействия процессов, выполняемых на модели произвольной многопроцессорной конфигурации над общей памятью. Опыт такой работы поможет обучаемым видеть и понимать разные явления, происходящие при взаимодействии потоков, предвидеть проблемы и накапливать рецепты решения проблем при подготовке и отладке программ. Для этих целей предлагается принцип неизменяемости данных, характерный для функционального программирования, распространить на работу с переменными в общей памяти и преобразовать его в принцип восстановимости данных, используя протоколы изменения значений переменных. Важную роль играют профилактика «фантомной» памяти и защита данных от некорректного стороннего доступа. Проблема освобождения общей памяти может быть решена как композиция методов «сборки мусора» и учета кратности доступа к переменным из потоков, взаимодействующих через использование данных, хранящихся в общей памяти. До полноты картины нужны еще команды управления процессами, но это уже не входит в тему данной статьи.

Список литературы

1. **Кнут Д. Э.** Искусство программирования. М.: Вильямс, 2007. Т. 1, вып. 1: MMIX – RISC-компьютеры нового тысячелетия. 160 с. ISBN 978-5-8459-1163-6
2. **Вирт Н.** Построение компиляторов. М.: ДМК Пресс, 2010. ISBN 978-5-94074-585-3, 0-201-40353-6
3. **Айлиф Дж.** Принципы построения базовой машины. М.: Мир, 1973. 119 с.
4. **Хендерсон П.** Функциональное программирование. М.: Мир, 1983. 349 с.
5. **Котов В. Е.** МАРС: архитектуры и языки для реализации параллелизма // Системная информатика. Новосибирск: Наука, 1991. Вып. 1: Проблемы современного программирования. С. 174–194.
6. **Иртегов Д. В.** Введение в операционные системы. СПб.: БХВ-Петербург, 2008. 1040 с.: ил. ISBN 978-5-94157-695-1
7. **Пеппер П., Экснер Ю., Зюдхольд М.** Функциональный поход к разработке программ с развитым параллелизмом // Системная информатика. Новосибирск: Наука, 1995. Вып. 4: Методы теоретического и системного программирования. С. 334–360.
8. **Грабер М.** Введение в SQL. М.: Лори, 1996. 377 с.
9. **Лавров С. С., Городняя Л. В.** Функциональное программирование. Интерпретатор языка Лисп // Компьютерные инструменты в образовании. 2002. № 5. С. 49–58.
10. **Эванс Б., Гоф Дж., Ньюленд К.** Java: оптимизация программ. Практические методы повышения производительности приложений в JVM. М.: Диалектика, 2019. 448 с. ISBN 978-5-907114-84-5
11. **Gorodnyaya L.** Method of paradigmatic analysis of programming languages and systems. In: CEUR Workshop Proceedings, 2020, no. 2543, pp. 149–158.
12. **Gorodnyaya L.** On the presentation of the results of the analysis of programming languages and systems. In: CEUR Workshop Proceedings, 2018, no. 2260, pp. 152–166.
13. **Cann D. C.** SISAL 1.2: A Brief Introduction and tutorial. Preprint UCRL-MA-110620. Lawrence Livermore National Lab. Livermore, California, 1992, May, 128 p.
14. **Backus J.** Can programming be liberated from the von Neumann style? A functional stile and its algebra of programs. In: Commun. ACM 21, 1978, no. 8, pp. 613–641.
15. **Филд А., Харрисон П.** Функциональное программирование / Пер. под ред. В. А. Горбатова. М. Мир, 1993. 638 с.
16. **Городняя Л. В.** Основы функционального программирования. М.: Интернет-университет информационных технологий, 2004. 272 с.

17. **Дейкстра Э.** Дисциплина программирования. М.: Мир, 1978. 275 с.
18. **Городняя Л. В.** Язык параллельного программирования СИНХРО, предназначеный для обучения. Новосибирск, 2016. Препринт ИСИ СО РАН № 180. 30 с.
19. **Баранов С. Н., Колодин М. Ю.** Феномен Форта // Системная информатика. Новосибирск: Наука, 1995. Вып. 4: Методы теоретического и системного программирования. С. 193–271.
20. **Ластовецкий А. Л.** Предварительное сообщение о языке программирования mpC // Вопросы кибернетики: Приложения системного программирования / Научный совет по комплексной проблеме «Кибернетика» РАН. М., 1995. С. 20–39.

References

1. **Knut D. E.** Iskusstvo programmirovaniya [The Art of Computer Programming]. Moscow, Vilyams Publ., 2007, vol. 1, iss 1: MMIX – A RISC Computer for the New Millennium, 160 p. (in Russ.) ISBN 978-5-8459-1163-6
2. **Virt N.** Postroyeniye kompilyatorov. Moscow, DMK Press, 2010. (in Russ.) ISBN 978-5-94074-585-3, 0-201-40353-6
3. **Aylif J.** Printsipy postroyeniya bazovoy mashiny. Moscow, Mir, 1973, 119 p. (in Russ.)
4. **Henderson P.** Funktsional'noye programmirovaniye. Moscow, Mir, 1983, 349 p. (in Russ.)
5. **Kotov V. E.** MARS: arkhitektury i yazyki dlya realizatsii parallelizma. In: Sistemnaya informatika. Novosibirsk, Nauka, 1991, iss. 1, pp. 174–194. (in Russ.)
6. **Irtegov D. V.** Vvedeniye v operatsionnyye sistemy. St. Petersburg, BKHV-Peterburg, 2008, 1040 p.: il. (in Russ.) ISBN 978-5-94157-695-1
7. **Pepper P., Eksner Yu., Zyudkhold M.** Funktsional'nyy pokhod k razrabotke programm s razvitym parallelizmom. On: Sistemnaya informatika. Novosibirsk, Nauka, 1995, iss. 4, pp. 334–360. (in Russ.)
8. **Graber M.** Vvedeniye v SQL. Moscow, Lori, 1996, 377 p. (in Russ.)
9. **Lavrov S. S., Gorodnyaya L. V.** Funktsional'noye programmirovaniye. Interpretator yazyka Lisp. *Komp'yuternyye instrumenty v obrazovanii*, 2002, no. 5, pp. 49–58. (in Russ.)
10. **Evans B., Gof Dzh, Nyulend K.** Java: optimizatsiya programm. Prakticheskiye metody povysheniya proizvoditel'nosti prilozheniy v JVM. Moscow, Dialektika Publ., 2019, 448 p. (in Russ.) ISBN 978-5-907114-84-5
11. **Gorodnyaya L.** Method of paradigmatic analysis of programming languages and systems. In: CEUR Workshop Proceedings, 2020, no. 2543, pp. 149–158.
12. On the presentation of the results of the analysis of programming languages and systems. In: CEUR Workshop Proceedings, 2018, no. 2260, pp. 152–166.
13. **Cann D. C.** SISAL 1.2: A Brief Introduction and tutorial. Preprint UCRL-MA-110620. Lawrence Livermore National Lab. Livermore, California, 1992, May, 128 p.
14. **Backus J.** Can programming be liberated from the von Neumann style? A functional stile and its algebra of programs. In: Commun. ACM 21, 1978, no. 8, pp. 613–641.
15. **Fild A., Harrison P.** Funktsional'noye programmirovaniye. Trans., ed. by V. A. Gorbatov. Moscow, Mir, 1993, 638 p. (in Russ.)
16. **Gorodnyaya L. V.** Osnovy funktsional'nogo programmirovaniya. Moscow, Internet-Universitet Informatsionnykh tekhnologiy, 2004, 272 p. (in Russ.)
17. **Deykstra E.** Distsiplina programmirovaniya. Moscow, Mir, 1978, 275 p. (in Russ.)
18. **Gorodnyaya L. V.** Yazyk parallel'nogo programmirovaniya SINKHRO, prednaznachenyy dlya obucheniya. Novosibirsk, 2016, preprint ISI SO RAN № 180, 30 p. (in Russ.)
19. **Baranov S. N., Kolodin M. Yu.** Fenomen Forta. In: Sistemnaya informatika. Novosibirsk, Nauka, 1995, iss. 4, pp. 193–271. (in Russ.)

20. **Lastovetsky A. L.** Predvaritel'noye soobshcheniye o yazyke programmirovaniya mpC. In: Voprosy kibernetiki: Prilozheniya sistemnogo programmirovaniya. Moscow, 1995, pp. 20–39. (in Russ.)

Информация об авторе

Лидия Васильевна Городняя, кандидат физико-математических наук, доцент

Information about the Author

Lydia V. Gorodnyaya, Candidate of Sciences (Physics and Mathematics), Associate Professor

Статья поступила в редакцию 10.10.2021;
одобрена после рецензирования 01.12.2021; *принята к публикации 01.12.2021*
The article was submitted 10.10.2021;
approved after reviewing 01.12.2021; *accepted for publication 01.12.2021*

Научная статья

УДК 004.89
DOI 10.25205/1818-7900-2021-19-4-36-49

Исследование SLAM-фреймворков для монокулярных мобильных роботов в проекте Duckietown

Александра Денисовна Девятовская [1]
Никита Евгеньевич Бирючков [2]
Татьяна Викторовна Лях [3]
Константин Владимирович Чайка [4]

[1–3] Новосибирский государственный университет
Новосибирск, Россия

[4] Санкт-Петербургский государственный электротехнический университет
Санкт-Петербург, Россия

[4] Лаборатория алгоритмов мобильных роботов JetBrains Research
Санкт-Петербург, Россия

[1] a.devyatovskaya@g.nsu.ru, https://orcid.org/0000-0002-0583-4202
[2] n.biryuchkov@g.nsu.ru, https://orcid.org/0000-0002-1755-9527
[3] t.liakhv@g.nsu.ru, https://orcid.org/0000-0001-9148-946X
[4] pro100kot14@gmail.com, https://orcid.org/0000-0001-5778-9266

Аннотация

Статья посвящена оценке применимости SLAM фреймворков для задачи мобильных роботов проекта Duckietown. Проведен сравнительный анализ существующих SLAM алгоритмов и фреймворков, были отобраны фреймворки с учетом всех ограничений, накладываемых роботами проекта. Приведены практические результаты апробации фреймворка OpenVSLAM как на данных реального окружения Duckietown, так и на данных симулятора Duckietown.

Ключевые слова

Duckietown, low-cost платформа, SLAM, локализация, мобильные системы

Благодарности

Работа выполнена при поддержке JetBrains Research

Для цитирования

Девятовская А. Д., *Бирючков Н. Е.*, *Лях Т. В.*, *Чайка К. В.* Исследование SLAM-фреймворков для монокулярных мобильных роботов в проекте Duckietown // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 4. С. 36–49. DOI 10.25205/1818-7900-2021-19-4-36-49

© Девятовская А. Д., Бирючков Н. Е., Лях Т. В., Чайка К. В., 2021

SLAM in Duckietown Simulator Using the OpenVSLAM Framework

Alexandra D. Devyatovskaya [1], **Nikita E. Biryuchkov** [2]
Tatyana V. Liakh [3], **Konstantin V. Chaika** [4]

[1–3] Novosibirsk State University
Novosibirsk, Russian Federation

[4] Saint Petersburg Electrotechnical University
St. Petersburg, Russian Federation

[4] Mobile Robot Algorithms Laboratory JetBrains Research
St. Petersburg, Russian Federation

[1] a.devyatovskaya@g.nsu.ru, https://orcid.org/0000-0002-0583-4202
[2] n.biryuchkov@g.nsu.ru, https://orcid.org/0000-0002-1755-9527
[3] t.liakhv@g.nsu.ru, https://orcid.org/0000-0001-9148-946X
[4] pro100kot14@gmail.com, https://orcid.org/0000-0001-5778-9266

Abstract

The article is devoted to evaluating the applicability of SLAM frameworks for the task of mobile robots of the Duckietown project. The paper provides a comparative analysis of existing SLAM algorithms and frameworks and selects frameworks taking into account all the constraints imposed by the project robots. The practical results of testing OpenVSLAM framework on the Duckietown environment and Duckietown simulator data are presented.

Keywords

Duckietown, low-cost platforms, SLAM, localization, mobile systems

Acknowledgements

The work was carried out with the support of JetBrains Research

For citation

Devyatovskaya A. D., Biryuchkov N. E., Lyakh T. V., Chaika K. V. SLAM in Duckietown Simulator Using the OpenVSLAM Framework. *Vestnik NSU. Series: Information Technologies*, 2021, vol. 19, no. 4, p. 36–49. (in Russ.) DOI 10.25205/1818-7900-2021-19-4-36-49

Введение

На сегодняшний день активно развивается исследовательский проект под названием «Duckietown» [1]. Это платформа, которая имитирует реальный город и используется для разработки и изучения алгоритмов управления автономными автомобилями. Он состоит из двух частей – роботов (duckiebots) и города (duckietown), по которому они передвигаются. В городе есть размеченные дороги, светофоры, дорожные знаки и препятствия. Тренажер отражает особенности индустриальных автопилотируемых систем с ограниченными ресурсами: роботы оснащены только камерами, а также имеют слабую вычислительную мощность. Разработчики используют ее для отладки и тестирования SLAM алгоритмов для промышленных роботов [1] и автопилотирования.

Алгоритмы построения карты местности и одновременного позиционирования (Simultaneous Localization And Mapping – SLAM) используются для решения задач локализации объекта и построения карты окружающей территории в сфере мобильной робототехники. SLAM выполняет минимизацию ошибок определения позиции камеры и вычисление инверсной глубины ключевых точек. За счет этого выравниваются траектория движения робота (камеры) и обозреваемое пространство, на основе которого формируется карта мира. Также технологии компьютерного зрения используются во многих областях промышленности и науки [2], в том числе и в робототехнике.

[1] Duckietown. URL: https://www.duckietown.org/ (дата обращения 31.03.2021).

SLAM алгоритмы – хорошо изученная область. Множество исследований описывают различные backend (TORO [3], ORB_SLAM [4], DSO_SLAM [5]) и frontend (Fast SLAM [2], DP-SLAM [3], Loop Closure [6]) SLAM [7] алгоритмов. Было проведено множество тестов для этих алгоритмов, их комбинаций в различных условиях освещенности, сложности рельефа местности, вычислительных мощностях роботов и типах камеры. Также, существует широкий спектр SLAM-фреймворков, таких как ISam [8], GMapping [9], OpenVSLAM [4], Cartographer [10], OV2Slam [5]. Сообществом проекта Duckietown также ведется разработка SLAM-фреймворков, таких как CSlam [6] и Lane-based SLAM [7].

Duckietown накладывает свои требования на используемые алгоритмы и фреймворки. Большинство разработчиков создают фреймворки для машин, использующих большое количество сенсоров (LIDAR [8], одометрия) и камер. Однако оснащение каждой машины или робота множеством датчиков может быть дорогостоящим для предпринимателя. Duckiebots имитируют промышленные low-cost автопилотируемые системы с ограниченными ресурсами: роботы оснащаются только камерами и имеют слабую вычислительную мощность. Данные особенности симулятора и его роботов накладывают ограничение на использование большинства стандартных алгоритмов и фреймворков SLAM. В результате задача SLAM в проекте Duckietown до сих пор не решена.

В статье был проведен анализ и апробация SLAM фреймворков для использования в тренажере duckiebots с учетом всех ограничений.

Анализ SLAM алгоритмов

Существует две большие группы SLAM алгоритмов – SLAM frontend и SLAM backend.

SLAM frontend решает следующие задачи.

1. Анализа и интеграции новых данных (data association). На этом этапе производится выделение особенностей (Feature/landmarks extraction) на вновь полученных данных. Особенности – это такие характеристики, которые легко могут быть выделены в среде и использоваться в дальнейшем для ориентации в пространстве.
2. Вычисления сдвига (Local Motion Estimation). Сдвиг положения выделенных особенностей на сцене можно определить, сопоставив набор особенностей, полученных на текущем шаге, с набором особенностей, полученных на предыдущем шаге. Так как особенности сами по себе стационарны, очевидно, что этот сдвиг – результат изменения положения камеры (робота). На основе этой информации можно выразить координаты камеры через систему линейных уравнений.
3. Обновления структуры (Features Integration), хранящей историю перемещений, где каждое состояние представляет собой глобальное положение робота и взаимное расположение обозреваемых особенностей на определенном промежутке времени. На этом этапе производится анализ, и полученные ранее данные добавляются в общую структуру для хранения информации о мире за все время исследования.

Существует множество frontend SLAM алгоритмов: ICP SLAM [11], Point Cloud SLAM [9], Fast Slam, DP_SLAM, Loop Closure. Для их сравнения были выбраны критерии: сложность алгоритма, устойчивость, погрешность (табл. 1). После анализа были отобраны три frontend

[2] FAST SLAM. URL: https://github.com/bushuhui/fastslamtps (дата обращения 31.03.2021).
[3] DP-SLAM. URL: https://users.cs.duke.edu/~parr/dpslam/ (дата обращения 31.03.2021).
[4] Tutorial Openvslam. URL: https://openvslam-community.readthedocs.io/en/latest/example.html (дата обращения 31.03.2021).
[5] OV2Slam. URL: https://github.com/ov2slam/ov2slam (дата обращения 31.03.2021).
[6] CSLAM. URL: https://github.com/duckietown/duckietown-cslam (дата обращения 31.03.2021).
[7] Lane-based SLAM. URL: https://github.com/mandanasmi/lane-slam (дата обращения 31.03.2021).
[8] LIDAR. URL: https://velodynelidar.com/what-is-lidar/ (дата обращения 31.03.2021).
[9] Point Cloud SLAM Overview. URL: https://www.mathworks.com/help/vision/ug/point-cloud-registration-workflow.html (дата обращения 31.03.2021).

алгоритма, которые могут быть использованы в проекте Duckietown, и проведен их сравнительный анализ. Были отброшены алгоритм ICP SLAM, так как он требует больших вычислительных мощностей, и алгоритм Point Cloud SLAM, так как требует дополнительных сенсоров.

Таблица 1

Сравнение SLAM frontend алгоритмов

Table 1

Comparison of SLAM frontend algorithms

Критерий	Loop Closure	DP_SLAM	Fast SLAM
Сложность	Логарифмическая	Логарифмическая	Логарифмическая
Устойчивость	Не может работать длительное время из-за сложности алгоритма и накапливающегося количества ошибок	Неточность при гладкой текстуре, не имеющей углов, появление шумов, может выдавать ошибки на мелких сильно удаленных объектах	Неточность на сложной карте и при обработке большого количества данных
Погрешность, %	~ 10	~ 5–10	~ 20

1. Fast Slam [10]

Этот алгоритм основывается на идее Мерфи. Во фреймворке используется фильтр частиц Рао – Блэквелла для построения гипотез о текущем положении робота и фильтр Калмана для отслеживания положений наперед заданных меток. Одной из проблем при составлении карты является возможная идентичность встречающихся на карте объектов. Чтобы более точно определять местоположение робота и пространство вокруг него, в рассматриваемом методе вводятся ориентиры (метки), которые уникальны для каждого объекта на карте и позволяют различать местность. В Fast SLAM одна большая карта рассматривается как совокупность локальных подкарт, что позволяет убрать зависимость ориентиров друг от друга и таким образом значительно сократить время пересчета оценки состояния системы.

Сложность: O(MlogK), где M – число частиц в фильтре частиц, а K – число меток.

2. DP_Slam [11]

DP-SLAM также основан на идее Мерфи. Он тоже использует фильтры частиц, однако не требует каких-либо предопределенных меток. Проблема сопоставления реальных объектов на карте с собранными данными в этом подходе решается путем хранения в памяти множества промежуточных карт. Карта представляется в виде сетки с заполнением ячеек, занятых препятствиями. Хранить такую карту удобно в виде массива, где элементы, отражающие положение препятствий, имеют значение 1, а все остальные – 0. Таким образом, обе части задачи – локализация и составление карты – решаются одним и тем же фильтром частиц. Карты хранятся в памяти в специальном виде, называемом «распределенные частицы» (distributed particle, DP), что позволяет управлять сотнями и тысячами вероятных карт-кандидатов и вероятных позиций одновременно. Алгоритм DP-SLAM достаточно прост в реализации, поскольку он не использует дополнительных эвристических методов разрешения циклов. Кроме того, он обладает высокой производительностью. Алгоритм в наихудшем случае про-

[10] FAST SLAM. URL: https://github.com/bushuhui/fastslamtps (дата обращения 31.03.2021).
[11] DP-SLAM. URL: https://users.cs.duke.edu/~parr/dpslam/ (дата обращения 31.03.2021).

демонстрировал лого квадратичную сложность в зависимости от количества отсчетов и линейную сложность в зависимости от обозреваемой лазерным сенсором площади.

3. Loop Closure [6]

Loop closure – обнаружение петель. Обособленно от задач SLAM стоит вопрос, как отслеживать ситуации, когда робот возвращается туда, где уже побывал. Одно из решений – так называемая Bags of Binary Words. Каждому кадру ставится в соответствие дескриптор (BRIEF descriptor, вычисляемый на основе визуальных особенностей изображения). Для хранения информации об изображениях на основе данных для обучения формируется словарь в виде дерева, содержащий «слова» для представления дескрипторов и их веса (отражающих, насколько часто они встречались в наборе изображений для обучения). Для формирования «слова» производится поиск визуальных особенностей на основе данных для тренинга и их последующая группировка (с помощью метода k-среднего). Каждый последующий уровень дерева получается путем повторения данной операции с дескриптором родительского узла. В результате при анализе новых поступивших данных производится быстрое (дистанция Хэмминга) разложение идентификатора текущего кадра в линейную комбинацию узлов дерева, где в роли коэффициентов выступают их веса – вероятности.

SLAM backend решает следующие задачи.

1. Сравнения данных локального сдвига и особенностей. С одной стороны, у нас есть результаты локального сдвига, на основании которых мы можем вычислить новое положение робота. С другой – мы также можем определить позицию робота относительно особенностей сцены. В силу погрешностей датчиков и алгоритмов определения местоположения при попытке совместить подсчитанное разными методами расположение робота может возникнуть погрешность. Таким образом, существует вероятность возникновения погрешности в определении локального положения.
2. Обновления общего представления о карте мира и траектории робота (Global Optimization). На основе вновь полученных данных производится уточнение текущих представлений о мире – пересчет позиций робота (state estimation), особенностей и вероятность появления их на карте с учетом погрешности. После всех вычислений обновляется общее представление мира – местоположение робота (state update) и координаты особенностей (landmark update).

Также существует множество backend slam алгоритмов: SIFT SLAM [12], TORO, ORB_SLAM, DSO_SLAM. Для их сравнения были выбраны критерии: сложность алгоритма, устойчивость, погрешность, оптимальные условия применения и сфера применимости (табл. 2). После анализа были отобраны три backend алгоритма, которые могут быть использованы на тренажерах Duckietown, и проведен их сравнительный анализ. SIFT SLAM был отброшен, так как требует мощной видеокарты.

1. TORO [3]

Данный метод определяет конфигурация графа, при которой вероятность наблюдаемых особенностей будет максимальна. Подходы к решению подобных задач –Sparse Bundle Adjustments, Structure from motion, Visual SLAM. В основу TORO лег метод стохастического градиентного спуска – (SGD – stochastic gradient descent). Суть данного метода в последовательной оптимизации всего графа для каждого ограничения (взаимного расположения особенностей и робота). Для объединения данных оптимизации с нескольких ограничений используются специальные коэффициенты для корректировки значений остатка. Ключевое достижение метода – представление данных для SGD – в TORO используется дерево параметров в качестве индексов для линейного уравнения поз робота.

Таблица 2

Сравнение SLAM backend алгоритмов

Table 2

Comparison of SLAM backend algorithms

Критерий	TORO	ORB_SLAM	DSO_SLAM
Сложность	Линейная	Константная	Линейная
Устойчивость	Устойчивость к плохим начальным конфигурациям за счет минимизации ошибок на основе градиентного спуска	Неточность при гладкой текстуре, не имеющей углов, появление шумов, может выдавать ошибки на мелких сильно удаленных объектах	Необходимость хорошей инициализации; невысокая скорость, за счет которой идет более точная картинка
Погрешность, %	~ 5–15	~ 15	~ 10–15
Оптимальные условия применения	Хорошо работает как с большими, так и с мелкими объектами на сцене, хуже работает при плохой освещенности	Monocular, Stereo or RGB-D cameras, желательна хорошая освещенность	Желательна хорошая освещенность, но неплохо работает и при плохой освещенности за счет удаления факторов погрешности от геометрических и фотометрических перспектив
Сфера применимости	3D-реконструкция улиц, дорог, внутренней планировки домов	3D-реконструкция сцены в самых разнообразных средах, начиная от небольших лабораторных карт и заканчивая автомобилем, объезжающим несколько городских кварталов	3D-реконструкция улиц, дорог, внутренней планировки домов

2. ORB_SLAM [4]

Это непрямой метод, базирующийся на поиске ключевых точек с помощью FAST детектора и поиска ориентированных дескрипторов. Он производится в несколько этапов: отслеживание, построение локальной карты, loop closure, релокализация, глобальная оптимизация. Этот метод достаточно эффективен, однако, поскольку он использует детектор углов, то дает недостаточно точные результаты, несмотря на хорошо реализованный алгоритм замыкания петель. Плюсы: быстродействие; легко удаляются шумы. Недостатками являются неточность при гладкой текстуре, не имеющей углов; использование малой части информации изображений.

3. DSO_SLAM [5]

Метод SLAM использует прямую одометрию и создает разреженную карту местности. Создание разреженной карты делает его эффективнее (LSD SLAM), так как повышает быстродействие и уменьшает аппаратные расходы. Метод требует достаточно серьезной начальной инициализации, геометрическую и фотометрическую калибровку. Поиск ключевых то-

чек ведется по наибольшему градиенту, применяется фотометрическая коррекция изображения. К достоинствам DSO_SLAM, можно отнести использование более полной информации о изображении, в связи с чем точное нахождение ключевых точек и построение карты. К недостаткам – необходимость хорошей инициализации, невысокую скорость, однако возможность распараллеливания.

Анализ SLAM-фреймворков

SLAM алгоритмы являются базовыми схемами действия для решения определенных SLAM задач. На их базе создаются SLAM-фреймворки – готовые программы или библиотеки, которые могут включать в себя несколько SLAM алгоритмов или их улучшенные под конкретный фреймворк модификации.

Проведенный анализ сравнения SLAM backend и fronted алгоритмов повлиял на выбор SLAM-фреймворков для изучения. В конечную выборку для изучения применимости на роботах проекта Duckitown были включены только те фреймворки, которые основываются на отобранных алгоритмах. Во время составления выборки были рассмотрены такие SLAM-фреймворки, как GMapping, KartoSLAM [13], HectorSLAM [14], LagoSLAM [15], CoreSLAM [16], VSLAM, Hector_mapping.

После исследования были отобраны и проанализированы пять SLAM-фреймворков, которые подходят по используемым SLAM алгоритмам, активно использовались в исследовательских проектах и доказали свою работоспособность [17] (табл. 3).

1. GMapping [9]

Библиотека, описанная и поддерживаемая в официальной документации ROS. Использует высокоэффективный Rao-Blackwellized фильтр частиц для изучения сеточных карт по данным лазерного диапазона. Этот подход использует фильтр частиц, в котором каждая частица несет индивидуальную карту окружающей среды. Соответственно, ключевой вопрос состоит в том, как уменьшить количество частиц. В этом SLAM алгоритме используются адаптивные методы уменьшения числа частиц в фильтре частиц с Rao- Blackwellized для изучения сеточных карт. Также в нем предлагается подход к вычислению точного распределения предложений, учитывающий не только движение робота, но и самые последние наблюдения. Это резко уменьшает неопределенность относительно позы робота на этапе прогнозирования фильтра.

2. VSLAM [12]

Библиотека, поддерживаемая ROS. В официальной документации описана как экспериментальная. Компоненты библиотеки:

- обнаружение / сопоставление признаков;
- поиск ключевых точек на изображении;
- сопоставление с ключевыми точками на других изображениях;
- visual odometry;
- крупномасштабная оптимизация по 3D точечным позициям и позициям камер;
- Loop closure;
- поиск совпадений между текущим кадром и набором предыдущих кадров.

[12] VSLAM. URL: http://wiki.ros.org/vslam (дата обращения 31.03.2021).

Таблица 3

Сравнение SLAM-фреймворков

Table 3

Comparison of SLAM frameworks

Критерий	GMapping	VSLAM	Hector-Mapping	OpenVSLAM	Cartographer_ros
Требования	Linux, GCC 3.3/4.0.x, odometry sensors, laser range-finder	Camera, Linux/Unix	LIDAR systems, Linux/Unix	Linux/Unix, Camera	64-bit, modern CPU, 16 GB RAM, Ubuntu 18.04, 20.04, gcc version 7.5.0, 9.3.0
Возможность использования вместе с IMU	+	–	+	–	+
Возможность использования вместе с датчиками одометрии	+	+	–	–	+
Использует	Rao-Blackwellized фильтр частиц, visual odometry	Visual odometry, Sparse bundle adjustment, Loop closure	Gauss-Newton Approach	Gauss-Newton Approach, ORB_SLAM, Proclaim и UcoSLAM	Loop closure,Visual odometry
Где используется	Мобильные роботы – платформы, обязательно наличие камеры	Мобильные роботы – платформы, обязательно наличие камеры	Беспилотные наземные роботы, транспортные средства, портативные картографические устройства и регистрируемые данные с БПЛА	Мобильные роботы – платформы, обязательно наличие камеры	Мобильные роботы – платформы

3. Hector-mapping [13]

Библиотека, описанная и поддерживаемая в официальной документации ROS. Это подход SLAM, который может быть использован без одометрии. Он использует высокую частоту обновления современных лидарных систем, таких как Hokuyo UTM-30LX, и обеспечивает 2D-оценку позиции при высокой скорости сканирования датчиков (40 Гц для UTM-30LX). Хотя система не обеспечивает явной возможности замыкания цикла, она достаточно точна для многих реальных сценариев. Система успешно применяется на беспилотных наземных роботах, беспилотных наземных транспортных средствах, портативных картографических устройствах и регистрируемых данных с БПЛА. Hector SLAM полагается на сопоставление сканирования, использует подход Гаусса – Ньютона и достаточно точен, чтобы не требовать loop closure и визуальной одометрии.

4. OpenVSLAM [14]

Это монокулярная, стерео и визуальная система RGB-D Slam. Примечательными особенностями являются:

- фреймворк совместим с различными типами моделей камер и может быть легко настроен для других моделей камер. При необходимости пользователи могут легко реализовать компоненты для работы с различными моделями камер (например, dual fisheye, catadioptric);
- созданные карты можно хранить и загружать, а затем OpenVSLAM может локализовать новые изображения на основе готовых карт;
- система полностью модульная. OpenVSLAM разработан путем инкапсуляции нескольких функций в отдельные компоненты с помощью простых для понимания API;
- OpenVSLAM основан на indirect SLAM алгоритме с дополнительными функциями, такими как ORB_SLAM, Proclaim и UcoSLAM.

5. Cartographer_ros [10]

Cartographer – это система, которая обеспечивает одновременную локализацию и создание карт в реальном времени в формате 2D и 3D на нескольких платформах и конфигурациях датчиков. Портативные лазерные дальномеры и синхронная локализация, создание карт – эффективный метод получения готовых карт этажей зданий. Создание и визуализация поэтажных планов в реальном времени помогает оценить качество и охват данных.

Duckietown на данный момент поддерживает собственные SLAM-фреймворки, такие как CSlam и Lane-based SLAM. Они работают очень медленно, требуют специального внешнего оборудования и не подходят для использования на больших площадях, например в складах или большой площади умного города.

1. CSlam

Требуемое оборудование:

- Duckiebot;
- Watch Tower для наблюдения за полем;
- AprilTag знаки.

Цель cSLAM – объединить наблюдения сторожевых башен и duckiebots в единую систему оптимизаций, которая пытается максимально точно предсказать текущее и прошлое местоположение duckiebots. Затем это визуализируется на 3D-модели города. Плюсы: модульность, возможность расширения (больше роботов, больше территория города), относительно удобный интерфейс, функция выявления ошибок. Минусы: зависимость от освещения и погодных условий, все AprilTag должны быть хорошо различимы, погрешность 22 %.

[13] Hecor_mapping. URL: http://wiki.ros.org/hector_mapping (дата обращения 31.03.2021).

[14] Tutorial Openvslam. URL: https://openvslam-community.readthedocs.io/en/latest/example.html (дата обращения 31.03.2021).

2. Lane-based SLAM [15]

Идея – использовать семантические изображения с камеры «уткобота». Линии распознаются как часть дороги. Есть три различных типа линий: желтые линии в центре дороги, белые линии, расположенные по бокам дороги, и красные линии, которые являются стоплиниями. Затем их используют в качестве предварительного знания о «дакитауне». Используется одометрия для оценки положения позиции робота. В основе – фильтр spurious lines. Комбинация одометрии и обнаружения линий. Используемые модули: Line detector module, Line descriptor module, Ground projection module, Line sanity module, Odometry module, Show map module. Плюсы: модульность, совместимость с ROS, визуализация карты с цветом линий. Минусы: не проработан до конца, может давать большую погрешность (30 %)

По результатам проведенного анализа были отобраны те фреймворки, которые позволяют использовать в качестве датчика монокулярную камеру, расположенную на Duckiebot и не требуют дополнительных датчиков. Из приведенных выше фреймворков этим требованиям соответствуют VSLAM и OpenVSLAM. После более детального изучения этих двух фреймворков было решено тестировать только OpenVSLAM, так как разработка VSLAM-фреймворка была заброшена.

Тестирование OpenVSLAM на стандартном видео dataset Duckietown

В эксперименте фреймворк настроен в соответствии с конфигурацией камеры Raspberry pi. Были выставлены настройки: --frame-skip arg (=1) – интервал пропуска кадров, --no-sleep – не ожидать следующий кадр в режиме реального времени, частота кадров в секунду 30 fps, разрешение кадров 1920×1080. На вход подавался видеодатасет, записанный с камеры duckiebot. Робот двигался по карте спроектированного города и ориентировался по специальной разметке, нанесенной на объекты.

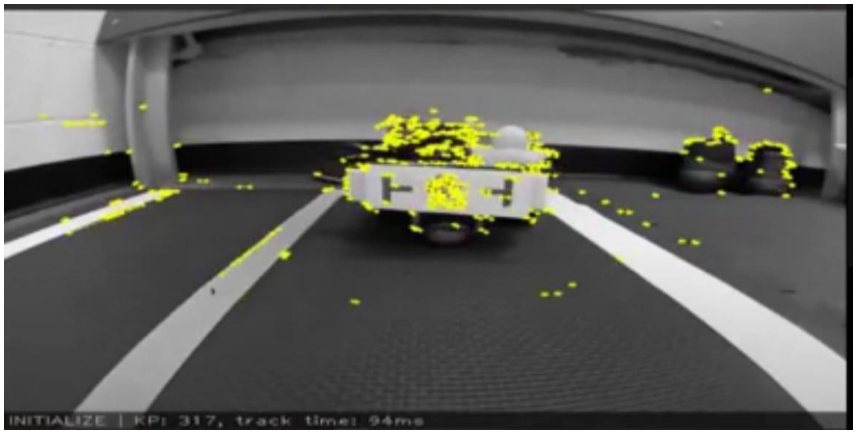


Рис. 1. Результат выставления меток на карте Duckietown
Fig. 1 Result of marking on the Duckietown map

В ходе эксперимента было установлено, что фреймворк с высокой точностью способен расставлять метки, обозначенные желтым цветом, на кадрах видеопотока. Результат работы OpenVSLAM представлен на рис. 1. При этом алгоритму не удалось построить карту местности, по которой двигался робот. При каждом резком повороте камеры Raspberry pi, фреймворк сбрасывал часть карты, которая была построена, и начинал строить маршрут сначала.

[15] Lane-based SLAM. URL: https://github.com/mandanasmi/lane-slam (дата обращения 31.03.2021).

Для устранения отказа в работе OpenVSLAM, настройки, указанные выше, варьировались в соответствии с возможными режимами. Флаг --no-sleep был отключен. В этом случае алгоритм также сбрасывал карту через каждые 3 секунды.

Тестирование OpenVSLAM на стандартном фото dataset Duckietown

В эксперименте видеодатасет был отформатирован и разделен на набор фотографий. Настройки OpenVSLAM выставлены в соответствии с камерой Raspberry pi, частота кадров 30 fps.

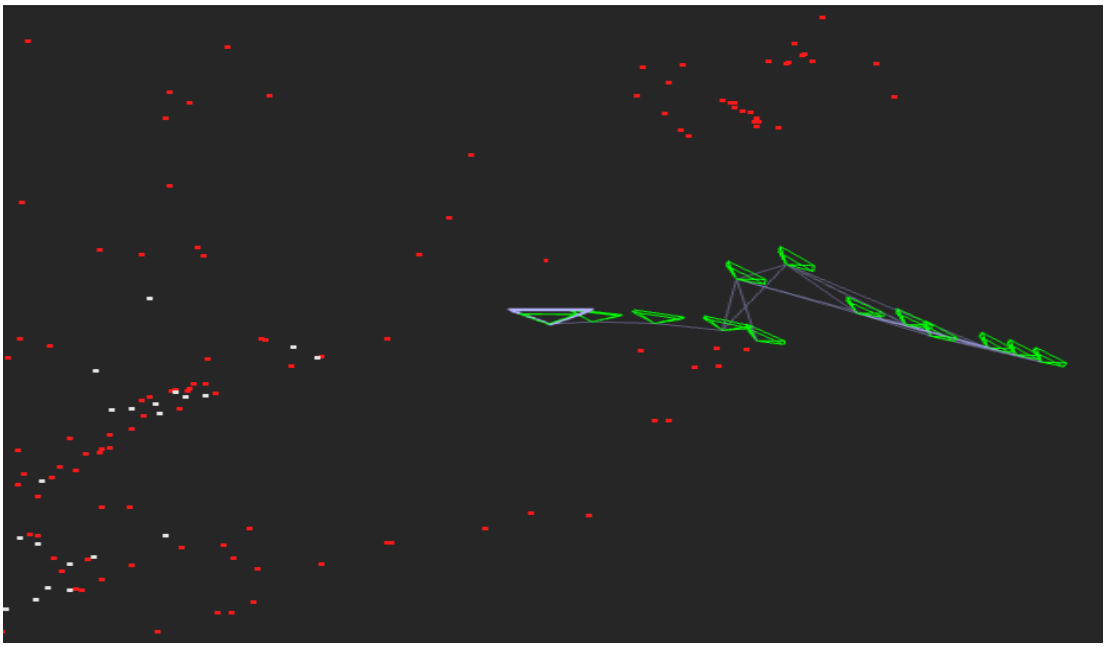


Рис. 2. Результат отрисовки карты фреймворком
Fig. 2. The result of the framework's rendering of the map

В результате работы фреймворка была построена виртуальная карта (рис. 2), повторяющая траекторию движения мобильного робота по местности Duckietown. Итоговая траектория отрисована зелеными треугольниками. Также алгоритм с высокой точностью определял свое текущее положение и выставлял соответствующую метку, обозначенную синим цветом. Каждая красная точка, выставленная OpenVSLAM, отображает, объект, находящийся перед камерой робота.

В ходе выполнения алгоритма был определен замкнутый цикл в маршруте duckiebot.

Апробация OpenVSLAM на данных симулятора Duckietown

В эксперименте был использован симулятор окружения Duckietown, который входит в перечень стандартных инструментов для работы с мобильными роботами проекта. Для работы с симулятором выбран скрипт, написанный на языке python. После запуска симулятора в стандартном режиме был записан видеопоток с виртуальной камеры duckiebot. С учетом предыдущих экспериментов Dataset разделен на набор фотографий, которые подавались на вход OpenVSLAM. Настройки, как и в предыдущих экспериментах, установлены в соответствии с конфигурацией камеры Raspberry pi.

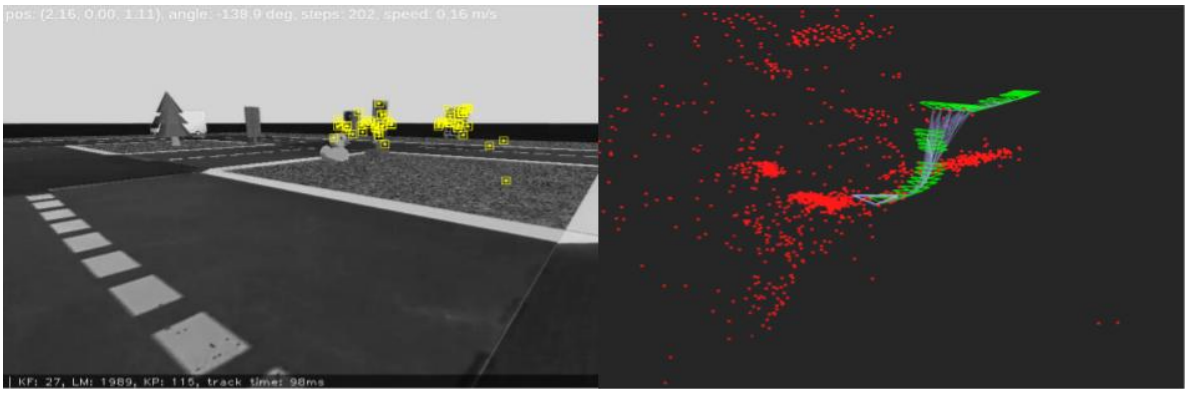


Рис. 3. Результат выставления меток на карте имитатора Duckietown и отрисовки карты фреймворком
Fig. 3. The result of marking on the Duckietown imitator map and of the framework's rendering of the map

В результате выполнения алгоритма OpenVSLAM расставил метки на каждом из объектов имитационного города Duckietown (рис. 3). Также была построена карта виртуальной местности уткогорода. Фреймворк с высокой точностью указывал положение препятствий на маршруте, а также свое местоположение.

Заключение

В статье проведено сравнительное исследование алгоритмов, использующихся для локализации мобильных роботов, оснащенных одной только монокулярной камерой и имеющих малые вычислительные мощности.

С учетом ограничений, которые накладывает используемое оборудование, установлено, что поставленным требованиям удовлетворяет фреймворк OpenVSLAM. Разработан алгоритм настройки фреймворка OpenVSLAM для работы с датасетами с Duckietown. Фреймворк был апробирован на стандартном видео, снятом в помещении, на видеодатасетах, полученных с официального сайта, и имитатора Duckietown. Лучший результат фреймворк OpenVSLAM показал на наборе фотографий, полученных с имитатора Duckietown.

В дальнейшей работе планируется адаптировать OpenVSLAM для работы на роботах проекта в режиме реального времени.

Список литературы

1. **Лях Т. В., Зюбин В. Е., Гаранина Н. О.** Автоматизированная верификация алгоритмов управления сложными технологическими объектами на программных имитаторах // Вестник НГУ. Серия: Информационные технологии, 2018. Т. 16, № 4. С. 85–94.
2. **Малышев А. Г., Полыгалов А. С., Алямкин С. А.** Автоматическое тегирование изображений одежды // Вестник НГУ. Серия: Информационные технологии. 2020. Т. 18, № 2. С. 54–61. DOI 10.25205/1818-7900-2020-18-2-54-61
3. **Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard.** Non-linear Constraint Network Optimization for Efficient Map Learning. *IEEE Transactions on Intelligent Transportation Systems*, 2009, vol. 10, no. 3, pp. 428–439.
4. **Raúl Mur-Artal, and Juan D. Tardós.** ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. In: ArXiv preprint arXiv:1610.06475, 2016.
5. **Jakob Engel, Prof. Vladlen Koltun, Prof. Daniel Cremers.** DSO: Direct Sparse Odometry. In: ArXiv preprint arXiv:1610.06475, 2017.

6. **Kin Leong Ho, Paul Newman.** Loop closure detection in SLAM by combining visual and spatial appearance. *Robotics Auton. Syst*., 2006, vol. 54, pp. 740–749.
7. **Baichuan Huang, Jun Zhao, Jingbin Liu**. A Survey of Simultaneous Localization and Mapping with an Envision in 6G Wireless Networks. In: arXiv:1909.05214v2, 2019.
8. **Michael Kaess, Ananth Ranganathan and Frank Dellaert**. iSAM: Incremental Smoothing and Mapping. *IEEE Transactions on Robotics (TRO)*, 2008, vol. 24, no. 6, pp. 1365–1378.
9. **Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard.** Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 2007, vol. 23, pp. 34–46.
10. **Hess W., Kohler D., Rapp H., and Andor D.** Real-Time Loop Closure in 2D LIDAR SLAM. In: Robotics and Automation (ICRA), IEEE International Conference, 2016, pp. 1271–1278.
11. **Tiar R., Lakrouf M., and Azouaoui O.** Fast ICP-SLAM for a Bi-Steerable Mobile Robot in Large Environments. In: IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2015, pp. 1–6.
12. **Tonghui Wang, Guoyun Lv, Shikai Wan, gHaili Li, Baicen Lu** SIFT Based Monocular SLAM with GPU Accelerated. CHINACOM Springer, 2018. DOI 10.1007/978-3-319-78139-6_2
13. **Kurt Konolige, Giorgio Grisetti, Rainer Kummerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent.** Efficient sparse pose adjustment for 2d mapping. In: IEEE / RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 22–29.
14. **Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and UweKlingauf**. A flexible and scalable slam system with full 3d motion estimation. In: IEEE International Symposium on Safety, Security, and Rescue Robotics, 2011, pp. 155–160.
15. **Luca Carlone, Rosario Aragues, Jos ´e A Castellanos, and Basilio Bona.** A linear approximation for graph-based simultaneous localization and mapping. *Robotics: Science and Systems*, 2012, no. 7, pp. 41–48.
16. **Steux B. and Hamzaoui O.** A slam algorithm in less than 200 lines c-language program. In: Proc. of the Control AutomationRobotics & Vision (ICARCV). Singapore, 2010, pp. 7–10.
17. **Baichuan Huang, Jun Zhao, Jingbin Liu** A Survey of Simultaneous Localization and Mapping. In: ArXiv:abs/1909.05214, 2019.

References

1. **Lyakh T. V., Zyubin V. E., Garanina N. O.** Automated verification of control algorithms for complex technological objects on software simulators. *Vestnik NSU. Series: Information Technology*, 2018, vol. 16, no. 4, pp. 85–94. (in Russ.)
2. **Malyshev A. G., Polygalov A. S., Alyamkin S. A.** Automatic tagging of clothing images. *Vestnik NSU. Series: Information Technology*, 2020, vol. 18, no. 2, pp. 54–61. DOI 10.25205/1818-7900-2020-18-2-54-61
3. **Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard.** Non-linear Constraint Network Optimization for Efficient Map Learning. *IEEE Transactions on Intelligent Transportation Systems*, 2009, vol. 10, no. 3, pp. 428–439.
4. **Raúl Mur-Artal, and Juan D. Tardós.** ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. In: ArXiv preprint arXiv:1610.06475, 2016.
5. **Jakob Engel, Prof. Vladlen Koltun, Prof. Daniel Cremers.** DSO: Direct Sparse Odometry. In: ArXiv preprint arXiv:1610.06475, 2017.
6. **Kin Leong Ho, Paul Newman.** Loop closure detection in SLAM by combining visual and spatial appearance. *Robotics Auton. Syst*., 2006, vol. 54, pp. 740–749.
7. **Baichuan Huang, Jun Zhao, Jingbin Liu**. A Survey of Simultaneous Localization and Mapping with an Envision in 6G Wireless Networks. In: arXiv:1909.05214v2, 2019.

8. **Michael Kaess, Ananth Ranganathan and Frank Dellaert**. iSAM: Incremental Smoothing and Mapping. *IEEE Transactions on Robotics (TRO)*, 2008, vol. 24, no. 6, pp. 1365–1378.
9. **Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard.** Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 2007, vol. 23, pp. 34–46.
10. **Hess W., Kohler D., Rapp H., and Andor D.** Real-Time Loop Closure in 2D LIDAR SLAM. In: Robotics and Automation (ICRA), IEEE International Conference, 2016, pp. 1271–1278.
11. **Tiar R., Lakrouf M., and Azouaoui O.** Fast ICP-SLAM for a Bi-Steerable Mobile Robot in Large Environments. In: IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2015, pp. 1–6.
12. **Tonghui Wang, Guoyun Lv, Shikai Wan, gHaili Li, Baicen Lu** SIFT Based Monocular SLAM with GPU Accelerated. CHINACOM Springer, 2018. DOI 10.1007/978-3-319-78139-6_2
13. **Kurt Konolige, Giorgio Grisetti, Rainer Kummerle, Wolfram Burgard, Benson Limketkai, and Regis Vincent.** Efficient sparse pose adjustment for 2d mapping. In: IEEE / RSJ International Conference on Intelligent Robots and Systems, 2010, pp. 22–29.
14. **Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and UweKlingauf**. A flexible and scalable slam system with full 3d motion estimation. In: IEEE International Symposium on Safety, Security, and Rescue Robotics, 2011, pp. 155–160.
15. **Luca Carlone, Rosario Aragues, Jos ´e A Castellanos, and Basilio Bona.** A linear approximation for graph-based simultaneous localization and mapping. *Robotics: Science and Systems*, 2012, no. 7, pp. 41–48.
16. **Steux B. and Hamzaoui O.** A slam algorithm in less than 200 lines c-language program. In: Proc. of the Control AutomationRobotics & Vision (ICARCV). Singapore, 2010, pp. 7–10.
17. **Baichuan Huang, Jun Zhao, Jingbin Liu** A Survey of Simultaneous Localization and Mapping. In: ArXiv:abs/1909.05214, 2019.

Информация об авторах

Александра Денисовна Девятовская, студентка
Никита Евгеньевич Бирючков, студент
Татьяна Викторовна Лях, кандидат технических наук
Константин Владимирович Чайка, аспирант

Information about the Authors

Alexandra D. Devyatovskaya, Bachelor Student
Nikita E. Biryuchkov, Bachelor Student
Tatyana V. Liakh, Candidate of Sciences (Engineering)
Konstantin V. Chaika, Post-Graduate Student

Статья поступила в редакцию 11.09.2021;
одобрена после рецензирования 01.12.2021; *принята к публикации 01.12.2021*
The article was submitted 11.09.2021;
approved after reviewing 01.12.2021; *accepted for publication 01.12.2021*

Научная статья

УДК 004.912
DOI 10.25205/1818-7900-2021-19-4-50-66

Разработка чат-ботов для поддержки поиска по контенту веб-сайтов на основе тематических и жанровых характеристик

Владислав Дмитриевич Рублев [1]
Елена Анатольевна Сидорова [2]

[1] Новосибирский государственный университет
Новосибирск, Россия

[2] Институт систем информатики им. А. П. Ершова
Сибирского отделения Российской академии наук
Новосибирск, Россия

[1] v.rublev@g.nsu.ru, https://orcid.org/0000-0001-8236-0461
[2] lsidorova@iis.nsk.su, https://orcid.org/0000-0001-8731-3058

Аннотация

Рассматривается подход к созданию интеллектуальных помощников в виде чат-ботов, поддерживающих информационный поиск на основе предварительной жанровой и тематической кластеризации контента веб-сайтов. Решаются задачи поиска необходимой информации и обеспечения информационной поддержки пользователя, организации обратной связи для улучшения качества поиска. Особенностью подхода является использование жанровых моделей, разрабатываемых для заданного типа ресурса (образовательный, информационный и т. п.), на основе которых осуществляется жанровая структуризация контента конкретного сайта. Полученные жанровые структуры позволяют более точно определять границы тематических кластеров, относящиеся к теме поискового запроса пользователя. Для обеспечения обратной связи с пользователем разработан простой сценарий, позволяющий не просто уточнить запрос, но и неявно получить информацию о том, что именно не устроило пользователя в результирующей выдаче. Проведено экспериментальное исследование на платформе Telegram, полученные результаты сравнивались с поисковой системой Яндекс.

Ключевые слова

интеллектуальный помощник, поисковая система, информационный поиск, жанровая модель сайта, жанровая сегментация, тематическая кластеризация

Для цитирования

Рублев В. Д., Сидорова Е. А. Разработка чат-ботов для поддержки поиска по контенту веб-сайтов на основе тематических и жанровых характеристик // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 4. С. 50–66. DOI 10.25205/1818-7900-2021-19-4-50-66

© Рублев В. Д., Сидорова Е. А., 2021

Development of Chatbots to Support Web Site Content Search Based on Thematic and Genre Characteristics

Vladislav D. Rublev [1], **Elena A. Sidorova** [2]

[1] Novosibirsk State University
Novosibirsk, Russian Federation

[2] A. P. Ershov Institute of Informatics Systems
of the Siberian Branch of the Russian Academy of Sciences
Novosibirsk, Russian Federation

[1] v.rublev@g.nsu.ru, https://orcid.org/0000-0001-8236-0461
[2] lsidorova@iis.nsk.su, https://orcid.org/0000-0001-8731-3058

Abstract

The paper considers an approach to creating intelligent assistants in the form of chatbots that support information search based on preliminary genre and thematic clustering of website content. The tasks of finding the necessary information and providing information support to the user, organizing feedback to improve the quality of the search are being solved. A feature of the approach is the use of genre models developed for a given type of resource (educational, informational, etc.), on the basis of which genre structuring of the content of a particular site is carried out. The resulting genre structures allow you to more accurately determine the boundaries of thematic clusters related to the topic of the user's search query. To provide feedback to the user, a simple script has been developed that allows not only to clarify the request, but also to implicitly get information about what exactly did not suit the user in the resulting output. An experimental study was conducted on the Telegram platform, the results were compared with the Yandex search engine.

Keywords

intelligent assistant, search engine, information search, genre model of the site, genre segmentation, thematic clustering

For citation

Rublev V. D., Sidorova E. A. Development of Chatbots to Support Web Site Content Search Based on Thematic and Genre Characteristics. *Vestnik NSU. Series: Information Technologies*, 2021, vol. 19, no. 4, p. 50–66. (in Russ.) DOI 10.25205/1818-7900-2021-19-4-50-66

Введение

Бурное развитие информационных технологий, в частности развитие сети Интернет, породило большое количество электронной информации. Простота и доступность создания и распространения данных привели к тому, что появился огромный поток нужной и ненужной информации. В настоящее время развивается информационный поиск, так как из большого количества данных необходимо находить только ту информацию, в которой заинтересован пользователь. Одним из способов улучшения качества поиска является использование методов анализа контента интернет-источников на основе знаний.

Для поиска с учетом семантики запроса существуют специальные поисковые системы. Функционал данных продуктов варьируется, начиная от переранжирования результатов поиска других поисковых систем и до полнотекстового поиска информации в проиндексированных текстах по ключевым запросам пользователей с учетом морфологических особенностей, синтаксиса и семантики слов. Также системы могут выполнять поиск с учетом синонимов и родственных слов, которые сгруппированы в соответствии с их семантическим значением. Так, система «Нигма» [1, с. 70–73] представляет собой метапоисковую систему, обеспечивающую поиск текстовой информации с учетом смысла запроса, заданного на естественном языке. Данный сервис также поддерживает формирование списка документов, разделенного на несколько кластеров, чтобы пользователь мог уточнить, в каком кластере продолжить поиск, тем самым улучшив качество поиска. Также существуют поисковые системы, основанные на тематическом кластерном анализе, например «Carrot2» [2]. Данная система предлагает два специализированных алгоритма кластеризации: Lingo – алгоритм,

основанный на сингулярном разложении, и STC – метод суффиксных деревьев. Поиск в системе «Yippy» [1, с. 74–77] основан на IBM Watson – суперкомпьютере, у которого есть технологии обработки естественного языка и который способен анализировать сложные, неструктурированные данные и даже понимать профессиональный сленг. «AskNet» [1] состоит из двух подсистем, позволяющих как поиск информации в Интернете, так и поиск информации на компьютерах пользователей в корпоративной сети. Система отличается от других поисковых систем тем, что на запрос пользователя она выдает не только ссылки на документы и ресурсы, а еще и текстовую информацию, являющуюся ответом на вопрос пользователя. «Hakia» [3] содержит свою лингвистическую базу данных, в которой слова подразделяются на различные «смыслы», которые они передают. Она извлекает все возможные запросы, относящиеся к контенту (используя свою базу данных), и они становятся путями к исходному документу. И далее независимо ранжирует контент на основе дополнительного анализа предложений. Также для определения релевантности использует достоверность и возраст контента.

Несмотря на то что множество из приведенных систем учитывают семантические особенности текста, они не используют предварительную тематическую кластеризацию и не учитывают жанровые особенности индексируемых веб-ресурсов. Также следует отметить, что большинство таких систем ориентировано на работу с текстами на английском языке.

Методы информационного поиска сегодня используются в разных приложениях, и одно из самых популярных приложений – мессенджер со встроенным интеллектуальным помощником, или чат-ботом. Существует 6 основных методов построения чат-ботов [2]: методы, основанные на правилах, поиске, генеративный подход, ансамблевые методы, обоснованное обучение и интерактивное обучение. Системы, основанные на правилах [4], обучаются на основе предопределенной иерархии правил, которые определяют, как преобразовать вводимые пользователем данные в ответ или действие. Основанные на поиске методы [5] применяются сегодня в большинстве чат-ботов. Такие системы работают с помощью ориентированных графов и обучены предоставлять наилучший возможный ответ из своей базы данных заранее определенных ответов. Вместо того чтобы использовать заранее определенные ответы, разговорный чат-бот, использующий генеративные методы [6], получает большое количество данных (реальных диалогов) и обучается генерировать новый диалог, который на них похож. Современные разговорные чат-боты, которые могут говорить на любую тему, были созданы с помощью ансамблевых методов [7], которые в зависимости от контекста используют некоторую комбинацию подходов на основе правил, поиска и генеративного подхода. Интеллектуальный помощник с использованием обоснованного обучения [8], анализируя вводимый пользователем запрос, генерирует нейронную сеть, которая настраивается для этого конкретного запроса и задачи. Такой интеллектуальный помощник лучше «обоснован» благодаря его способности учиться и использовать представления знаний реального мира. Интерактивное машинное обучение – это алгоритмы и интеллектуальные структуры пользовательского интерфейса, которые упрощают машинное обучение благодаря взаимодействию с человеком. Эта разработка позволяет компьютерам учиться у людей, «взаимодействуя» с ними на естественном языке и «наблюдая» за ними.

В данной работе предлагается подход, который интегрирует разные методы поиска, основанные на кластерном анализе, использует жанровые модели и опирается на интернет-жанр сайта. И для апробации предложенного подхода реализуется чат-бот для поиска по сайтам образовательных учреждений г. Новосибирска.

Для анализа, построения жанровой модели сайта и проведения экспериментов был составлен корпус сайтов общеобразовательных учреждений размером 9 760 текстов, который содержит контент 208 сайтов.

[1] Официальный сайт вопросно-ответной поисковой системы AskNet. URL: http://asknet.ru/.
[2] Technical Approaches for Building Conversational AI. URL: https://www.topbots.com/building-conversational-ai/.

Модель представления сайта определенного жанра

В данном подходе поиск базируется на предварительной индексации сайта на основе его жанровой структуры.

Каждый сайт обладает чертами, которые определяются спецификой сферы деятельности и формируются благодаря сходству тематики, композиции и стиля, что соответствует классическому определению речевого жанра, сформулированному М. М. Бахтиным [9]. Жанр – это типовая модель построения речевого целого. Для каждого типа сайта может быть определена жанровая модель, представляющая его «типическую воспроизводимую жанровую форму». Каждая жанровая модель представляет собой общую структуру сайтов, принадлежащих этой модели. Таким образом, контент каждого сайта может быть разбит на жанровые фрагменты, представляющие некоторые аспекты содержания. На основе анализа корпуса текстов разработана жанровая модель сайта образовательного учреждения, верхний уровень которой представлен в табл. 1.

Таблица 1

Спектр жанров фрагментов текста в зависимости от жанра сайта

Table 1

The range of genres of text fragments depending on the genre of the site

	Жанр сайта: образовательное учреждение	
	высшее	среднее
Жанр фрагмента	Описание научного учреждения	Прием в школу
	Описание факультетов	Родителям
	Описание кампуса	Учителям
	Поступающим	Итоговое сочинение
	Обучающимся	ГИА
	Выпускникам	
	Сотрудникам	
	Аспирантам	
	Новостная лента	
	Комментарий	
	Описание мероприятия	

Контент (содержательная часть) сайта представляет собой последовательность текстовых блоков. Для определения жанра этих блоков используется набор жанровых маркеров и язык маркеров, который позволяет указать термины, их комбинацию или перечисление. Для описания жанровой модели используется язык, предложенный в работе [10], который позволяет на основе жанровых маркеров составлять описание аспектов содержания любого жанрового блока. Жанровая модель содержит наборы маркеров, каждый набор описывает некоторый аспект содержания, для каждого жанра описываются типичные аспекты содержания и указывается, в какой части html-разметки их искать.

Шаблон для описания аспектов содержания:

Аспект содержания: [“Маркер 1”][“Маркер 2”][“Маркер 3”][“Маркер 4”, “Маркер 5”].

Шаблон для описания жанров:

“**Идентификатор жанра**”: [<Аспект содержания 1, тег>] [<Аспект содержания 2, тег > <Аспект содержания 3, тег >].

Примеры аспектов содержания:

_поступающим*: ["поступать"]["абитуриент"]["прием"]["приемная комиссия"]["приемная кампания"]["правило приема"]["рейтинговый список"]["стоимость обучения"]*

_уровниОбразов*: ["бакалавриат"]["специалитет"]["магистратура"]["аспирантура"]*

Пример описания жанра страницы:

*"**поступающим**":* [*<_поступающим, text>*][*<_поступающим, all><_уровниОбразов, all>*].

Разработанная жанровая модель используется для выделения жанровых фрагментов в тексте.

Тематическая кластеризация текстов

После сегментации текстового контента сайта осуществляется его тематическая кластеризация – выделение тематических кластеров, по которым в дальнейшем будет осуществляться поиск.

Кластеризация (кластерный анализ) – это задача группирования множества объектов в группы, называемые кластерами, чтобы внутри каждой группы оказались «похожие» элементы, существенно отличающиеся от элементов других групп. Предметом данного исследования являются методы кластеризации текстов.

Для кластеризации контента и анализа запроса пользователя необходим словарь, поэтому в системе извлечения предметной лексики KLAN [11] был создан словарь терминов. Эта система поддерживает основные этапы анализа текста: синтаксический, семантический и морфологический. Словарь, созданный на основе корпуса сайтов общеобразовательных учреждений, содержит однословные и многословные термины, его размер 17 500 терминов, для которых собрана статистика.

По способу организации кластеров алгоритмы кластеризации можно классифицировать на плоские и иерархические. Был использован алгоритм k-средних [12], вследствие того что обладает высокой скоростью обучения и агломеративной иерархической кластеризации [13], потому что позволяет хорошо интерпретировать результат.

Каждый документ представлен вектором в пространстве R^n, где n – размерность словаря. Для получения такого представления используется статистическая мера tf-idf [14], которая показывает частоту встречаемости термина в документе. Расстояние между векторами можно вычислять по различным метрикам, были опробованы соответственно косинусное сходство и Евклидова метрика:

$$\rho(a,b) = \frac{\sum_{k=1}^{n} a_k \times b_k}{\sqrt{\sum_{k=1}^{n} a_k^2} \times \sqrt{\sum_{k=1}^{n} b_k^2}},$$

$$\rho(a,b) = \sqrt{\sum_{k=1}^{n} (a_k - b_k)^2}.$$

Косинусное сходство статистически хуже работает на собранном корпусе текстов, поэтому в данном исследовании используется Евклидова метрика.

Метод k-средних – это итеративный алгоритм (рис. 1), который основан на минимизации суммарного квадратичного отклонения точек кластеров от центров этих кластеров.

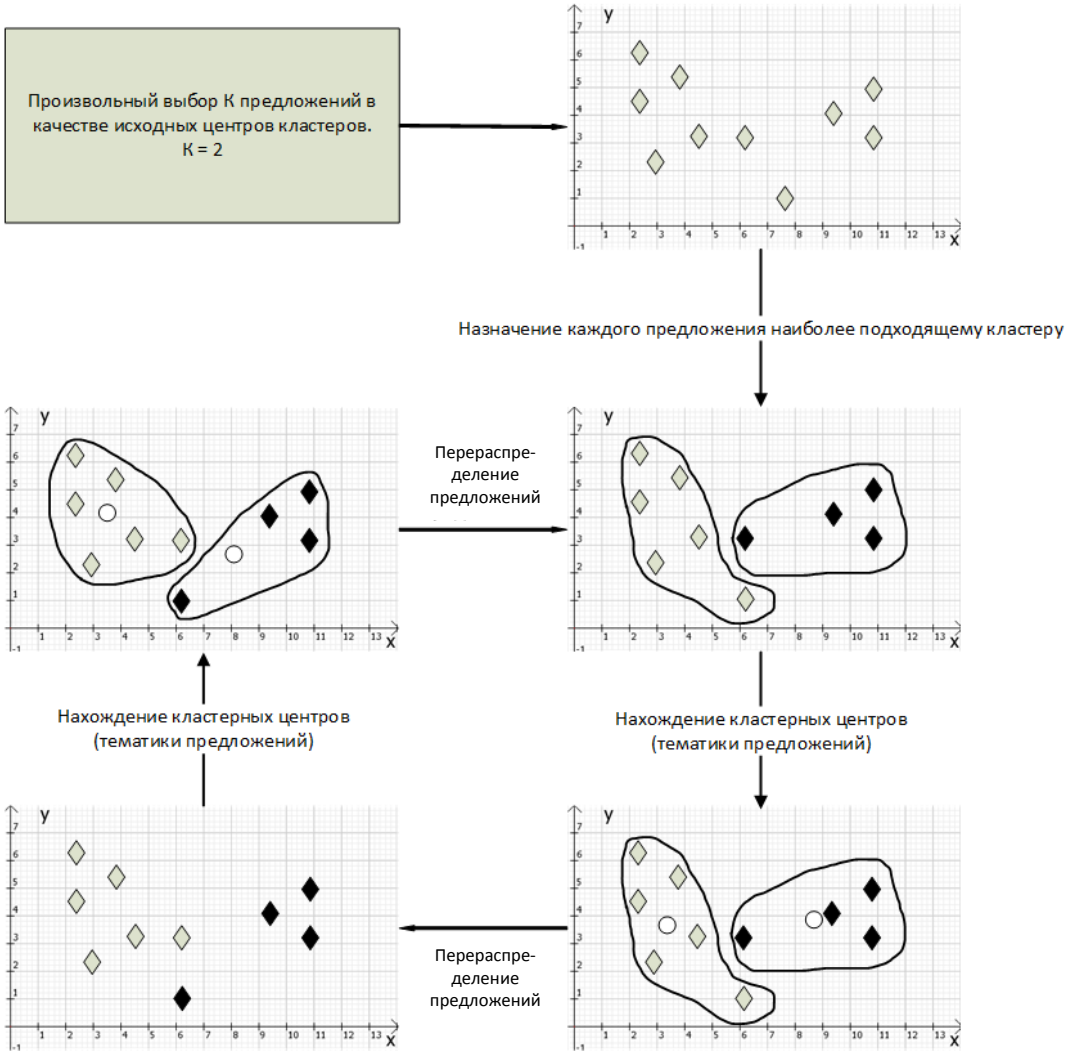


Рис. 1. Метод k-средних
Fig. 1. K-means clustering

Центроид (центр кластера) – вектор, элементы которого представляют средние значения, вычисленные по всем предложениям (представленным в векторном виде) из кластера. Из множества входных данных выбирается *k* предложений, которые будут начальными центрами кластеров (каждый кластер соответствует своей теме), для каждого предложения определяется ближайший центр кластера и на каждой итерации перевычисляется центроид каждого кластера (S_j), с учетом координат новых предложений:

$$\mu_j = \frac{1}{|S_j|} \sum_{x^{(j)} \in S_j} x^{(j)}.$$

Далее предложения снова разбиваются на кластеры в соответствии с тем, какой из новых центроидов оказался ближе. Алгоритм выполняется до тех пор, пока на каком-то шаге не произойдет изменения внутрикластерного расстояния или не будет достигнут порог по коли-

честву итераций. Таким образом, центроид является векторным представлением тематики предложений, входящих в кластер.

Агломеративные алгоритмы работают по принципу «снизу вверх» (рис. 2): в начале работы помещают каждое предложение в отдельный кластер, а затем происходит объединение двух ближайших во все более крупные, пока все кластеры не сольются в один или не будет найдено необходимое число кластеров. Таким образом строится дерево разбиений.

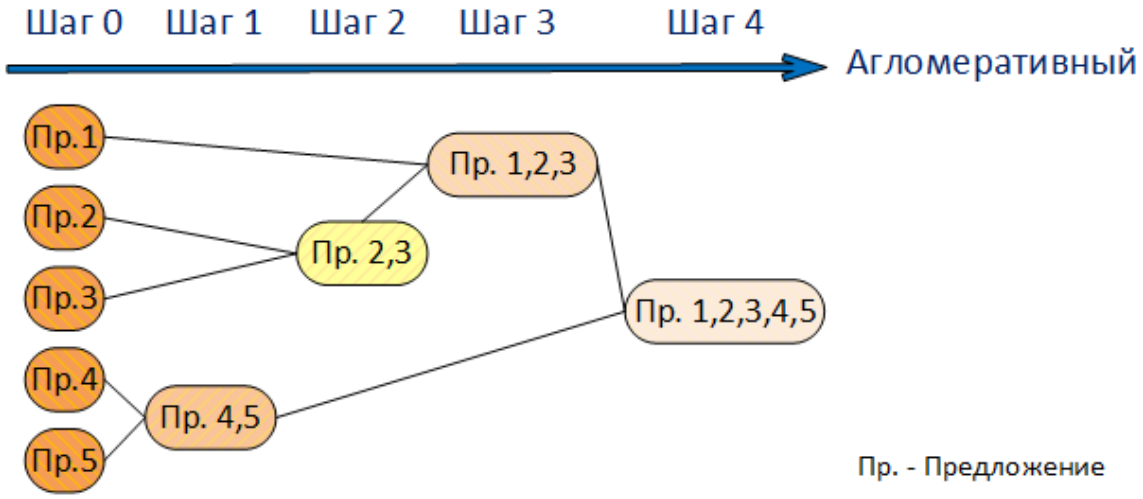


Рис. 2. Иерархическая кластеризация
Fig. 2. Hierarchical clustering

Для вычисления расстояний между кластерами чаще всего пользуются следующими методами:

1) метод одиночной связи – расстояние между двумя кластерами определяется как минимальное расстояние между элементами этих кластеров:

$$\min\{\rho(a,b):\ a\in A,\ b\in B\};$$

2) метод полной связи – расстояние между двумя кластерами определяется как максимальное расстояние между элементами этих кластеров:

$$\max\{\rho(a,b):\ a\in A,\ b\in B\};$$

3) метод средней связи – расстояние между кластерами определяется как среднее расстояние между элементами этих кластеров:

$$\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B}\rho(a,b);$$

4) центроидный метод – расстояние между кластерами полагается равным расстоянию между центроидами кластеров:

$$\|\mu_A-\mu_B\|.$$

В работе используется метод средней связи, потому что он лучше работает на собранном корпусе текстов.

Существует большое количество мер для оценки качества кластеризации и проведено множество их сравнений [15–17]. Одной из таких мер является “Silhouette score” [18], которая статистически значимо показывает результаты лучше, чем остальные меры. Вычисляется отдельно для каждого объекта по формуле

$$s = \frac{b - a}{\max\{a, b\}},$$

где a – среднее расстояние от выбранного объекта до объектов из его кластера, b – среднее расстояние от выбранного объекта до объектов ближайшего кластера (не содержащего выбранный объект). Для оценки качества кластеризации вычисляется среднее значение “Silhouette score” по всем объектам. С помощью этой меры была проведена оценка методов кластеризации, и результаты получились неоднозначные, так как исход сильно зависел от входных данных.

Поиск по сайту

Поиск информации, необходимой пользователю, осуществляется на основе предварительного индексирования, построенного с помощью жанрового анализа и тематической кластеризации.

На рис. 3 представлена архитектура поисковой системы, которая включает два основных модуля: подсистема предварительной обработки и поисковая подсистема.

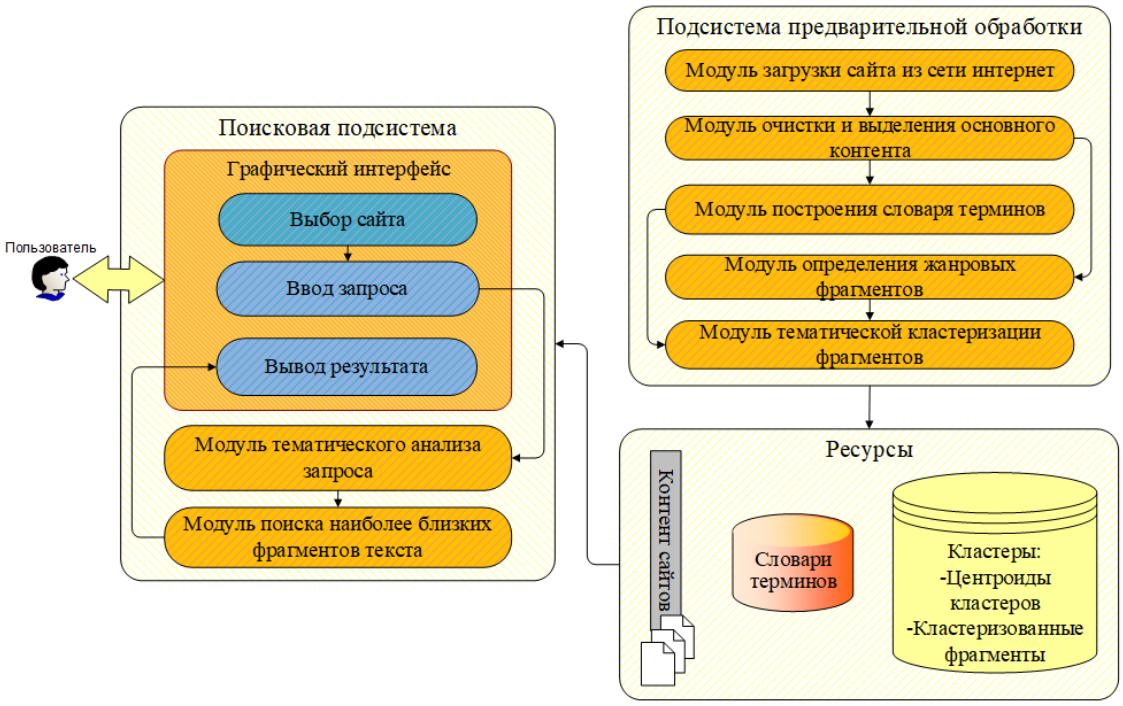


Рис. 3. Архитектура поисковой системы
Fig. 3. Search Engine architecture

Предварительная обработка сайта делается заранее и состоит из следующих этапов:

1) загрузка веб-сайта из сети Интернет – рекурсивно находятся все ссылки на страницы сайта и скачиваютсяж

2) очистка и выделение основного контента – на всех страницах сайта находятся и удаляются данные, которые не являются основным контентом (футер, выпадающее меню и т. д.), также удаляются все html-теги, кроме разрешенных: h1, h2, h3, h4, h5, h6, a, b, ul, ol, li;

3) генерация словаря сайта – полученные файлы с основным контентом сайта загружаются в словарную систему, где в автоматическом режиме находятся все термины и для них собирается статистика;

4) предварительная обработка текста – производится лемматизация, удаление стоп-слов и слов, которые встречаются слишком редко;

5) жанровая классификация – на базе разработанной жанровой модели находится количество жанров на странице, если их несколько, то страница разбивается на фрагменты от заголовка до заголовка, и определяются жанры этих фрагментов. Фрагменты одного жанра, идущие подряд объединяются в один, а фрагменты, жанр которых определить не удалось, выбрасываются из рассмотрения;

6) тематическая кластеризация – выбирается один из доступных алгоритмов кластеризации, текст представляется в векторном виде с помощью меры tf-idf, определяется количество кластеров, на которое будет разбиваться текст в зависимости от его жанра и объема. Производится кластеризация выбранным методом на заданное количество кластеров;

7) сохранение результатов – сохраняются результат кластеризации, текстовая коллекция и словарь сайта.

В результате индексации сайт Новосибирского государственного университета содержит 153 страницы, 998 кластеров и 4 661 предложение. Сайт Московского государственного университета имени М. В. Ломоносова состоит из 123 страниц, 858 кластеров и 2 512 предложений.

Для поиска ответов на запросы пользователя необходимо:

1) произвести анализ запроса пользователя – при помощи словаря сайта строится вектор запроса, состоящий из нулей и единиц, где единица – слово содержится в запросе, нуль – иначе;

2) найти и выдать пользователю фрагменты, наиболее релевантные запросу, – с помощью Евклидовой метрики находятся расстояния от вектора запроса до центров кластеров, берется n ближайших кластеров и определяется, каким текстам из коллекции они соответствуют.

Информационный поиск, представленный выше, используется для создания интеллектуальных помощников, реализованных в виде чат-ботов.

Информационный чат-бот

Информационный чат-бот – программное обеспечение, которое имитирует диалог с пользователем на естественном языке и используется для оперативного поиска информации по заданной тематике. Чат-боты позволяют общаться с помощью сообщений на сайтах, в мессенджерах или мобильных приложениях.

Для повышения качества информационной поддержки пользователя разработан сценарий взаимодействия с чат-ботом (рис. 4).

Сценарий состоит из следующих этапов. Пользователь задает поисковый запрос чат-боту. Чат-бот отправляет результат поиска и предлагает пользователю уточнить запрос (если пользователя не устраивает результат) или начать новый поиск. Если пользователь хочет уточнить запрос, то чат-бот предлагает ему один из трех параметров поиска (поиск с исключением предыдущих результатов, расширение или сужение выборки поиска). Вместе с параметром поиска пользователю предлагается уточнить запрос (написать новый или исправить предыдущий). Далее чат-бот возвращается в начало сценария и отвечает пользователю на вопрос с учетом выбранного параметра поиска. Благодаря цикличности сценария удается добиться итеративного улучшения качества поиска.

Для апробации подхода в качестве интерфейса был выбран мессенджер Telegram в силу своей популярности, простоты и бесплатного доступа к API.

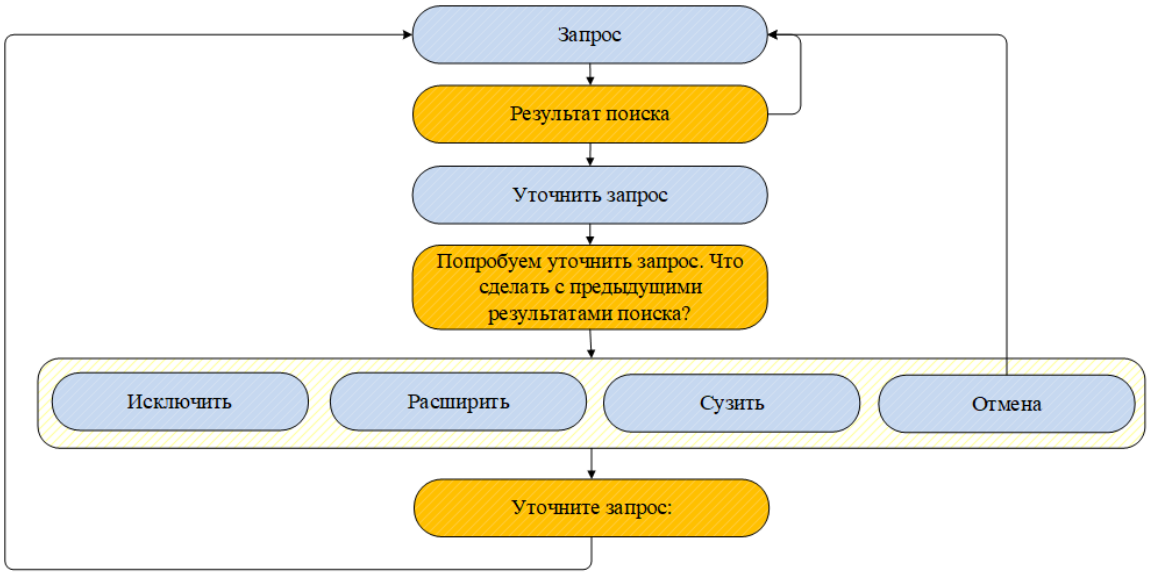


Рис. 4. Сценарий взаимодействия пользователя с чат-ботом
Fig. 4. Scenario of user interaction with a chatbot

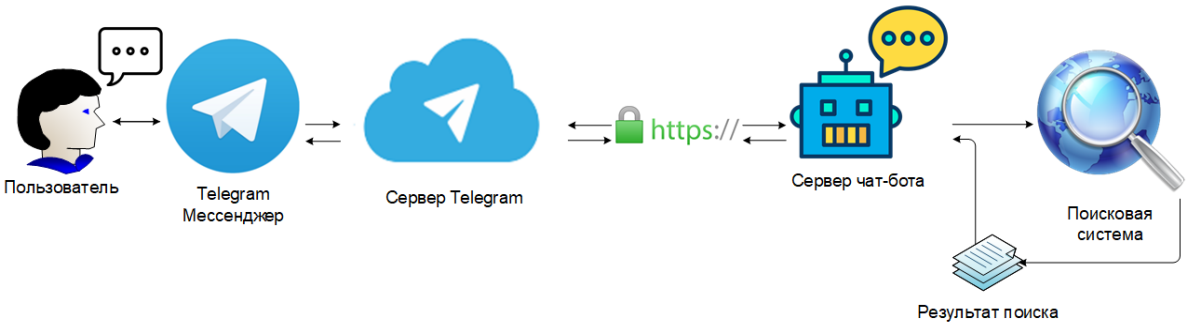


Рис. 5. Схема работы интеллектуального помощника на базе мессенджера Telegram
Fig. 5. The scheme of the intelligent assistant based on the Telegram messenger

Боты – специальные аккаунты в Telegram, созданные для того, чтобы автоматически обрабатывать и отправлять сообщения. Пользователи могут взаимодействовать с ботами при помощи сообщений, отправляемых через обычные или групповые чаты. Для реализации чат-ботов Telegram предоставляет Telegram Bot API [3]. Логика бота контролируется при помощи HTTPS запросов к этому API. Сообщения и запросы, отправленные пользователями чат-боту, передаются программному обеспечению, работающему на сервере (рис. 5). Всё шифрование и связь с Telegram API промежуточный сервер Telegram обрабатывает самостоятельно. Все запросы к этому серверу должны быть следующего вида:

https://api.telegram.org/bot<token>/METHOD_NAME?Param1=<p1>&ParamN=<pn>,

[3] Telegram Bot API. URL: https://core.telegram.org/bots/api/.

где

<token> – уникальный идентификатор чат-бота,

METHOD_NAME – название метода, который необходимо вызвать у бота,

Param1, ParamN – набор параметров вызываемого метода (может отсутствовать),

p1, pn – значения параметров.

Ответ на все запросы приходит в виде json-объекта, в котором присутствует булево поле *ok*, которое истинно в случае успешного запроса, и результат его выполнения можно увидеть в поле «result».


```
{
"ok":true,
"result":
{
"message_id":594,
"from": {"id":457981543,
          "is bot":true,
          first_name":"RublevBot",
          "username":"RubleffBot",
          "language_code": "ru"},
"chat":{"id":254438520,
         "first_name":"Vladislav",
         "last name":"Rublev",
         "username":"spac1k",
         "type": "private"},
"date":1617858474,
"text":"Привет"
}
}
```


Если сделать запрос с методом *send_message* (отправить сообщение), то поле «result» будет содержать информацию о номере сообщения, отправителе, получателе, дате запроса, а также текст отправленного сообщения.

В случае ошибки (*ok: false*) в поле *description* будет указана причина ошибки, а в поле *error_code* указан код ошибки:


```
{
"ok":false,
"error_code":400,
"description":"Bad Request: message text is empty"
}
```


Разработанный поисковый чат-бот на запрос пользователя производит информационный поиск и отправляет результат в виде трех сообщений с фрагментами текста и гиперссылками на источник (рис. 6). Если пользователя не устраивают результаты поиска, то он может уточнить запрос (нажав кнопку «Уточнить запрос»). Далее пользователю необходимо выбрать один из трех параметров поиска: «Исключить предыдущий результат», «Расширить выборку», «Сузить выборку (искать в найденном)».

После выбора параметра пользователь уточняет сам запрос (рис. 7) и в ответ получает результат повторного поиска с параметром.

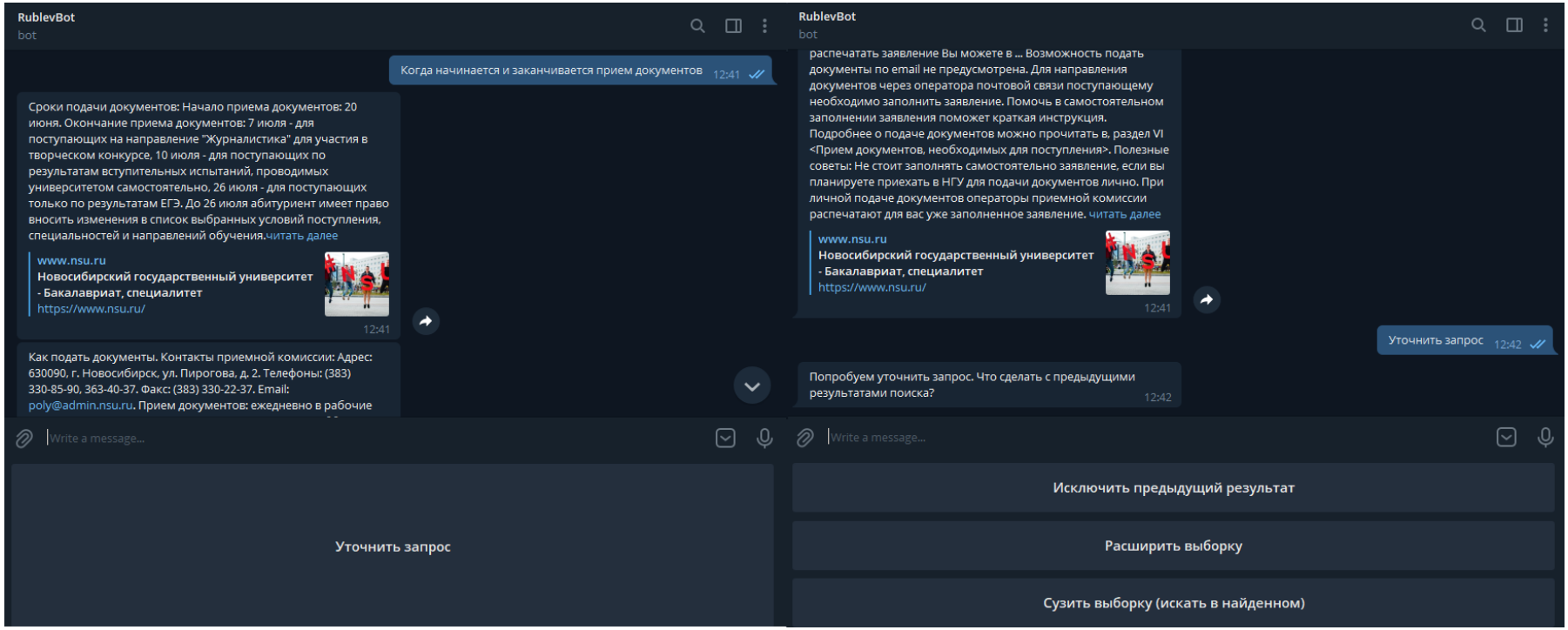


Рис. 6. Пример работы чат-бота после запроса пользователя
Fig. 6. An example of a chatbot working after a user request

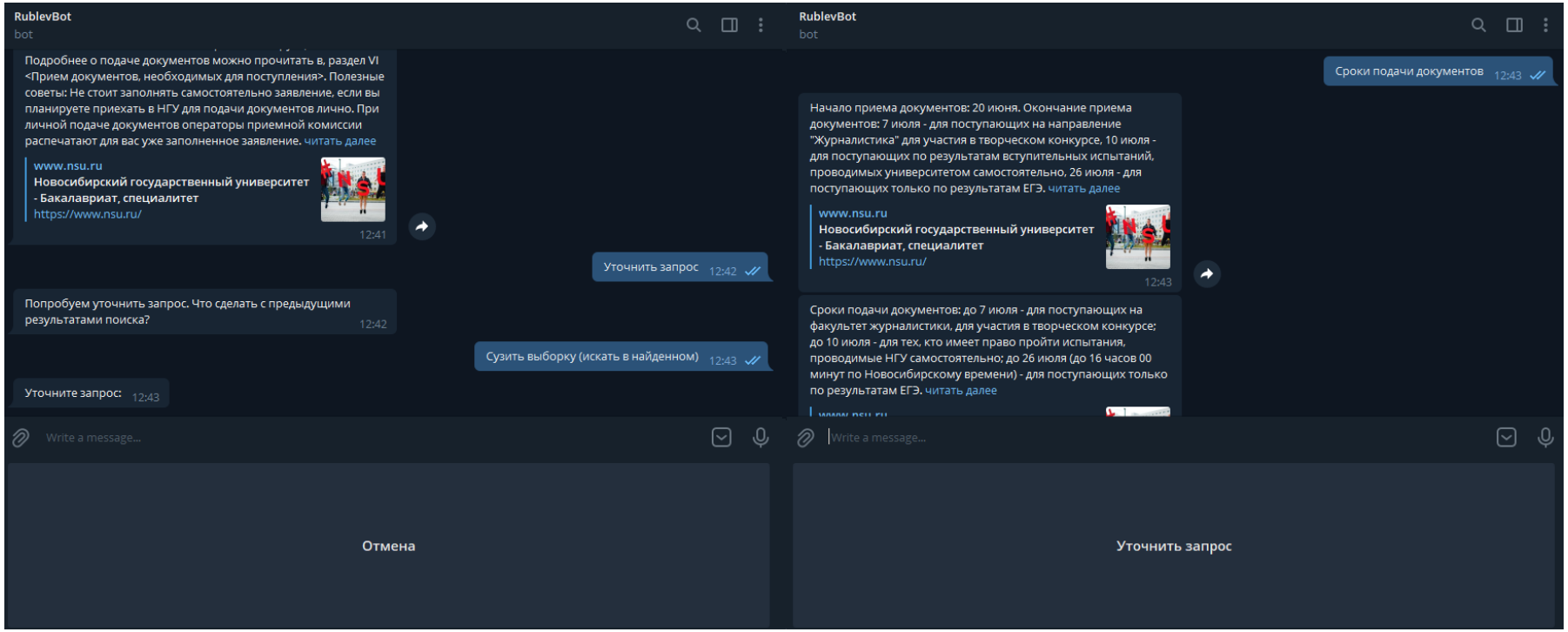


Рис. 7. Пример работы чат-бота во время уточнения запроса
Fig. 7. An example of a chatbot working during request refinement

Экспериментальное исследование

Качество работы чат-бота при поиске оценивалось в сравнении с поисковой системой компании «Яндекс». Поиск осуществлялся по сайту Новосибирского государственного университета. В качестве поисковых запросов были взяты часто задаваемые вопросы (19 вопросов) с сайта Московского государственного технического университета им. Н. Э. Баумана [4].

Пусть rel – множество всех релевантных документов, det – множество найденных документов. Для оценки качества использовались стандартные меры для оценки информационного поиска [19] – полнота, точность, F-мера и оригинальность.

1. Полнота определяет, насколько хорошо система находит нужные пользователю документы, представляет собой отношение найденных релевантных документов к общему количеству релевантных документов:

$$R = \frac{\mathrm{rel} \cap \mathrm{det}}{\mathrm{rel}}.$$

2. Точность определяет способность системы выдавать пользователю только релевантные документы, вычисляется как отношение найденных релевантных документов к общему количеству найденных документов:

$$P = \frac{\mathrm{rel} \cap \mathrm{det}}{\mathrm{det}}.$$

3. F-мера представляет собой взвешенное гармоническое среднее полноты и точности, позволяет придать различный вес полноте и точности, если необходимо отдать приоритет одной из этих метрик:

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}}, \ \alpha \in [0,1].$$

При α =1/2 получается сбалансированная F-мера и вычисляется она по следующей формуле:

$$F = \frac{2PR}{P+R}.$$

4. Оригинальность определяет количество различных результатов поиска (документы с разным контентом).

Средние значения этих показателей были вычислены по всем вопросам и первым 3-м результатам выдачи поисковых систем. Оценка этих значений проводилась вручную [20]. Полученные результаты отражены в табл. 2.

Таблица 2

Оценка качества поиска, %

Table 2

Evaluation of search quality, %

	Полнота	Точность	F-мера	Оригинальность
«Яндекс»	82,16	91,83	86,73	71,84
Чат-бот	83,37	92,75	87,81	80,92

[4] Официальный сайт МГТУ им. Н. Э. Баумана. Ответы на часто задаваемые вопросы. URL: https://bmstu.ru/abitur/general/qanda/.

При сравнении результатов поиска, полученных разработанным интеллектуальным помощником и системой компании «Яндекс», можно сделать вывод, что помощник немного превосходит систему «Яндекс» для поиска по конкретным сайтам. При этом обе поисковые системы предоставляют возможность найти ответы на основные вопросы пользователя, но не гарантируют получения всей необходимой информации. Но в отличие от поисковика «Яндекс» чат-боту благодаря «диалогу» с пользователем удается добиться итеративного улучшения качества поиска.

Заключение

В работе представлен подход к созданию информационных помощников для поиска в контенте веб-сайтов на основе жанровой модели и предварительной тематической кластеризации текстового контента. Предлагаемый подход позволяет найти необходимую информацию, организовать обратную связь с пользователем и обеспечить итеративное улучшение результатов поиска. Особенностью подхода является использование жанровой информации о сайте, на основе которой осуществляется жанровая сегментация, которая позволяет более точно структурировать контент. Над множеством жанровых сегментов осуществляется тематическая кластеризация, целью которой является выделение и группировка фрагментов текста, относящихся к одной области. Дальнейший поиск осуществляется стандартными методами. Результаты экспериментов показывают, что добавление таких особенностей улучшает результаты поиска, и, следовательно, предложенный подход можно применять для улучшения качества поиска, например, в метапоисковых системах (смешивания и переранжирования результатов поиска других поисковых систем).

Разработанная система хорошо масштабируется, в частности созданные ресурсы применимы для произвольных образовательных сайтов, а для того чтобы настроить систему на другие типы сайтов, достаточно написать новую жанровую модель и проиндексировать заданные сайты нового типа (для этого в системе разработан независимый модуль индексации).

Список литературы

1. **Кутовенко А.** Профессиональный поиск в интернете. СПб.: Питер, 2011. 252 с.
2. **Stanislaw Osinski, Dawid Weiss**. Carrot2 Project. In: Carrot2 – Open Source Search Results Clustering Engine. URL: http://project.carrot2.org/.
3. **Radhakrishnan Arun.** Hakia’s Semantic Search : The Answer to Poor Keyword Based Relevancy. *Search Engine Journal*. URL: https://www.searchenginejournal.com/hakias-semantic-search-the-answer-to-poor-keyword-based-relevancy/5246/.
4. **Nimavat K., Champaneria T.** Chatbots: an overview of types, architecture, tools and future possibilities. *Int. J. Sci. Res. Dev.*, 2017, pp. 1019–1024.
5. **Wu Y., Wu W., Xing C., Zhou M., Li Z.** Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots. In: ArXiv:11612.01627, 2017.
6. **Kapočiūtė-Dzikienė J.** A Domain-Specific Generative Chatbot Trained from Little Data. *Applied Sciences*, 2020, vol. 10, p. 2221.
7. **Heriberto Cuayáhuitl, Donghyeon Lee, Seonghan Ryu, Yongjin Cho, Sungja Choi, Satish Indurthi, Seunghak Yu, Hyungtak Choi, Inchul Hwang, Jihie Kim.** Ensemble-based deep reinforcement learning for chatbots. *Neurocomputing*, 2019, vol. 366, pp. 118–130.
8. **Kim Sihyung, Kwon Oh-Woog, Kim Harksoo.** Knowledge-Grounded Chatbot Based on Dual Wasserstein Generative Adversarial Networks with Effective Attention Mechanisms. *Applied Sciences*, 2020, vol. 10.
9. **Бахтин М. М.** Проблема речевых жанров // Эстетика словесного творчества. М.: Искусство, 1986. С. 250–296.

10. **Кононенко И. С., Сидорова Е. А.** Жанровые аспекты классификации веб-сайтов // Программная инженерия. 2015. № 8. С. 32–40.
11. **Сидорова Е. А.** Комплексный подход к исследованию лексических характеристик текста // Вестник СибГУТИ. 2019. № 3. С. 80–88.
12. **MacQueen J. B.** Some Methods for classification and Analysis of Multivariate Observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, 1967, pp. 281–297.
13. **Guo J., Hartung S., Komusiewicz C. et al.** Exact algorithms and experiments for hierarchical tree clustering. In: Proceedings of the TwentyFourth AAAI Conference on Artificial Intelligence (AAAI-10), 2010, pp. 1–6.
14. **Manwar A., Mahalle H., Chinchkhede K. et al.** A vector space model for information retrieval: a matlab approach. *Indian Journal of Computer Science and Engineering*, 2012, no. 3, pp. 222–230.
15. **Erendira Rendon, Itzel Abundez, Alejandra Arizmendi et al.** Internal versus external cluster validation indexes. *International Journal of Computers and Communications*, 2011, vol. 5, no. 1, pp. 27–34.
16. **Yanchi Liu, Zhongmou Li, Hui Xiong et al.** Understanding of internal clustering validation measures. In: IEEE International Conference on Data Mining, 2010, pp. 911–916. DOI 10.1109/tsmcb.2012.2220543
17. **Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza et al.** An extensive comparative study of cluster validity indices. *Pattern Recognition*, 2013, vol. 46, no. 1, pp. 243–256. DOI 10.1016/j.patcog.2012.07.021
18. **Rousseeuw Peter J.** Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 1987, vol. 20, pp. 53–65. DOI 10.1016/0377-0427(87)90125-7
19. **Sirotkin P. F.** On Search Engine Evaluation Metrics. In: ArXiv:abs/1302.2318, 2013, pp. 24–26.
20. **Белозеров В. Н.** Эффективность систем Яндекс и Гугл для поиска учебного материала // Вестник МГУКИ. 2015. № 1. С. 208–213.

References

1. **Kutovenko A.** Professional internet search. St. Petersbug, Peter, 2011, 252 p. (in Russ.)
2. **Stanislaw Osinski, Dawid Weiss**. Carrot2 Project. In: Carrot2 – Open Source Search Results Clustering Engine. URL: http://project.carrot2.org/.
3. **Radhakrishnan Arun.** Hakia's Semantic Search : The Answer to Poor Keyword Based Relevancy. *Search Engine Journal*. URL: https://www.searchenginejournal.com/hakias-semantic-search-the-answer-to-poor-keyword-based-relevancy/5246/.
4. **Nimavat K., Champaneria T.** Chatbots: an overview of types, architecture, tools and future possibilities. *Int. J. Sci. Res. Dev.*, 2017, pp. 1019–1024.
5. **Wu Y., Wu W., Xing C., Zhou M., Li Z.** Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots. In: ArXiv:11612.01627, 2017.
6. **Kapočiūtė-Dzikienė J.** A Domain-Specific Generative Chatbot Trained from Little Data. *Applied Sciences*, 2020, vol. 10, p. 2221.
7. **Heriberto Cuayáhuitl, Donghyeon Lee, Seonghan Ryu, Yongjin Cho, Sungja Choi, Satish Indurthi, Seunghak Yu, Hyungtak Choi, Inchul Hwang, Jihie Kim.** Ensemble-based deep reinforcement learning for chatbots. *Neurocomputing*, 2019, vol. 366, pp. 118–130.
8. **Kim Sihyung, Kwon Oh-Woog, Kim Harksoo.** Knowledge-Grounded Chatbot Based on Dual Wasserstein Generative Adversarial Networks with Effective Attention Mechanisms. *Applied Sciences*, 2020, vol. 10.

9. **Bahtin M. M.** The problem of speech genres. In: Estetika slovesnogo tvorchestva [Aesthetics of Verbal Creation]. Moscow, Iskusstvo, 1986, pp. 250–296. (in Russ.)
10. **Kononenko I. S., Sidorova E. A.** Genre aspects of website classification. Software Engineering, 2015, no. 8, pp. 32–40.
11. **Sidorova E. A.** A comprehensive approach to the study of lexical characteristics of the text. *Vestnik SibSUTI*, 2019, no. 3, pp. 80–88.
12. **MacQueen J. B.** Some Methods for classification and Analysis of Multivariate Observations. In: Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, 1967, pp. 281–297.
13. **Guo J., Hartung S., Komusiewicz C. et al.** Exact algorithms and experiments for hierarchical tree clustering. In: Proceedings of the TwentyFourth AAAI Conference on Artificial Intelligence (AAAI-10), 2010, pp. 1–6.
14. **Manwar A., Mahalle H., Chinchkhede K. et al.** A vector space model for information retrieval: a matlab approach. *Indian Journal of Computer Science and Engineering*, 2012, no. 3, pp. 222–230.
15. **Erendira Rendon, Itzel Abundez, Alejandra Arizmendi et al.** Internal versus external cluster validation indexes. *International Journal of Computers and Communications*, 2011, vol. 5, no. 1, pp. 27–34.
16. **Yanchi Liu, Zhongmou Li, Hui Xiong et al.** Understanding of internal clustering validation measures. In: IEEE International Conference on Data Mining, 2010, pp. 911–916. DOI 10.1109/tsmcb.2012.2220543
17. **Olatz Arbelaitz, Ibai Gurrutxaga, Javier Muguerza et al.** An extensive comparative study of cluster validity indices. *Pattern Recognition*, 2013, vol. 46, no. 1, pp. 243–256. DOI 10.1016/j.patcog.2012.07.021
18. **Rousseeuw Peter J.** Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 1987, vol. 20, pp. 53–65. DOI 10.1016/0377-0427(87)90125-7
19. **Sirotkin P. F.** On Search Engine Evaluation Metrics. In: ArXiv:abs/1302.2318, 2013, pp. 24–26.
20. **Belozerov V. N.** The efficiency of the engines Yandex and Google to search for educational material. *Vestnik MGIK*, 2015. no. 1, pp. 208–213.

Информация об авторах

Владислав Дмитриевич Рублев, студент магистратуры
Елена Анатольевна Сидорова, кандидат физико-математических наук, старший научный сотрудник

Information about the Authors

Vladislav D. Rublev, Master's Student
Elena A. Sidorova, Candidate of Sciences (Physics and Mathematics), Senior Researcher

Статья поступила в редакцию 01.08.2021;
одобрена после рецензирования 01.10.2021; *принята к публикации 01.12.2021*
The article was submitted 01.08.2021;
approved after reviewing 01.10.2021; *accepted for publication 01.12.2021*

Обзорная статья

УДК 539.3
DOI 10.25205/1818-7900-2021-19-4-67-84

Развитие САПР для решения задач механики с использованием МКЭ

Аркадий Николаевич Соловьев [1]
Ростислав Викторович Киричевский [2]

[1] Донской государственный технический университет
Ростов-на-Дону, Россия

[2] Луганский государственный университет имени Владимира Даля
Луганск, Украина

[1] solovievarc@gmail.com, https://orcid.org/0000-0001-8465-5554
[2] rost71@mail.ru, https://orcid.org/0000-0002-4557-8469

Аннотация

Представленный обзор существующих отечественных и зарубежных систем автоматизированного проектирования (САПР) показал, что применение в их структуре метода конечных элементов (МКЭ) остается вполне актуальным перед другими методами в скорости вычислений, достаточной точности и программной реализации в структуре САПР. Для решения задач механики эластомеров и композитов на их основе более полно представлен обзор разработанного отечественного программного комплекса МИРЕЛА+. Этот комплекс решает многие задачи механики деформации твердого тела: имеет специальную направленность на решение задач диссипативного разогрева, параметров механики разрушения массивных эластомерных элементов конструкций и тонкослойных резинометаллических элементов с трещинами с изменяющимися физико-механическими и теплофизическими параметрами в условиях циклического деформирования. Не многие из отечественных программ смогут это реализовать, а лицензии зарубежных программ стоят десятки тысяч долларов.

В работе также приведены способы дискретизации конечными элементами и алгоритм построения системы разрешающих уравнений, используемый в МИРЕЛА+, выражения для определения компонентов тензора преобразования координат, тензора деформаций в центрах КЭ в декартовой системе координат. Приведены примеры визуального представления в трехмерном изображении с помощью цветной картины, где каждому оттенку или цвету соответствует определенный диапазон числовых значений функции.

Ключевые слова

метод конечных элементов, программный комплекс, САПР, напряженно-деформированное состояние

Для цитирования

Соловьев А. Н., Киричевский Р. В. Развитие САПР для решения задач механики с использованием МКЭ // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 4. С. 67–84. DOI 10.25205/1818-7900-2021-19-4-67-84

© Соловьев А. Н., Киричевский Р. В., 2021

Development of a Finite Element Method and Its Application in a CAD

Arkady N. Soloviev [1], **Rostislav V. Kirichevsky** [2]

[1] Don State Technical University
Rostov on Don, Russian Federation

[2] V. Dahl Lugansk State University
Lugansk, Ukrain

[1] solovievarc@gmail.com, https://orcid.org/0000-0001-8465-5554
[2] rost71@mail.ru, https://orcid.org/0000-0002-4557-8469

Abstract

The presented review of existing domestic and foreign CAD systems showed that the use of FEM in their structure remains quite relevant before other methods in the speed of calculations, sufficient accuracy and software implementation in the CAD structure. To solve the problems of mechanics of elastomers and composites based on them, an overview of the developed domestic software MIRELA+ is presented in more detail. This complex solves many problems of solid mechanics: it has a special focus on solving problems of dissipative heating, parameters of fracture mechanics of massive elastomeric structural elements and thin-layer rubber-metal elements with cracks with changing physical, mechanical and thermophysical parameters under cyclic deformation. Not many of the domestic programs can implement this, and licenses for foreign programs cost tens of thousands of dollars.

This paper also presents methods of discretization by finite elements and an algorithm for constructing a system of resolving equations used in MIRELA+, expressions for determining the components of the coordinate transformation tensor, the strain tensor at the FE centers in a Cartesian coordinate system. Examples of visual representation in a three-dimensional image using a color picture are given, where each shade or color corresponds to a certain range of numerical values of the corresponding function.

Keywords

finite element method, program complex, CAD, CAE, stress-strain state

For citation

Solovev A. N., Kirichevsky R. V. Development of a Finite Element Method and Its Application in a CAD. *Vestnik NSU. Series: Information Technologies*, 2021, vol. 19, no. 4, p. 67–84. (in Russ.) DOI 10.25205/1818-7900-2021-19-4-67-84

Введение

Одним из наиболее важных аспектов автоматизации проектирования является замена дорогостоящего и длительного экспериментального исследования опытного образца численным экспериментом, суть которого состоит в построении и исследовании с помощью компьютера математической модели проектируемого объекта. Кроме того, на практике не всегда имеется возможность испытаний опытных образцов, например, в авиастроении, так как это может привести к весьма серьезным экономическим затратам, а иногда и к катастрофическим последствиям.

При проектировании сложных инженерных и строительных конструкций наиболее важным элементом анализа является исследование их напряженно-деформированного состояния, что приводит к необходимости автоматизации решения с помощью ЭВМ задач механики деформируемого твердого тела. В силу сложности физической природы исследуемых явлений и их математического описания решения получают при весьма существенных ограничениях относительно свойств материалов, конструктивных форм, граничных и начальных условий. Поэтому для моделирования и анализа напряженно-деформированного состояния на практике часто используют различные численные методы.

Обзор численных методов дискретизации объектов механики

Обычно численные методы основываются на дискретизации статических и математических моделей решаемых задач. В методах дискретизации рассматриваемая сплошная система с бесконечным числом степеней свободы аппроксимируется системой с конечным числом степеней свободы. Дифференциальные и интегральные уравнения, описывающие поведение сплошной среды, в методах дискретизации заменяются системой алгебраических уравнений, неизвестными в которой являются узловые значения разрешающей функции.

Среди них особое место занимают метод конечных разностей (МКР), вариационно-разностный метод (ВРМ) и метод конечных элементов (МКЭ) [1–11]. Помимо универсальности, обусловленной общими принципами механики, эти методы объединяет дискретная математическая модель исследуемого объекта. Состояние описывается системой алгебраических уравнений, решение которой определяет дискретное множество значений разрешающей функции в заранее намеченных точках (узлах) области. Однако путь «исследуемый объект – математическая модель» для каждого метода различен, что в значительной степени определяет области приложения методов, принципы доказательства существования и сходимости решений, процедуры реализации на ЭВМ.

Тот или другой метод дискретизации выбирается в зависимости от класса решаемых задач. Например, МКР широко используется в динамике жидкости и газа, в теории упругости, связанной с временными координатами, а МКЭ наиболее применим в механике деформированного твердого тела, строительной механике, в стационарных задачах теплопередачи и гидродинамики.

С точки зрения механики МКЭ базируется на замене исследуемого объекта совокупностью конечного числа дискретных элементов, соединенных между собой в отдельных узлах. Статические свойства каждого элемента определяются на основе вводимых механических гипотез, а искомые усилия (перемещения) – из условия кинематической (статической) совместимости системы. Приведенная трактовка обусловливает такую последовательность проведения исследования по МКЭ:

1) назначение расчетных узлов, в которых определяются величины разрешающей функции, и расчленение исследуемого объекта на конечные элементы желаемой формы;

2) установление зависимостей между усилиями и перемещениями в «контактных» узлах элемента, т. е. построение матриц жесткости (податливости);

3) составление системы алгебраических уравнений, выражающих кинематическую (статическую) совместимость деформации исследуемого объекта;

4) решение составленных уравнений и вычисление значений разрешающей функции в расчетных узлах;

5) определение компонентов напряженно-деформированного состояния исследуемой системы на основе найденных значений разрешающей функции.

Наиболее важными являются первые два пункта, определяющие количество и расположение расчетных узлов, форму конечных элементов (КЭ) и гипотезы о распределении перемещений или напряжений в области КЭ. От рационального решения этих вопросов зависит успех решения задачи в целом.

Если рассматривать МКЭ, для которого внутри КЭ перемещения аппроксимируются, например, полиномиальными функциями, а контакт на границах элементов осуществляется с учетом соблюдения условия неразрывности, то этот вариант МКЭ обладает медленной сходимостью в силу того, что полиномиальные функции не включали в себя слагаемые, описывающие жесткие смещения КЭ. Как следует из работ А. С. Сахарова [1; 2], этот эффект существеннее проявляется при использовании криволинейных КЭ, и учет жестких смещений КЭ следует рассматривать не как необходимое условие сходимости, а как важное средство повышения эффективности МКЭ при расчете тел криволинейной формы.

В процессе эксплуатации стандартной схемы МКЭ в форме метода перемещений, наряду с проблемой жестких смещений КЭ, было замечено и другое негативное свойство матрицы жесткости (МЖ), имеющее в настоящее время название «эффекта ложного сдвига». Суть его заключается в том, что при изгибе тонких пластин и оболочек, моделируемых трехмерными КЭ, значительно возрастают погрешности, связанные с появлением фиктивных сдвиговых деформаций.

Для устранения этих двух недостатков стандартных схем МКЭ в форме метода перемещений А. С. Сахаровым, В. В. Киричевским была разработана моментная схема конечных элементов (МСКЭ) [2; 3]. Эта схема позволяет учесть основные свойства жестких смещений как для изопараметрических, так и для криволинейных КЭ изотропных упругих тел. Суть ее заключается в отбрасывании определенных членов разложения деформаций, реагирующих на жесткие смещения и на появляющиеся фиктивные сдвиговые деформации. При этом точные уравнения связи деформаций и перемещений заменяются приближенными.

Во многих случаях сложность исследуемого объекта (нерегулярная структура, наличие отверстий, весьма неравномерный градиент разрешающей функции) делает необходимым использование в трехмерных задачах искривленных трехмерных КЭ. В целях упрощения учета геометрии элементов целесообразнее использовать криволинейные координаты. Однако такому уточненному учету геометрии противостоит трудность вычисления интегралов сложной структуры, зависящих от геометрических характеристик.

Успешное применение МКЭ в линейных задачах предопределило дальнейшее приложение этого метода к нелинейным задачам.

В большинстве рассмотренных работ по МКЭ коэффициенты МЖ представлялись в декартовой системе координат. Использование таких, уже готовых МЖ для исследования нерегулярных областей с криволинейными поверхностями становится затруднительным и вряд ли может давать хорошие результаты. Вопросы построения МЖ МКЭ в настоящее время достаточно изучены. Например, в работе [4] показано влияние аппроксимирующих функций при построении МЖ КЭ на скорость сходимости МКЭ. В ряде сложных задач, для исследования которых применяются криволинейные элементы произвольной формы, нет надобности в построении МЖ в численной форме. Целесообразнее иметь алгебраическое выражение МЖ, записанное в ковариантной форме, не зависящей от системы координат. В этом случае увеличивается эффективность работы вычислительных комплексов.

Решение сколь угодно простой задачи по МКЭ невозможно без привлечения компьютерных систем. Большинство существующих систем автоматизированного проектирования (САПР) ориентировано на использование МКЭ. К настоящему времени на основе МКЭ создано большое количество вычислительных программ и их число продолжает увеличиваться.

Обзор САПР

Вычислительные программы, разработанные до 1960 г., были в большинстве своем основаны на использовании метода сил в матричной форме с ручным вводом исходной информации и числом неизвестных, не превышающим 100. Развитие самолетостроения и ракетостроения в 1960–1970-е гг. усложнило задачи по проблемам прочности, таким образом, возрасли требования к большей детализации и точности дискретизации конструкций. Это привело к расчетным схемам с большим числом неизвестных. В работах этого периода отсутствовала четкая классификация программ, и сообщалось только об их применении без описания принципов работы.

Первый вычислительный комплекс программ в нашей стране был разработан в ПНИЛТПК КИСИ под руководством Д. В. Вайнберга в 1962–1964 гг. и назывался ВК-1 [5]. Переход к массовому расчету элементов конструкций потребовал модификации комплекса ВК-1 с целью полной автоматизации многосерийного счета, и новый вариант комплекса получил название ВК-2 [6]. Уже в то время комплекс позволял решать линейные и нелинейные

задачи статики, динамики, колебаний и устойчивости сложных конструкций на основе вариационно-разностного метода [6–8] и разработанных итерационных алгоритмов решения больших систем алгебраических уравнений [9].

Разработанный комплекс был ориентирован на ЭВМ М20/М220 и отличался значительной эффективностью благодаря рациональному построению алгоритмов разностных аппроксимаций и удачной логической структуре. Несмотря на малую оперативную память применяемых ЭВМ, успешно решались задачи с числом неизвестных до 600. Это позволило решить ряд новых, к тому времени еще не решенных задач строительной механики и теории упругости [4; 10].

Интересно отметить, что первые попытки решения задач статики сложных комбинированных систем также относятся к этому периоду [11]. Результатом опытной эксплуатации комплекса ВК-2 стал анализ обнаруженных трудностей и недостатков, что помогло определить задачи, которые решались в процессе создания вычислительного комплекса второго поколения – ПРОЧНОСТЬ-1 [12; 13]. Этот комплекс был реализован на быстродействующей ЭВМ БЭСМ-6, и в основу его реализации был положен более прогрессивный численный метод – МКЭ. В дальнейшем модернизированный вариант комплекса получил название ПРОЧНОСТЬ-75 [14–16] и был адаптирован для ЕС ЭВМ [17].

Один из первых зарубежных вычислительных комплексов ASKA (автоматическая система кинематического анализа) [18; 19] был создан в ФРГ под руководством Дж. Аргириса. Принципиальная идея ASKA состояла в создании универсального и гибкого инструмента для расчета МКЭ, который можно было бы использовать для решения практических задач механики без составления дополнительных программ.

ASKA тогда состояла примерно из 120 подпрограмм, каждая из которых могла быть загружена в оперативную память ЭВМ как некоторый логический объект. Здесь реализован принцип расчленения сложных объектов на суперэлементы, организована библиотека конечных элементов, содержащая большое количество одномерных, двумерных и трехмерных КЭ с числом степеней свободы в узле, достигающим 32. Большое внимание в системе уделено компактным способам задания входных данных, методам переработки и хранения информации, а также поиску и диагностике различных ошибок, встречающихся в процессе решения задач. Комплекс рассчитан на применение различных типов ЭВМ с большим быстродействием и оперативной памятью.

Французскими исследователями была создана универсальная система программ TITUS [20], которая позволяла решать широкий класс задач теории упругости, строительной механики и сопротивления материалов. Система была ориентирована на пользователя, где основное внимание уделено подготовке исходной информации, свободной от какого-либо формата и операции вывода данных. Автоматическое разбиение на элементы производится при помощи различных процедур, связанных с конкретным видом конструкции. Однако задача простого описания разбиения на элементы любой геометрической формы осталась нерешенной.

С точки зрения программирования в системе TITUS нет априорного ограничения на число соединений, элементов, характеристик. Ограничивается только число видов нагрузки (до 10), для которых можно вести расчеты одновременно. Все остальные ограничения связаны с типом используемой ЭВМ. В системе организовано «динамическое управление» используемой центральной памяти, что дает возможность смешанной обработки информации, т. е. использования двух ЭВМ, что особенно эффективно при решении больших по объему задач, где предварительная обработка может быть произведена на малой ЭВМ, а основная вычислительная работа – на большой. Все промежуточные результаты записываются на различные носители. Исходные данные задачи загружаются в упакованной форме с помощью системы вычислительных адресов, что позволило отказаться от обычно используемой в Фортране предварительной резервации массива.

Система программ SESAM-69 [21; 22], созданная в Норвегии и предназначенная для прочностного анализа судовых конструкций на основе суперэлементной дискретизации [23], представляет собой собрание программных модулей, которые могут комбинироваться различным образом. Благодаря блочной структуре в систему, по мере необходимости, можно вводить новые программные блоки.

Суперэлементный подход хорошо сочетается с модульной структурой и компактным программированием. Матрица жесткости для КЭ, так же как и результирующая матрица, разбивается на субматрицы приемлемых размеров, которые могут последовательно обрабатываться центральным процессором ЭВМ. Раздельная обработка левой и правой частей позволяет производить обработку нескольких правых частей одновременно. Операции над субматрицами выполняет набор стандартных подпрограмм, написанных как на Фортране, так и на ассемблере. Объединение суперэлементов осуществляется с помощью топологического описания. Возможности системы хорошо демонстрировались на примере расчета танкера ESSO NORWAY [23].

Далее, рассмотрим некоторые зарубежные и отечественные вычислительные комплексы, предназначенные для расчета пространственных конструкций на основе МКЭ, часть из которых являются программами общего назначения, возникшие ранее, модернизированные и не утратившие своего значения к данному моменту.

Большой универсальный комплекс общего назначения NASTRAN [24–26] разработан в исследовательских центрах NASA (США) и изначально был предназначен для решения задач американской аэрокосмической промышленности. Он реализовывал МКЭ в форме перемещений и насчитывал около 750 подпрограмм, написанных в основном на Фортране и частично на ассемблере. Метод подконструкций (суперэлементов) используется на основе специально разработанного языка абстрактных матричных операций – DMAP. Особое внимание уделено рациональным методам задания исходной информации, методам хранения и решения больших систем уравнений. Сейчас NASTRAN – 57-летняя система конечно-элементного расчета и анализа конструкций – предоставляет возможность расчета напряженно-деформированного состояния, запасов прочности, собственных частот и форм колебаний, анализ устойчивости, исследование установившихся и неустановившихся динамических процессов, решение задач теплопередачи, анализ сложного контактного взаимодействия, возможность моделирования различных типов материалов, включая композитные и гиперупругие. В состав расширенных функций входит технология суперэлементов, включая продвинутые методы динамических конденсаций, модальный синтез и развитые методы анализа динамики сложных структур на основе суперэлементов [27].

Популярность среди образовательных и исследовательских организаций получили пакеты программ с открытым исходным кодом. Так, Tahoe, Elmer, CalculiX и др. позволяют пользователям реализовывать свои алгоритмы анализа, не прибегая к созданию модулей пре- и постпроцессора [28]. Например, свободная платформа SALOME кроме внутреннего языка Python дает возможность написания модулей на C++ [1].

По настоящее время одним из наиболее известных и широко применяемых программных комплексов является ANSYS [29–31]. Данная программная система позволяет численно решать пространственные задачи статического и динамического нагружения, задачи с учетом геометрической нелинейности, устойчивости, механики разрушения и целый ряд тому подобных классов задач. Кроме задач механики деформированного твердого тела ANSYS позволяет решать также ряд задач гидродинамики, акустики и др.

Более подробные обзоры других современных САПР (CATIA, COSMOS, MECHANICA, PATRAN, ABAQUS, LS-DYNA, PRO/ENGINEER, ADINA, HyperWorks и др.) можно найти, например, в работах [32–35].

[1] SALOME. URL: https://www.salome-platform.org/ (дата обращения 05.05.2020).

На первых этапах практической реализации МКЭ решение сложных задач значительно осложнялось из-за несоответствия количества подготовленных данных возможностям их переработки [36; 37]. Позже эти проблемы приобрели первостепенное значение, но их решение происходит медленно. Одной из таких проблем на пути автоматического генерирования исходных данных является деление конструкции на элементы. Автоматическое образование сетки координат, очевидно, существенно, если используются тетраэдральные элементы, однако при применении их возникают затруднения, которые можно преодолеть, заранее объединив тетраэдры в призмы и шестигранные элементы. Ряд разработчиков используют комбинированные схемы МКЭ, более эффективные (с сокращением времени вычислений и улучшением точности результата) гибридные МКЭ, поскольку для ускорения циклов анализа необходима адаптация модели САПР. Этап адаптации заключается в упрощении геометрии модели САПР за счет исключения деталей (отверстий, фасок, углового сечения и т. п.) и граней [38].

В странах бывшего СНГ также достигнуты большие успехи в разработке вычислительных комплексов программ для ЭВМ на основе МКЭ: ПРОЧНОСТЬ-1, КАСКАД-2, МИРАЖ, СУПЕР, ЛИРА, МОРЕ, РАССУДОК, ГАММА, КОДЕТОМ, МИРЕЛА+, КОМПАС, COMSOL и др. [35].

Украино-российская разработка ЛИРА в виде программного комплекса предназначена для проектирования и расчета строительных и машиностроительных конструкций. Для расчета модели данный комплекс имеет несколько процессоров: линейный (работа материала конструкций в линейно-упругой постановке) и нелинейный (учитывает физическую нелинейность материала в рамках теории упругости и в упругопластической постановке, геометрическую нелинейность и конструктивную нелинейность), а также вспомогательные процессоры (используются для исследования основной расчетной модели). Расчетные процессоры используют МКЭ в перемещениях для определения напряженно-деформированного состояния конструкции с большим выбором конечных элементов. Согласно описанию, при дискретизации рассматриваемых объектов и переходе к системам уравнений, данный программный комплекс не накладывает ограничения на количество узлов и элементов, что позволяет добиться высокой точности [39].

Программный комплекс FORTU-FEM разработан для автоматизации анализа сложных механических процессов на базе МКЭ и позволяет решать широкий класс задач машиностроения. Эта САПР позволяет генерировать в автоматическом режиме дискретные модели сложных трехмерных тел, проводить конечно-элементный анализ и визуализировать результаты. Данный программный комплекс имеет встроенный объектно-ориентированный язык программирования FORTU-3, который является методом описания общей постановки задачи и метода ее расчета. В процессе расчета пользователь имеет возможность выбирать тип КЭ, вид функционала, который минимизируется. В состав FORTU-FEM входят две подсистемы: подсистема дискретизации произвольной геометрической области на КЭ и подсистема анализа, позволяющая эффективно исследовать полученные численные результаты [40].

Обзор программного комплекса МИРЕЛА+

Программный комплекс МИРЕЛА+ разработан для автоматизации анализа задач механики эластомеров на основе МКЭ. Вычислительный комплекс МИРЕЛА+ [41; 42] является дальнейшим развитием вычислительной системы КОДЕТОМ и предназначен для исследования прочности, долговечности и разрушения конструкций из эластомерных и композитных материалов в условиях линейного и нелинейного вязкоупругого деформирования.

Каждый модуль в процессоре МИРЕЛА+, как и многих других САПР, реализует определенный вид расчета некоторым методом, жестко заложенным на этапе его разработки. Это приводит к тому, что большинство таких САПР являются закрытыми для пользователя, что не позволяет ему выполнять анализ задач, методы решения которых не были предусмотрены

и реализованы разработчиками конкретного программного комплекса. В этом случае инженер-проектировщик должен либо самостоятельно программировать численную схему решения задачи, либо использовать другую систему автоматизации анализа напряженно-деформированного состояния. Каждый из этих этапов представляет собой самостоятельную сложную задачу, требующую для решения значительных временных и вычислительных затрат.

Пример конечно-элементной модели эластомерных элементов конструкций и визуальное представление результатов расчета их напряженно-деформированного состояния в системе МИРЕЛА+ изображен на рис. 1. Структура вычислительного комплекса МИРЕЛА+ представлена на рис. 2.

Расчет конструкций МКЭ в МИРЕЛА+ осуществляется в виде трех взаимосвязанных последовательных процессов:

1) подготовка исходных данных – конечно-элементная дискретизация рассчитываемого объекта, его топология и кинематические и силовые граничные условия, физико-механические характеристики материала;

2) численный расчет конечно-элементной модели – вычисление коэффициентов матрицы жесткости конечных элементов, формирование глобальной системы разрешающих уравнений и ее решение;

3) обработка результатов решения – вычисление параметров напряженно-деформированного и температурного состояния конструкции; их визуальное представление в виде таблиц, графиков, двумерных либо трехмерных изображений.

Эти процессы численной реализации выполняются тремя подсистемами – препроцессором, процессором и постпроцессором соответственно. Препроцессор представляет собой блок подпрограмм, обеспечивающих задание геометрии исследуемого объекта, структура которого представлена на рис. 3.

Для задания геометрии имеется библиотека фигур стандартных типоразмеров: призматические, цилиндрические, сферические, конические и тела вращения. Геометрия нестандартных объектов задается посредством опорных точек, которые являются точками изменений в очертании фигуры.

Для описания конструкции, как уже упоминалось, вводятся две системы координат – базисная декартовая система координат $z^{k'}$, в которой задаются геометрические координаты узлов, поля нагрузок и граничных условий и местная криволинейная система координат x^i, в которой задается нумерация и сеточные координаты узлов (рис. 4).

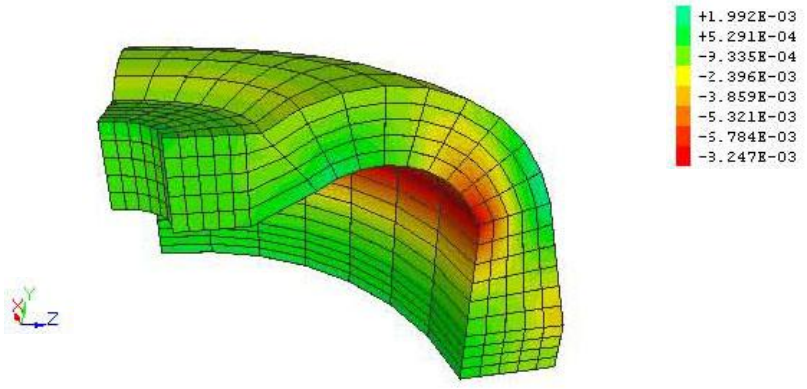


Рис. 1. Конечно-элементная модель резинового амортизатора
Fig. 1. A finite element model of a rubber shock absorber

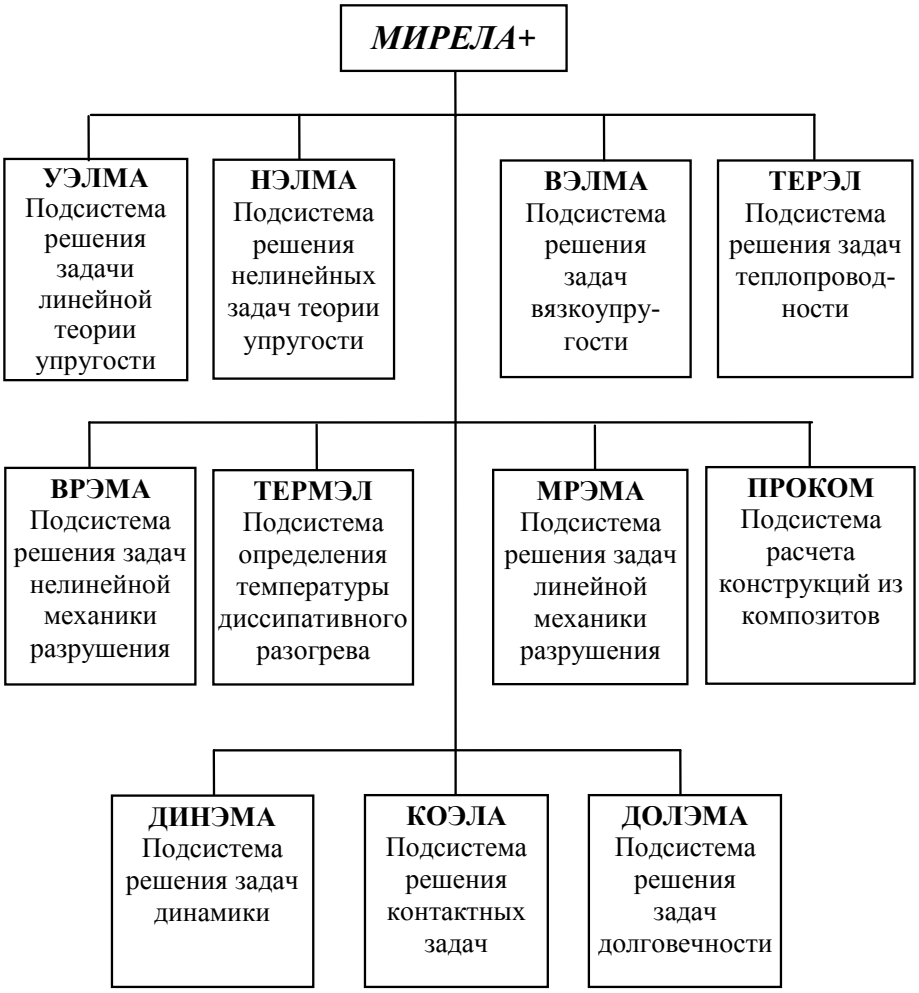


Рис. 2. Структура вычислительнго комплекса МИРЕЛА+
Fig. 2. The structure of the computing complex MIRELA+

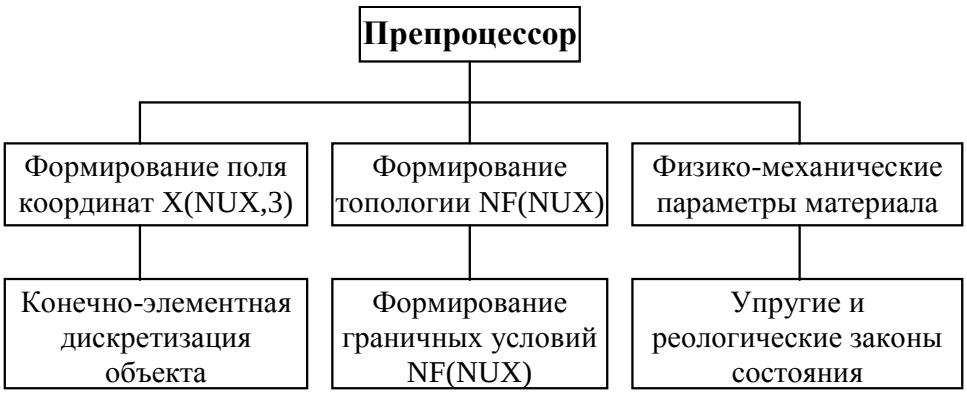


Рис. 3. Структура препроцессора
Fig. 3. Structure of the preprocessor

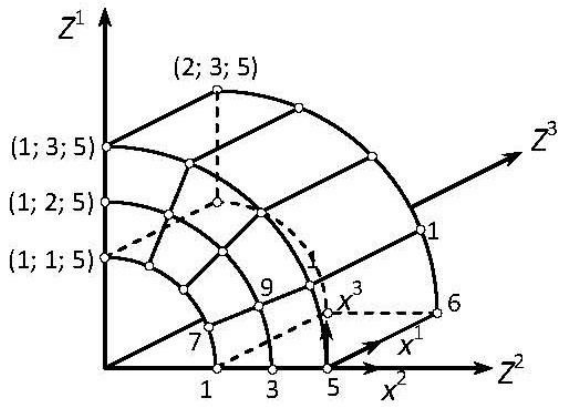


Рис. 4. Пространственная и сеточная нумерация узлов
Fig. 4. Spatial and grid node numbering

Блок процессора занимает центральное место при решении задач МКЭ, это ядро системы. Он включает в себя подпрограммы вычисления коэффициентов матрицы жесткости или теплопроводности конечных элементов, температурных нагрузок. Существует два подхода к формированию глобальной матрицы жесткости конструкции. Первый: матрица формируется из матриц жесткости КЭ с последующим суммированием по одноименным узлам. Второй: система уравнений формируется по строкам из коэффициентов матрицы жесткости КЭ, примыкающих к текущему узлу.

Алгоритм построения системы разрешающих уравнений следующий.

1. Определяется действительный номер индексной решетки

$$N^d(I,J,K) = N14 = N^c + I - 2 + M1(J-2) + M1 \cdot M2(K-2),$$

где N^c – номер центрального узла сеточной решетки; I, J, K – сеточные координаты узла; $M1, M2$ – размеры сетки разбиения на КЭ.

2. Вычисляются коэффициенты МЖ КЭ, примыкающих к узлу $N14$. Для этого каждому узлу КЭ задается смещение по m-му направлению, и вычисляется реакция в узле N^d по n-му направлению $(m, n = 1,2,3)$. Ввиду симметрии МЖ вычисляется не 576, а только 300 коэффициентов.

3. Определяются относительные номера узлов индексной решетки по сеточным координатам элемента ($i, j, k = 1,2;\quad i', j', k' = 1,2$)

$$l = i + i' + 3(j + j') + 9(k + k') - 25.$$

4. Значения коэффициентов заносятся в массив $RM^{(N)}(m,n,l)$. Для одного узла сеточной области вычисляется $3 \times 3 \times 27 = 243$ коэффициента. При одноименных вариациях перемещений N-го узла коэффициенты суммируются

$$RM^{(N)}(m,n,l) = \sum_{r=1}^{E} RM_r^{(N)}(m,n,l),$$

где E – число КЭ, примыкающих к узлу $N14$ (рис. 5).

5. Выполняется рассылка коэффициентов матрицы в матрицу системы разрешающих уравнений

$$K_{ij} \leftarrow RM^{(N)}(m,n,l).$$

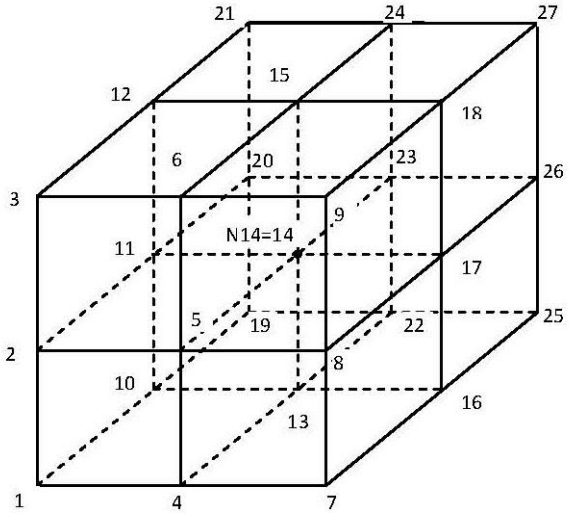


Рис. 5. Пространственная индексация КЭ решетки
Fig. 5. Spatial indexing of the grid

Между индексами существует следующее соответствие:

- действительный номер – $N = INT\left(\frac{i-1}{3}+1\right)$;
- направление реакции – $m = i - 3(N-1)$;
- номер смещаемого узла – $N^c = INT\left(\frac{j-1}{3}+1\right)$;
- направление смещаемого узла – $m = j - 3(N^c - 1)$;
- относительный номер индексной решетки $l = N^c - N + 14$.

Система линейных алгебраических уравнений, построенная таким образом, симметричная и имеет ленточный вид

$$\left[K^{ij}\right]\left\{u_j\right\} = \left\{P^i\right\}. \quad (1)$$

Вторая задача блока – это решение системы (1) модифицированным блочным методом Гаусса для ленточных, симметричных матриц либо другим точным, приближенным методом (комплекс допускает расширение).

Результатом решения системы (1) является вектор узловых перемещений $\left\{u_j\right\}$. По вычисленному вектору узловых перемещений $\left\{u_j\right\}$ определяются компоненты тензора деформаций ε_{ij} в центрах КЭ в декартовой системе координат:

$$\varepsilon_{11} = \Delta_1 u_1 c_1^1 + \Delta_1 u_2 c_1^2 + \Delta_1 u_3 c_1^3;$$
$$\varepsilon_{22} = \Delta_2 u_1 c_2^1 + \Delta_2 u_2 c_2^2 + \Delta_2 u_3 c_2^3;$$
$$\varepsilon_{22} = \Delta_3 u_1 c_2^1 + \Delta_3 u_2 c_2^2 + \Delta_3 u_3 c_2^3;$$
$$\varepsilon_{12} = \varepsilon_{21} = \frac{1}{2}(\Delta_2 u_1 c_1^1 + \Delta_2 u_2 c_1^2 + \Delta_2 u_3 c_1^3 + \Delta_1 u_1 c_2^1 + \Delta_1 u_2 c_2^2 + \Delta_1 u_3 c_2^3);$$

$$\varepsilon_{13}=\varepsilon_{31}=\frac{1}{2}(\Delta_3 u_1 c_1^1+\Delta_3 u_2 c_1^2+\Delta_3 u_3 c_1^3+\Delta_1 u_1 c_3^1+\Delta_1 u_2 c_3^2+\Delta_1 u_3 c_3^3);$$

$$\varepsilon_{23}=\varepsilon_{32}=\frac{1}{2}(\Delta_3 u_1 c_2^1+\Delta_3 u_2 c_2^2+\Delta_3 u_3 c_2^3+\Delta_2 u_1 c_3^1+\Delta_2 u_2 c_3^2+\Delta_2 u_3 c_3^3).$$

Определяются компоненты тензора преобразования координат $c_j^{i'}$.

Для линейного элемента градиенты деформаций $\Delta_i u_j$ вычисляются в центре КЭ (рис. 6) через узловые перемещения $U(N,I)$ по формулам:

$$\Delta_i U_1=\frac{1}{4}(U(2,i)+U(4,i)+U(6,i)+U(8,i)-U(1,i)-U(3,i)-U(5,i)-U(7,i));$$

$$\Delta_i U_2=\frac{1}{4}(U(3,i)+U(4,i)+U(7,i)+U(8,i)-U(1,i)-U(2,i)-U(5,i)-U(5,i));$$

$$\Delta_i U_3=\frac{1}{4}(U(5,i)+U(6,i)+U(7,i)+U(8,i)-U(1,i)-U(2,i)-U(3,i)-U(4,i)).$$

Компоненты тензора напряжений также вычисляются в центре КЭ по зависимостям:

$$\sigma^{11}=2\mu\varepsilon_{11}+\lambda\theta;\quad \theta=(\varepsilon_{11}+\varepsilon_{22}+\varepsilon_{33})/3;$$

$$\sigma^{22}=2\mu\varepsilon_{22}+\lambda\theta;$$

$$\sigma^{33}=2\mu\varepsilon_{33}+\lambda\theta;$$

$$\sigma^{12}=\sigma^{21}=2\mu\varepsilon_{12};\ \sigma^{13}=\sigma^{31}=2\mu\varepsilon_{13};\ \sigma^{23}=\sigma^{32}=2\mu\varepsilon_{23}.$$

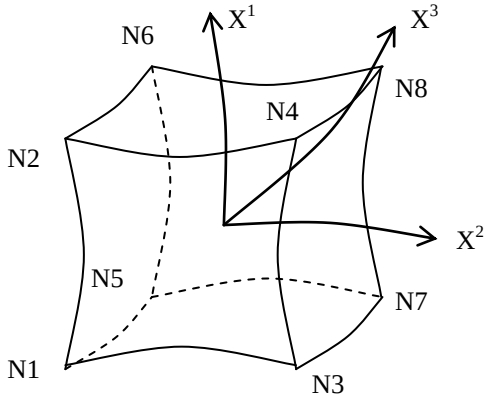


Рис. 6. Пространственная нумерация узлов КЭ
Fig. 6. Spatial numbering of FE nodes

Результаты расчета конструкции могут быть представлены в виде таблиц перемещений, деформаций и напряжений в узлах и центрах КЭ дискретной области, а также графически в виде муаровых полос, изолиний или поверхностей функций перемещений и напряжений по объему или в заданных сечениях. Визуальное представление осуществляется в виде двумерного или трехмерного изображения с помощью полутоновой или цветной картины, где каждому оттенку или цвету соответствует определенный диапазон числовых значений функции.

Несмотря на то, что ANSYS, например, решает более широкий круг задач: служит для анализа поведения материалов и конструкций при краткосрочных значительных и критических нагрузках, высоких давлениях или взрывах, для моделирования деформаций и разрушения материала, для проведения прочностных и усталостных расчетов, для моделирования взаимодействия жидкостей, газов и твердых тел, фазовых переходов, распространения ударных волн, для моделирования пластического и гиперупругого нелинейного поведения материала, проведения гармонических, сейсмических и вибрационных расчетов, моделирования контактного взаимодействия и анализа кинематики механизмов, – МИРЕЛА+ кроме того, что также решает многие задачи механики деформации твердого тела, имеет специальную направленность на решение задач диссипативного разогрева, параметров механики разрушения массивных эластомерных элементов конструкций и тонкослойных резинометаллических элементов с трещинами с изменяющимися физико-механическими и теплофизическими параметрами в условиях циклического деформирования. Не многие из отечественных программ это смогут реализовать, а лицензии зарубежных программ стоят десятки тысяч долларов.

Заключение

Проведенный обзор существующих САПР показывает, что применение в их структуре МКЭ остается вполне актуальным перед другими методами в скорости вычислений, достаточной точности и программной реализации в структуре САПР. Для решения задач механики эластомеров и композитов на их основе комплекс МИРЕЛА+ эффективно справляется с поставленными задачами в области линейного и нелинейного вязкоупругого деформирования, исследования температурных полей диссипативного разогрева, параметров механики разрушения, долговечности работы конструкций при различных видах нагружений.

Список литературы

1. **Сахаров А. С.** Модификация метода Ритца для расчета массивных тел на основе полиномиальных разложений с учетом жестких смещений // Сопротивление материалов и теория сооружений. 1974. Вып. 23. С. 61–70.
2. **Сахаров А. С.** Моментная схема метода конечных элементов (МСКЭ) с учетом жестких смещений // Сопротивление материалов и теория сооружений. 1974. Вып. 24. С. 147–156.
3. **Сахаров А. С., Кислоокий В. Н., Киричевский В. В. и др.** Метод конечных элементов в механике твердых тел. Киев, Вища школа, 1982. 480 с.
4. **Киричевский Р. В., Скринникова А. В.** Влияние аппроксимирующих функций при построении матрицы жесткости конечного элемента на скорость сходимости МКЭ // Вестник Том. гос. ун-та. Матем. и мех. 2019. № 57. С. 26–37.
5. **Вайнберг Д. В., Синявский А. Л.** Дискретный анализ в теории упругости // Численные методы расчета пространственных конструкций. Киев: КИСИ, 1968. С. 5–38.
6. **Вайнберг Д. В.** Численные методы в теории оболочек и пластин // Тр. IV Всесоюз. конф. по теории оболочек и пластин. Ереван, 1966. С. 206–215.
7. **Ворошко П. П.** Вариационно-разностный метод решения трехмерных задач в теории упругости // Численные методы расчета пространственных конструкций. Киев: КИСИ, 1968. С. 209–215.
8. **Ворошко П. П., Сахаров А. С.** Построение разностных уравнений теории упругости и их получение на ЭВМ // Сопротивление материалов и теория сооружений. 1966. Вып. 4. С. 174–190.

9. **Ильченко Е. Н., Сахаров А. С.** О решении больших систем уравнений при расчете пластин и оболочек // Сопротивление материалов и теория сооружений. 1972. Вып. 16. С. 259–263.
10. **Геращенко В. М.** Расчет пластинчатых и коробчатых систем // Численные методы расчета пространственных конструкций. Киев: КИСИ, 1968. С. 49–70.
11. **Гоцуляк Е. А. и др.** Блочный метод расчета комбинированных систем // Численные методы расчета пространственных конструкций. Киев: КИСИ, 1968. С. 197–208.
12. **Кислоокий В. Н., Сахаров А. С.** Проблемно-ориентированный вычислительный комплекс прочностных расчетов пространственных конструкций // Организация и методика строительного проектирования с применением вычислительной и организационной техники. М.: ЦНИПИАСС, 1974. Вып. 2. С. 10–14. (ГОССТРОЙ СССР, Серия X)
13. **Вайнберг Д. В. и др.** Система математического обеспечения расчета пространственных конструкций «Прочность-1» // Организация и методика строительного проектирования. М.: ЦНИПИАС, 1972. Вып. 2. С. 6–10. (ГОССТРОЙ СССР, Серия X)
14. **Исаханов Г. В. и др.** Система математического обеспечения прочностных расчетов пространственных конструкций // Проблемы прочности. 1978. № 11. С. 59–61; № 12. С. 25–28.
15. **Гончаренко И. Е. и др.** Проблемно-ориентированные языки пользователя. Формальное описание. ПРОЧНОСТЬ-75. Система математического обеспечения расчетов пространственных конструкций. Киев: Республиканский фонд алгоритмов и программ АН УССР, 1975. Т. 4. 550 с.
16. **Гончаренко И. Е. и др.** Комплекс прикладных программ расчета оболочечных конструкций. Листинг. ПРОЧНОСТЬ-75. Система математического обеспечения расчетов пространственных конструкций. Киев: Республиканский фонд алгоритмов и программ АН УССР, 1975. Т. 5. 357 с.
17. **Бесценный Ю. Г. и др.** Система автоматизации комплексного расчета пространственных конструкций на ЕС ЭВМ // Комплексный расчет зданий и сооружений с применением ЭВМ. Киев: КИСИ, 1978. С. 55–58.
18. **Скрим Э., Рой Дж. Р.** Автоматическая система кинематического анализа // Расчет упругих конструкций с использованием ЭВМ: В 2 т. / Пер. с англ., под ред. А. П. Филина. Л.: Судостроение, 1974. Т. 2. С. 36–67.
19. **Argyris J. H.** ASKA: automatic system for kinematic analysis – a universal system for structural analysis based on the matrix displacement (finite element) method. *Nucl. Eng. Des.*, 1969, no. 2, pp. 441–447.
20. **Launay P. et al.** The three-dimensional thermoelastic computer code “TITUS”. In: Prepr. 1st Int. Conf. Struct. Mech. React. Technol. Berlin, Amsterdam, Amsterdam e.a., 1971, no. 5, pp. M5-4/1–M5-4/21.
21. **Araldsen P. O., Egeland O.** General description of SESAM-69 super element structural analysis (Program) modules. *European Shipbuilding*, 1971, no. 2, pp. 21–35.
22. **Egeland O., Araldsen P. O.** SESAM-69 – A general purpose finite element method program. *Intern. J. of Computers and Structures*, 1974, no. 1, pp. 41–68.
23. **Araldsen P. O.** The application of the superelement method in analysis and design of ship structures and machinery components. In: National Symp. on Computerized structural Analysis and Design. Norway, March, 1972, pp. 2–93.
24. **Butler T. G., Michel D.** NASTRAN. A summary of the functions and capabilities of the NASA structural analysis computer system. Washington, 1971, 22 p. (NASA SP-260)
25. **McNeal R. H., McCormic C. W.** The NASTRAN computer program for structural analysis. *Comput. and Struct.*, 1971, no. 1, pp. 32–35.
26. **Tocher J. L., Herness E. D.** A critical view of NASTRAN. *Numerical and Computer Methods in Structural Mechanics*, 1973, pp. 151–174.

27. **Чумаченко Е. Н. и др.** Математическое моделирование в нелинейной механике (обзор программных комплексов для решения задач моделирования сложных систем). М., 2009. 23 с.
28. **Faas D., Vance J. M.** Interactive deformation through mesh-free stress analysis in virtual reality. *Mechanical Engineering Conference Presentations, Papers, and Proceedings*, 2008, p. 46.
29. **Фролов Д.** Обзор возможностей ANSYS Mechanical для решения инженерных задач // САПР и графика. 2010. № 11. URL: http://www.caeexpert.ru/sites/default/files/obzor_vozmozhnostey_ansys_mechanical_dlya_resheniya_inzhenernyh_zadach.pdf.
30. **Чигарёв А. В., Кравчук А. С., Смалюк А. Ф.** ANSYS для инженеров: Справ. пособие. М.: Машиностроение-1, 2004. 512 с.
31. **Черпаков А. В. и др.** Моделирование колебаний при импульсном воздействии многослойной конструкции в комплексе Ansys // Инженерный вестник Дона. 2019. № 6. URL: http:// ivdon.ru/ru/magazine/archive/n6y2019/6057.
32. **Селяков М. Ю.** Отечественные и зарубежные CAD/CAM системы // Успехи современного естествознания. 2011. № 7. С. 193–197.
33. **Иванов С. Е.** Интеллектуальные программные комплексы для технической и технологической подготовки производства. Часть 5: Системы инженерного расчета и анализа деталей и сборочных единиц: Учеб.-метод. пособие / Под ред. Д.Д. Куликова. СПб.: СПбГУ ИТМО, 2011. 48 с.
34. **Павлов С.** CAE – технологии в 2013 году: обзор достижений и анализ рынка // CAD/CAM/CAE Observer. 2014. № 4 (88). URL: http://www.cadcamcae.lv/N88/08-18.pdf.
35. **Калинин А. В., Хвалин А. Л.** Применение метода конечных элементов в современных системах автоматизированного проектирования // Гетеромагнитная микроэлектроника. 2019. № 26. С. 41–51.
36. **Городецкий А. С. и др.** Метод конечных элементов: теория и численная реализация. Киев: Факт, 1997. 137 с.
37. **Киричевский Р. В.** Численное моделирование температурных полей диссипативного разогрева конструкций из эластомеров с трещинами. Киев: Наук. дум., 1998. 120 с.
38. **Mounir H., Nizar A., Borhen L., Benamara A., Deneux D.** FEM Simulation Based on CAD Model Simplification: A Comparison Study between the Hybrid Method and the Technique Using a Removing Details. *Design and Modeling of Mechanical Systems*, 2013, pp. 587–596. DOI 10.1007/978-3-642-37143-1_70
39. **Городецкий Д. А. и др.** Программный комплекс ЛИРА-САПР 2013: Учеб. пособие. Киев; Москва: Электронное издание, 2013. 376 с.
40. **Mezhuev V., Lavrik V., Ravi S.** Development and application of the problem-oriented language FORTU for the design of non-standard mechanical constructions. *Journal of the Serbian Society for Computational Mechanics*, 2015, no. 9 (2), pp. 1–9. DOI 10.5937/jsscm1502001M
41. **Киричевский В. В. и др.** Метод конечных элементов в вычислительном комплексе «МІРЕЛА+». Киев: Наук. дум., 2005. 403 с.
42. **Киричевский В. В. и др.** Развитие метода конечных элементов и его применение в САПР // Вестник Запорож. нац. ун-та. Физ.-мат. науки. 2006. № 1. С. 38–56.

References

1. **Sakharov A. S.** Modifikatsiya metoda Rittsa dlya rascheta massivnikh tel na osnove polinomialnykh razlozhenii s uchetom zhestkikh smeshchenii. *Soprotivlenie materialov i teoriya sooruzhenii*, 1974, no. 23, pp. 61–70. (in Russ.)
2. **Sakharov A. S.** Momentnaya skhema metoda konechnykh elementov (MSKE) s uchetom zhestkikh smeshchenii. *Soprotivlenie materialov i teoriya sooruzhenii*, 1974, no 24, pp. 147–156. (in Russ.)

3. **Sakharov A. S., Kislooky V. N., Kirichevsky V. V. et al.** Metod konechnykh elementov v mekhanike tverdykh tel. Kiev, Vischa shkola, 1982, 480 p. (in Russ.)
4. **Kirichevsky R. V., Skrinnikova A. V.** Vliyanie approksimiruyushchikh funktsii pri postroenii matritsi zhestkosti konechnogo elementa na skorost skhodimosti MKE. *Vestnik Tomsk. gos. un-ta. Matem. i mekh.*, 2019, no. 57, pp. 26–37. (in Russ.)
5. **Vainberg D. V., Sinyavsky A. L.** Diskretnii analiz v teorii uprugosti. In: Chislennie metodi rascheta prostranstvennykh konstruktsii. Kiev, KISI Press, 1968, pp. 5–38. (in Russ.)
6. **Vainberg D. V.** Chislennie metodi v teorii obolochek i plastin. In: Trudi IV Vsesoyuznoi konf. po teorii obolochek i plastin. Erevan, 1966, pp. 206–215. (in Russ.)
7. **Voroshko P. P.** Variatsionno-raznostnyi metod resheniya trekhmernykh zadach v teorii uprugosti. In: Chislennye metody rascheta prostranstvennykh konstruktsii. Kiev, KISI Press, 1968, pp. 209–215. (in Russ.)
8. **Voroshko P. P., Sakharov A. S.** Postroenie raznostnykh uravnenii teorii uprugosti i ikh poluchenie na EVM. *Soprotivlenie materialov i teoriya sooruzhenii*, 1966, no. 4, pp. 174–190. (in Russ.)
9. **Ilchenko E. N., Sakharov A. S.** O reshenii bolshikh sistem uravnenii pri raschete plastin i obolochek. *Soprotivlenie materialov i teoriya sooruzhenii*, 1972, no. 16, pp. 259–263. (in Russ.)
10. **Geraschenko V. M.** Raschet plastinchatykh i korobchatykh system. Chislennye metody rascheta prostranstvennykh konstruktsii. Kiev, KISI Press, 1968, pp. 49–70. (in Russ.)
11. **Gotsulyak E. A. et al.** Blochnii metod rascheta kombinirovannih system. In: Chislennye metody rascheta prostranstvennykh konstrukysii. Kiev, KISI Press, 1968, pp. 197–208. (in Russ.)
12. **Kislooky V. N., Sakharov A. S.** Problemno-orientirovannyi vichislitelnyi kompleks prochnostnykh raschetov prostranstvennykh konstruktsii. In: Organizatsiya i metodika stroitelnogo proektirovaniya s primeneniem vichislitelnoi i organizacionnoi tekhniki. Moscow, CNIPIASS, 1974, iss. 2, pp. 10–14. (in Russ.)
13. **Vainberg D. V. et al.** Sistema matematicheskogo obespecheniya rascheta prostranstvennykh konstruktsii “Prochnost_1”. In: Organizatsiya i metodika stroitelnogo proektirovaniya. Moscow, CNIPIAS, 1972, iss. 2, pp. 6–10. (in Russ.)
14. **Isakhanov G. V. et al.** Sistema matematicheskogo obespecheniya prochnostnykh raschetov prostranstvennykh konstruktsii. *Problemy prochnosti*, 1978, no. 11, pp. 59–61; no. 12, pp. 25–28. (in Russ.)
15. **Goncharenko I. E. et al.** Problemno-orientirovannye yaziki polzovatelya. Formalnoe opisanie. PROChNOST-75. Sistema matematicheskogo obespecheniya raschetov prostranstvennykh konstruktsii. Kiev, Respublikanskii fond algoritmov i programm AN USSR, 1975, vol. 4, 550 p. (in Russ.)
16. **Goncharenko I. E. et al.** Problemno-orientirovannye yaziki polzovatelya. Formalnoe opisanie. PROChNOST-75. Sistema matematicheskogo obespecheniya raschetov prostranstvennykh konstruktsii. Kiev, Respublikanskii fond algoritmov i programm AN USSR, 1975, vol. 5, 357 p. (in Russ.)
17. **Bestsenny Yu. G. et al.** Sistema avtomatizatsii kompleksnogo rascheta prostranstvennykh konstruktsii na ES EVM. In: Kompleksnyi raschet zdanii i sooruzhenii s primeneniem EVM. Kiev, KISI, 1978, pp. 55–58. (in Russ.)
18. **Skrim E., Roi Dj. R.** Avtomaticheskaya sistema kinematicheskogo analiza. In: Raschet uprugikh konstruktsii s ispolzovaniem EVM. In 2 vols. Trans from Eng., ed. by A. P. Filin. Leningrad, Sudostroenie, 1974, vol. 2, pp. 36–67. (in Russ.)
19. **Argyris J. H.** ASKA: automatic system for kinematic analysis – a universal system for structural analysis based on the matrix displacement (finite element) method. *Nucl. Eng. Des.*, 1969, no. 2, pp. 441–447.

20. **Launay P. et al.** The three-dimensional thermoelastic computer code “TITUS”. In: Prepr. 1st Int. Conf. Struct. Mech. React. Technol. Berlin, Amsterdam, Amsterdam e.a., 1971, no. 5, pp. M5-4/1–M5-4/21.
21. **Araldsen P. O., Egeland O.** General description of SESAM-69 super element structural analysis (Program) modules. *European Shipbuilding*, 1971, no. 2, pp. 21–35.
22. **Egeland O., Araldsen P. O.** SESAM-69 – A general purpose finite element method program. *Intern. J. of Computers and Structures*, 1974, no. 1, pp. 41–68.
23. **Araldsen P. O.** The application of the superelement method in analysis and design of ship structures and machinery components. In: National Symp. on Computerized structural Analysis and Design. Norway, March, 1972, pp. 2–93.
24. **Butler T. G., Michel D.** NASTRAN. A summary of the functions and capabilities of the NASA structural analysis computer system. Washington, 1971, 22 p. (NASA SP-260)
25. **McNeal R. H., McCormic C. W.** The NASTRAN computer program for structural analysis. *Comput. and Struct*., 1971, no. 1, pp. 32–35.
26. **Tocher J. L., Herness E. D.** A critical view of NASTRAN. *Numerical and Computer Methods in Structural Mechanics*, 1973, pp. 151–174.
27. **Chumachenko E. N. et al.** Matematicheskoe modelirovanie v nelineinoi mehanike (obzor programmnykh kompleksov dlya resheniya zadach modelirovaniya slozhnykh sistem). Moscow, 2009, 23 p. (in Russ.)
28. **Faas D., Vance J. M.** Interactive deformation through mesh-free stress analysis in virtual reality. *Mechanical Engineering Conference Presentations, Papers, and Proceedings*, 2008, p. 46.
29. **Frolov D.** Obzor vozmozhnostei ANSYS Mechanical dlya resheniya inzhenernykh zadach. *SAPR i grafika*, 2010, no. 11. (in Russ.) URL: http://www.caeexpert.ru/sites/default/files/obzor_vozmozhnostey_ansys_mechanical_dlya_resheniya_inzhenernyh_zadach.pdf.
30. **Chigarev A. V., Kravchuk A. S., Smalyuk A. F.** ANSYS dlya injenerov. Sprav. posobie. Moscow, Mashinostroenie-1, 2004, 512 p. (in Russ.)
31. **Cherpakov A. V. et al.** Modelirovanie kolebanii pri impulsnom vozdeistvii mnogosloinoi konstrukcii v komplekse Ansys. *Inzhenernyi vestnik Dona*, 2019, no. 6. (in Russ.) URL: http://ivdon.ru/ru/magazine/archive/n6y2019/6057.
32. **Selyakov M. Yu.** Otechestvennie i zarubejnie CAD/SAM sistemi. *Uspehi sovremennogo estestvoznaniya*, 2011, no. 7, pp. 193–197. (in Russ.)
33. **Ivanov S. E.** Intellektualnie programmnie kompleksi dlya tekhnicheskoi i tekhnologicheskoi podgotovki proizvodstva. Chast 5: Sistemy inzhenernogo rascheta i analiza detalei i sborochnykh edinits. Uchebno-metodicheskoe posobie. Ed. by D. D. Kulikov. St. Petersburg, SPbGU ITMO, 2011, 48 p. (in Russ.)
34. **Pavlov S.** CAE – tekhnologii v 2013 godu: obzor dostizhenii i analiz rynka. *CAD/CAM/CAE Observer*, 2014, no. 4 (88). (in Russ.) URL: http://www.cadcamcae.lv/N88/08-18.pdf.
35. **Kalinin A. V., Khvalin A. L.** Primenenie metoda konechnykh elementov v sovremennykh sistemakh avtomatizirovannogo proektirovaniya. *Geteromagnitnaya mikroelektronika*, 2019, no. 26, pp. 41–51. (in Russ.)
36. **Gorodetsky A. S. et al.** Metod konechnykh elementov: teoriya i chislennaya realizatsiya. Kiev, Fakt, 1997, 137 p. (in Russ.)
37. **Kirichevsky R. V.** Chislennoe modelirovanie temperaturnykh polei dissipativnogo razogreva konstruktsii iz elastomerov s treschinami. Kiev, Naukova Dumka, 1998, 120 p. (in Russ.)
38. **Mounir H., Nizar A., Borhen L., Benamara A., Deneux D.** FEM Simulation Based on CAD Model Simplification: A Comparison Study between the Hybrid Method and the Technique Using a Removing Details. *Design and Modeling of Mechanical Systems*, 2013, pp. 587–596. DOI 10.1007/978-3-642-37143-1_70
39. **Gorodetsky D. A. et al.** Programmnii kompleks LIRA-SAPR 2013. Uchebnoe posobie. Kiev, Moscow, Elektronnoe izdanie, 2013, 376 p. (in Russ.)

40. **Mezhuev V., Lavrik V., Ravi S.** Development and application of the problem-oriented language FORTU for the design of non-standard mechanical constructions. *Journal of the Serbian Society for Computational Mechanics*, 2015, no. 9 (2), pp. 1–9. DOI 10.5937/jsscm1502001M
41. **Kirichevsky V. V. et al.** Metod konechnykh elementov v vichislitelnom komplekse "MIRELA+". Kiev, Naukova Dumka, 2005, 403 p. (in Russ.)
42. **Kirichevsky V. V. et al.** Razvitie metoda konechnykh elementov i ego primenenie v SAPR. *Vestnik Zaporozhskogo nats. un-ta. Fiz.-mat. nauki*., 2006, no. 1, pp. 38–56. (in Russ.)

Информация об авторах

Аркадий Николаевич Соловьев, доктор физико-математических наук, профессор
Ростислав Викторович Киричевский, кандидат технических наук, доцент

Information about the Authors

Arkady N. Soloviev, Doctor of Sciences (Physics and Mathematics), Professor
Rostislav V. Kirichevsky, Candidate of Technical Sciences, Associate Professor

Статья поступила в редакцию 11.09.2021;
одобрена после рецензирования 01.12.2021; *принята к публикации 01.12.2021*
The article was submitted 11.09.2021;
approved after reviewing 01.12.2021; *accepted for publication 01.12.2021*

Научная статья

УДК 004.89:004.4
DOI 10.25205/1818-7900-2021-19-4-85-95

Разработка интеллектуального редактора SPARQL-запросов

Ирина Алексеевна Турова [1]
Игорь Сергеевич Постаногов [2]

[1, 2] Пермский государственный национальный исследовательский университет
Пермь, Россия

[1] turovaia@yandex.ru
[2] ipostanogov@outlook.com, https://orcid.org/0000-0002-7791-4789

Аннотация

Обсуждается вопрос создания редактора SPARQL-запросов – запросов к онтологиям в форматах RDF и OWL. Потребность в удобном и функциональном инструменте для составления SPARQL-запросов постоянно растет в связи с распространением концепции семантического веба. Приводится анализ возможностей, а также общих подходов к реализации существующих редакторов SPARQL-запросов. Предлагается концепция инструмента, одновременно предоставляющего интеллектуальные дополняющие подсказки, а также возможности построения визуального представления запроса и сравнения результатов выполнения запросов с использованием различных программ-ризонеров и без них. Языковая поддержка реализуется в соответствии с современным подходом к предоставлению инструментальной поддержки языков программирования, основанного на протоколе LSP. Приводятся скриншоты работы редактора.

Ключевые слова

SPARQL-запросы, онтологии, OWL, языковой сервер, LSP, ризонер, визуальное представление запроса

Для цитирования

Турова И. А., Постаногов И. С. Разработка интеллектуального редактора SPARQL-запросов // Вестник НГУ. Серия: Информационные технологии. 2021. Т. 19, № 4. С. 85–95. DOI 10.25205/1818-7900-2021-19-4-85-95

Development of Intelligent SPARQL Query Editor

Irina A. Turova [1], **Igor S. Postanogov** [2]

[1, 2] Perm State University
Perm, Russian Federation

[1] turovaia@yandex.ru
[2] ipostanogov@outlook.com, https://orcid.org/0000-0002-7791-4789

Abstract

The paper discusses the development of SPARQL query editor. This is an actual question because of the growth of Semantic Web data – the data presented in RDF/OWL formats. A comparative analysis of different types of editors and their main features is provided. In this paper, we propose a description of a SPARQL editor that combines three most useful features: intelligent completions, query visualization comparison of query results provided by different reasoners or without them. The editor provides SPARQL support as LSP service, this approach is considered a modern way to implement language support. This paper also presents the editor's screenshots.

Keywords

SPARQL queries, ontologies, OWL, language server, LSP, reasoner, query visualization

© Турова И. А., Постаногов И. С., 2021

For citation

Turova I. A., Postanogov I. S. Development of Intelligent SPARQL Query Editor. *Vestnik NSU. Series: Information Technologies*, 2021, vol. 19, no. 4, p. 85–95. (in Russ.) DOI 10.25205/1818-7900-2021-19-4-85-95

Введение

В настоящее время активно развиваются технологии искусственного интеллекта. Создаются интеллектуальные бизнес-решения, также интеллектуальные системы широко используются в образовательных целях. Технологии искусственного интеллекта являются одним из приоритетных направлений развития государства [1]. Востребованы как методы машинного обучения, так и интеллектуальные системы, основанные на знаниях.

Большую популярность в настоящее время приобрели системы, основанные на онтологиях, – одной из форм представления знаний. Microsoft использует онтологически управляемые решения для создания цифровых двойников умных городов [2]. Онтологии активно используются в медицине: многие системы поддержки принятия врачебных решений основываются на онтологиях [1].

Официальным стандартом для представления онтологий является формат OWL 2, который рекомендован консорциумом Всемирной паутины. SPARQL – язык запросов к онтологиям. Для работы со знаниями инженерам необходимо составлять запросы на языке SPARQL, позволяющие извлекать факты из онтологий, проверять истинность некоторых гипотез, добавлять новые факты в онтологию.

Статистические исследования [2; 3] SPARQL-запросов к разным точкам доступа SPARQL, в которых пользовательские запросы рассматриваются отдельно от программных, показывают, что пользовательские запросы структурно более сложные, чем программные. В работе рассматриваются возможности различных редакторов SPARQL-запросов, а также формулируются требования к собственному редактору SPARQL-запросов для удовлетворения максимального количества потребностей пользователей.

Обзор редакторов SPARQL-запросов

За последние десятилетия многие исследования были посвящены разработке различных редакторов для составления SPARQL-запросов. Наибольшее количество публикаций описывает визуальные редакторы, которые позволяют пользователям составлять запросы, добавляя на рабочую область графические элементы, соответствующие различным частям запроса.

Многие авторы, разрабатывая подобные редакторы, используют аналогичные по назначению определенным в SPARQL графические блоки, а не придумывают визуальный язык с нуля. Однако набор поддерживаемых возможностей обычно значительно меньше, чем в SPARQL. Например, редактор RDF Explorer [4] поддерживает составление запросов, состоящих только из основного графового шаблона запроса [3], где концепты могут быть заменены переменными. Авторы работы позиционируют RDF Explorer как редактор для пользователей, не являющихся экспертами.

Поскольку обычно визуальные редакторы SPARQL-запросов значительно ограничены в своей выразительности, более сложные запросы не могут быть сформулированы с их помощью. Хотя такие редакторы и подходят для начинающих пользователей, могут возникнуть трудности при постепенном изучении конструкций языка, поскольку пользователь в процес-

[1] Указ Президента РФ от 10 октября 2019 г. № 490 «О развитии искусственного интеллекта в Российской Федерации»

[2] URL: https://techcommunity.microsoft.com/t5/internet-of-things/smart-cities-ontology-for-digital-twins/ba-p/2166585

[3] URL: https://www.w3.org/TR/rdf-sparql-query/#BasicGraphPatterns

се обучения столкнется с необходимостью либо искать более выразительный визуальный интерфейс, либо переходить на текстовый язык запросов. Некоторые инструменты, например SPARQLinG [5] и NITELIGHT [6], по своей выразительности близки к полной спецификации SPARQL 1.0, но даже небольшие визуальные запросы выглядят громоздко, что затрудняет их использование начинающими пользователями, а для более продвинутых пользователей составление текстового запроса может оказаться предпочтительней.

Текстовые редакторы SPARQL-запросов не ограничивают пользователя в выразительности языка. Для начинающих пользователей могут быть реализованы дополняющие подсказки, которые упростят процесс составления запросов. Продвинутые пользователи смогут использовать любые возможности языка, даже если для некоторых из них не будет реализовано активной поддержки.

Большинство существующих текстовых редакторов, например YASGUI [7; 8] и основанный на нем Flint SPARQL-editor [4], предоставляют базовую поддержку при составлении SPARQL-запросов: подсветку и валидацию синтаксиса запроса. Помимо подсказок ключевых слов дополняющие подсказки обычно составляются на основе всех концептов и отношений онтологий. Редактор SparQLed [9] основан на редакторе Flint SPARQL-editor и предоставляет дополняющие подсказки на основе содержимого онтологии, положения вызова подсказки в запросе, а также уже присутствующих в запросе триплетов, но в настоящее время проект закрыт.

Многие инструменты для работы с онтологиями также позволяют составить и выполнить SPARQL-запрос к онтологиям, однако зачастую эти возможности достаточно ограничены. Среди таких инструментов можно выделить Protégé – самый популярный и широко используемый редактор онтологий (около 300 000 пользователей [5]), который обладает богатым, расширяемым за счет разработки и подключения плагинов набором функциональных возможностей. Protégé также имеет возможность составления SPARQL-запросов, что позволяет работать с онтологиями и SPARQL-запросами в единой интегрированной среде. Однако редактор SPARQL-запросов, предоставляемый Protégé, не реализует никакой поддержки пользователей в процессе составления запросов и представляет собой простое текстовое поле, что затрудняет разработку запросов.

Stardog Studio [6] – IDE для работы с графами знаний (в том числе и онтологиями) и SPARQL-запросами к ним. Stardog Studio имеет возможность составления SPARQL-запросов и предоставляет поддержку пользователей на этапе составления запросов – присутствует подсветка синтаксиса, валидация запроса с учетом синтаксиса и дополнение запроса ключевыми словами SPARQL и ранее использованными в запросе токенами (переменными, названиями элементов онтологии). Иначе говоря, при формировании подсказок не используются знания из целевой онтологии. В отличие от приведенных выше текстовых редакторов поддержка SPARQL выделена в модуль, отдельный от Stardog Studio, но поставляется вместе с ней. Этот модуль представляет собой языковой сервер, реализующий все функции для поддержки SPARQL и взаимодействующий с клиентом (Stardog Studio) по протоколу LSP [7] (Language Server Protocol).

Помимо концептов и отношений онтологии могут содержать аксиомы, которые могут использоваться для вывода новых фактов на основе присутствующих в онтологии. Ризонер – программное средство для вывода новых фактов из онтологии на основе аксиом. Ризонеры различаются набором поддерживаемых аксиом. Удобной возможностью для редактора SPARQL-запросов может стать сравнение результатов выполнения запросов к одной онтологии с использованием нескольких ризонеров. Существуют работы [10; 11], описывающие системы для автоматизированного сравнения показателей работы нескольких ризонеров

[4] URL: https://github.com/TSO-Openup/FlintSparqlEditor/.
[5] URL: https://protege.stanford.edu/community.php
[6] URL: https://www.stardog.com/studio/.
[7] URL: https://microsoft.github.io/language-server-protocol/.

на пользовательских онтологиях, но ни один из рассмотренных редакторов SPARQL-запросов не предоставляет такой возможности.

Среди рассмотренных редакторов не было найдено ни одного инструмента, одновременно предоставляющего возможности составления текстового SPARQL-запроса, просмотра визуального представления запроса и сравнения результатов выполнения запроса с разными ризонерами. Визуальные редакторы SPARQLinG и NITELIGHT предоставляют возможность конвертации визуального запроса в SPARQL, однако текстовый вариант запроса доступен только для чтения.

Концепция редактора SPARQL-запросов

На основании обзора рассмотренных ранее редакторов было решено разработать собственный текстовый редактор SPARQL-запросов, который должен иметь следующие возможности:

- дополняющие подсказки на основе контекста вызова и содержимого онтологии;
- визуальное представление запроса с возможностью исследования связанных элементов онтологии;
- сравнение результатов выполнения запроса к одной и той же онтологии с использованием различных ризонеров.

Ключевой особенностью разрабатываемого решения является наличие всех этих возможностей в одном инструменте. Кроме того, эти возможности будут вынесены в отдельные модули, которые в дальнейшем могут быть легко переиспользованы. Каждый модуль будет реализован в виде сервиса, клиентом для этих сервисов будет выступать расширение для редактора Visual Studio Code [8], в интерфейсе которого пользователю будет предложено работать со SPARQL-запросами.

Далее будут подробнее описаны особенности реализации модуля языкового сервера и правила составления дополняющих подсказок, а также модули визуального представления и сравнения ризонеров. В качестве сквозного примера рассматривается SPARQL-запрос к фрагменту географических данных онтологии Dbpedia (данные трех стран о столицах и официальных языках и восьми городов) для получения всех городов стран, в которых официальным языком является французский (рис. 1).

```
1  # sparql : http://localhost:3030/countries_ds
2  PREFIX countries: <http://www.semanticweb.org/irina/ontologies/2021/5/countries#>
3  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4
5  SELECT ?city
6  WHERE {
7    ?country a countries:Country .
8    ?city a countries:City .
9    ?country countries:city ?city .
10   ?lang a countries:Language .
11   ?lang rdfs:label "French"@en .
12   ?country countries:officialLanguage ?lang .
13 }
14
```

Рис. 1. Пример SPARQL-запроса
Fig. 1. Sample SPARQL query

[8] URL: https://code.visualstudio.com

Языковой сервер

Основной модуль редактора – языковой сервер, реализующий поддержку пользователя при составлении текстового SPARQL-запроса. Кроме генерации дополняющих подсказок этот модуль отвечает за подсветку и валидацию синтаксиса SPARQL-запроса. Для взаимодействия этого модуля с остальной системой можно использовать протокол LSP.

Language Server Protocol (LSP) – протокол для взаимодействия редакторов кода или IDE с языковыми серверами. Протокол основан на JSON-RPC – одном из протоколов удаленного вызова процедур – и определяет форматы сообщений для основных функций поддержки языков программирования. LSP поддерживается большинством современных редакторов кода и IDE.

В ходе проведенного исследования удалось найти один языковой сервер, реализующий поддержку SPARQL по протоколу LSP, – сервер, реализованный компанией «Stardog» в рамках разработки редактора Stardog Studio. Данное программное обеспечение предоставляет базовую поддержку языка SPARQL: подсветка и проверка синтаксиса, дополнение введенного запроса на основе синтаксиса (ключевыми словами) и ранее введенных токенов, однако отсутствуют подсказки не присутствующих в запросе элементов онтологии. Поскольку сервер обладает базовой структурой для реализации полной поддержки SPARQL, было решено реализовать собственный языковой сервер на основе существующего, а в качестве клиента для демонстрации использовать редактор Visual Studio Code.

Использование LSP в качестве протокола для коммуникации между языковым клиентом и языковым сервером позволит в дальнейшем переиспользовать языковой сервер во многих других редакторах, часть из которых использует LSP в качестве основной модели для подключения языковой поддержки (например, Visual Studio Code), другие имеют возможность подключения языковых серверов по протоколу LSP (например, Intellij IDEA, Sublime Text) с помощью плагинов для этих сред.

Дополняющие подсказки на основе содержимого онтологии

Наличие дополняющих подсказок в современных текстовых редакторах является неотъемлемой возможностью. Большинство рассмотренных выше редакторов реализует дополняющие подсказки ключевых слов SPARQL и элементов онтологии, но в основном без учета контекста. При отсутствии в редакторе SPARQL-запросов функции дополнения элементами онтологии пользователю приходится запоминать структуру онтологии либо параллельно с составлением запроса использовать средства для навигации по онтологии. Наличие дополняющих подсказок, основывающихся на структуре целевой онтологии, позволяет совместить эти два процесса. В работе [12] обсуждается необходимость интеллектуальных (использующих структуру онтологии) дополняющих подсказок для разных групп пользователей SPARQL.

Базовым содержимым SPARQL-запроса является основной графовый шаблон – набор триплетов, участвующих в запросе. Каждый триплет состоит из субъекта, предиката и объекта, в роли которых могут выступать элемент онтологии, переменная или константа (литерал). Для упрощения составления корректного графового шаблона необходимо поддержать как дополнение имеющейся части триплета недостающими элементами онтологии, непротиворечащими ее структуре и другим триплетам запроса, а также проверку каждого триплета из графового шаблона на соответствие некоторому факту в онтологии. Кроме этого, полезной будет подсказка о несвязности графового шаблона, так как, вероятнее всего, SPARQL-запрос с несвязанным графовым шаблоном был составлен по ошибке.

В зависимости от места вызова дополняющей подсказки в триплете предлагаемые для дополнения элементы должны отличаться. В качестве дополняющих элементов могут высту-

пать как элементы онтологии, так и переменные. Формирование дополняющих подсказок происходит в три этапа:

1) определение типа дополняющей подсказки в зависимости от места вызова;

2) формирование и выполнение вспомогательного SPARQL-запроса для определения первичных дополняющих подсказок;

3) формирование вторичных дополняющих подсказок, содержащих переменные, на основе результатов вспомогательного SPARQL-запроса.

Рассмотрим алгоритм составления дополняющих подсказок для субъекта последнего триплета рассматриваемого в разделе SPARQL-запроса (рис. 2).

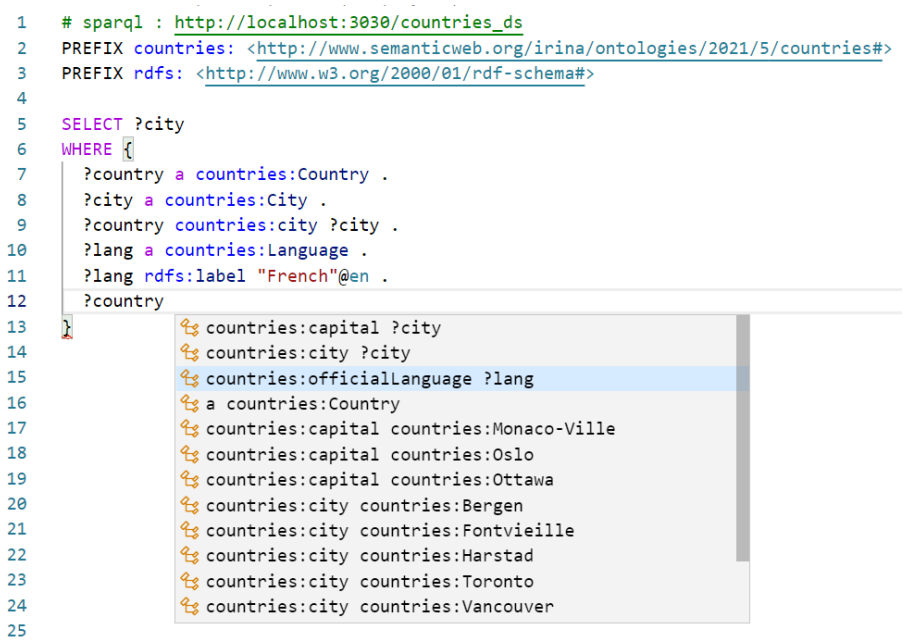


Рис. 2. Дополняющие подсказки
Fig. 2. Completion hints

При вызове подсказки курсор находится в 12-й строке после переменной *?country*, перед которой находится токен «.» (точка), который означает конец триплета. Преобразование текста запроса в последовательность токенов, а также определение их типов осуществляется с помощью лексического анализатора библиотеки Millan, разработанной компанией «Stardog» для своего языкового сервера и основанной на Chevrotain [9] – популярной библиотеке для создания парсеров синтаксических деревьев по заданной грамматике. Таким образом, можно понять, что в текущем триплете присутствует только субъект в виде переменной, а предикат и объект требуют дополнения.

Так как в данном примере на месте субъекта стоит переменная, а не конкретный элемент онтологии, то в общем случае дополняющими элементами могут быть все присутствующие в онтологии комбинации предикат-объект. Однако выше в запросе присутствует триплет *?country a countries:Country*, определяющий тип допустимых значений переменной *?country*. Таким образом, поиск может быть сужен до тех комбинаций предикат-объект, субъектом для которых выступает элемент типа *Country*. Поскольку дополнение SPARQL-запроса на основе структуры онтологии требует получения из онтологии элементов, отвечающих некоторым критериям, удобно для формирования подсказок использовать SPARQL-запросы, сформиро-

[9] URL: https://sap.github.io/chevrotain/docs/.

ванные программно и выполняемые в фоновом режиме, к той же самой онтологии, к которой пользователь составляет запрос в редакторе. Помимо поиска комбинаций предикат-объект также с помощью вспомогательного SPARQL-запроса определяется и тип объекта для формирования вторичных подсказок (рис. 3).

```
1  PREFIX countries: <http://www.semanticweb.org/irina/ontologies/2021/5/countries#>
2
3  SELECT DISTINCT ?p ?o ?oc
4  WHERE {
5    ?s ?p ?o .
6    ?s a countries:Country .
7    ?o a ?oc
8  }
```

Рис. 3. Вспомогательный SPARQL-запрос
Fig. 3. Service SPARQL query

Приведем результаты выполнения вспомогательного SPARQL-запроса (единообразные результаты заменены многоточием):

?p	?o	?oc
capital	Monaco-Ville	City
capital	Oslo	City
capital	…	City
city	Bergen	City
city	Toronto	City
city	…	City
officialLanguage	Norwegian	Language
officialLanguage	French	Language
officialLanguage	…	Language
type	Country	Class

Первичные дополняющие подсказки составлены на основе уникальных пар значений переменных *?p* и *?o*. Вторичные дополняющие подсказки в данном примере составляются путем подмены объекта (значения переменной *?o*) в первичных подсказках на подходящую переменную. Для определения подходящих переменных используется тип значения объекта (значение переменной *?oc*), и среди всех используемых в запросе переменных выбираются те, значения которых ограничены нужным типом. В рассматриваемом примере на основе пар из предиката *capital* и объектов типа *City* будет составлена дополняющая подсказка *capital ?city*, так как значение переменной *?city* ограничено типом *City* с помощью триплета *?city a countries:City*. Аналогично будет составлена подсказка *officialLanguage ?lang*.

Визуальное представление

Помимо дополняющих подсказок и проверки корректности фактов в запросе для поиска семантических ошибок удобно использовать графическое представление запроса. Тело SPARQL-запросов типа SELECT представляет собой графовый шаблон, поэтому он естественным образом представляется в виде графа, где вершинам и ребрам в соответствие поставлены элементы онтологии или переменные из запроса. Поскольку протокол LSP не поддерживает работу с графическими элементами, передача информации о графе на клиент происходит посредством протокола HTTP.

Наличие в редакторе дополняющих подсказок позволяет составлять запросы, не обращаясь за информацией о структуре онтологии к редакторам онтологий. Однако полезной возможностью может стать получение дополнительной информации о структуре онтологии из визуального представления без необходимости редактировать SPARQL-запрос с целью вызова необходимой подсказки. Поскольку онтология может содержать большое количество концептов, которые могут быть достаточно сильно связаны между собой отношениями, чтобы не перегружать визуальное представление, выводятся только концепты и отношения, связанные с выбранной пользователем вершиной, которая может быть как вершиной, так и элементом онтологии (рис. 4). В случае если выбран элемент онтологии, в качестве соседних отображаются концепты и отношения, связанные с этим элементом. А в случае если выбрана переменная, – концепты и отношения, связанные с любым из возможных значений этой переменной.

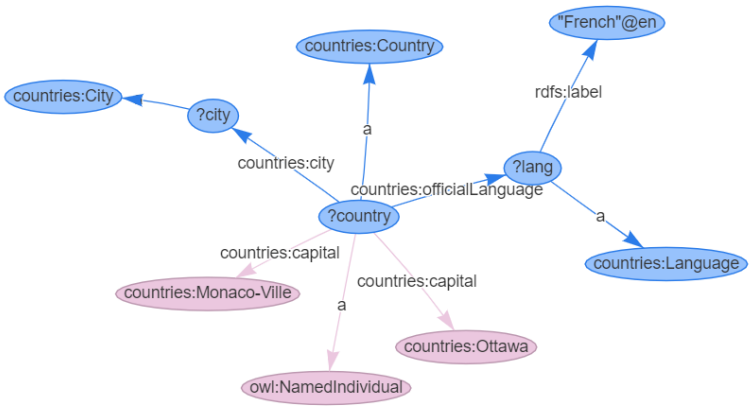


Рис. 4. Визуальное представление SPARQL-запроса
Fig. 4. Vizualization of SPARQL query

Сравнение результатов нескольких ризонеров

При выполнении SPARQL-запроса к онтологии в результат могут попасть только те факты, которые непосредственно присутствуют в описании онтологии. Чтобы учесть факты, которые могут быть выведены с помощью аксиом онтологии, необходимо использовать ризонер. Ризонеры различаются набором поддерживаемых типов аксиом, временем выполнения, подходом к выводу новых фактов (жадный вывод с кэшированием или ленивый вывод по необходимости). Для сравнения нескольких ризонеров с целью выбора ризонера, наиболее подходящего под задачу, а также в учебных целях для демонстрации различных техник логического вывода удобной возможностью редактора SPARQL-запросов представляется выполнение запроса к онтологии и сравнение результатов, полученных несколькими ризонерами.

Для реализации этой возможности необходимо иметь несколько точек доступа SPARQL: для исходной онтологии и по одной для каждого ризонера, который будет работать с этой онтологией. Как для удаленных онтологий (если они открыто доступны), так и для локальных задача подключения к ним ризонеров решается созданием собственных точек доступа с необходимой онтологией и ризонером.

После выполнения запросов ко всем точкам доступа необходимо предоставить возможность сравнения результатов. Поскольку результат SELECT-запросов можно рассматривать как таблицу, то задача сводится к сравнению нескольких таблиц. Наглядным образом пред-

ставить сравнение нескольких таблиц достаточно сложно, поэтому было решено организовать попарное сравнение результатов различных ризонеров и исходной онтологии.

При выполнении рассматриваемого SPARQL-запроса к точке доступа онтологии без использования ризонера в результат не попадут города-столицы, так как отношение *capital* наследуется от отношения *city*, а в онтологии для столиц отсутствуют факты с использованием отношения *city*. Однако при использовании ризонера города-столицы будут присутствовать в результате, так как ризонер учитывает транзитивность отношения наследования. В таблице выводится результат сравнения результатов, полученных с ризонером (все пять строк), относительно результатов без ризонера (три строки) (рис. 5).

Select one or two tables to compare: countries_ds | countries_ds_OWLFBRule | Daff table

@@	city
	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Fontvieille
+++	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Monaco-Ville
+++	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Ottawa
	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Toronto
	http://www.semanticweb.org/irina/ontologies/2021/5/countries#Vancouver

Рис. 5. Сравнение результатов ризонеров
Fig. 5. Comparison of reasoners results

Архитектура редактора SPARQL-запросов

Полная система состоит из языкового клиента, языкового сервера и точек доступа SPARQL (рис. 6). Сторонние модули отмечены пунктирными границами, а разработанные самостоятельно – сплошными. Модули языкового сервера не зависят от используемого в качестве клиента редактора кода или IDE. При необходимости использовать другой инструмент в качестве клиента необходимо будет реализовать модули, объединенные в языковой клиент, с использованием API другого редактора.

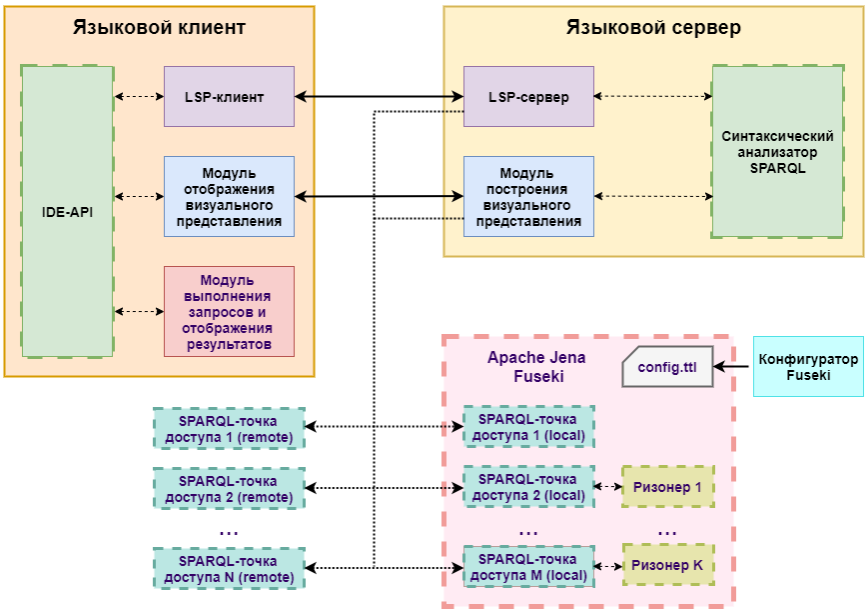


Рис. 6. Архитектура системы
Fig. 6. System architecture

Редактор может работать с любыми удаленными точками доступа SPARQL, такими как Dbpedia, Wikidata и др. Для работы с локальными онтологиями используется SPARQL-сервер Fuseki, который позволяет настроить несколько точек доступа с разными онтологиями и ризонерами. Для упрощения настройки Fuseki было разработано приложение для генерации файла конфигураций Fuseki, позволяющее в графическом интерфейсе выбрать файлы с онтологиями и необходимые ризонеры.

Заключение

В работе был проведен обзор редакторов SPARQL-запросов, проанализированы возможности различных типов редакторов. На основании обзора сформулированы требования к собственному редактору.

Разработанный редактор реализует три основные возможности для работы со SPARQL: дополняющие подсказки, визуальное представление запроса и сравнение результатов выполнения. Ключевой особенностью разработанного редактора является его модульность, каждый модуль реализует одну из возможностей. Такое архитектурное решение позволяет переиспользовать отдельные модули в других инструментах.

Разработанный редактор может быть использован как в образовательных целях на занятиях, посвященных работе с онтологиями, изучению SPARQL или работе ризонеров, так и в практических и исследовательских целях.

Список литературы / References

1. **Dissanayake P. I., Colicchio T. K., Cimino J. J.** Using clinical reasoning ontologies to make smarter clinical decision support systems: a systematic review and data synthesis. *Journal of the American Medical Informatics Association*, 2020, vol. 27, no. 1, pp. 159–174. DOI 10.1093/jamia/ocz169
2. **Rietveld L., Hoekstra R.** Man vs machine: Differences in SPARQL queries. In: Proceedings of the 4th USEWOD Workshop on Usage Analysis and the Web of Data, ESWC, 2014.
3. **Warren P., Mulholland P.** Using SPARQL – the practitioners' viewpoint. In: European Knowledge Acquisition Workshop. Cham., 2018, pp. 485–500. DOI 10.1007/978-3-030-03667-6_31
4. **Vargas H., Buil-Aranda C., Hogan A., López C.** RDF Explorer: A Visual SPARQL Query Builder. In: International Semantic Web Conference. Cham., 2019, pp. 647–663. DOI 10.1007/978-3-030-30793-6_37
5. **Hogenboom F., Milea V., Frasincar F., Kaymak U.** RDF-GL: A SPARQLBased Graphical Query Language for RDF. In: Emergent Web Intelligence: Advanced Information Retrieval, 2010, pp. 87–116. DOI 10.1007/978-1-84996-074-8_4
6. **Smart R., Russell A., Braines D., Kalfoglou Y., Bao J., Shadbolt R.** A Visual Approach to Semantic Query Design Using a Web-Based Graphical Query Designer. In: Knowledge Engineering and Knowledge Management (EKAW), 2008, pp. 275–291. DOI 10.1007/978-3-540-87696-0_25
7. **Rietveld L., Hoekstra R.** YASGUI: not just another SPARQL client. In: Extended Semantic Web Conference. Berlin, 2013, pp. 78–86. DOI 10.1007/978-3-642-41242-4_7
8. **Rietveld L., Hoekstra R.** The YASGUI family of SPARQL clients. *Semantic Web*, 2017, vol. 8, no. 3, pp. 373–383. DOI 10.3233/sw-150197
9. **Campinas S., Perry T. E., Ceccarelli D., Delbru R., Tummarello G.** Introducing RDF graph summary with application to assisted SPARQL formulation. In: 23rd International Workshop on Database and Expert Systems Applications, 2012, pp. 261–266. DOI 10.1109/dexa.2012.38

10. **Gardiner T., Horrocks I., Tsarkov D.** Automated benchmarking of description logic reasoners. In: Proceedings of the International Workshop on Description Logics (06) CEUR, 2006, vol. 189, pp. 167–174.
11. **Alaya N., Yahia S.B., Lamolle M.** RakSOR: Ranking of ontology reasoners based on predicted performances. In: IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), 2016, pp. 1076–1083. DOI 10.1109/ictai.2016.0165
12. **Rafes K., Abiteboul S., Cohen-Boulakia S., Rance B.** Designing scientific SPARQL queries using autocompletion by snippets. In: IEEE 14th International Conference on e-Science, 2018, pp. 234–244. DOI 10.1109/escience.2018.00038

Информация об авторах

Ирина Алексеевна Турова, студентка магистратуры
Игорь Сергеевич Постаногов, старший преподаватель

Information about the Authors

Irina A. Turova, Master's Student
Igor S. Postanogov, Senior Lecturer

Статья поступила в редакцию 09.09.2021;
одобрена после рецензирования 01.11.2021; *принята к публикации 01.12.2021*
The article was submitted 09.09.2021;
approved after reviewing 01.11.2021; *accepted for publication 01.12.2021*

Правила оформления текста рукописи

Авторы представляют статьи на русском или английском языке объемом от 0,5 авторского листа (20 тыс. знаков) до 1 авторского листа (40 тыс. знаков), включая иллюстрации (1 иллюстрация форматом 190 × 270 мм = 1/6 авторского листа, или 6,7 тыс. знаков). Публикации, превышающие указанный объем, допускаются к рассмотрению только после индивидуального согласования с редакцией журнала.

Текст рукописи должен быть представлен в редколлегию в виде файла MS Word (.doc, .docx). Гарнитура Times New Roman, размер шрифта 11, межстрочный интервал 1, размеры полей – стандартные значения текстового редактора. Форматирование – выравнивание по ширине страницы, переносы слов включены, каждый новый абзац начинается с красной строки. Не допускается ручное форматирование абзацев (пробелами, лишними переводами строк, разрывами страниц).

Структура статьи

- Индекс УДК (универсальной десятичной классификации). Выравнивание по левому краю
- Название статьи. Выравнивание по центру, полужирный шрифт
- ФИО авторов (полностью). Выравнивание по центру, полужирный шрифт
- Места работы всех авторов. Выравнивание по центру, курсив
- Адреса электронной почты, ORCID авторов
- Аннотация статьи
- Ключевые слова, не более 10
- Благодарности, сведения о финансовой поддержке
- Название статьи **на английском языке**. Выравнивание по центру, полужирный шрифт
- ФИО авторов **на английском языке** (полностью). Выравнивание по центру, полужирный шрифт
- Места работы авторов **на английском языке**. Выравнивание по центру, курсив
- Аннотация статьи **на английском языке (Abstract),** 200–250 слов
- Ключевые слова **на английском языке (Keywords)**, не более 10
- Благодарности, сведения о финансовой поддержке **на английском языке**, если есть соответствующий раздел на русском языке **(Acknowledgements)**
- Основной текст
- Список литературы / **References**
- Сведения об авторах

Требования к оформлению основного текста и иллюстративных материалов

Основной текст должен быть представлен в структурированном виде, рекомендуется использовать подзаголовки – например: Введение, Методика..., Выводы, Результаты, Заключение.

Подзаголовки отделяются и набираются полужирным шрифтом. В целях выделения частей текста и отдельных слов и словосочетаний допускается использование курсива или полужирного шрифта. Подчеркивание, разрядка, изменение основного кегля и выделение цветом не используются.

Иллюстрации к рукописи статьи должны быть приложены в виде отдельных файлов. При этом в тексте должно содержаться включенное изображение с указанием имени файла. Все иллюстрации, содержащие схемы, графики, алгоритмы и т. п., должны быть представлены в векторном виде (.ai, .eps, .cdr). Скриншоты и другие растровые изображения должны быть представлены в максимально высоком качестве, без каких-либо потерь и искажений (.jpg, .tif). Все иллюстрации должны иметь подрисуночную подпись – свое название. Надписи к таблицам и подписи к иллюстрациям приводятся **на двух языках (русском и английском).**

Примеры:

Рис. 1. Диаграмма производительности...
Fig. 1. Performance diagram...

Таблица 1

Сравнение алгоритмов...

Table 1

Comparison of algorithms...

Нумерация последовательная и неразрывная от начала статьи. Не допускается использование других наименований, кроме «Рис.» / «Fig.», «Таблица» / «Table», и усложнение нумерации (например, «Рис. 3.2.»). Ссылка на иллюстрацию в тексте должна быть приведена в круглых скобках, например: (рис. 1), (табл. 1).

Формулы должны быть набраны с использованием редактора MathType либо встроенного редактора формул MS Word. Кегль основных символов – 11, греческие символы набираются прямым шрифтом, латинские – курсивом. Нумеруются только те формулы, на которые автор ссылается в тексте.

Abstract

Аннотация статьи на английском языке (Abstract) не должна быть дословным переводом русскоязычной аннотации. Раздел Abstract, как и основной текст, должен быть структурирован, в нем должно содержаться описание цели работы, методов исследования, научной значимости, выводов / результатов. Требуется качественный перевод на английский язык (при необходимости просим авторов обращаться к профессиональным переводчикам). **Объем Abstract 200–250 слов.**

Список литературы / References

Список литературы и список литературы на английском языке (References) размещаются в общем разделе. Рекомендуемое количество цитируемых в статье источников – не менее 10, в список желательно включать ссылки на актуальные работы по теме исследования, особенно в иностранных периодических изданиях.

В тексте статьи ссылки на литературу указываются цифрами в квадратных скобках, при необходимости указываются номера страниц, например: [2; 3. С. 15].

Список литературы нумеруется в порядке цитирования и оформляется в соответствии с ГОСТ Р 7.0.5-2008 на библиографическое описание (знаки тире в описании опускаются). Ссылки на неопубликованные работы, а также на Интернет-ресурсы (кроме электронных изданий, поддающихся библиографическому описанию) оформляются в виде сноски.

В Список литературы ссылки на источники следует включать на оригинальном языке опубликования. Каждый источник должен быть также оформлен на английском языке (References) по международному стандарту для публикаций в области информатики IEEE Style со следующими отличиями:

- инициалы авторов указываются после фамилии;
- название статьи не берется в кавычки, отделяется точкой;
- отсутствует союз «and» перед фамилией последнего автора;
- в диапазоне страниц – удвоенная «p» (например, «pp. 2–9»);

- год издания указывается после места издания (для книг) и сразу после названия журнала (для периодики).

Перевод источника на английский язык:

- если источник имеет выходные данные на английском языке, то для формирования References **следует использовать именно эти данные**;
- если оригинальная публикация не содержит выходных данных на английском языке, то допускается транслитерация названия материала на латинский алфавит в сочетании с переводом на английский язык в квадратных скобках. В конце описания указывается, на каком языке написана эта работа, например, (in Russ.). При транслитерации можно воспользоваться Интернет-ресурсом http://ru.translit.ru/, рекомендуется выбрать стандарт BSI. Место издания не транслитерируется, указывается полностью на английском языке, например: Moscow. Название издательства / издателя, как правило, транслитерируется. Для журналов, у которых есть официальное название на английском языке, – использовать его (проверить на сайте журнала, или, например, в библиотеке WorldCat), если названия на английском языке нет, использовать транслитерацию по системе BSI. Не следует самостоятельно переводить названия журналов.

Если у цитируемого источника есть **цифровой идентификатор DOI** (https://search.crossref.org), его требуется обязательно указывать в конце библиографической ссылки.

Примеры оформления ссылок. Каждый источник в том же пункте дублируется на английском языке (References).

Источник на русском языке, перевод на английский доступен в метаданных статьи

1. **Журавлев С. С., Рудометов С. В., Окольнишников В. В., Шакиров С. Р.** Применение модельно-ориентированного проектирования к созданию АСУ ТП опасных промышленных объектов // Вестник НГУ. Серия: Информационные технологии. 2018. Т. 16, № 4. С. 56–67. DOI 10.25205/1818-7900-2018-16-4-56-67

Zhuravlev S. S., Rudometov S. V., Okolnishnikov V. V., Shakirov S. R. Model-Based Design Approach for Development Process Control Systems of Hazardous Industrial Facilities. *Vestnik NSU. Series: Information Technologies*, 2018, vol. 16, no. 4, pp. 56–67. (in Russ.) DOI 10.25205/1818-7900-2018-16-4-56-67

Источник на английском языке. Оформляем согласно требованиям для References. Приводим только 1 раз.

2. **Telnov V. I.** Optimization of the Beam Crossing Angle at the ILC for E + e- and yy Collisions. *Journal of Instrumentation*, 2018, vol. 13, no. 03, pp. P03020–P03020. DOI 10.1088/1748-0221/13/03/p03020

Метаданные источника доступны только на русском языке

3. **Жижимов О. Л., Федотов А. М., Шокин Ю. И.** Технологическая платформа массовой интеграции гетерогенных данных // Вестник НГУ. Серия: Информационные технологии. 2013. Т. 11, вып. 1. С. 24–41.

Zhizhimov O. L., Fedotov A. M., Shokin Yu. I. Tekhnologicheskaya platforma massovoi integratsii geterogennykh dannykh [Technology Platform for the Mass Integration of Heterogeneous Data]. *Vestnik NSU. Series: Information Technologies*, 2013, vol. 11, no. 1, pp. 24–41. (in Russ.)

Сведения об авторах

Последний раздел статьи – информация об авторе / авторах **на русском и английском языках**:

- ФИО полностью, ученая степень, ученое звание;
- идентификаторы автора, такие как ResearcherID (всем авторам рекомендуется использовать данные сервисы для ведения актуального списка своих публикаций);

- контактный телефон (не публикуется).

Если статья представляется на английском языке, необходимо приложить перевод на русский язык названия, аннотации, ключевых слов, сведений об авторе.

Доставка материалов

Материалы предоставляются в редакцию по электронной почте inftech@vestnik.nsu.ru.

Порядок рецензирования

Все статьи сначала проходят проверку на заимствование и только после этого отправляются на рецензирование. Редакционный совет не допускает к публикации материал, если имеется достаточно оснований полагать, что он является плагиатом.

Тип рецензирования статей – двухуровневое, одностороннее анонимное («слепое»).

Для каждой статьи редколлегией выбираются рецензенты, научная деятельность которых связана с темой представленного материала. Ответственный секретарь журнала обращается к ним с просьбой дать экспертную оценку статье либо помочь организовать рецензирование.

Рецензии для журнала «Вестник НГУ. Серия: Информационные технологии» составляются по единой схеме и подразумевают оценку по следующим критериям: соответствие тематике журнала, оригинальность и значимость результатов, качество изложения материала.

Заполненный бланк рецензии высылается на электронный адрес редакции. В зависимости от экспертных заключений статья может быть принята редакционным советом к опубликованию, рекомендована автору к доработке (с последующим повторным рецензированием либо без него) или отклонена (с предоставлением автору мотивированного отказа). Автору на электронный адрес высылается текст рецензии без указания ФИО рецензента и его контактных данных.

Все рецензии хранятся в редакции журнала не менее 5 лет. Редколлегия журнала обязуется при поступлении соответствующего запроса направлять копии рецензий в Министерство науки и высшего образования Российской Федерации.