Программные системы: теория и приложения

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ПРОГРАММНОЕ И АППАРАТНОЕ ОБЕСПЕЧЕНИЕ РАСПРЕДЕЛЕННЫХ И СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМ

УДК 004.272.45
doi 10.25209/2079-3316-2023-14-4-3-23

Организация взаимодействия активных объектов однородных цифровых структур

Геннадий Георгиевич **Стецюра**✉
Институт проблем управления имени В. А. Трапезникова РАН, Москва, Россия
✉*gstetsura@mail.ru*

Аннотация. В статье расширены возможности взаимодействия активных устройств (объектов) однородных цифровых систем. Система состоит из объектов, которые расположены в пределах не более нескольких десятков метров и организуют свое взаимодействие только собственными средствами. Объекты могут быть стационарными и мобильными с произвольным и изменяемым во времени взаимным расположением объектов. Однородность системы означает отсутствие внешнего управления и равные возможности объектов в организации их взаимодействия. Связи между объектами беспроводные с использованием оптических или радиосигналов. Сигналы любого объекта-источника непосредственно поступают ко всем объектам-приемникам. Объект получает право передачи сигналов детерминировано в соответствии со значением его приоритета, задаваемым динамически или статически. Предложенные структура и способы взаимодействия позволяют объектам кроме обычного для распределенных систем обмена сообщениями выполнять распределенные групповые (ассоциативные) операции. В них объекты одновременно устраняют группу конфликтов доступа к общему каналу обмена данными, определяют состояние всех объектов системы и синхронно выполняют совместные действия объектов, реагируя на непредвиденно изменяющиеся внешние условия. Группа одновременно участвующих в групповой операции объектов выбирается с указанием набора критериев, которым должны обладать объекты. Однородность системы существенно упрощает ее техническую реализацию, но включение в систему неоднородности при усложнении системы ускоряет выполнение групповых операций. Поэтому однородную систему предлагается использовать в основном для связи между активными периферийными устройствами и связи этих устройств с более сложными компьютерами кластера.

Ключевые слова и фразы: компьютерные кластеры, децентрализованное управление, быстрая синхронизация цифровых объектов, внутрисетевые вычисления, распределенные групповые (ассоциативные) операции

Для цитирования: Стецюра Г. Г. *Организация взаимодействия активных объектов однородных цифровых структур* // Программные системы: теория и приложения. 2023. **Т. 14**. № 4(59). С. 3–23. https://psta.psiras.ru/read/psta2023_4_3-23.pdf

Введение

Статья рассматривает организацию взаимодействия компьютеров и других содержащих цифровые средства активных устройств, например, периферийных устройств компьютеров, объединенных в однородную структуру. Далее всех активных участников структуры назовем объектами. Активность объектов означает, что они способны по собственной инициативе взаимодействовать между всеми объектами системы. Однородность структуры подразумевает равные возможности объектов при взаимодействии и отсутствие внешнего устройства управления взаимодействием. В состав объектов могут входить также цифровые устройства, взаимодействующие с внешней средой.

Объекты могут быть стационарными и мобильными. В обоих вариантах каналы связи между объектами беспроводные с обменом по ним оптическими или радиосигналами.

Сигналы любого объекта-источника непосредственно поступают ко всем объектам-приемникам. Управление передачей сигналов детерминировано. Объект-источник получает право передачи сигналов с учетом значения его динамического или статического приоритета.

Использование беспроводного канала связи не только в мобильных, но и в стационарных системах объясняется расширением функций предлагаемых средств взаимодействия объектов по сравнению с функциями известных средств связи.

Сейчас задача сетевых средств компьютерных систем устройств состоит в доставке сообщений источников приемникам. Но от средств доставки сообщений можно перейти к средствам *взаимодействия*, выполняющим дополнительно управление совместными действиями объектов и распределенные вычисления. В статье с ориентацией на однородные системы показано, что ряд таких практически широко востребованных функций реализуется простыми средствами и дает существенное их ускорение за счет одновременной доставки сообщений группы источников группе приемников.

Автор настоящей статьи в 2022 г. в статье [1] показал возможность создания средств взаимодействия с указанными выше расширенными возможностями для сложных суперкомпьютерных систем. Особенность их в том, что в дополнение к передаче сообщений группа объектов системы с высокой точностью *одновременно* получает и обрабатывает одноименные разряды сообщений других групп объектов без использования компьютеров. Время выполнения таких распределенных операций не

зависит от количества участвующих в них объектов или зависит логарифмически. Показано, что эти возможности существенно ускоряют ряд важных операций распределенного управления совместными действиями компьютеров распределенных систем и распределенных вычислений. В результате средства передачи сообщений дополнительно действуют как средства управления и вычислений. Для этого систему потребовалось сделать *неоднородной*, ввести дополнительно группу не имеющих вычислительных средств ретрансляторов сигналов, в которых и выполняется распределенная обработка данных, находящихся в сообщениях.

Такой результат получен в результате точного измерения удаленности группы объектов системы от ретранслятора. Высокую точность измерений дает применение разработанного в CERN высокоточного способа измерения [2], стандартизованного IEEE в 2019 г. [3].

Взаимодействие в статье [1] включает управление порядком обмена сообщениями и ряд операций по децентрализованной распределенной обработке данных непосредственно средствами взаимодействия объектов. Объекты выбирают поочередную передачу сообщений или передачу с учетом приоритетов источников. Средства взаимодействия детерминировано устраняют конфликты при одновременной попытке передать сообщения, восстанавливают управляемость системой при отключении ряда объектов. Кроме этого, они выполняют ряд видов распределенной обработки данных. Синхронизация действий объектов выполняется непосредственно объектами.

Существенно подчеркнуть отличие задач, решенных в CERN, от синхронизации взаимодействия объектов в статье. В CERN измерения точно согласовывают показания часов группы объектов без обращения к внешнему эталону времени. После этого одно из устройств задает группе момент времени для начала совместных действий объектов. Но синхронизация часов не решает другую очень востребованную задачу. Совместные действия группы *однородных* объектов должны начаться с возможно меньшей задержкой после возникновения событий, потребовавших выполнения действий объектов. Это основное требование для любых средств управления и основная функция обратной связи.

Приведенное решение легко перенести на компьютерный кластер простым сведением группы ретрансляторов к единственному ретранслятору. Однако для взаимодействия простых объектов и компьютеров с простыми объектами требуется дополнение – система взаимодействующих объектов должна быть однородной, не содержать ретрансляторы, и быть технически просто реализуемой. Требование однородности особенно существенно

для взаимодействия стационарных объектов с мобильными объектами и мобильных объектов между собой, так как для них существенно усложняется использование выделенных ретрансляторов, обрабатывающих данные.

В предлагаемой статье получена существенно более простая реализация по сравнению с решениями [1], устранены неоднородные компоненты систем. При этом скорость передачи данных практически не уменьшается, но распределенные вычисления выполняются существенно медленнее, что для областей применения однородных решений допустимо.

Таким образом, в статье [1] и настоящей статье рассмотрены цифровые системы, существенно различающиеся по сложности и точности синхронизации.

В статье предполагается, что операции взаимодействия будут выполняться без вмешательства во внутреннюю структуру объединяемых устройств, например, средствами интерфейсных (сетевых) карт объектов. Несмотря на основную ориентацию статьи на взаимодействие с простыми распределенными объектами, в ней показана возможность применения групповых операций для связей в пределах отдельного чипа, печатной платы.

Сделаем несколько замечаний. Для связи персональных компьютеров самым распространенным средством взаимодействия служит Ethernet. Он не может применяться для выполнения требующих синхронизации распределенных операций, так как использует случайный способ доступа, причем доступ предоставляется одновременно не группе, а единственному источнику. Известен и широко применяется детерминированный способ взаимодействия объектов – приборный интерфейс I^2C и его развитие I^3C фирмы Philips. Фирма позиционирует I^2C как интерфейс для средств автоматизации, но он по своей структуре мог бы использоваться для взаимодействия компьютерных объектов. Например, подобный интерфейс был разработан и опубликован как «Децентрализованное приоритетное управление (ДПУ)» в Институте проблем управления РАН в 1970 г. за 12 лет до I^2C и применялся в АСУ ТП для выполнения распределенных вычислений (см. раздел 2.3 статьи). Но в ДПУ множество объектов было упорядочено – объекты соединялись проводным каналомв цепочку. В рассматриваемой статье и в [1] требовалось множество объектов не упорядочивать в пространстве, что было получено, но существенно усложнило оборудование.

Для связи объектов вычислительной техники применяются многие средства, известный пример – InfiniBand, но они не ориентированы

на одновременную синхронную доставку сообщений группы источников каждому приемнику из группы приемников, требуемую в настоящей статье.

Результаты статьи распределены следующим образом. В разделе 1 рассмотрена структура связей в однородной системе. В разделе 2 показан переход от асинхронной к синхронной системе взаимодействия объектов. Указаны все компоненты выполнения этого перехода. Раздел 3 рассматривает специальное представление данных – логические шкалы. В разделе 4 показано применение ассоциативного управления. Раздел 5 посвящен системам с малыми расстояниями между объектами – системам, размещенным на плате и на чипе.

1. Организация связей в однородной системе объектов

Так как предполагается, что объекты могут быть не только стационарными, но и мобильными, то основным способом обмена сообщениями между объектами будет обмен оптическими или радиосигналами по беспроводным каналам связи.

Система связей объектов имеет очень простую структуру (рисунок 1).

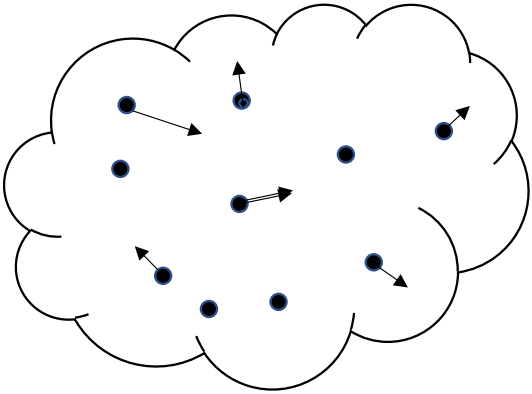


Рисунок 1. Однородная структура связей объектов

Здесь показана система взаимодействующих объектов – компьютеров кластера и (или) активных периферийных устройств. Допускается взаимодействие групп таких устройств в любых сочетаниях. Произвольная часть группы объектов стационарная, другие объекты могут перемещаться в произвольных направлениях с разными скоростями в пределах заданных границ системы.

Для взаимодействия объект должен иметь специализированное средство связи –сетевую карту, приспособленную для управления обменом сообщениями в соответствии с предлагаемыми в статье правилами. Предполагается, что сетевая карта имеет источник сигналов объектов и их приемник. Периферийные устройства активны и по своей инициативе обращаются к компьютерам кластера и другим периферийным устройствам. Отсутствие внешних посредников означает, что сигнал любого источника начнет поступать к любому приемнику объекта через интервал времени не больше времени переноса сигнала между наиболее взаимно удаленными объектами $T = L/c$. Здесь L – расстояние между этими объектами, c – скорость света.

В однородной структуре невозможно создать разброс начала реакции объектов на поступающие сообщения меньше T. Для многих применений – это приемлемое значение. Например, при $L = 30$ м, $T \approx 100$ нс. Но для распределенных систем, от компьютеров которых требуется предельно точная и быстрая синхронизация совместных операций, сравнимая с достижимой в сосредоточенных системах, требуется организация, приведенная в [1].

Заметим, при отсутствии прямой видимости объектов потребуется обычным способом транслировать сигналы, делая их доступными для удаленных объектов.

2. Переход от асинхронных к синхронно взаимодействующим объектам

Для перехода группы объектов к упорядоченному взаимодействию объекты вначале должны синхронизовать совместные действия по выбору способа взаимодействия, рассмотренного в следующем разделе. После этого начинается взаимодействие по команде, созданной одним из объектов или одновременно группой объектов.

2.1. Процесс начала синхронизации

Раздел основан на результатах статьи [4]. Будем считать, что задан интервал времени ${}^*T$, длительности не менее T – времени переноса сигнала в пределах всей системы объектов, предполагаемый неизвестным. Если в течение ${}^*T$ объект не получает сигналы от других объектов, то в системе нет обмена сообщениями и объекты могут начать процесс синхронизации для выполнения сеанса обмена сообщениями.

Таким образом, система начинает синхронизацию полностью децентрализовано с задержкой ${}^{*}T$. Для начала синхронизации объект, *не получив сигналы* объектов в течение интервала ${}^{*}T$, посылает сигнал S длительности ${}^{*}T$ и ожидает появления сигнала ${}^{*}S$ момента завершения сигнала S. Выполнение ограничения ${}^{*}T \geqslant T$ обеспечивает при совмещении во времени сигналов S, поступающих от группы объектов, единственность сигналов S и ${}^{*}S$. Момент завершения объектом сигнала S (т.е. сигнал ${}^{*}S$), разрешает объектам приступить к следующим шагам по организации взаимодействия объектов, приведенным в разделе 2.2.

2.2. Выбор способа обмена сообщениями

При обычной сетевой связи передаваемые источниками сообщения часто не коррелированы, и выбор действий источников по упорядочению передачи сообщений диктует система связи. Например, система решает будут объекты передавать сообщения поочередно или с учетом динамических приоритетов.

В рассматриваемых средствах взаимодействия, напротив, есть большая зависимость в передаваемых объектами сообщениях, так как они часто решают общую задачу. Поэтому полезно предоставить объектам быстрый способ коллективного выбора характера взаимодействия.

Для этого рассмотрим процедуру коллективного выбора объектами между некоторыми альтернативными действиями A_0 и A_1. Но вначале рассмотрим действия объектов при получении сигнала ${}^{*}S$, иллюстрируемого рисунком 2.

В разделе 2.2 и следующих разделах потребуется, чтобы во времени пересекались только сигналы, имеющие одинаковую смысловую нагрузку (например, одноименные двоичные разряды сообщений). Необходимые для этого условия иллюстрирует рисунок 2.

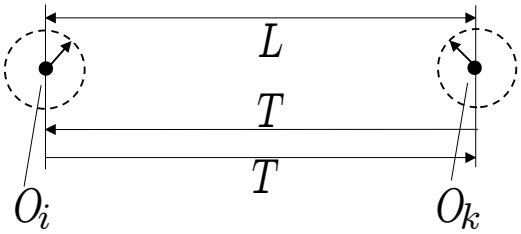


Рисунок 2. Общее условие корректной передачи сигналов

Здесь показаны два объекта-источника – O_i и O_k, наиболее взаимно удаленных в системе. Расстояние между ними равно L; время переноса

сигнала $T = L/c$, где $c-$скорость света. Объекты посылают ненаправленные сигналы длительности $\leqslant T$.

Пусть в системе появляется условие (в частности, появление сигнала ${}^*S$), разрешающее объекту O_i передачу сигнала. Посланный O_i сигнал поступит к объекту O_k через интервал времени T, к любому другому объекту системы через интервал ${}^*T \leqslant T$. Так как длительность посылаемых объектами сигналов выбрана $\leqslant T$, то сигнал объекта O_i выйдет за пределы системы (исчезнет) не позже интервала $2T$ после его отправки объектом. К объекту O_k сигнал от O_i поступит через интервал T и исчезнет еще через интервал ${}^*T \leqslant T$. Объект O_k также может послать свой сигнал при приходе ${}^*S$. Это произойдет через интервал T после передачи сигнала объектом O_i. По отношению к моменту получения ${}^*S$ в O_i сигнал от O_k исчезнет через интервал $3T$. Поэтому, если требуется, чтобы сигналы от O_i и O_k гарантировано пересекались во времени, объекты должны их посылать при получении ${}^*S$.

Сигналы любых других пар объектов также будут пересекаться и исчезнут в пределах $3T$. Если T неизвестен, то замена его на ${}^*T \geqslant T$ не изменит результат, полученный для T.

Чтобы сигналы объектов не пересекались, объекты должны передавать их в разных интервалах $3T$ (или $3{}^*T$).

Теперь перейдем к коллективному выбору объектов между альтернативными действиями A_0 и A_1.

Для участия в выборе объекты используют беспроводные сигналы частоты f_0 для информирования всех объектов о предпочтении выбора A_0 и f_1 о предпочтении выбора A_1. Для передачи одного сигнала, как отмечалось, объекты используют два интервала, каждый длительности ${}^*T$ или T, если последний известен. Любой объект при получении * начнет передачу сигнала длительностью $\leqslant^* T$. Не позднее следующего интервала ${}^*T$ сигнал исчезнет, а все объекты получат сигналы объектов, информирующие о предпочтении их выбора на данном шаге. Для выбора объекты должны учесть инструкцию системы по выбору, которая определяет решение в соответствии с полученными объектами сигналами f_0 и f_1. Перед выбором инструкция должна быть известна объектам.

Рассмотрим пример. Пусть объектам требуется выбрать между поочередной передачей объектами сообщений (выбор A_0) и передачей сообщений источников в соответствии с текущими значениями приоритетов (выбор A_1). Объектам инструкция системы известна.

Допустим, система предлагает приоритетный выбор при единогласном выборе объектов на приоритетный доступ, иначе должен быть поочередный

доступ. Пусть объекты получили сигнал * и сообщают о своем решении посылкой сигнала f_0 (выбор A_0) или f_1 (выбор A_1). Тогда, при наличии только f_1 все объекты работают по приоритетам, при наличии только f_0 или f_1 и f_0 выполняется поочередный доступ. Таким образом, инструкция системы должна предложить выбор, который учитывает неизвестные ей заранее предпочтения объектов. При начальном запуске инструкция системы задана, но в процессе работы объект, получивший право передавать сообщения может предложить свою инструкцию или цепочку инструкций с учетом задаваемой объектам цепочкой операций.

2.3. Детали выполнения поочередной и приоритетной передачи сообщений

Поочередная передача сообщений. Предполагается, что всем объектам предварительно присвоены порядковые номера (адреса), и объект завершает сообщение распознаваемым всеми объектами стоп-признаком, характерным для объекта (именем или порядковым номером объекта).

Первый по порядку объект при получении $^*S$ передает сообщение на высокой скорости, ограниченной только техническими возможностями объектов, и завершает его стоп-признаком, так как длительность сообщений объектов не унифицирована. Каждый следующий объект передает сообщение после обнаружения стоп-признака с номером непосредственного предшественника. Если каждый объект работоспособен и обязательно передает сообщение, то сканирование объектов завершено.

Пусть теперь произвольным объектов разрешено не передавать сообщения. Первый объект без каких-либо условий передает сообщение. Каждый следующий после первого объект с порядковым номером k ожидает завершения передачи $k-1$ сообщения и затем передает сообщение. Если при этом объект обнаруживает отсутствие сигналов в течение интервалов длительности 3^*T, то объект считает каждый такой пустой интервал сообщением источника с очередным порядковым номером. Так как количество объектов известно, передача будет завершена после передачи всех ожидаемых сообщений. Дополнительно при этом процесс определяет номера не передающих сигналы объектов.

Передача сообщений с учетом приоритетов. В современных протоколах *USB I3C* и *MIPI I3C* используются давно известные способы учета приоритета [5]. Объекты, претендующие на обмен сообщениями, одновременно побитно передают двоичные коды своих приоритетов. Вначале все объекты передадут общее сообщение, содержащее старший бит кода приоритета и проверят значение принятых сигналов. Если приняты

только сигналы, представляющие двоичную единицу или одновременно единицу и ноль, то считается, что получена единица и только объекты, пославшие единицу, передадут сообщение, содержащее следующий бит адреса и т.д. После передачи всех битов адреса сообщение передаст единственный объект со старшим значением приоритета.

Код приоритета объекта может быть представлен числом, состоящим из двух групп разрядов. В группе старших разрядов указано текущее значение приоритета, в группе младших разрядов порядковый номер объекта. Такое решение более гибко выбирает реакцию объектов на события.

2.4. Два вида формирования общего сообщения группой объектов

Пусть объекты при получении специального сообщения-команды могут выбрать один из двух видов передачи сообщений – передачу сообщений одного за другим или с пересечением одноименных двоичных разрядов сообщений. Первый вариант использован в вариантах раздела 2.3

Во втором варианте специальное сообщение разрешает заранее выбранной группе объектов передавать свои сообщения в едином для группы сообщении, посылая в пределах времени $3T$ одновременно очередной одноименный разряд своих сообщений. В результате устраняется зависимость длины сообщения от количества объектов в группе. Второй вид передачи сообщений основной для операций следующих разделов статьи.

Подведем итог раздела 2. Раздел дает быстрое детерминированное управление передачей сообщений объектами. В разделе 2.1 объекты асинхронной посылкой общего сигнала S получают право передавать сообщения. В разделах 2.2 и 2.3 детально рассмотрена поочередная передача сообщений и передача с учетом приоритетов. В разделе 2.4 представлены два вида передачи группой объектов сообщений в едином для группы сообщении. В нем объекты размещают свои сообщения одно за другим или в виде только одного сообщения с совмещением в нем одноименных разрядов сообщений объектов. Этим завершено изложение всех базовых механизмов, обеспечивающих быстрое полностью децентрализованное управление взаимодействием цифровых объектов.

3. Специальный формат данных – логические шкалы

Логические шкалы и ассоциативные операции следующего раздела применялись автором в неоднородных системах, например, в [1]. В этой статье показано их применение в однородных системах.

Структура логической шкалы. В статье логическая шкала (для краткости – шкала), это формат данных, используемый объектами для одновременной обработки данных в распределенных операциях. Шкалы, посылаемые одновременно несколькими объектами в общей распределенной операции, содержат равное количество разрядов. Каждый разряд шкалы приписан одному из известных всем объектам признаков (критериев), характеризуемых шкалой.

В статье шкала, это часть связки – шкалы и групповой операции, синхронно обрабатывающей группу поставляемых объектами шкал. Синхронность обработки объединяет группу шкал в одну групповую операцию или синхронную последовательность групповых операций. В разделе 2 уже применялся вариант такой синхронной последовательности при выделении объекта с максимальным значением кода приоритета. В нем каждая шкала – двоичный разряд значения приоритета, а вся групповая операция состояла из синхронной последовательной обработки всех разрядов кода как единого целого. Конечный результат применения связки состоит в повышении скорости реакции группы объектов на возникающие события.

В шкале объекту для представления значения двоичного разряда можно использовать четыре варианта сигналов: ноль передается сигналом f_0, единица сигналом f_1, при отсутствии активности объект не передает в разряд сигналы (сигнал Z), Наличие в разряде f_0 и f_1 – сигнал W. В зависимости от конкретной операции смысл сигналов Z и W может быть разным. Например, сигнал Z – отсутствие сигналов покажет, что объект не может приписать значение признаку. Сигнал W полезен при задании действия группе объектов и разрешит участвовать в операции объектам при любом значении указанного в W разряда шкалы (маска разряда).

В частности, шкалы ускоряют работу с числами, представленными в системе с произвольным основанием. В такой шкале каждая цифра числа задается шкалой с количеством двоичных разрядов, соответствующим основанию системы. Шкала содержит только одну единицу в разряде, соответствующим значению цифры. Например, цифре 7 в десятичной системе соответствует шкала 001000000. Такое представление цифр полезно, например, для поиска максимального и минимального значения числа. Если при совмещении шкал десятичных цифр появится шкала 001001101, то из шкалы следует, что в совмещении участвовали максимальная цифра 7 и минимальная цифра 1.

В следующем разделе рассмотрено применение логических шкал в распределенных групповых (ассоциативных) операциях.

4. Ассоциативные операции управления действиями объектов однородных систем

В задачах обработки данных ассоциативные операции впервые появились в ассоциативных запоминающих устройствах для одновременного поиска в массиве данных. Ассоциативными операции названы потому, что обращение к данным выполняется не по их адресам, а по характерным признакам. В соответствии с предыдущим текстом ассоциативные операции могут быть названы и групповыми.

Для выделения отличий известных ассоциативных устройств и операций от предлагаемых в статье для однородных систем сопоставим новые решения с первыми структурами сосредоточенных ассоциативных устройств, обеспечивающих переход от ассоциативной памяти к ассоциативным процессорам [6, 7]. На рисунке 3 показана структура, представленная в [7] и ряде других последующих публикаций. Она наиболее соответствует содержанию раздела 4.

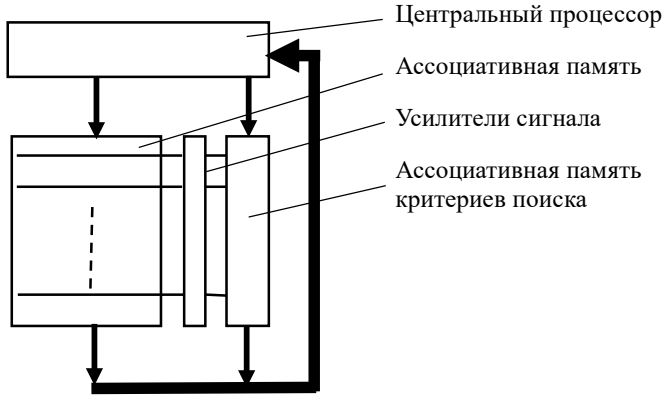


Рисунок 3. Ассоциативная память / вычислительное устройство

Здесь строки ассоциативной памяти содержат логические шкалы, приведенные в разделе 3. Каждый разряд шкалы состоит из двух битов, заполняемых любым сочетанием единиц и нулей, показанным в разделе 3. Запись в разряде комбинации 10 интерпретируется как логическая единица, запись 01 – логический ноль. Таким образом нули активные – передаются наличием сигнала. Часть разрядов в поиске можно игнорировать (маскировать), используя запись W. Ассоциативная память критериев поиска состоит из одного или группы столбцов.

Устройство работает следующим образом. Центральный процессор структуры посылает одновременно на все разряды строк массива ассоциативной памяти слово поиска строк (слов) в массиве, соответствующих слову поиска. Найденные строки через усилители одновременно посылают сигнал в выбранный процессором столбец ассоциативной памяти критериев поиска. В результате в столбце будет отмечена группа строк, соответствующих условию поиска. На общем выходе всех строк массива ассоциативной памяти появится строка суммарного состояния столбцов массива, которая поступает в процессор. Затем процессор выполняет новые поиски с записями в другие столбцы памяти критериев. Новые поиски используют сформированные ранее разряды строк памяти критериев как часть нового слова поиска.

Как в разделе 3 цифры чисел можно задавать в системе с основанием, отличным от двоичного, и представлять соответствующими логическими шкалами, выполняя с ними логические операции, поиск максимума / минимума, поиск чисел, расположенных в заданном интервале.

Затем появились значительно более сложные операции в ассоциативных процессорах и компьютерах. Они одновременно обращались к расположенных в разных частях системы данным, и находили данные, соответствующие условиям поиска или близкие к ним. Все приведенные системы были сосредоточенными.

Можно создать распределенные компьютерные системы, конкурирующие по производительности с сосредоточенными системами при доступности высокой точности синхронизации взаимодействия компонентов этих систем [**1**]. В работе [**1**] и нескольких предыдущих публикациях автора распределенные системы содержат непосредственно (без посредников) доступные **всем** объектам ретрансляторы сигналов, каждый из которых собирает синхронизованные побитно или поочередно сообщения группы объектов в одну точку-ретранслятор. Затем ретрансляторы, как единственные источники, выполняют групповую обработку данных и рассылают общее сообщение групп всем объектам – приемникам сообщений группы. Это позволяет синхронизовать объекты и избавиться от медленной индивидуальной рассылки сообщений. В ряде важных операций получена независимость длительности операции от количества участвующих в ней объектов. Каждая выполняемая в распределенной системе ассоциативная операция одновременно обрабатывает большое количество данных, но в общем случае из-за разной сложности решаемых объектами задач вся система объектов или ее отдельные группы действуют асинхронно. Это выдвигает две решаемые в [**1**] задачи.

Первая задача – продвижение распределенной асинхронной обработки данных по мере их готовности в разных объектах. Вторая задача – доставка требуемых данных **одновременно** всем потребителям данных. Эти две задачи существенны для вычислительных задач типа dataflow, а также для многих эвристических задач и задач реального времени.

В однородных системах этой статьи ассоциативные операции по организации и способам применения мало отличаются от операций в неоднородных системах, но реализуются проще. Существенное отличие только во времени исполнения – группа объектов передает одноименный двоичный разряд сообщений в пределах интервала длительности $3T$. Ниже кратко рассмотрим каждую из ассоциативных операций и опишем типовую схему из применения.

Рассмотрим следующие распределенные групповые (ассоциативные) операции, одновременно использующие группу аргументов: групповые поразрядные операции *логическое И*, *логическое ИЛИ*; групповой поиск max или min; групповые арифметические *сложение* и *вычитание*. Операции в качестве аргументов используют логические шкалы.

В групповой операции *поразрядное логическое И* объекты считают общим значением всех принятых в пределах интервала T битов данных единицу, если среди битов не было нулей (сигналов f_0). Соответственно значение операции *поразрядное логическое ИЛИ* равно нулю, если среди битов не было единиц (сигналов f_1).

В групповом поиске max или min объекты действуют с двоичными числами подобно доступу по приоритетам (раздел 2.3) и с числами в системе с произвольным значением основания подобно разделу 3. Вначале все объекты передадут общее сообщение, содержащее в пределах T старшую цифру числа, и выделят объекты, передавшие наибольшую цифру. После передачи всех цифр чисел будет выделено максимальное значение числа. Подобным способом объекты находят минимум.

Выполнение групповых арифметических операций *сложение* и *вычитание*. В операциях применим представление чисел шкалами в соответствии с разделом 3. Используем десятичную систему счисления. Каждая цифра числа задается шкалой с количеством двоичных разрядов, соответствующим основанию системы, и содержит только одну единицу в разряде, соответствующим значению цифры. Над шкалами, представляющими цифры выполним аналого-цифровые операции. Добавим в интерфейсные карты объектов элемент аналого-цифровой преобразователь (АЦП). Пусть цифры передаются в виде указанных выше шкал и посылаемые объектами сигналы имеют равную энергию. Тогда АЦП при совмещении

в ретрансляторе одноименных разрядов шкалы сформирует численное значение суммарной энергии сигналов в каждом разряде, и одновременно все объекты выполнят суммирование.

Приведем пример: на АЦП поступает от группы объектов общая шкала 0(3)00(4)00(6)0. Здесь в скобках указаны значения, которые АЦП сформирует для объекта. Цифра 3 в восьмом разряде шкалы появилась при одновременном поступлении в АЦП этом разряде трех сигналов f_1 от объектов. Такой же смысл имеют цифры 4 и 6 в пятом и втором разрядах шкалы. Получив от АЦП такие значения, все объекты вычисляют: $(8 \times 3) + (5 \times 4) + (2 \times 6) = 56$. Здесь нет зависимости времени исполнения от количества участников операции.

Операции с АЦП требуют высокой стабильности мощности посылаемых объектами сигналов. В [8] приведен простой светодиодный источник со стабильностью выходной мощности лучше 50 ppm/°C. Это достаточно для одновременного суммирования сигналов нескольких тысяч источников, что существенно превышает требования задач настоящей статьи.

Следующее важное действие – слежение за завершением асинхронных работ объектов. Подготовка к ассоциативной операции, выполнение операции, а также дополнительных, индивидуальных для объекта операций может требовать разное время. Поэтому каждый объект должен оповещать все объекты о завершении локальных действий. Наиболее быстро сообщить о завершении работ позволяет следующий вариант барьерной синхронизации. Каждый объект, начав локальные действия, посылает непрерывный сигнал Br всем объектам и снимает его по завершению локальной работы. Завершение Br всеми объектами они воспринимают как новый сигнал ${}^*Br$ завершения всех работ. На сигнал ${}^*Br$ объекты реагируют, как на ${}^*S$, и начинают без обращения к центру совместное выполнение очередной команды.

В системе могут быть объекты-исполнители, непосредственно воздействующие на внешнюю среду. Логические шкалы, при соответствующей интерпретации значения их разрядов, исполнители воспримут как команды, которые требуется выполнять одновременно. Как показано выше, в однородной системе разброс считающихся одновременными действий объектов не превышает T. От однородной системы не требуется очень высокая точность синхронизации, поэтому объекты, включая исполнителей могут незначительно менять расположение.

Кроме применений шкал в групповых операциях они полезны в рассмотренных в предыдущих разделах операциях управления действиями объектов.

Пример такой операции – групповое устранение конфликтов объектов с учетом прав передачи сообщений. Задается шкала с количеством разрядов, равным количеству объектов. Каждый претендующий на передачу сообщения объект в своей части общего сообщения объектов в выделенный ему двоичный разряд шкалы заносит единицу. Объекты принимают последовательность шкал и изложенными выше способами формируют общую шкалу объектов. После этого объекты упорядоченно передают сообщения с учетом единиц в разрядах шкалы.

В заключение раздела вернемся к рисункам 1 и 3. Распределенная ассоциативная структура не отличается от распределенной структуры на рисунке 1. Структура связей сохранена, добавлен ряд ассоциативных операций. Отличие от сосредоточенной ассоциативной структуры существенное – каждый распределенный объект содержит собственный компьютер с обычной памятью или с ассоциативной памятью, подобной памяти на рисунке 3.

Наличие в объекте компьютера позволяет расширить набор ассоциативных операций, на программном уровне комбинируя приведенные выше операции или создавая совершенно новые.

В структуре рисунка 1 объекты в динамике выбирают текущего лидера. В результате в однородной системе в каждый момент времени объекты формируют новую распределенную древовидную структуру. В ней каждый объект подобен строке ассоциативной памяти на рисунке 3, но строке с собственными вычислительными средствами и гибким набором операций.

Автор полагает, что ассоциативные операции могут быть полезны в эвристических задачах и задачах с элементами dataflow, требующими одновременное общение групп объектов без использования адресов и имен комплексов данных, лишь удовлетворяющих набору соответствующих требований. В системах реального времени применение ассоциативных операций ускоряет реакцию на возникающие события.

5. Системы с малыми расстояниями между объектами

Несмотря на то, что статья рассматривает распределенные на значительные расстояния однородные системы, важно отметить полезный вариант систем, объединяющих возможности неоднородных и однородных способов взаимодействия. Это системы с малыми расстояниями между объектами, имеющие при однородности точность, близкую к получаемой в неоднородных системах.

Пусть, например, имеется однородная система с модулями (чипами), расположенными на печатной плате с максимальным расстоянием 30 см между ними, и требуется организовать взаимодействие модулей. Сигнал переместится в таких пределах за время T, равное 1 нс. Для чипа с расстояниями порядка 3 см время переноса сигнала составит 0,1 нс. Эти времена достаточно малы и позволяют создать для многих приложений однородные по управлению структуры, способные по скорости реакции конкурировать с более сложными неоднородными системами.

Приведем пример существенного упрощения. В статье [1] рассмотрена структура суперкомпьютера с неоднородной структурой и ориентированными оптическими связями, объединяющими его активные компоненты. Возможности этой структуры отмечены во введении к настоящей статье. Структура имеет группу специализированных коммутаторов, которые позволяют при обращении приемников к коммутатору доставлять приемникам сообщение источников *за счет энергии сигналов* приемников. Структура позволяет выполнить одновременную перестройку всех связей между объектами с наносекундными задержками, что позволяет выполнять перестройку связей в пределах текущей задачи за время перехода от одной операции к следующей. Быстрая перестройка потребовала высокой точности измерений расстояний между объектами в соответствии с решениями CERN, вошедшими в международный стандарт [3].

При малых расстояниях между объектами можно создать подобные системы упрощенно, отказом от измерений расстояний. При этом структура взаимодействия сохраняется, скорость и гибкость реорганизации связей сохраняются, но неизбежно уменьшение точности синхронизации по сравнению с [1] с замедлением групповых операций, в значительной степени компенсируемое малым временем переноса сигналов. Здесь, как и в [1], допустимы два вида реализации.

Первый вид использует ретрансляторы с ретрорефлектором, который принимает сигналы группы источников и отправляет их группе приемников. За счет энергии приемников в ретрансляторах модуляторы данными в сигналах источников модулируют сигналы приемников. Структуры с единственным ретранслятором известны и применялись на практике (см. ссылки в [1]). Публикации по структурам с многими ретрансляторами не обнаружены. Также нет данных о использовании в таких структурах быстрых переключателей, выбирающих направление оптического сигнала источника на требуемый ретранслятор из группы ретрансляторов. Эти вопросы требуют дальнейшей проработки.

Вторая известная, более простая реализация требует наличия в источниках генераторов сигналов с набором частот, соответствующим

количеству ретрансляторов. Каждый из ретрансляторов принимает сигналы своей частоты и ретранслирует их на другой, также характерной только для него частоте. Здесь все сигналы ненаправленные и не требуется иметь переключатели направления сигналов объектов. Такая структура реализуема при небольшом количестве ретрансляторов.

6. Заключение

В рассмотренном взаимодействии активных устройств (объектов) однородных цифровых систем получены следующие результаты. Система состоит из объектов, которые организуют свое взаимодействие без привлечения внешних управляющих средств.

Предполагается ограниченная взаимная удаленность объектов в пределах нескольких десятков метров. Отсутствие внешнего управления сохраняет работоспособность системы при отключении любых ее объектов. Также произвольная группа указанных объектов совместными действиями объединяется в однородную систему. Объекты могут быть стационарными и мобильными. В последнем случае взаимодействие должно выполняться оптическими или радиосигналами.

Указанные возможности достигаются простыми в реализации техническими средствами, но ценой следующего ограничения в точности синхронизации действий объектов. Происходящие в разных объектах системы события считаются одновременными, если они удалены не более чем на интервал времени $T = L/c$, где L – максимальное расстояние между объектами, c – скорость света.

Один из видов реализации требуемых средств – создание интерфейсных карт, объединяющих в единую систему персональные компьютеры или компьютеры кластеров.

Предполагаемая основная область использования – управление взаимодействием объектов с выбором их не по адресам, а по требуемой текущей совокупности свойств. В частности, такая задача характерна для ряда эвристических задач, ряда задач dataflow и для систем, работающих в реальном времени. Кроме этого предложения статьи могут быть полезны в области периферийных вычислений (edge computing) для выполнения распределенных задач управления и вычислений вне компьютеров.

Список литературы

[1] Стецюра Г. Г. *Синхронное выполнение групповых операций в распределенных компонентах суперкомпьютеров и компьютерных кластерах* // Программные

системы: теория и приложения.– 2022.– Т. **13**.– № 4(55).– С. 3–24. ↑4, 5, 6, 8, 12, 15, 19

[2] Lipinski M., Włostowski T., Serrano J., Alvarez P. *White Rabbit: a PTP application for robust sub-nanosecond synchronization* // *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication* (Munich, Germany, 2011).– Pp. 25–30. ↑5

[3] Girela-López F., López-Jiménez J., Jiménez-López M., Rodríguez R., Ros E., Díaz J. *IEEE 1588 high accuracy default profile: applications and challenges* // IEEE Access.– 2020.– Vol. **8**.– Pp. 45211–45220. ↑5, 19

[4] Стецюра Г. Г. *Децентрализованная автономная синхронизация процессов взаимодействия мобильных объектов* // Проблемы управления.– 2020.– № 6.– С. 46–56. ↑8

[5] Прангишвили И. В., Подлазов В. С., Стецюра Г. Г. *Локальные микропроцессорные вычислительные сети.*– М.: Наука.– 1984.– 176 с. ↑11

[6] Стецюра Г. Г. *Новый способ построения запоминающих устройств* // Доклады АН СССР.– 1960.– Т. **132**.– № 6.– С. 1291–1294. ↑14

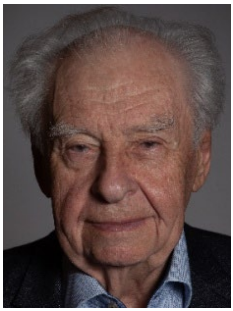
[7] Falkoff A. D. *Algorithms for parallel-search memories* // Journal of the ACM.– 1962.– Vol. **9**.– No. 4.– Pp. 488–511. ↑14

[8] Bosiljevac M., Downing J., Babić D. *Reaching <100 ppm/K output intensity temperature stability with single-color light-emitting diodes* // Applied Optics.– 2016.– Vol. **55**.– No. 32.– Pp. 9060–9066. ↑17

Поступила в редакцию 14.06.2023;
одобрена после рецензирования 07.07.2023;
принята к публикации 12.09.2023;
опубликована онлайн 10.10.2023.

Рекомендовал к публикации *к.ф.-м.н. С. А. Романенко*

Информация об авторе:



Геннадий Георгиевич Стецюра

д. т. н., профессор, гл. н. с. ИПУ РАН. Область интересов: информатика, computing. Основные работы в области распределенных многокомпонентных цифровых систем. Более 150 публикаций и более 20 патентов и авторских свидетельств на изобретения.

0000-0003-4606-4424

e-mail: gstetsura@mail.ru

Автор заявляет об отсутствии конфликта интересов.

PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article HARDWARE AND SOFTWARE FOR DISTRIBUTED AND SUPERCOMPUTER SYSTEMS

UDC 004.272.45
10.25209/2079-3316-2023-14-4-3-23

The organization of interaction the active objects of homogeneous digital structures

Gennady Georgievich **Stetsyura**
V. A. Trapeznikov Institute of Control Sciences of RAS, Moscow, Russia
gstetsura@mail.ru

Abstract. This article extends the possibilities of interaction of active devices (objects) in homogeneous digital systems. The system consists of objects that are located within a few tens of meters and organize their interaction only by their own means. Objects can be stationary and mobile with arbitrary and time-varying mutual arrangement The homogeneity of the system means the absence of external control and equal opportunities of objects in the organization of their interaction. Communications between objects are wireless using optical or radio signals. Signals from any source object go directly to all receiver objects. The object receives the right to transmit signals deterministically according to its priority value, either dynamically or statically. The proposed structure and interaction methods allow objects to perform distributed group (associative) operations in addition to the usual distributed system message exchange. In them, objects simultaneously eliminate a group of conflicts of access to the common channel of data exchange, determine the state of all objects of the system, and synchronously perform joint actions of objects, reacting to unforeseen changing external conditions. A group of objects simultaneously participating in a group operation is selected by specifying a set of criteria that the objects must possess. The homogeneity of the system greatly simplifies its technical implementation, but the inclusion of heterogeneity in the system will accelerate the performance of group operations when becoming more complex. Therefore, the homogeneous system is proposed to be used mainly for communication between active peripheral devices and those with more complex cluster computers. (*In Russian*).

Key words and phrases: computer clusters, decentralized control, fast synchronization of digital objects, intranet computing, distributed associative operations, edge computing

2020 *Mathematics Subject Classification*: 65Y05; 68Q10

For citation: Gennady G. Stetsyura. *The organization of interaction the active objects of homogeneous digital structures*. Program Systems: Theory and Applications, 2023, **14**:4(59), pp. 3–23. (*In Russ.*). https://psta.psiras.ru/read/psta2023_4_3-23.pdf

References

[1] G. G. Stetsyura. "Synchronous execution of group operations in distributed supercomputer components and computer clusters", *Program Systems: Theory and Applications*, **13**:4(55) (2022), pp. 25–46.

[2] M. Lipinski, T. Włostowski, J. Serrano, P. Alvarez. "White Rabbit: a PTP application for robust sub-nanosecond synchronization", *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication* (Munich, Germany, 2011), pp. 25–30.

[3] F. Girela-López, J. López-Jiménez, M. Jiménez-López, R. Rodríguez, E. Ros, J. Díaz. "IEEE 1588 high accuracy default profile: applications and challenges", *IEEE Access*, **8** (2020), pp. 45211–45220.

[4] G. G. Stetsyura. "Decentralized autonomic synchronization of interaction processes of mobile objects", *Control Sciences*, 2020, no. 6, pp. 46–56 (in Russian).

[5] I. V. Prangishvili, V. S. Podlazov, G. G. Stetsyura. *Local Microprocessor-Based Computer Networks*, Nauka, M., 1984 (in Russian), 176 pp.

[6] G. G. Stetsyura. "A new method to build memory devices", *Proceedings of the Academy of Sciences*, **132**:6 (1960), pp. 1291–1294 (in Russian).

[7] A. D. Falkoff. "Algorithms for parallel-search memories", *Journal of the ACM*, **9**:4 (1962), pp. 488–511.

[8] M. Bosiljevac, J. Downing, D. Babić. "Reaching <100 ppm/K output intensity temperature stability with single-color light-emitting diodes", *Applied Optics*, **55**:32 (2016), pp. 9060–9066.

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ, НЕЙРОННЫЕ СЕТИ

УДК 004.89:81'322
DOI 10.25209/2079-3316-2023-14-4-25-45

Метод классификации аспектов аргументации в русскоязычных текстах

Ирина Николаевна **Фищева**[1], Татьяна Анатольевна **Пескишева**[2], Валерия Сергеевна **Головизнина**[3], Евгений Вячеславович **Котельников**[4]✉
Вятский государственный университет, Киров, Россия
[4]✉*kotelnikov.ev@gmail.com*

Аннотация. Автоматический анализ аргументации в текстах привлекает в последние годы внимание исследователей в связи с широким диапазоном приложений, в частности, в анализе научных и юридических текстов, новостных статей, политических дебатов, студенческих эссе и социальных медиа. Новая задача в этой области – анализ аргументации с учетом аспектов, где под аспектом понимается свойство объекта, относительно которого строится довод. Учет аспектов позволяет уточнить направленность аргументации и понимание аргументационной структуры, а также может быть использован для генерации высококачественных и специфичных для выбранных аспектов доводов.

В статье предлагается метод классификации аспектов аргументации в текстах на русском языке, построение на его основе и исследование моделей классификации аспектов аргументации с использованием машинного обучения и нейронных сетей. Впервые сформирован русскоязычный текстовый корпус, включающий 1426 предложений и размеченный по 16 аспектам аргументации, построена нейросетевая языковая модель классификации аргументов ArgBERT и обучены модели Random Forest для классификации аспектов аргументации. Качество классификации на основе Random Forest составляет в среднем F1=0,6373. Наилучшее качество разработанные модели демонстрируют для аспектов «Безопасность», «Влияние на здоровье», «Влияние на психику», «Отношение властей» и «Уровень жизни» (F1-мера выше 0,75).

Ключевые слова и фразы: анализ аргументации, текстовые корпуса, нейросетевые языковые модели, машинное обучение, Random Forest, аспекты аргументации

Благодарности: Исследование выполнено за счет *гранта Российского научного фонда № 22-21-00885*

Для цитирования: Фищева И. Н., Пескишева Т. А., Головизнина В. С., Котельников Е. В. *Метод классификации аспектов аргументации в русскоязычных текстах* // Программные системы: теория и приложения. 2023. **Т. 14**. № 4(59). С. 25–45. https://psta.psiras.ru/read/psta2023_4_25-45.pdf

Введение

Аргументация – вид вербальной, социальной и рациональной деятельности, направленной на убеждение разумного критика в приемлемости некоторой точки зрения посредством приведения совокупности подтверждающих или опровергающих доводов [1]. Автоматический анализ аргументации (argument mining) – область компьютерной лингвистики, целью которой является обнаружение аргументов, представленных в неструктурированных текстах, и извлечение связей между аргументами [2, 3]. Помимо теоретического интереса внимание к анализу аргументации связано с широким диапазоном приложений, в частности, при изучении мнений пользователей на основе анализа социальных медиа [4], в анализе юридических текстов [5], научных публикаций [6], рецензий на научные статьи [7], политических дебатов [8], новостных статей [9] и студенческих эссе [10].

В области анализа аргументации в настоящее время ведутся активные исследования:

- каждый год проводится международный семинар *Workshop on Argument Mining*;
- постоянно растет число публикаций: Поисковый запрос *«Argument Mining» в Google Scholar* дает следующие результаты: 2017 год – 216 статей, 2018 год – 281 статья, 2019 год – 394 статьи, 2020 год – 473 статьи, 2021 год – 645 статей, 2022 год – 697 статей;
- проводятся соревнования программных систем по данной тематике: SemEval в 2016 году [11], Touché в 2020 и 2021 годах [12, 13], в том числе для русского языка – RuArg-2022 [14].

Однако большинство работ проводится на материале английского языка; исследований для русского языка за последние годы было относительно немного [14–17]. При анализе аргументации решаются следующие основные задачи:

- определение фрагментов текста, содержащих аргументацию;
- распознавание утверждений и доводов, выявление связей между ними;
- построение общей аргументационной структуры; определение качества аргументации.

Ключевым подходом здесь, как и в других областях обработки естественного языка, стало применение нейросетевых языковых моделей, таких как BERT [18] и GPT [19]. Недавно была поставлена новая задача в этой области – анализ аргументации с учетом аспектов (aspect-based argument mining), где под аспектом понимается свойство объекта, относительно

которого строится довод [20]. Учет аспектов позволяет, с одной стороны, уточнить направленность аргументации и понимание аргументационной структуры; с другой стороны, может быть использован для генерации высококачественных и специфичных для выбранных аспектов доводов [21]. Модели, учитывающие аспекты при анализе аргументации в русскоязычных текстах, насколько нам известно, отсутствуют; есть работы только для английского языка [20, 22]. Поэтому основными целями настоящей статьи являются разработка метода классификации аспектов аргументации в текстах на русском языке, а также исследование моделей классификации аспектов аргументации, построенных на основе данного метода с использованием машинного обучения и нейронных сетей. Предлагаемый метод включает этапы формирования базы данных, поиска потенциальных аргументов, формирование корпуса аспектов и построение моделей классификации.

Вклад нашей работы заключается в следующем:

- разработан метод классификации аспектов аргументации;
- построена нейросетевая языковая модель ArgBERT, позволяющая осуществлять бинарную классификацию предложений на «довод»/«не довод»;
- сформирован текстовый корпус, включающий 1426 предложений и размеченный по 16 аспектам;
- обучены модели Random Forest, позволяющие классифицировать предложения по аспектам аргументации.

Статья имеет следующую структуру. Сначала рассматривается связанная с аргументацией терминология, используемая в работе. Затем приведен обзор работ, посвященных анализу аргументации в текстах и учету аспектов. Далее описывается предлагаемый метод классификации аспектов и продемонстрированы результаты экспериментов по применению этого метода на реальных текстах. В Заключении формулируются основные выводы и результаты работы.

1. Терминология

Анализ аргументации проводится для текста, посвященного определенной теме. В тексте упоминаются один или несколько *целевых объектов*. Автор текста выражает некоторую *точку зрения* по отношению к целевому объекту, которая включает определенную позицию. *Позиция* задаётся шкалой, содержащей, например, значения «за», «против» и «нейтрально».

Анализируемый текст можно разделить на фрагменты, которые имеют единое аргументационное значение – ***аргументативные дискурсивные***

единицы (АДЕ) [3]. В роли АДЕ могут выступать части предложения, целые предложения или несколько предложений. В настоящей работе в качестве АДЕ рассматриваются отдельные предложения. АДЕ, выражающую позицию автора, называют *утверждением*, оно, как правило, находится в начале или в конце текста.

АДЕ, поддерживающее утверждение, является *доводом «за»*, а опровергающее – *доводом «против»* (контрдоводом). *Аргументом* называют совокупность взаимосвязанных АДЕ, подтверждающих или опровергающих некоторую точку зрения. Минимальный аргумент включает одно утверждение и, по крайней мере, один довод.

Каждый довод описывает определенный аспект (или аспекты) целевого объекта [21]. *Аспект* – свойство целевого объекта, упоминаемого в заданном утверждении, которое выражается в тексте словом или словосочетанием.

Описанные понятия иллюстрируются на рисунке 1:

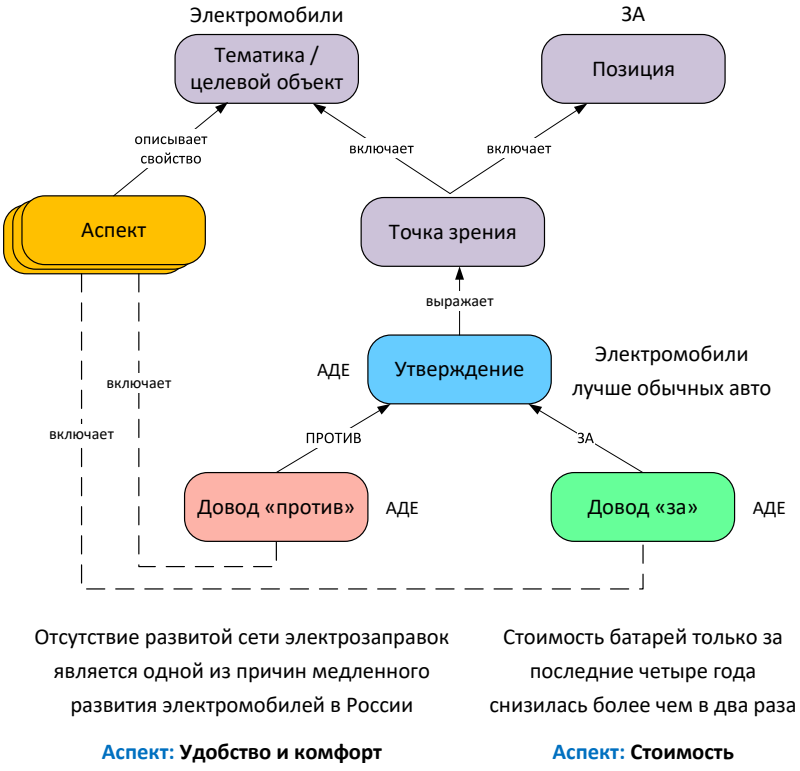


Рисунок 1. Основные понятия в области анализа аргументации

Утверждение: Электромобили лучше обычных авто.

Довод «за»: Стоимость батарей только за последние четыре года снизилась более чем в два раза.

Довод «против»: Отсутствие развитой сети электрозаправок является одной из причин медленного развития электромобилей в России.

Аспекты: «Стоимость» (для довода «за»), «Удобство и комфорт» (для довода «против»).

2. Обзор предыдущих работ

В работах [10, 23] рассматриваются тексты на английском языке и применяются традиционные модели машинного обучения бинарных и многоклассовых классификаторов для определения роли АДЕ в аргументационной структуре. Для обучения моделей авторы использовали различные признаки: лексические, синтаксические, контекстуальные, векторные представления предложений; оценивали их влияние на качество обучения.

Авторы работ [15, 16] учитывали специфику текстов на русском языке для определения того, является предложение доводом «за» или «против», а также создали словари признаков. Было исследовано влияние на качество классификации предложения, добавление в вектор признаков информации о предыдущем и следующем предложении.

В работе [17] используется комбинированный подход к частичному извлечению аргументативной структуры текстов на русском языке с использованием шаблонов индикаторов аргументации, созданных лингвистом и автоматически расширенных.

В статье [21] была предложена нейросетевая модель Arg-CTRL для управляемой генерации текстов на основе известной модели CTRL [25]. Управление порождением текста осуществляется с помощью специальных управляющих кодов, включающих тему, позицию и аспект аргумента. Для учета аспектов применялись информационно-поисковая система и модель BERT.

В нашей работе впервые предлагается метод классификации аспектов аргументации для русскоязычных текстов, в котором модели на основе BERT используются для формирования векторных представлений текстов, а также для отбора предложений, потенциально содержащих аргументы.

3. Метод классификации аспектов

Схема предлагаемого метода классификации аспектов представлена на рисунке 2 и состоит из четырех этапов.

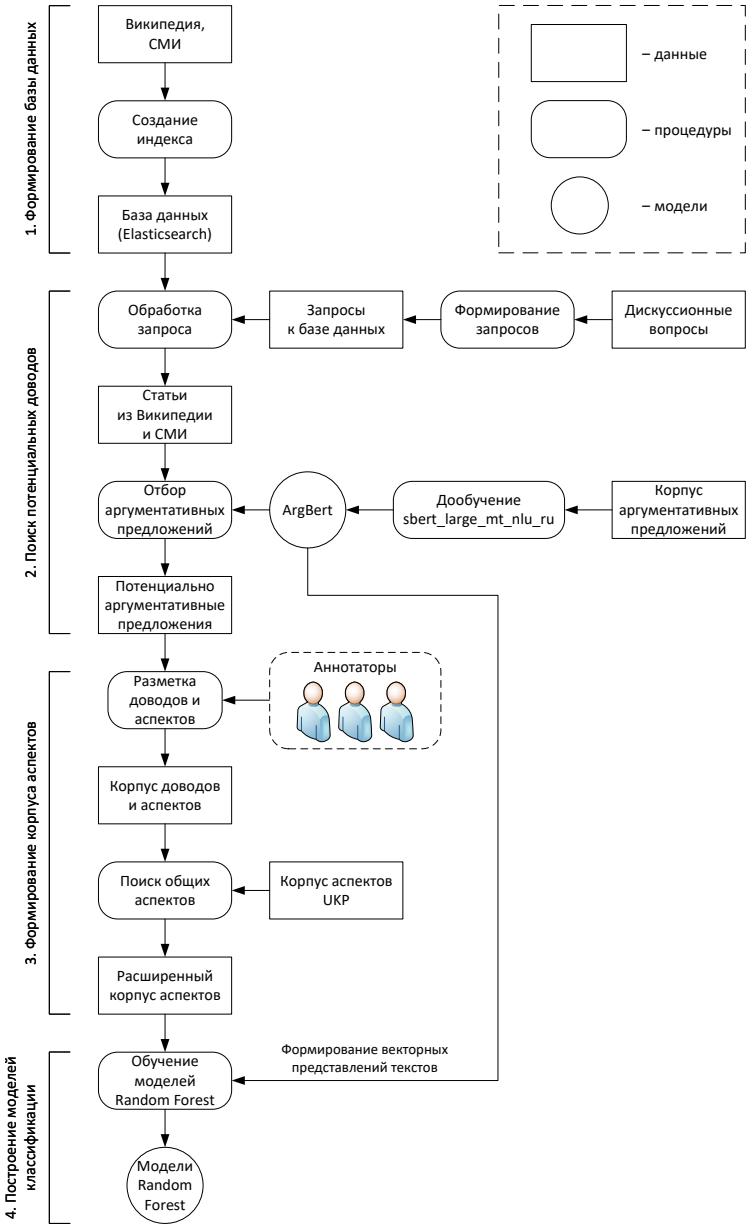


Рисунок 2. Схема метода классификации аспектов

На первом этапе формируется текстовая база данных материалов СМИ и Википедии на основе поисковой системы Elasticsearch [**26**], с помощью которой можно быстро находить тексты требуемой тематики.

На втором этапе из найденных текстов по заданным тематикам отбираются предложения, потенциально содержащие доводы, с помощью специально обученной нейросетевой языковой модели ArgBERT.

На третьем этапе отобранные предложения размечают аннотаторы: содержит ли предложение аргументацию, является ли оно доводом или контрдоводом, и какой аспект отражает. Сформированный размеченный корпус расширяется с помощью переведенной версии корпуса UKP Argument Aspect Detection [**21**].

На последнем этапе строятся модели Random Forest для классификации аспектов на основе обучения с использованием сформированного корпуса.

В следующем разделе подробно рассматриваются отдельные этапы предлагаемого метода и полученные с его помощью экспериментальные результаты.

4. Результаты экспериментов

4.1. Формирование базы данных

База данных была создана на основе статей из русскоязычной Википедии (3 994 609 статей) и новостных статей сайта *Lenta.ru* (800 976 статей). Из статей извлекалась только текстовая информация (рисунки, формулы, ссылки на источники удалялись). Для каждой статьи сохранялся идентификатор, URL-адрес, название статьи.

Затем были построены два полнотекстовых индекса в документо-ориентированной СУБД Elasticsearch [**26**], позволяющие быстро находить тексты по ключевым словам с учетом морфологии.

4.2. Поиск потенциальных доводов

Тематики, по которым существуют спорные точки зрения, были выбраны с помощью сайта *ВЦИОМ*, на котором публикуются аналитические социологические обзоры по самым значимым для российского общества проблемам. Дискуссионные вопросы по выбранным тематикам приведены в таблице 1. По каждому вопросу была определена позиция, зафиксированная в утверждении, которую будут доказывать или опровергать найденные доводы.

Таблица 1. Выбранные тематики и соответствующие им дискуссионные вопросы и утверждения

№	Тематика	Дискуссионный вопрос	Утверждение
1	Свободные деньги	Свободные деньги: тратить или откладывать?	Свободные деньги лучше тратить, чем откладывать
2	Пенсионные сбережения	Надо ли откладывать деньги на пенсию?	Нужно делать пенсионные сбережения
3	Криптовалюта	Следует ли использовать криптовалюту?	Нужно использовать криптовалюту и инвестировать в неё
4	Супермаркеты и продуктовые рынки	Где лучше покупать продукты: в супермаркетах или на продуктовых рынках?	Продукты лучше покупать в супермаркете, а не на рынке
5	Покупки в интернете	Можно ли делать покупки в интернете?	Следует делать покупки в интернете
6	Фриланс	Что лучше: фриланс или работа по найму?	Работа фриланс лучше работы по найму
7	Удаленная работа	Что лучше: работа в офисе или удаленная работа?	Удаленная работа предпочтительнее работы в офисе
8	Электросамокаты	Следует ли ввести ограничения для электросамокатов?	Следует ввести ограничения для электросамокатов
9	Электромобили	Что лучше: электромобили или обычные авто?	Электромобили лучше обычных авто
10	Онлайн-образование	Может ли онлайн-образование конкурировать с традиционным образованием?	Онлайн-образование может конкурировать с традиционным образованием
11	Донорство крови	Опасно ли донорство крови?	Донорство безопасно
12	Киберспорт	Следует ли киберспорт сделать олимпийским видом спорта?	Киберспорт следует сделать олимпийским видом спорта
13	Детские лагеря	Следует ли отправлять ребенка в детский лагерь?	Детский лагерь влияет на ребенка позитивно
14	Детские гаджеты	Следует ли давать гаджеты детям?	Гаджеты влияют на детей позитивно
15	Детские видеоблоги	Следует ли поощрять детей создавать видеоблоги?	Следует поощрять детей создавать видеоблоги
16	Шутеры	Опасны ли видеоигры-шутеры?	Видеоигры-шутеры безопасны
17	Бумажные и электронные книги	Что лучше: бумажные книги или электронные книги?	Бумажные книги лучше электронных

По каждой тематике были сформированы запросы к Elasticsearch, содержащие часть утверждения и шаблоны для учета синонимов с целью повышения вариативности запросов. Например, запросы и шаблоны, сформированные для тематики «Свободные деньги», показаны в таблице 2.

Таблица 2. Пример поисковых запросов по тематике «Свободные деньги» и используемые при поиске шаблоны

Запросы	Шаблоны
откладывать свободные деньги тратить свободные деньги лучше откладывать свободные деньги лучше тратить свободные деньги выбрать тратить свободные деньги стратегия оптимальная откладывать свободные деньги стратегия оптимальная тратить свободные деньги	тратить = [тратить | потратить] откладывать = [откладывать | вкладывать | сберегать | сохранять | вложить | инвестировать | копить] лучше = [лучше | выгоднее | предпочтительнее]

Таким образом, например, при поиске статей по запросу «лучше откладывать свободные деньги», выполнялся поиск «лучше откладывать свободные деньги», «лучше вкладывать свободные деньги», «выгоднее откладывать свободные деньги» и т. п.

В результате поиска было найдено 520 статей: 258 – по документам Википедии и 262 – по документам Lenta.ru. Найденные статьи были разделены на предложения при помощи библиотеки Razdel проекта *Natasha*[URL]. Полученные предложения классифицированы на основе специально обученной для этой цели модели ArgBERT на два класса – «довод»/«не довод». В качестве базовой модели для ArgBERT была использована модель типа BERT [18] *sbert_large_mt_nlu_ru*[URL]. Она была дообучена нами для решения задачи бинарной классификации «довод»/«не довод» на объединении переводных версий [16] корпусов с разметкой по аргументации: ArgMicro [27], Persuasive Essays [10] и UKP

Sentential Argument Mining Corpus [28]. Эта дообученная модель получила название ArgBERT и *предоставлена в общий доступ*.

В результате классификации предложений на основе модели ArgBERT было выделено в качестве потенциальных аргументов 15,4% от общего количества найденных предложений. Мы случайным образом отобрали для разметки ровно 5 000 потенциально аргументативных предложений, сохраняя полученные исходно в ходе поиска пропорции по аспектам. Было отобрано 4 247 предложений из Википедии и 753 предложения из Lenta.ru.

4.3. Формирование корпуса аспектов

На третьем этапе предложения, предположительно содержащие доводы, были размечены тремя аннотаторами. Разметка осуществлялась по трем параметрам:

(*1*) является ли предложение доводом по отношению к заданному утверждению;

(*2*) если предложение является доводом, то выражена позиция «за» или «против»;

(*3*) если предложение является доводом, то какой упоминается аспект.

В качестве аргументационных аннотаторы рассматривали такие предложения, которые потенциально могут быть использованы для убеждения оппонента в дискуссии относительно данного утверждения. Решение принималось большинством аннотаторов (то есть как минимум два аннотатора должны быть согласны).

Для формирования перечня аспектов была проведена пилотная разметка (100 предложений из Википедии, 100 предложений из Lenta.ru). По результатам этой разметки был сформирован первоначальный перечень аспектов, который уточнялся в ходе основного этапа разметки. Было выделено 16 универсальных аспектов, каждый из которых мог относиться к любой рассматриваемой тематике, например, «Безопасность», «Влияние на здоровье», «Стоимость».

Из 5 000 предложений были размечены как аргументационные 548 предложений (11%). Для одного аргументационного предложения аннотатор мог выделить от одного до трех аспектов. В результате для

аргументационных предложений было выделено 820 аспектов. Наиболее частыми аспектами оказались «Безопасность» (133 предложения), «Надежность» (90), «Удобство и комфорт» (88).

Полный перечень аспектов с указанием их частоты приведен в таблице 3.

Таблица 3. Частотность аспектов в размеченном корпусе (в одном предложении могло быть выделено до трех аспектов)

№	Аспект	Частота, предложения
1	Безопасность	133
2	Надежность	90
3	Удобство и комфорт	88
4	Отношение властей	78
5	Перспективы	72
6	Влияние на здоровье	56
7	Стоимость	55
8	Влияние на психику	51
9	Эффективность	39
10	Доходность	34
11	Уровень жизни	26
12	Юридический аспект	26
13	Экологичность	23
14	Общение с людьми	22
15	Популярность	15
16	Качество	12
Всего:		820

Согласие между аннотаторами, вычисленное по метрике Fleiss' kappa [**30**], составило 0,6458 по предложениям из Lenta.ru, 0,6249 по предложениям из Википедии и 0,6427 по всему корпусу, что соответствует значительной (substantial) степени согласия [**31**]. Примеры разметки приведены в таблице 4.

Таблица 4. Примеры разметки

Основное утверждение	Предложение	Довод	За/против	Аспект
Электромобили лучше обычных авто	Электрокары широко применяются на предприятиях для перевозки грузов внутри цехов благодаря отсутствию вредных выхлопов, на аэродромах и железнодорожных вокзалах.	да	за	Экологичность
	Ремни безопасности в электрокарах, как считают специалисты Tesla, могут неожиданно расстегнуться.	да	против	Надежность
	Среди положительных характеристик отмечены, прежде всего, хорошие ходовые данные.	нет	–	–
Бумажные книги лучше электронных	Обвинители настаивали на том, что программа Advanced eBook Processor сродни тем, которые можно использовать для изготовления нелегальных копий электронных книг.	да	за	Юридический аспект
	В одну такую электронную книгу должны поместиться до двух тысяч обычных.	да	против	Удобство и комфорт
Гаджеты влияют на детей позитивно	Носимые датчики могут обнаруживать аномальные и непредвиденные ситуации, а также контролировать физиологические параметры и симптомы.	да	за	Влияние на здоровье
	Шестиклассники, которые пять дней провели без своих смартфонов, стали гораздо лучше понимать эмоциональный настрой своих собеседников, выяснили американские психологи.	да	против	Влияние на психику, Общение с людьми

В корпусе UKP Argument Aspect Detection [21] также содержится разметка по аспектам. Однако способ выделения аспектов отличается от нашей работы. В [21] аннотаторы выделяли аспект как часть анализируемого предложения, так что в итоге сформировалось 6 747 аспектов, б**о**льшая часть которых встречается только один раз (5 429 – 80,4%) и только 117 (1,7%) – более шести раз.

Мы сопоставили наиболее частотные (которые встречаются более шести раз) аспекты из работы [21] нашим универсальным аспектам. Затем наш корпус аспектов был дополнен 890 переведенными предложениями из корпуса UKP Argument Aspect Detection для аспектов с установленной связью. Итоговый расширенный корпус включает 1 426 уникальных предложений, аспекты в которых отмечены 1 764 раза.

4.4. Построение моделей классификации

Для классификации предложений по упоминаемым в них аспектам была использована модель машинного обучения Random Forest («случайный лес»), показавшая наилучшие результаты в предварительных экспериментах среди других моделей машинного обучения. Модель Random Forest представляет собой ансамбль большого количества решающих деревьев [32]. Для формирования векторного представления предложений применялась построенная модель ArgBERT. Также в качестве классификатора была протестирована двухслойная нейронная сеть (двухслойный персептрон) с использованием класса BertForSequenceClassification из библиотеки *transformers*. Для сравнения с указанными моделями были использованы простейшие базовые классификаторы для каждого аспекта, применявшие элементарное решающее правило, в соответствии с которым любое предложение относилось к данному аспекту.

Тестирование осуществлялось на основе перекрестной проверки по 10 блокам с сохранением распределения предложений по аспектам. Для каждого аспекта строилась своя модель Random Forest, которая отличала данный аспект от всех остальных. Таким образом, для каждой модели Random Forest формировался отдельный корпус: в него с меткой 1 входили предложения, содержащие данный аспект, а с меткой 0 – предложения, в которых данный аспект отсутствовал. Модель BertForSequenceClassification обучалась распознавать 16 аспектов одновременно в режиме многозначной классификации (multi-label classification); для неё в ходе предварительных экспериментов было установлено оптимальное количество эпох, равное 10.

В таблице 5 приведены результаты экспериментов для простейших базовых моделей (всего 16 моделей), модели BertForSequenceClassification (единая модель для всех аспектов) и моделей Random Forest (всего 16 моделей).

Таблица 5. Результаты классификации по аспектам (F1-мера)

Аспект	Базовые классификаторы	BertForSequence-Classification	Random Forest
Безопасность	0,1970	0,7307	*0,7517*
Влияние на здоровье	0,0936	0,7854	*0,8006*
Влияние на психику	0,0860	0,6648	*0,7643*
Доходность	0,0589	0,1967	*0,5052*
Качество	0,0216	0,0000	*0,4944*
Надежность	0,1424	0,5558	*0,6472*
Общение с людьми	0,0389	0,0000	*0,5489*
Отношение властей	0,1258	0,6194	*0,7604*
Перспективы	0,1172	0,2462	*0,6219*
Популярность	0,0268	0,0000	*0,4929*
Стоимость	0,0920	0,6332	*0,6945*
Удобство и комфорт	0,1396	0,5402	*0,6722*
Уровень жизни	0,0457	0,5890	*0,7638*
Экологичность	0,0406	*0,6050*	0,5215
Эффективность	0,0671	0,2200	*0,6181*
Юридический аспект	0,0457	0,2467	*0,5386*
В среднем	0,0837	0,4146	*0,6373*

Из таблицы 5 видно, что модели Random Forest превосходят модель BertForSequenceClassification в среднем на 22,2 процентных пункта (0,6373 против 0,4146). Построение отдельной модели для каждого аспекта оказывается более выгодной стратегией, чем единая модель для всех аспектов. Модель BertForSequenceClassification оказалась лучше только для аспекта «Экологичность».

Для пяти аспектов («Безопасность», «Влияние на здоровье», «Влияние на психику», «Отношение властей» и «Уровень жизни») качество классификации в соответствии с F1-мерой оказалось выше 0,75 (от 0,7517 для «Безопасности» до 0,8006 для «Влияния на здоровье»). Два аспекта

распознаются с качеством, ниже 0,5: «Качество» (0,4944) и «Популярность» (0,4929). Значения F1-меры для базовых классификаторов достаточно низкие и составляют от 0,0216 до 0,1970 (в среднем 0,0837).

Также было исследовано влияние расширения исходного корпуса аспектов за счет добавления предложений из корпуса UKP Argument Aspect Detection. Такое расширение привело к повышению F1-меры для Random Forest почти на 2 процентных пункта: с 0,6186 до 0,6373, а для BertForSequenceClassification – более чем на 9 процентных пунктов: с 0,3224 до 0,4146. Таким образом, добавление переведенных аргументативных предложений позволяет повысить качество классификации аспектов.

Заключение

Таким образом, в настоящей работе на основе предложенного метода был впервые сформирован русскоязычный текстовый корпус, включающий 1426 предложений и размеченный по 16 аспектам аргументации, разработаны языковая модель ArgBERT для классификации аргументов и модели Random Forest для классификации аспектов аргументации.

Предложенные корпус и модели возможно применять для поиска и анализа аргументации в текстах. В частности, сформированный текстовый корпус может быть использован для обучения и тестирования новых моделей. Модель ArgBERT позволяет осуществлять отбор аргументативных предложений. Построенные модели Random Forest возможно использовать для автоматической разметки предложений по аспектам с учетом того, что одному предложению может соответствовать несколько аспектов (многозначная классификация). Наилучшее качество разработанные модели демонстрируют для аспектов «Безопасность», «Влияние на здоровье», «Влияние на психику», «Отношение властей» и «Уровень жизни» (выше 0,75 по F1-мере).

Сформированный текстовый корпус и построенные модели *предоставлены в общий доступ*[URL].

В работе [**33**] показано, что на основе дообучения относительно небольшой языковой модели (760 млн параметров) с использованием полученного корпуса возможно осуществлять генерацию аргументативных предложений с заданными аспектами, в том числе для новых, неизвестных языковой модели аспектов. Из сгенерированных языковой моделью предложений более 50% являются доводами с заданными аспектами.

В дальнейших исследованиях предполагается расширить сформированный корпус аспектов на основе предложенного метода, а также использовать данный корпус для дообучения открытых больших языковых моделей типа LLaMA [34] (с размерами 7, 13, 30 и 65 млрд параметров), которые потенциально могут существенно повысить долю правильно генерируемых доводов с заданными аспектами.

Список литературы

[1] van Eemeren F. H., Grootendorst R., Johnson R. H., Plantin C., Willard C. A. *Fundamentals of Argumentation Theory. A Handbook of Historical Backgrounds and Contemporary Developments.*– New York–London: Routledge Taylor& Francis Group.– 1996.– ISBN 978-1-136-68803-4. ↑26

[2] Lawrence J., Reed C. *Argument mining: a survey* // Computational Linguistics.– 2020.– Vol. **45**.– No. 4.– Pp. 765–818. ↑26

[3] Stede M., Schneider J. *Argumentation Mining*, Synthesis Lectures on Human Language Technologies.– Vol. **40**.– Morgan & Claypool.– 2018.– ISBN 978-3-031-01041-5.– xv+175 pp. ↑26, 28

[4] Addawood A. A., Bashir M. N. *What is your evidence? A study of controversial topics on social media* // *Proceedings of the Third Workshop on Argument Mining*, ArgMining-2016 (Berlin, Germany).– ACL.– 2016.– Pp. 1–11. ↑26

[5] Lippi M., Palka P., Contissa G., Lagioia F., Micklitz H.-W., Sartor G., Torroni P. *CLAUDETTE: An automated detector of potentially unfair clauses in online terms of service* // Artificial Intelligence and Law.– 2019.– Vol. **27**.– Pp. 117–139. ↑26

[6] Green N. L. *Towards mining scientific discourse using argumentation schemes* // Argument & Computation.– 2018.– Vol. **9**.– No. 2.– Pp. 121–135. ↑26

[7] Hua X., Nikolov M., Badugu N., Wang L. *Argument mining for understanding peer reviews* // *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*– V. 1: *Long and Short Papers*, NAACL 2019 (Minneapolis, Minnesota).– ACL.– 2019.– Pp. 2131–2137. ↑26

[8] Roush A., Balaji A. *DebateSum: A large-scale argument mining and summarization dataset* // *Proceedings of the 7th Workshop on Argument Mining.*– ACL.– 2020.– Pp. 1–7. ↑26

[9] El Baff R., Wachsmuth H., Al-Khatib K., Stein B. *Analyzing the persuasive effect of style in news editorial argumentation* // *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.*– ACL.– 2020.– Pp. 3154–3160. ↑26

[10] Stab C., Gurevych I. *Identifying argumentative discourse structures in persuasive essays* // *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP-2014 (Doha, Qatar).– ACL.– 2014.– Pp. 46–56. ↑26, 29, 33

[11] Mohammad S., Kiritchenko S., Sobhani P., Zhu X., Cherry C. *SemEval-2016 task 6: Detecting stance in tweets // Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval-2016 (San Diego, California).– 2016.– Pp. 31–41. ↑26

[12] Bondarenko A., Hagen M., Potthast M., Wachsmuth H., Beloucif M., Biemann C., Panchenko A., Stein B. *Touche: First shared task on argument retrieval // Proceedings of the 42nd European Conference on Information Retrieval*, ECIR 2020, Advances in Information Retrieval.– vol. **12036**.– 2020.– Pp. 517–523. ↑26

[13] Bondarenko A., Gienapp L., Frobe M., Beloucif M., Ajjour Y., Panchenko A., Biemann C., Stein B., Wachsmuth H., Potthast M., Hagen M. *Overview of Touché 2021: Argument retrieval // Experimental IR Meets Multilinguality, Multimodality, and Interaction*, CLEF 2021, Lecture Notes in Computer Science.– vol. **12880**, Cham: Springer.– 2021.– ISBN 978-3-030-85250-4.– Pp. 450–467. ↑26

[14] Kotelnikov E., Loukachevitch N., Nikishina I., Panchenko A. *RuArg-2022: Argument mining evaluation*, Papers from the Annual International Conference "Dialogue-2022" (Moscow, June 15–18, 2022), Computational Linguistics and Intellectual Technologies.– vol. **21**.– ISBN 978-5-7281-3205-9.– Pp. 333–348. ↑26

[15] Fishcheva I. N., Goloviznina V. S., Kotelnikov E. V. *Traditional machine learning and deep learning models for argumentation mining in Russian texts*, Papers from the Annual International Conference "Dialogue-2021", Computational Linguistics and Intellectual Technologies.– vol. **20**.– 2021.– ISBN 978-5-7281-3032-1.– Pp. 246–258. ↑26, 29

[16] Fishcheva I. N., Kotelnikov E. V. *Cross-lingual argumentation mining for Russian texts*, 8th International Conference "Analysis of Images, Social networks and Texts" (AIST 2019), Lecture Notes in Computer Science.– vol. **11832**, Cham: Springer.– 2019.– ISBN 978-3-030-37333-7.– Pp. 134–144. ↑26, 29, 33

[17] Salomatina N. V., Kononenko I. S., Sidorova E. A., Pimenov I. S. *Identification of connected arguments based on reasoning schemes "from expert opinion"*, International Conference «Marchuk Scientific Readings 2020» (MSR-2020), dedicated to the 95th anniversary of the birthday of RAS Academician Guri I. Marchuk (October 19–23, 2020, Akademgorodok, Novosibirsk, Russia), Journal of Physics: Conference Series.– vol. **1715**.– 2021.– id. 012013.– 11 pp. ↑26, 29

[18] Devlin J., Chang M.-W., Lee K., Toutanova K. *BERT: Pre-training of deep bidirectional transformers for language understanding // Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics.*– V. 1: *Long and Short Papers* (Minneapolis, Minnesota).– ACL.– 2019.– Pp. 4171–4186. ↑26, 33

[19] Brown T., Mann B., Ryder N., Subbiah M., Kaplan J. D., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D., Wu J., Winter C., Hesse Ch., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner Ch., McCandlish S., Radford A., Sutskever I., Amodei D. *Language models are few shot learners*, NeurIPS 2020, Advances in Neural Information Processing Systems.– vol. **33**.– 2020.– ISBN 9781713829546.– Pp. 1877–1901. ↑26

[20] Ruckdeschel M., Wiedemann G. *Boundary detection and categorization of argument aspects via supervised learning // Proceedings of the 9th Workshop on Argument Mining* (Online and in Gyeongju, Republic of Korea).– International Conference on Computational Linguistics.– 2022.– Pp. 126–136. URL ↑27

[21] Schiller B., Daxenberger J., Gurevych I. *Aspect-controlled neural argument generation , aspect-controlled neural argument generation // Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL 2021.– ACL.– 2021.– Pp. 380–396. doi ↑27, 28, 29, 31, 37

[22] Jurkschat L., Wiedemann G., Heinrich M., Ruckdeschel M., Torge S. *Few-shot learning for argument aspects of the nuclear energy debate // Proceedings of the 13th Language Resources and Evaluation Conference*, LREC-2022 (Marseille, France).– European Language Resources Association.– 2022.– Pp. 663–672. URL ↑27

[23] Stab C., Gurevych I. *Recognizing insufficiently supported arguments in argumentative essays // Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.– V. 1: *Long Papers*, EACL-2017 (Valencia, Spain).– ACL.– 2017.– Pp. 980–990. URL ↑29

[24] Fishcheva I. N., Osadchiy D., Bochenina K. O., Kotelnikov E. V. *Argumentative text generation in economic domain*, Papers from the Annual International Conference "Dialogue-2022" (Moscow, June 15–18, 2022), Computational Linguistics and Intellectual Technologies.– vol. **21**.– ISBN 978-5-7281-3205-9.– Pp. 211–222. doi ↑

[25] Keskar N. S., McCann B., Varshney L. R., Xiong C., Socher R. *CTRL: A conditional transformer language model for controllable generation.*– 2019.– 18 pp. arXiv 1909.05858 ↑29

[26] Gormley C., Tong Z. *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine.*– O'Reilly Media Inc..– 2015.– ISBN 978-1449358549.– 721 pp. ↑31

[27] Peldszus A., Stede M. *Joint prediction in MST-style discourse parsing for argumentation mining // Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP 2015 (Lisbon, Portugal).– ACL.– 2015.– Pp. 938–948. doi ↑33

[28] Stab C., Miller T., Schiller B., Rai P., Gurevych I. *Cross-topic argument mining from heterogeneous sources* // *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP-2018 (Brussels, Belgium).– ACL.– 2018.– Pp. 3664–3674. ↑34

[29] Manning C. D., Raghavan P., Schütze H. *Introduction to Information Retrieval.*– Cambridge University Press.– 2008.– ISBN 978-0521865715.– 506 pp. ↑

[30] Fleiss J. L. *Measuring nominal scale agreement among many raters* // Psychological Bulletin.– 1971.– Vol. **76**.– No. 5.– Pp. 378–382. ↑35

[31] Artstein R., Poesio M. *Inter-coder agreement for computational linguistics* // Computational Linguistics.– 2008.– Vol. **34**.– No. 4.– Pp. 555–596. ↑35

[32] Breiman L. *Random forests* // Machine Learning.– 2001.– Vol. **45**.– Pp. 5–32. ↑37

[33] Goloviznina V. S., Fishcheva I. N., Peskisheva T. A., Kotelnikov E. V. *Aspect-based argument generation in Russian*, Papers from the Annual International Conference "Dialogue" (2023) (June 14–16, 2023), Computational Linguistics and Intellectual Technologies.– vol. **22**, Supplementary volume.– Pp. 117–129. ↑39

[34] Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M.-A., Lacroix T., Rozière B., Goyal N., Hambro E., Azhar F., Rodriguez A., Joulin A., Grave E., Lample G. *LLaMA: Open and efficient foundation language models.*– 2023.– 27 pp. arXiv 2302.13971 ↑40

Поступила в редакцию 01.07.2023;
одобрена после рецензирования 19.07.2023;
принята к публикации 20.07.2023;
опубликована онлайн 19.10.2023.

Рекомендовал к публикации *д.ф.-м.н. Н. В. Лукашевич*

Информация об авторах:

Ирина Николаевна Фищева

Старший преподаватель кафедры прикладной математики и информатики Вятского государственного университета. Область научных интересов: автоматическая обработка текстов, машинное обучение, анализ и генерация аргументов в текстах на естественном языке, разработка программного обеспечения.

0000-0002-6941-2009
e-mail: fishchevain@gmail.com

Татьяна Анатольевна Пескишева

к. техн. н., доцент кафедры прикладной математики и информатики Вятского государственного университета. Научные интересы: автоматическая обработка текстов, машинное обучение, операционные системы, компьютерные сети.

0009-0000-9843-0911

e-mail: peskisheva.ta@gmail.com

Валерия Сергеевна Головизнина

Магистр по профилю «Машинное обучение и анализ данных». Научные интересы: машинное обучение, обработка естественного языка, анализ аргументации.

0000-0003-1167-2606

e-mail: golovizninavs@gmail.com

Евгений Вячеславович Котельников

Д. техн. н., профессор кафедры прикладной математики и информатики Вятского государственного университета. Научные интересы: обработка естественного языка, машинное обучение, языковые модели, анализ аргументации.

0000-0001-9745-1489

e-mail: kotelnikov.ev@gmail.com

Вклад авторов: *И. Н. Фищева* – 25% (формирование базы данных, разметка корпуса, написание черновой версии, доработка и редактирование); *Т. А. Пескишева* – 25% (обучение моделей Random Forest, разметка корпуса, написание черновой версии, доработка и редактирование); *В. С. Головизнина* – 25% (обучение модели ArgBERT, разметка корпуса, написание черновой версии, доработка и редактирование); *Е. В. Котельников* – 25% (идея, методология, написание черновой версии, доработка и редактирование, наставничество, администрирование).

Авторы заявляют об отсутствии конфликта интересов.

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article ARTIFICIAL INTELLIGENCE, INTELLIGENCE SYSTEMS, NEURAL NETWORKS

UDC 004.89:81'322
10.25209/2079-3316-2023-14-4-25-45

Method for classifying aspects of argumentation in Russian-language texts

Irina Nikolaevna **Fishcheva**[1], Tatiana Anatolevna **Peskisheva**[2], Valeriya Sergeevna **Goloviznina**[3], Evgeny Vyacheslavovich **Kotelnikov**[4]✉
Vyatka State University, Kirov, Russia
[4]✉*kotelnikov.ev@gmail.com*

Abstract. Argumentation mining in texts has attracted the attention of researchers in recent years due to a wide range of applications, in particular, in the analysis of scientific and legal texts, news articles, political debates, student essays and social media. Recently, a new task has been set in this area — aspect-based argumentation mining, where an aspect is defined as a property of the object, regarding which the argument is being built. Accounting for the aspects allows, on the one hand, to clarify the direction of the argumentation and understanding of the argument structure; on the other hand, it can be used to generate high-quality and aspect-specific arguments.

The article proposes a method for classifying aspects of argumentation in texts in Russian. On its basis we train and study the models for classifying aspects of argumentation using machine learning and neural networks. For the first time, a Russian-language text corpus was formed, including 1,426 sentences and marked by 16 aspects of argumentation, a neural network language model ArgBERT for classifying arguments was built, and Random Forest models were trained to classify aspects of argumentation. The classification performance obtained on the basis of Random Forest models is 0.6373 by F1-score. The developed models demonstrate the best performance for the aspects "Safety", "Impact on health", "Influence on the psyche", "Attitude of the authorities" and "Standard of living" (F1-score is higher than 0.75).

Key words and phrases: argumentation mining, text corpora, neural network language models, machine learning, Random Forest, aspects of argumentation

2020 *Mathematics Subject Classification*: 68T07; 68T50

Acknowledgments: The study was supported by *Russian Science Foundation grant No. 22-21-00885*

For citation: Irina N. Fishcheva, Tatiana A. Peskisheva, Valeriya S. Goloviznina, Evgeny V. Kotelnikov. *Method for classifying aspects of argumentation in Russian-language texts*. Program Systems: Theory and Applications, 2023, **14**:4(59), pp. 25–45. https://psta.psiras.ru/read/psta2023_4_25-45.pdf

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья МЕТОДЫ ОПТИМИЗАЦИИ И ТЕОРИЯ УПРАВЛЕНИЯ

УДК 517.977:004.421.2
10.25209/2079-3316-2023-14-4-47-66

Метод минимаксного улучшения для неоднородных дискретных систем

Ирина Викторовна **Расина**[1]✉, Александр Олегович **Блинов**[2]

[1] Институт программных систем им. А. К. Айламазяна РАН, Веськово, Россия
[1] Федеральный исследовательский центр «Информатика и управление» РАН, Москва, Россия
[2] Российский государственный социальный университет, Москва, Россия

[1]✉*irinarasina@gmail.com*

Аннотация. Рассматривается класс двухуровневых дискретных неоднородных систем (ДНС) для случая, когда все однородные подсистемы нижнего уровня не только связаны общим функционалом, но имеют и свои собственные цели. Подобные системы широко распространенных на практике (экономика, экология), а также возникают в процессе численного решения задач оптимизации при дискретизации непрерывных управляемых систем.

Предлагается метод улучшения управления второго порядка, для вывода которого используется обобщение достаточных условий оптимальности В. Ф. Кротова. Приводятся иллюстративные примеры.

Ключевые слова и фразы: неоднородные дискретые системы, промежуточные критерии, достаточные условия оптимальности, метод улучшения управления

Благодарности:

[1] Работа выполнена при финансовой поддержке Российского Научного Фонда (проект №21-11-00202)

Для цитирования: Расина И. В., Блинов А. О. *Метод минимаксного улучшения для неоднородных дискретных систем* // Программные системы: теория и приложения. 2023. **Т. 14**. № 4(59). С. 47–66. `https://psta.psiras.ru/read/psta2023_4_47-66.pdf`

Введение

С 80-х годов прошлого века ведутся систематические исследования систем управления неоднородной структуры, для которых характерны самые разнообразные названия: системы переменной структуры [1], дискретно-непрерывные системы [2], логико-динамические системы [3], [4], импульсные системы [5], гибридные системы [6].Наиболее часто употребляемым является термин гибридные системы. Кроме того существует еще одна разновидность систем неоднородной структуры, в которых представлен человеческий фактор. Таковы многоуровневые предприятия, учреждения, коллективы. Для них был введен специальный термин: активные системы. Представление о начальной стадии их исследования можно получить в [7]. Начато оно было в 60-е годы 20 века. Подробная информация о разновидностях таких систем и достигнутых результатах по решению задач управления содержится в обзоре [8]. Поскольку для гибридных систем классические методы оптимального управления непосредственно неприменимы, то предложены многочисленные варианты необходимых и достаточных условий оптимальности управления, отражающие различные подходы к объекту исследования. Один из возможных подходов состоит в обобщении для них достаточных условий оптимальности Кротова [9]. В [10] сформулированы общие условия оптимальности для абстрактной динамической системы как многошаговой, операторы которой на разных шагах допускают различную интерпретацию. В [2, 11, 12] предложена и развита математическая модель дискретно-непрерывной системы (ДНС) в виде конкретизации указанной абстрактной модели [10], применимая для широкого класса задач управления неоднородными процессами, и для нее получен аналог достаточных условий Кротова для непрерывных и дискретных систем. При таком подходе строится иерархическая модель, в которой нижний уровень представляет собой описания однородных процессов на отдельных этапах, а верхний уровень связывает эти описания в единый процесс и управляет функционированием всей системы в целом. В различных задачах управления, в частности в задачах оптимизации, оба уровня рассматриваются во взаимодействии. Подчеркнем, что на нижнем уровне на каждом из этапов фигурировали непрерывные управляемые системы.

В данной работе рассматривается модель, в которой и на нижнем уровне действуют дискретные управляемые системы. Для краткости будем называть их НДС [11]. Такие системы могут рассматриваться как самостоятельные «дискретно-дискретные», примерами могут служить эколого-экономические модели при смене инновационной политики на различных этапах времени. Подобные системы возникают и в качестве вспомогательных для ДНС с учетом естественной дискретизации

непрерывных подсистем в реальных вычислениях [13]. При проведении дискретизации на отдельных шагах можно задавать управляющие воздействия в разных видах (кусочно-постоянное управление, непрерывная функция времени и т.д.) с учетом специфики конкретной задачи. В итоге возникает неоднородная дискретная система [14].

1. Неоднородные дискретные процессы с промежуточными критериями

Рассмотрим двухуровневую модель, в которой нижний уровень составляют дискретные динамические системы однородной структуры. На верхнем уровне фигурирует дискретная модель общего вида

$$x(k+1) = f(k, x(k), u(k)), \quad k \in \mathbf{K} = \{k_I, k_I+1, \ldots, k_F\}, \quad u \in \mathbf{U}(k,x), \tag{1}$$

где k – номер шага (этапа), k_I – начальный шаг, k_F – конечный шаг, x и u – соответственно переменные состояния и управления произвольной природы (возможно различной) для различных k, $\mathbf{U}(k,x)$ – заданное при каждом k и x множество, f – оператор.

На некотором подмножестве $\mathbf{K}' \subset \mathbf{K}$, $k_F \notin \mathbf{K}'$, $u(k)$ интерпретируется как пара $(u^v(k), m^d(k))$. Здесь $m^d(k)$ – процесс $(x^d(k,t), u^d(k,t))$, $t \in \mathbf{T}(k, z(k))$, $m^d(k) \in \mathbf{D}^d(k, z(k))$. Через $\mathbf{D}^d$ обозначено множество допустимых процессов m^d, удовлетворяющих системе

$$x^d(k, t+1) = f^d(k, z, t, x^d(k,t), u^d(k,t)), \quad t \in \mathbf{T} = \{t_I(z), t_I(z)+1, \ldots t_F(z)\}, \tag{2}$$

где x^d и u^d – соответственно переменные состояния и управления произвольной природы, $t_I(z)$ и $t_F(z)$ – начальный и конечный шаги на каждом этапе, номер которого указан в z; f^d – оператор,

$$x^d \in \mathbf{X}^d(k,z,t), \quad u^d \in \mathbf{U}^d(k,z,t,x^d), \quad z = (k, x, u^v).$$

Вектор $z = (k, x, u^d)$ – характеристика воздействия системы верхнего уровня на нижний, играющая на нижнем роль параметров. В ней $u^v = u^v(k)$ – управляющее воздействие верхнего уровня на нижний на каждом этапе. Для системы (2) на множестве $\mathbf{T}$ задана промежуточная цель в виде функционала, который необходимо минимизировать:

$$I^k = \sum_{\mathbf{T}(z) \setminus t_F(z)} f^k(t, x^d(k,t), u^d(k,t)) \to \inf.$$

Здесь $\mathbf{X}^d(k,z,t)$ и $\mathbf{U}^d(k,z,t,x^d)$ – заданные при каждом t, z и x^d множества. В функционале I^k верхний индекс k указывает номер этапа. Оператор

правой части (1) на множестве $\mathbf{K}'$ сводится к следующему:

$$f(k,x,u)=\theta(z,\gamma^d(z)),\qquad \text{где}\quad \gamma^d=(t_I,x_I^d,t_F,x_F^d)\in\mathbf{\Gamma}^d(k,z),$$
$$\mathbf{\Gamma}^d(z)=\{\gamma^d: t_I=\tau(k,z),t_F=\vartheta(k,z),x_I^d=\xi(k,z),x_F^d\in\mathbf{\Gamma}_F^d(k,z)\}.$$

Пусть $\tilde{m}(k)=(x(k),u(k),x^d(k,t),u^d(k,t))$. Управление

$$u(k)=(u^v(k),m^d(k))$$

состоит из фиксированных на шаге u^v и x и дискретно изменяемой части m^d.

Решение описанной двухуровневой системы $m=\{\bar{m}(k):k\in\mathbf{K}'\}$ есть *неоднородный-дискретный процесс*. Совокупность таких решений обозначаем через $\mathbf{D}$ и называем *множеством допустимых неоднородных-дискретных процессов*.

На множестве $\mathbf{D}$ процессов удовлетворяющих (1), (2), рассматривается задача оптимального управления о минимизации концевого функционала $I=F(x(k_F))$ при фиксированных $k_I=0$, k_F, $x(k_I)$ и дополнительных ограничениях $x(k)\in\mathbf{X}(k)$.

2. Достаточные условия оптимальности

Справедливы следующие теоремы [11]:

Теорема 1. *Пусть имеются последовательность процессов* $\{m_s\}\subset\mathbf{D}$ *и функционалы* φ, φ^d, *такие что:*

(1) $R(k,x_s(k),u_s(k))\to\mu(k),\quad k\in\mathbf{K}$;
(2) $R^d(z_s,t,x_s^d(t),u_s^d(t))-\mu^d(z_s,t)\to 0,\quad k\in\mathbf{K}',t\in\mathbf{T}(z_s)$;
(3) $G^d(z_s,\gamma_s^d)-l^d(z_s)\to 0,\quad k\in\mathbf{K}'$;
(4) $G(x_s(t_F))\to l$.

Тогда последовательность $\{m_s\}$ *– минимизирующая для* I *на* $\mathbf{D}$.

Теорема 2. *Для любого элемента* $m\in\mathbf{D}$ *и любых функционалов* φ, φ^d *имеет место оценка*

$$I(m)-\inf_{\mathbf{D}} I\leqslant\Delta=I(m)-l.$$

Пусть имеются два процесса $m^{\mathrm{I}}\in\mathbf{D}$ *и* $m^{\mathrm{II}}\in\mathbf{E}$ *и функционалы* φ *и* φ^d *такие, что* $L\left(m^{\mathrm{II}}\right)<L\left(m^{\mathrm{I}}\right)=I\left(m^{\mathrm{I}}\right)$ *и* $m^{\mathrm{II}}\in\mathbf{D}$. *Тогда* $I(m^{\mathrm{II}})<I(m^{\mathrm{I}})$.

Здесь:

$$L = G(x(k_F)) - \sum_{\mathbf{K}\setminus\mathbf{K}'\setminus k_F} R(k, x(k), u(k)) + \sum_{\mathbf{K}'}\Big(G^d(z) - \\ - \sum_{\mathbf{T}(z)\setminus t_F} R^d(z, t, x^d(k,t), u^d(k,t))\Big),$$

$$G(x) = F(x(k_F)) + \varphi(k_F, x) - \varphi(k_I, x(k_I)),$$

$$R(k, x, u) = \varphi(k+1, f(k, x, u)) - \varphi(k, x),$$

$$G^d(k, z, \gamma^d) = -\varphi(k+1, \theta(k, z, \gamma^d)) + \varphi(k, x(k)) + \\ + \varphi^d(k, z, t_F, x^d_F) - \varphi^d(k, z, t_I, x^d_I),$$

$$R^d(k, z, t, x^d, u^d) = \varphi^d(k, z, t+1, f^d(k, z, t, x^d, u^d)) - \\ - f^k(t, x^d(k,t), u^d(k,t)) - \varphi^d(k, z, t, x^d),$$

$$\mu^d(k, z, t) = \sup_{x^d\in\mathbf{X}^d(k,z,t),\quad u^d\in\mathbf{U}^d(k,z,t,x^d)} R^d(k, z, t, x^d, u^d),$$

$$l^d(k, z) = \inf_{\gamma^d\in\mathbf{\Gamma}^d(k,z),\quad x^d\in\mathbf{X}^d(k,z,t_F)} G^d(k, z, \gamma^d),$$

$$\mu(k) = \begin{cases} \sup\limits_{x\in\mathbf{X}(k),\quad u\in\mathbf{U}(k,x)} R(k, x, u), & t \in \mathbf{K}\setminus\mathbf{K}', \\ -\inf\limits_{x\in\mathbf{X}(k),\quad u^v\in\mathbf{U}^v(k,x)} l^d(z), & k \in \mathbf{K}', \end{cases}$$

$$l = \inf_{x\in\mathbf{\Gamma}\cap\mathbf{X}(k_F)} G(x).$$

Здесь $\varphi(k, x)$ – произвольный функционал, $\varphi^d(k, z, t, x^d)$ – произвольное параметрическое семейство функционалов (с параметром z).

Заметим, что $L(m) = I(m)$ при $m \in \mathbf{D}$, т. е. при выполнении отброшенных связей $L(m)$ совпадает с $I(m)$.

3. Минимаксный метод улучшения

Предположим, что k_I, x_I, K, $t_I(k)$, $t_F(k)$ – заданы; $\mathbf{X}(k) = R^m(k)$, $\mathbf{X}^d(k,t) = R^n(k)$, $\mathbf{\Gamma}(z) = R^{2m}$, $\mathbf{\Gamma}^d(z) = R^{2n}(k)$, $x^d_I(k) = \xi(k, x(k))$, ограничения на переменные состояния обоих уровней отсутствуют, и подсистемы нижнего уровня не зависят от u^v, а используемые конструкции достаточных условий оптимальности таковы, что справедливы все ниже следующие операции.

За основу для построения метода будем использовать задачу улучшения элемента. Ее суть состоит в построении некоторого оператора $\omega : \mathbf{D} \to \mathbf{D}$, для которого $I(\omega(m)) \leqslant I(m)$ (монотонного по функционалу) [13]. Решение

такой задачи может быть реализовано не единственным образом. Далее предлагается один из возможных вариантов.

Итак, пусть задан элемент $m^{\mathrm{I}} \in \mathbf{D}$. Требуется найти элемент $m^{\mathrm{II}} \in \mathbf{D}$ такой, что $I(m^{\mathrm{I}}) \geqslant I(m^{\mathrm{II}})$. Поиск элемента m^{II} и соответственно функций $\varphi^{\mathrm{I}}(k, x(k))$, $\varphi^{d\mathrm{I}}(z, t, x^d)$ будем вести из выполнения условий:

$$R(k, x(k), u^I(k)) \to \min_x, \tag{3}$$

$$G(x) \to \max, \tag{4}$$

$$R^d(z, t, x^d(k, t), u^{d\mathrm{I}}(k, t)) \to \min_{x^d}, \tag{5}$$

$$R^d(z, t, x^d(k, t), u^{d\mathrm{I}}(k, t)) \to \min_x, \tag{6}$$

$$G^d(z, x^d_F, x^d_I) \to \max_x . \tag{7}$$

Пусть

$$\tilde{u}(k, x) = \arg \max_{u \in \mathbf{U}(k,x)} R(k, x(k), u(k)), \tag{8}$$

$$\tilde{u}^d(z, t, x^d) = \arg \max_{u^d \in \mathbf{U}^d(z,t,x^d)} R^d(z, t, x^d, u^d). \tag{9}$$

Тогда из заданной дискретно-непрерывной системы и начальных условий при полученных управлениях находятся функции $x^{\mathrm{II}}(k)$, $x^{d\mathrm{II}}(k, t)$ и программы управлений:

$$u^{\mathrm{II}}(k) = \tilde{u}(k, x^{\mathrm{II}}(k)), \qquad u^{d\mathrm{II}}(k, t) = \tilde{u}^d(k, t, x^{\mathrm{II}}(k), x^{d\mathrm{II}}(k, t)),$$

т. е. элемент m^{II} такой, что $I(m^{\mathrm{II}}) \leqslant I(m^{\mathrm{I}})$. Повторяя итерационно эти операции, получим улучшающую последовательность $\{m_s\}$.

В этом случае справедливо следующее утверждение.

Теорема 3. *Если элемент m^{I} не является решением задачи, то справедливо неравенство $L(m^{\mathrm{I}}) > L(m^{\mathrm{II}})$.*

Доказательство. Покажем, что $I(m^{\mathrm{II}}) - I(m^{\mathrm{I}}) < 0$, следуя работе [15]. Имеем

$$\begin{aligned} L(m^{\mathrm{II}}, \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) - L(m^{\mathrm{I}}, \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) = \\ = G(x^{\mathrm{II}}) - G(x^{\mathrm{I}}) - \\ - \sum_{\mathbf{K} \setminus \mathbf{K}' \setminus k_F} \Big(R(k, x^{\mathrm{II}}(k), u^{\mathrm{II}}(k), \varphi^{\mathrm{I}}) - R(k, x^{\mathrm{I}}(k), u^{\mathrm{I}}(k), \varphi^{\mathrm{I}}) \Big) + \\ + \sum_{\mathbf{K}'} (G^d(z^{\mathrm{II}}(k), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}})) - G^d(z^{\mathrm{I}}(k), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) - \end{aligned}$$

$$- \sum_{\mathbf{T}(z)\setminus t_F} \Big(R^d(z^{\mathrm{II}}(k), t, x^{d\mathrm{II}}(t), u^{d\mathrm{II}}(t), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) - \\ - R^d(z^{\mathrm{I}}(k), t, x^{d\mathrm{I}}(t), u^{d\mathrm{I}}(t), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) \Big) = \\ = \Delta_1 - \Delta_2 + \Delta_3 - \Delta_4,$$

где $\Delta_1 = G(x^{\mathrm{II}}) - G(x^{\mathrm{I}}) < 0$ в силу условия (5),

$$\Delta_2 = \sum_{\mathbf{K}\setminus\mathbf{K}'\setminus k_F} \Big(R(k, x^{\mathrm{II}}(k), u^{\mathrm{II}}(k), \varphi^{\mathrm{I}}) - R(k, x^{\mathrm{I}}(k), u^{\mathrm{I}}(k), \varphi^{\mathrm{I}}) \Big) = \\ = \sum_{\mathbf{K}\setminus\mathbf{K}'\setminus k_F} \Big(R(k, x^{\mathrm{II}}(k), u^{\mathrm{II}}(k), \varphi^{\mathrm{I}}) - R(k, x^{\mathrm{II}}(k), u^{\mathrm{I}}(k), \varphi^{\mathrm{I}}) \Big) + \\ + \sum_{\mathbf{K}\setminus\mathbf{K}'\setminus k_F} \Big(R(k, x^{\mathrm{II}}(k), u^{\mathrm{I}}(k), \varphi^{\mathrm{I}}) - R(k, x^{\mathrm{I}}(k), u^{\mathrm{I}}(k), \varphi^{\mathrm{I}}) \Big) > 0$$

согласно (4),

$$\Delta_3 = \sum_{\mathbf{K}'} (G^d(z^{\mathrm{II}}(k), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) - G^d(z^{\mathrm{I}}(k), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}})) < 0,$$

и

$$\Delta_4 = \sum_{\mathbf{T}(z)\setminus t_F} \Big(R^d(z^{\mathrm{II}}(k), t, x^{d\mathrm{II}}(t), u^{d\mathrm{II}}(t), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) - \\ - R^d(z^{\mathrm{I}}(k), t, x^{d\mathrm{I}}(t), u^{d\mathrm{I}}(t), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) \Big) = \\ = \sum_{\mathbf{T}(z)\setminus t_F} \Big(R^d(z^{\mathrm{II}}(k), t, x^{d\mathrm{II}}(t), u^{d\mathrm{II}}(t), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) - \\ - R^d(z^{\mathrm{II}}(k), t, x^{d\mathrm{II}}(t), u^{d\mathrm{I}}(t), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) \Big) + \\ + \sum_{\mathbf{T}(z)\setminus t_F} \Big(R^d(z^{\mathrm{II}}(k), t, x^{d\mathrm{II}}(t), u^{d\mathrm{I}}(t), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) - \\ - R^d(z^{\mathrm{I}}(k), t, x^{d\mathrm{I}}(t), u^{d\mathrm{I}}(t), \varphi^{\mathrm{I}}, \varphi^{d\mathrm{I}}) \Big)$$

в силу условий (6), (7) и (8). Тогда

$$L(m^{\mathrm{II}}) - L(m^{\mathrm{I}}) = \Delta_1 - \Delta_2 + \Delta_3 - \Delta_4 < 0.$$

□

Из теоремы следует, что при выполнении выше сформулированных условий можно построить такую улучшающую последовательность $\{m_s\}$, что $I(m_{s+1}) \leqslant I(m_s)$.

Перейдем непосредственно к методике поиска функций φ, φ^d. Воспользуемся принципом расширения [14] и теоремой 2. Условия (4), (5), (6), (7),

(8) означают, что функционал L, подсчитанный при управлениях $u^I(k)$, $u^{dI}(k,t)$, исследуется на максимум. Рассмотрим приращение функционала $L = I$, которое представим в виде:

$$\Delta L \approx \mathrm{d}\, G + \frac{1}{2}\,\mathrm{d}^2\, G - \sum_{\mathbf{K}\setminus\mathbf{K}'\setminus k_F}\left(\mathrm{d}\, R + \frac{1}{2}\,\mathrm{d}^2\, R\right) + \\ + \sum_{\mathbf{K}'\setminus k_F}\left(\mathrm{d}\, G^d + \frac{1}{2}\,\mathrm{d}^2\, G^d - \sum_{\mathbf{T}(z)\setminus t_F}\left(\mathrm{d}\, R^d + \frac{1}{2}\,\mathrm{d}^2\, R^d\right)\right)$$

или

$$\Delta L \approx G_x^T\Delta x + \frac{1}{2}\Delta x^T G_{xx}\Delta x - \sum_{\mathbf{K}\setminus\mathbf{K}'\setminus k_F}\left(R_x^T\Delta x + \frac{1}{2}\Delta x^T R_{xx}\Delta x\right) + \\ + \sum_{\mathbf{K}'\setminus k_F}\left(G_{x_F^d}^{c^T}\Delta x_F^d + \frac{1}{2}\Delta x_{F^T}^d G_{x_F^d x_F^d}^d + \right. \\ \left. + G_x^{d^T}\Delta x + \frac{1}{2}\Delta x^T G_{xx}^d\Delta x + \Delta x_{F^T}^d G_{x_F^d x}^d\Delta x\right) - \\ - \sum_{\mathbf{T}(z)\setminus t_F}\left(R_{x^d}^{d^T}\Delta x^d + R_x^{d^T}\Delta x + \frac{1}{2}\Delta x^{d^T} R_{x^d x^d}^d\Delta x^d + \right. \\ \left. + \Delta x^T R_{xx^d}^d\Delta x^d + \frac{1}{2}\Delta x^T R_{xx}^d\Delta x\right).$$

Здесь первые и вторые производные функций R, G, R^d, G^d подсчитаны при $u = u^{\mathrm{I}}$, $u^d = u^{d\mathrm{I}}$, а $\Delta x = x - x^{\mathrm{I}}(k)$, $\Delta x^d = x^d - x^{d\mathrm{I}}(k,t)$, $\Delta x_F^d = x_F^d - x_F^{d\mathrm{I}}$.

Для выполнения условий (4), (5), (6), (7), (8) достаточно положить:

$$G_x = 0, \quad R_x = 0, \quad R_{x^d}^d = 0, \quad G_{x_F^d}^d = 0, \quad G_x^d = 0, \quad R_x^d = 0, \tag{10}$$

$$G_{xx} = -\Lambda^1, \quad R_{xx} = \Lambda^2, \quad R_{x^d x^d}^d = \Lambda^3, \quad R_{xx^d}^d = 0, \quad G_{x_F^d x_F^d}^d = -\Lambda^4, \tag{11}$$

$$G_{xx_F^d}^d = 0, \quad G_{xx}^d = -\Lambda^5, \quad R_{xx}^d = -\Lambda^6. \tag{12}$$

Здесь $-\Lambda^1, \Lambda^2, \Lambda^3, -\Lambda^4, -\Lambda^5, -\Lambda^6$ – положительно определенные диагональные матрицы. Нетрудно видеть, что условия (10), (11), (12) представляют собой достаточные условия экстремума функций G, G^d, R, R^d [14]. Дополним условия (10), (11), (12) условиями первого и второго порядков стыковки уровней:

$$\frac{d}{dx}(\varphi(k+1,\theta(k,x,x_F^d,x_I^d)) - \varphi^d(k,x,t_F,x_F^d)) = 0, \tag{13}$$

$$\frac{d^2}{d^2x}(\varphi(k+1,\theta(k,x,x_F^d,x_I^d)) - \varphi^d(k,x,t_F,x_F^d)) = \Lambda^7, \tag{14}$$

где Λ^7 – положительно определенная диагональная матрица.

Зададим функции φ и φ^d в следующем виде:

$$\varphi(k,x) = \psi^T(k)x + \frac{1}{2}\Delta x^T \sigma(k)\Delta x,$$

$$\varphi^d(z,t,x^d) = \lambda^T(k,t)x + \psi^{d\mathrm{T}}(k,t)x^d + \frac{1}{2}\Delta x^{dT}\sigma^d(k,t)\Delta x^d + \\ + \frac{1}{2}\Delta x^T D(k,t)\Delta x + \Delta x^T \Lambda(k,t)\Delta x^d,$$

где $\psi(k)$, $\lambda(k,t)$, $\psi^d(k,t)$ – вектор-функции размера m,n, n, а $\sigma^d(k,t)$, $D(k,t)$, $\Lambda(k,t)$ – матрицы размера $n\times n$, $m\times m$, $m\times n$ соответственно. Тогда

$$\varphi_x = \psi(k), \quad \varphi_{xx} = \sigma(k), \quad \varphi^d_x = \lambda(k,t), \quad \varphi^d_{x^d} = \psi^d(k,t),$$
$$\varphi^d_{xx} = \sigma^d(k,t), \quad \varphi^d_{x^d x^d} = D(k,t), \quad \varphi^d_{x^d x} = \Lambda(k,t).$$

Кроме того, введем в рассмотрение функции

$$H(k,x(k),\psi(k+1),u(k)) = \\ = \psi^T(k+1)f(k,x(k),u(k)), k\in \mathbf{K}\setminus\mathbf{K}'\setminus k_F,$$
$$H(k,x(k),\psi(k+1),x(k_I),x(k_F)) = \\ = \psi^T(k+1)\theta(k,x(k),x(k_I),x(k_F)), k\in\mathbf{K}',$$
$$H^d(k,x(k),\psi^d(k,t),x^d(k,t),u^d(k,t)) = \\ = \psi^{d\mathrm{T}}(k,t)f^d(k,x(k),x^d(k,t),u^d(k,t)) - f^k(t,x^d(k,t),u^d(k,t)).$$

С учетом введенных обозначений из условий (12), (13), (14) имеем:

$$\psi(k_F) = -F_x,$$

$$\psi(k) = \begin{cases} H_x, & k\in\mathbf{K}\setminus\mathbf{K}'\setminus k_F, \\ H_x + (H^d_{x_I}\xi_x)^T - \lambda(k,t_F) + \\ \quad + \lambda(k,t_I) + \xi_x^T\psi^d(t_I), & k\in\mathbf{K}'\setminus k_F, \end{cases}$$

$$\lambda(k,t) = \lambda(k,t+1) + H^d_x, \quad \lambda(k,t_F) = H_x + \xi_x^T H_{x^d_I},$$

$$\psi^d = H^d_{x^d}, \quad \psi^d(k,t_F) = H_{x^d_F},$$

$$\sigma^d(k,t_F) = \theta^T_{x^d}(t_F)\sigma(k+1)\theta_{x^d}(t_F) + H_{x^d x^d} + \Lambda^4,$$

$$\sigma^d(k,t) = \sigma^d(k,t+1)f^d_{x^d} + f^{d\mathrm{T}}_{x^d}\sigma^d(k,t+1) + H^d_{x^d x^d} + \Lambda^5$$

$$D(k,t) = \frac{1}{2}\Lambda(k,t+1)f^d_{x^d x} + \frac{1}{2}(\Lambda(k,t+1)f^d_{x^d x})^T + \frac{1}{2}\Lambda(k,t+1)f^d_x - \\ - \frac{1}{2}(\Lambda(k,t+1)f^d_x)^T + H^d_{xx} + \Lambda^5,$$

$$D(k,t_F) = \theta_x^T\sigma(k+1)\theta_x + H_x + H_{xx^d}\xi_x + \\ + \theta_x^T\sigma(k+1)\theta_{x^d}\xi_x + \xi_x^T H_{x^dx}\xi_x + \xi_{xx}H_{x^d},$$

$$\Lambda(k,t) = \frac{1}{2}\Lambda(k,t+1)f^d_{x^d} + \frac{1}{2}(\Lambda(k,t+1)f^d_{x^d})^T + \frac{1}{2}f^{d\mathrm{T}}_x\sigma^d + \\ + \frac{1}{2}\sigma^d(k,t+1)f^d_x + H^d_{xx^d},$$

$$\Lambda(k,t_F) = \theta_x^T\sigma(k+1)\theta_{x^d}(t_F) + H_{xx^d}$$

$$\sigma(k) = \begin{cases} f_x^T\sigma(k+1)f_x + H_{xx} + \Lambda^2, & k \in \mathbf{K}\setminus\mathbf{K}'\setminus k_F, \\ H_{xx} + H_{xx^d}(t_I)\xi_x + \theta_x^T\sigma\,(k+1)\,\theta_x + \\ \quad + \xi_x^T\sigma^d\,(t_I)\,\xi_x + H^d_{x^d_I}(t_I)\xi_{xx} + \\ \quad + \xi_x^T\theta^T_{x^d}(t_I)\sigma(k+1)\theta_{x^d}(t_I)\xi_x + \\ \quad + \xi_x^T H_{x^dx^d}(t_I)\xi_x \\ \quad + H^d\xi_{xx}\theta(t_I) + \Lambda^5, & k \in \mathbf{K}'\setminus k_F, \end{cases}$$

$$\sigma(k_F) = -F_{xx} + \Lambda^1.$$

4. Алгоритм метода

1. Задаются произвольные функции $\varphi^{\mathrm{I}}(k,x(k))$ и $\varphi^{d\mathrm{I}}(z,t,x^d)$.

2. По формулам (10), (11) определяются управления $\tilde{u}(k,x)$, $\widetilde{u^d}(k,t,x^d)$.

3. Из уравнений дискретно-дискретного процесса (1)-(2) определяются траектории $x^{\mathrm{I}}, x^{d\mathrm{I}}$ и программы управления $u^{\mathrm{I}}(k), u^{d\mathrm{I}}(k,t)$. Тем самым определяется элемент m^{I} и вычисляется значение функционала I^{I}.

4. «Справа-налево» решается система векторно-матричных уравнений относительно вектор-функций ψ, ψ^d, λ и матриц σ, D, σ^d, Λ. Для определенности можно положить все λ^i_{jj} равными постоянной $\delta^i \le 0$, $i = 2,3,5$; $\delta^i \ge 0, i = 1,4$. Определяются новые функции $\varphi^{\mathrm{II}}(k,x(k))$ и $\varphi^{d\mathrm{II}}(z,t,x^d)$. Переход к пункту 2.

Замечание 1. Если улучшения функционала не произошло, то величины λ^i необходимо увеличить.

Замечание 2. Система векторно-матричных уравнений относительно вектор-функций ψ, ψ^d, λ и матриц σ, D, σ^d, Λ линейная и, следовательно, всегда имеет решение.

Замечание 3. При $\sigma = 0$, $\sigma^d = 0$, $\sigma^d = 0$, $\Lambda = 0$ получается алгоритм улучшения первого порядка.

Замечание 4. Параметры δ^i играют роль регуляторов близости соседних приближений.

5. Примеры

Пример 1. Пусть задана НДС, состоящая из двух этапов, обозначим их номерами 0 и 1. На нулевом этапе управляемый процесс описывается одним уравнением:

$$x^d(t+1) = -2x^d(t) + (u_1^d(t))^2, \quad x^d(0) = 1, \quad t = 0, 1, 2, 3,$$

Задан промежуточный функционал этапа:

$$I^0 = \frac{1}{2}(x^d(t))^2 + \frac{1}{3}(u_1^d)^3,$$

Здесь $f^d = -2x^d(t) + (u_1^d(t))^2$. Следующий первый этап характеризуется другим уравнением и своим промежуточным функционалом:

$$x^d(t+1) = (t - u_2^d)^2, \quad t = 4, 5, 6, \quad I^1 = \frac{1}{2}(x^d)^2 + u_2^d,$$

В данном случае $f^d = (t - u_2^d)^2$. Задан общий функционал задачи:

$$I = F = x^d(7) \to \min.$$

Нетрудно видеть, что момент окончания всего процесса k_F равен 2, тогда $\mathbf{K} = 0, 1, 2$. Смена описания процесса происходит один раз, следовательно $\mathbf{K}' = 1$. Поскольку роль связующей переменной на двух рассматриваемых этапах играет x^d, то в терминах этой переменной легко записать процесс верхнего уровня:

$$x(0) = x^d(0, 0), \quad x(1) = x^d(0, 4), \quad x(2) = x^d(1, 7), \quad x^d(1, 4) = x(1).$$

Так как множество $\mathbf{K}'$ состоит из одного элемента, то для удобства расчетов, модель может быть дополнена третьим мгновенным этапом, не имеющим протяженности во времени и состоящим лишь в передачи информации об окончании второго этапа на верхний уровень. Тогда $x(3) = x(2) = x^d(1, 7)$, $\mathbf{K}' = 1, 2$, $\theta(k) = x^d(k, t_F)$, $\xi(k) = x(k)$. Запишем задачу и конструкции в терминах НДС.

Имеем:

$$f^d(0, t) = -2x^d(t) + (u_1^d(t))^2, \quad f^d(1, t) = (t - u_2^d)^2,$$

$$f^d_{x^d}(0, t) = -2, \quad f^d_{x^d}(1, t) = 0,$$

$$H^d(0, t) = \psi^d(0, t+1)(-2x^d(t) + (u_1^d(t))^2) - \frac{1}{2}(x^d(t))^2 - \frac{1}{3}(u_1^d)^3,$$

$$H^d_{x^d}(0, t) = -2\psi^d(0, t+1) - x^d(t), \qquad H^d_{x^d x^d}(0, t) = -1,$$

$$H^d(1,t) = \psi^d(1,t+1)(t-u_2^d)^2 - \frac{1}{2}(x^d)^2 - u_2^d,$$

$$H^d_{x^d}(1,t) = -x^d(t), \qquad H^d_{x^d x^d}(1,t) = -1,$$

$$R^d(0,t) = \psi^d(0,t+1)(-2x^d(t) + (u_1^d(t))^2) +$$

$$+ \frac{1}{2}\sigma^d(0,t+1)((-2x^d(t) + (u_1^d(t))^2 - x^{d\mathrm{I}(t)})^2 -$$

$$- \frac{1}{2}(x^d(t))^2 - \frac{1}{3}(u_1^d)^3 - \psi^d(0,t)x^d(t) - \frac{1}{2}\sigma^d(0,t)(x^d(t) - x^{d\mathrm{I}}(t))^2,$$

$$R^d(1,t) = \psi^d(1,t+1)(t-u_2^d)^2 +$$

$$+ \frac{1}{2}\sigma^d(1,t+1)((t-u_2^d)^2 - x^{d\mathrm{I}}(t))^2 - \frac{1}{2}(x^d)^2 - u_2^d - \psi^d(1,t)x^d(t) -$$

$$- \frac{1}{2}\sigma^d(1,t)(x^d(t) - x^{d\mathrm{I}}(t))^2.$$

На обоих этапах управляющие воздействия, найденные из необходимых условий экстремума функций $R^d(k,t)$ являются результатом решения следующих уравнений:

$$2\psi^d(0,t+1) + 2\sigma^d(0,t+1)(-2x^d(t) + (\tilde{u}_1^d(t))^2 - x^{d\mathrm{I}(t)}) - \tilde{u}_1^d(t) = 0,$$

$$-2\psi^d(1,t+1)(t-\tilde{u}_2^d) - 2\sigma^d(1,t+1)((t-\tilde{u}_2^d)^2 - x^{d\mathrm{I}(t)})(t-\tilde{u}_2^d) - 1 = 0,$$

$$\psi(2) = -1, \qquad \sigma(2) = \Lambda^1, \qquad \psi(1) = \psi^d(1,4), \qquad \sigma(1) = \sigma^d(1,4) + \Lambda^5,$$

$$\psi^d(0,t) = -2\psi^d(0,t+1) - x^d(t), \qquad \psi^d(1,t) = -x^d(t),$$

$$\sigma^d(0,t) = -4\sigma^d(0,t+1) - 1 + \Lambda^3, \qquad \sigma^d(1,t) = -1 + \Lambda^3,$$

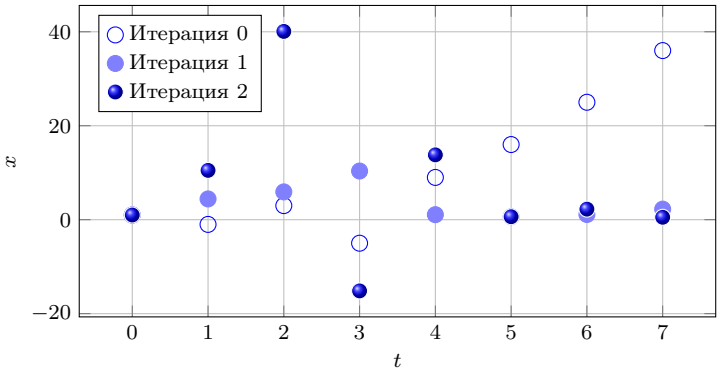
$$\sigma^d(0,3) = \sigma(2), \qquad \sigma^d(1,7) = \sigma(3).$$

Решение получено за 2 итерации. Изменение функционала по итерациям представлено в таблице 1.

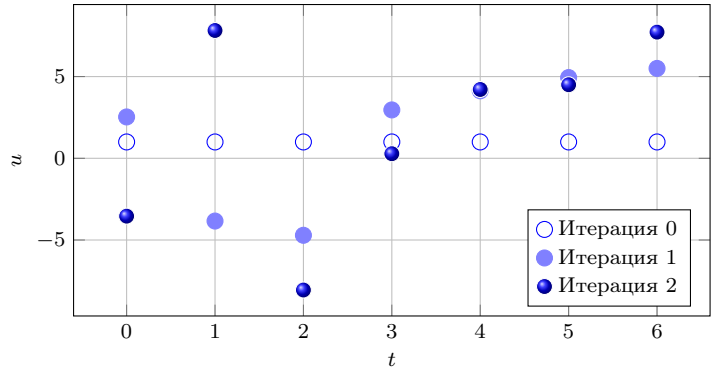
Таблица 1. Результаты расчетов примера 1

Итерация	$\Lambda^1 = \Lambda^3 = \Lambda^5$	$I = x^d(7) \to \min$
0	–	36
1	0	2.25
2	0.7	0.52

При расчетах выбор значения Λ^3 осуществлялся для обеспечения минимума функционала и находился перебором в диапазоне от 0 до 1 с шагом 0.1. Графики переменных состояния и управления приведены на рисунках 1*а* и 1*б*.



(*а*) Состояние



(*б*) Управление

Рисунок 1. Результаты расчетов примера 1

Пример 2.

Рассматривается следующая 2-этапная задача.

1-й этап:

$$x^d(t+1) = (x^d(t))^2 + u^{d1}, \qquad |u^{d1}| \leqslant 1,$$
$$x^d(0) = 0, \qquad f^0 = x^d, \qquad t = 0, 1, 2.$$

2-й этап:

$$x^d(t+1) = u^{d2} - (x^d)^2, \qquad f^1 = x^{d1}u^{d2}, \qquad |u^{d2}| \leqslant 2, \quad t = 3, 4, 5$$
$$I = x^d(6) \to \inf.$$

По аналогии с примером 1 представим эту систему в виде НДС.

Имеем $k = 0, 1, 2, 3$. Поскольку роль связующей переменной на двух рассматриваемых этапах играет x^d, то в терминах этой переменной легко записать процесс верхнего уровня.

Установим взаимосвязь между переменными. В начале процесса при $k = t = 0$, $x(0) = x^d(0) = 0$. Далее $x(1) = x^d(0, 2)$, $x^d(1, 3) = x(1)$. Тогда $I = x(2)$. Так как множество $\mathbf{K}'$ состоит из одного элемента, то для удобства расчетов, модель может быть дополнена третьим мгновенным этапом, не имеющим протяженности во времени и состоящим лишь в передаче информации об окончании второго этапа на верхний уровень. Тогда $x(3) = x(2) = x^d(1, 6)$; $\mathbf{K}' = 1, 2$; $\theta(k) = x^d(k, t_F)$; $\xi(k) = x(k)$.

Последний мгновенный 3-ий этап играет роль передатчика информации об окончании всего процесса и $I = x(3) = x(2)$.

Итак, получили следующую модель НДС:

$$
\begin{aligned}
k = 0: \quad & x^d(0, t+1) = (x^d(0,t))^2 + u^{d1}(0,t), \qquad |u^{d1}(0,t)| \leqslant 1, \\
& x(0) = x^d(0,0) = 0, \qquad f^0(0,t) = x^d(0,t), \qquad t = 0, 1, 2. \\
k = 1: \quad & x^d(1, t+1) = u^{d2} - (x^d)^2, \qquad |u^{d2}(1,t)| \leqslant 2, \\
& x(1) = x^d(0,1), \qquad f^1(1,t) = x^d(1,t) u^{d2}(1,t), \qquad t = 3, 4, 5, \\
k = 2, 3: \quad & x(2) = x(3) = x^d(1,2).
\end{aligned}
$$

Очевидно, что множество $\mathbf{K}' = 1, 2$, а функции $\theta(1) = x^d(0,1)$, $\xi(1) = x(1)$, $\theta(2) = x^d(1,2)$, $\xi(2) = x(2)$.

Заметим, что на обоих этапах процесс нижнего уровня не зависит от переменных состояния верхнего уровня, тогда $\lambda(0,t) = \lambda(1,t) = 0$, $\Lambda(0,t) = \Lambda(1,t) = 0$, $D(0,t) = D(1,t) = 0$.

$$
\begin{aligned}
H^d(0,t) &= \psi^d(0,t+1)((x^{d1}(t))^2 + u^{d1}) - x^d(t), \\
H^d_{x^d}(0,t) &= 2\psi^d(0,t+1)x^{d1}(t) - 1, \quad H^d_{x^d x^d}(0,t) = 2\psi^d(0,t+1), \\
H^d(1,t) &= \psi^d(1,t+1)(u^{d2} - (x^d)^2) - x^d(t)u^{d2}(t), \\
H^d_{x^d}(1,t) &= -2\psi^d(1,t+1)x^d(t) - u^{d2}(t), \quad H^d_{x^d x^d}(1,t) = -2\psi^d(1,t+1), \\
R^d(0,t) &= \psi^d(0,t+1)((x^d(t))^2 + u^{d1}) + \\
&\quad + \frac{1}{2}\sigma^d(0,t+1)((x^d(t))^2 + u^{d1} - x^{d\mathrm{I}}(t))^2 - \\
&\quad - (x^d(t) - \psi^d(0,t)x^d(t) - \frac{1}{2}\sigma^d(0,t)(x^d(t) - x^{d\mathrm{I}}(t))^2, \\
R^d(1,t) &= \psi^d(1,t+1)(u^{d2} - (x^d)^2) + \\
&\quad + \frac{1}{2}\sigma^d(1,t+1)((u^{d2} - (x^d)^2)^2 - x^{d\mathrm{I}}(t))^2 - \\
&\quad - x^d u_2^d - \psi^d(1,t)x^d(t) - \frac{1}{2}\sigma^d(1,t)(x^d(t) - x^{d\mathrm{I}}(t))^2.
\end{aligned}
$$

На обоих этапах воспользуемся необходимыми условиями экстремума функций $R^d(k,t)$ по переменным управления.Найдем производные от этой функции по управляющим переменным и приравняем к нулю. Соответственно по этапам имеем.

$$\psi^d(0,t+1)+\sigma^d(0,t+1)((x^d(t))^2+u^{d1}-x^{d\mathrm{I}}(t))=0,$$

Решение полученного уравнения обозначим через N_1.

$$N_1=\frac{\sigma^d(0,t+1)(x^{d\mathrm{I}}(t)-(x^d(t))^2)-\psi^d(0,t+1)}{\sigma^d(0,t+1)},$$

$$\tilde{u}_1^d(t)=\begin{cases}N_1, & N_1\in(-1,1),\\ -1, & N_1\le(-1),\\ 1, & N_1\ge 1.\end{cases}$$

По аналогии на втором этапе:

$$\psi^d(1,t+1)+\sigma^d(1,t+1)((u^{d2}(t)-(x^d)^2(t)^2-x^{d\mathrm{I}}(t))-x^d(t)=0,$$

$$N_2=\frac{\sigma^d(1,t+1)(x^{d\mathrm{I}}(t)+(x^d(t))^2)-\psi^d(1,t+1)+x^d(t)}{\sigma^d(1,t+1)},$$

$$\tilde{u}_2^d(t)=\begin{cases}N_2, & N_2\in(-2,2),\\ -2, & N_2\le(-2),\\ 2, & N_2\ge 2.\end{cases}$$

$$\psi(2)=-1,\ \sigma(2)=\Lambda^1,\ \psi(1)=\psi^d(1,3),\ \sigma(1)=\sigma^d(1,3)+\Lambda^5,$$

$$\psi^d(0,t)=)=2\psi^d(0,t+1)x^d(0,t)-1,$$

$$\psi^d(1,t)=-2\psi^d(1,t+1)x^d(1,t)-u^{d2}(1,t),$$

$$\sigma^d(0,t)=4\sigma^d(0,t+1)x^d(0,t)+2\psi(0,t+1)+\Lambda^3,$$

$$\sigma^d(1,t)=-4\sigma^d(1,t+1)-2\psi(1,t+1)+\Lambda^3,$$

$$\sigma^d(0,3)=\sigma(2),\qquad \sigma^d(1,6)=\sigma(3).$$

Результаты расчетов представлены в таблице 2 и на рисунках 2*а* и 2*б*.

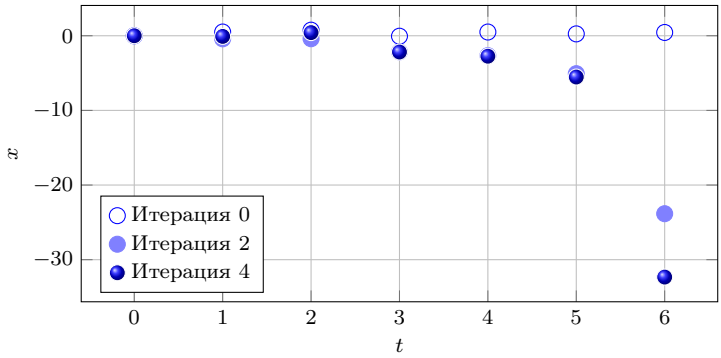
Решение получено за 4 итерации, изменение функционала по итерациям дано в таблице 2. При расчетах выбор значения Λ^3 осуществлялся для обеспечения минимума функционала и находился перебором в диапазоне от 0 до 1 с шагом 0.1. Графики переменных состояния и управления приведены на рисунках 2*а* и 2*б*.

6. Заключение

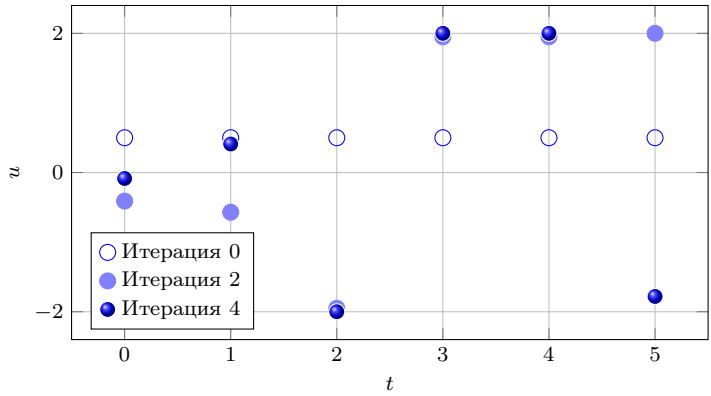
В работе предложен метод минимаксного улучшения второго порядка для неоднородных дискретных систем с промежуточными критериями и

Таблица 2. Результаты расчетов примера 2

Итерация	$\Lambda^1 = \Lambda^3 = \Lambda^5$	$I = x^d(6) \to \min$
0	-	0.44
1	0.5	-0.73
2	0.2	-23.84
3	0.7	-32.29
4	0.171	-32.33



(*а*) Состояние



(*б*) Управление

Рисунок 2. Результаты расчетов примера 2

сформулирован его алгоритм. Также приведена и доказана теорема об улучшаемости начального приближения.

Алгоритм апробирован на двух иллюстративных примерах. В первом на переменные управления не заданы ограничения, тогда как во втором примере они присутствуют. Решения достигаются за две, три итерации. Расчеты показывают, что присутствующие в сопряженной системе знакоопределенные матрицы являются дополнительным средством для уменьшения значения функционала. Методика их выбора пока отсутствует, использовался перебор числовых значений в заданном диапазоне. Проведенные расчеты подтверждают работоспособность метода.

Список литературы

[1] *Теория систем с переменной структурой*/ ред. С. В. Емельянов.– М.: Наука.– 1970.– 592 с. ↑48

[2] Гурман В. И. *К теории оптимальных дискретных процессов* // Автоматика и телемеханика.– 1973.– № 7.– С. 53–58. ↑48

[3] Васильев С. Н. *Теория и применение логико-управляемых систем* // *Труды 2-ой Международной конференции «Идентификация систем и задачи управления»*, SICPRO'03 (Москва, 29–31 января 2003), М.: Институт проблем управления им. В.А. Трапезникова РАН.– 2003.– С. 23–52. ↑48

[4] Бортаковский А. С. *Достаточные условия оптимальности управления детерминированными логико-динамическими системами*, Информатика. Сер. Автоматизация проектирования.– №2–3, М.: ВИМИ.– 1992.– С. 72–79. ↑48

[5] Миллер Б. М., Рубинович Е. Я. *Оптимизация динамических систем с импульсными управлениями*.– М.: Наука.– 2005.– ISBN 978-5-9710-5725-3.– 429 с. ↑48

[6] Lygeros J. *Lecture Notes on Hybrid Systems*.– Cambridge: University of Cambridge.– 2003.– 70 pp. URL ↑48

[7] Бурков В. Н. *Основы математической теории активных систем*.– М.: Наука.– 1977.– 255 с. ↑48

[8] Бурков В. Н., Новиков Д. А. *Теория активных систем (история развития и современное состояние)* // Пробл. управл.– 2009.– № 3.1.– С. 29–35. ↑48

[9] Кротов В. Ф., Гурман В. И. *Методы и задачи оптимального управления*.– М.: Наука.– 1973.– 448 с. ↑48

[10] Кротов В. Ф. *Достаточные условия оптимальности для дискретных управляемых систем* // ДАН СССР.– 1967.– Т. **172**.– № 1.– С. 18–21. ↑48

[11] Расина И. В., Гусева И. С. *Метод улучшения управления для неоднородных дискретных систем с промежуточными критериями* // Программные системы: теория и приложения.– 2018.– Т. **9**.– № 2.– С. 23–38. URL ↑48, 50

[12] Гурман В. И., Расина И. В. *О практических приложениях достаточных условий сильного относительного минимума* // Автоматика и телемеханика.– 1979.– № 10.– С. 12–18. ↑48

[13] Rasina I., Danilenko O. *Second-order improvement method for discrete-continuous systems with intermediate criteria*, 17th IFAC Workshop on Control Applications of Optimization (October 15–19, 2018, Yekaterinburg, Russia), IFAC-Papers Online.– vol. **51**.– No. 32.– 2018.– Pp. 184–188. ↑49, 51

[14] Расина И. В., Фесько О. В. *Достаточные условия относительного минимума для дискретно-непрерывных систем* // Программные системы: теория и приложения.– 2020.– Т. **11**.– № 2.– С. 47–59. ↑49, 53, 54

[15] Гурман В. И.,Трушкова Е. А. *Приближенные методы оптимизации управляемых процессов* // Программные системы: теория и приложения.– 2010.– Т. **4**.– № 4.– С. 85–104. ↑52

Поступила в редакцию 31.07.2023;
одобрена после рецензирования 11.10.2023;
принята к публикации 12.10.2023;
опубликована онлайн 28.10.2023.

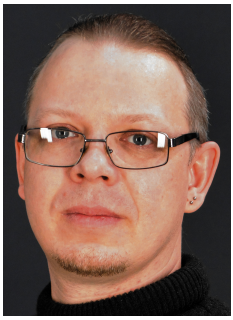
Рекомендовал к публикации *д.т.н. А. М. Цирлин*

Информация об авторах:

Ирина Викторовна Расина

д.ф.-м.н., г.н.с. Исследовательского центра системного анализа Института программных систем им. А. К. Айламазяна РАН и Федерального исследовательского центра «Информатика и управление» РАН, Москва, Российская Федерация, специалист в области моделирования и управления гибридными системами, автор и соавтор более 130 статей и 5 монографий

0000-0001-8939-2968
e-mail: irinarasina@gmail.com



Александр Олегович Блинов

к.т.н., доцент кафедры информационных технологий, искусственного интеллекта и общественно-социальных технологий цифрового общества факультета политических и социальных технологий ФГБОУ ВО «Российский государственный социальный университет», Москва, Российская Федерация

0000-0002-5713-2325
e-mail: aleblinov@yandex.ru

Все авторы сделали эквивалентный вклад в подготовку публикации. Авторы заявляют об отсутствии конфликта интересов.

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article OPTIMIZATION METHODS AND CONTROL THEORY

UDC 517.977:004.421.2
10.25209/2079-3316-2023-14-4-47-66

Minimax improvement method for inhomogeneous discrete systems

Irina Viktorovna **Rasina**[1], Alexander Olegovich **Blinov**[2]

[1] Ailamazyan Program Systems Institute of RAS, Ves'kovo, Russia
[1] Federal Research Center "Computer Science and Control" of RAS, Moscow, Russia
[2] Russian State Social University, Moscow, Russia
[1]✉*irinarasina@gmail.com*

Abstract. A class of two-level discrete inhomogeneous systems (DNS) is considered for the case when all homogeneous subsystems of the lower level are not only connected by a common functionality, but also have their own goals. Similar systems are widely used in practice (economics, ecology), and also arise in the process of numerically solving optimization problems when discretizing continuous control systems.

A second-order control improvement method is proposed, for the derivation of which a generalization of sufficient optimality conditions by V. F. Krotov. Illustrative examples are given. (*In Russian*).

Key words and phrases: heterogeneous discrete systems, intermediate criteria, sufficient optimality conditions, control improvement method

2020 *Mathematics Subject Classification*: 49M99; 49N10

Acknowledgments:

[1]The work was supported by the Russian Science Foundation (project no. 21-11-00202)

For citation: Irina V. Rasina, Alexander O. Blinov. *Minimax improvement method for inhomogeneous discrete systems*. Program Systems: Theory and Applications, 2023, **14**:4(59), pp. 47–66. (*In Russ.*). `https://psta.psiras.ru/read/psta2023_4_47-66.pdf`

References

[1] *Theory of Systems with Variable Structures*, ed. Emel'yanov S. V., Nauka, M., 1970 (in Russian), 592 pp.

[2] V. I. Gurman. "Theory of optimum discrete processes", *Autom. Remote Control*, **34**:7 (1973), pp. 1082–1087 (in Russian).

[3] S. N. Vassilyev. "Theory and application of logic-based controlled systems", *Proceedings of the International Conference Identification and Control Problems*, SICPRO'03 (Moskva, 29–31 yanvarya 2003, Institut problem upravleniya im. V.A. Trapeznikova RAN), Institute of control sciences, M., 2003, pp. 23–52 (in Russian).

[4] A. S. Bortakovskij. "Sufficient optimality conditions for control of deterministic logical-dynamic systems", Informatika, Ser. Computer Aided Design, 2–3, VIMI, M., 1992, pp. 72–79 (in Russian).

[5] B. M. Miller, E. Ya. Rubinovich. *Optimization of the Dynamic Systems with Pulse Controls*, Nauka, M., 2005, ISBN 978-5-9710-5725-3 (in Russian), 429 pp.

[6] J. Lygeros. *Lecture Notes on Hybrid Systems*, University of Cambridge, Cambridge, 2003, 70 pp.

[7] V. N. Burkov. *Fundamentals of the mathematical theory of active systems*, Nauka, M., 1977 (in Russian), 255 pp.

[8] V. N. Burkov, D. A. Novikov. "Active systems theory (history of development)", *Probl. upravl.*, 2009, no. 3.1, pp. 29–35 (in Russian).

[9] V. F. Krotov, V. I. Gurman. *Methods and Problems of Optimal Control*, Nauka, M., 1973 (in Russian), 448 pp.

[10] V. F. Krotov. "Sufficient conditions for the optimality of discrete control systems", *DAN SSSR*, **172**:1 (1967), pp. 18–21 (in Russian).

[11] I. V. Rasina, I. S. Guseva. "Control improvement method for non-homogeneous discretesystems with intermediate criterions", *Program Systems: Theory and Applications*, **9**:2 (2018), pp. 23–38 (in Russian).

[12] V. I. Gurman, I. V. Rasina. "On practical applications of conditions sufficient for a strong relative minimum", *Autom. Remote Control*, **40**:10 (1980), pp. 1410–1415.

[13] I. Rasina, O. Danilenko. "Second-order improvement method for discrete-continuous systems with intermediate criteria", 17th IFAC Workshop on Control Applications of Optimization (October 15–19, 2018, Yekaterinburg, Russia), IFAC-Papers Online, vol. **51**, 32, 2018, pp. 184–188.

[14] I. V. Rasina, O. V. Fesko. "Sufficient relative minimum conditions for discrete-continuous control systems", *Program Systems: Theory and Applications*, **11**:2 (2020), pp. 61–73.

[15] V. I. Gurman, E. A. Trushkova. "Approximate methods of control processes optimization", *Program Systems: Theory and Applications*, **4**:4 (2010), pp. 85–104 (in Russian).

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ПРОГРАММНОЕ И АППАРАТНОЕ ОБЕСПЕЧЕНИЕ РАСПРЕДЕЛЕННЫХ И СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМ

УДК 004.312.4
10.25209/2079-3316-2023-14-4-67-89

Метод построения циклических конвейеров

Игорь Алексеевич **Адамович**[1]✉, Юрий Андреевич **Климов**[2]

[1] Институт программных систем им. А. К. Айламазяна РАН, Веськово, Россия
[2] Институт прикладной математики им. М. В. Келдыша РАН, Москва, Россия
[1]✉*i.a.adamovich@gmail.com*

Аннотация. Одним из наиболее эффективных способов организации вычислений на ASIC или FPGA является построение неостанавливаемых конвейеров. Однако для некоторых вычислительных схем получаемый конвейер может оказаться слишком большим для имеющихся ресурсов ASIC или FPGA. Авторами предлагается метод построения циклических конвейеров, управление потоками данных в которых основано на счетчиках и не зависит от данных, передаваемых по конвейеру. Предложенный метод позволяет строить более компактные неостанавливаемые конвейеры со скважностью, равной количеству проходов по циклу, которые должны пройти данные, чтобы конвейер преобразовал их в искомый результат.

Ключевые слова и фразы: конвейер, ПЛИС, микросхема, скважность, очередь, кредит

Для цитирования: Адамович И. А., Климов Ю. А. *Метод построения циклических конвейеров* // Программные системы: теория и приложения. 2023. **Т. 14**. № 4(59). С. 67–89. https://psta.psiras.ru/read/psta2023_4_67-89.pdf

Введение

При разработке эффективных программно-аппаратных комплексов встречаются ситуации, в которых процессор общего назначения не способен обеспечить требуемую производительность при заданных рамках потребляемой мощности. Частым решением в таких условиях является перенос вычислительно-емкого ядра программы на одну или несколько ASIC- или FPGA-микросхем.

ASIC [1] (Application-Specific Integrated Circuit, интегральная схема специального назначения) — это интегральная схема, которая спроектирована для решения какой-то одной отдельной задачи и, в отличие от процессора общего назначения, не предназначена для решения любых задач. Благодаря специализации ASIC имеет оптимизированные на схемотехническом уровне соединения элементов, а также их расположение, что во многих случаях позволяет значительно увеличить производительность и эффективность комплекса. Возможность адаптировать ASIC на таком глубоком уровне к решению конкретной проблемы является основой для тонкой настройки и оптимизации всей системы. Кроме того, при равной вычислительной мощности ASIC зачастую потребляет значительно меньше энергии по сравнению с процессором общего назначения, когда измерение производится на одной и той же задаче. Следует отметить, что разработка и выпуск ASIC занимает много времени и обходится существенно дороже готовых процессоров общего назначения, поэтому круг применения ASIC достаточно узок.

Промежуточным вариантом между процессором общего назначения и ASIC является FPGA. FPGA [2] (Field Programmable Gate Array, программируемая пользователем вентильная матрица) — это программируемая микросхема, в которую загружается не программа для процессора общего назначения, а описание схемы. Она не такая быстрая как ASIC: работает на меньшей частоте, потребляет больше энергии, но обладает схожими с ASIC возможностями для оптимизации схемы соединений и расположения элементов, что в некоторых задачах дает значительные преимущества перед процессорами общего назначения.

Основное преимущество FPGA перед ASIC — программируемость. Схему соединений и расположение элементов в любой момент можно изменить. Благодаря этому свойству FPGA часто используют как прототип для будущей специализированной микросхемы: если при разработке допущена ошибка, то в FPGA ее легко исправить, в отличие от ASIC.

Простой перенос алгоритма на ASIC или FPGA может не дать ожидаемого прироста в производительности (см. например раздел 2 в [3]). Одним из наиболее используемых и эффективных методов адаптации

алгоритма к архитектуре ASIC/FPGA является метод конвейеризации [2]. Суть этого метода заключается в разбиении алгоритма на несколько последовательных этапов, позволяя системе обрабатывать несколько задач одновременно (по задаче на каждый этап), как на производственной линии.

При разработке конвейерной реализации алгоритма необходимо, с одной стороны, соблюсти баланс между необходимой производительностью и занимаемой конвейером площадью микросхемы и, с другой стороны, обеспечить поток данных через конвейер, когда блоки, выдающие данные в конвейер, и блоки, потребляющие данные из конвейера, могут по различным причинам быть не готовыми к выдаче или потреблению данных.

Как общие традиционные способы построения конвейеров [4], так и частные, например, применяемые для построения процессорных микроархитектур [5], часто могут оказываться непригодны из-за больших длин конвейеров.

В исследованиях мы столкнулись с необходимостью сконструировать алгоритм, потенциально реализуемый на нескольких конвейерах, но ресурсы FPGA не позволяли построить полную конвейерную схему. При этом в FPGA не было достаточно ресурсов для реализации традиционной промежуточной буферизации данных, и схема такой реализации оказалась слишком сложна. Поэтому мы поставили перед собой цель разработать новое решение обозначенной выше проблемы.

В настоящей статье предлагается новый, разработанный авторами способ построения конвейеров, который позволяет в некоторых случаях упростить разработку и существенно повысить тактовую частоту схемы по сравнению с традиционными вариантами. Новый способ эффективен при невозможности разместить полный конвейер в микросхеме. В этом случае мы предлагаем организовать в конвейере цикл и предлагаем локальный способ разрешения конфликтов в месте объединения потоков данных. В предлагаемом циклическом конвейере отсутствует возможность остановки (stall) конвейера, что и упрощает разработку, и повышает тактовую частоту разработанной схемы. Конвейеры без возможности остановки мы будем называть *неостанавливаемыми*.

Следует сказать, что широко используется [6, 7] способ построения систем с помощью кредитного механизма, который используется и в нашей работе. Схема [8] основывается на неуправляемом подходе к построению конвейеров и ее можно включить в нашу систему конвейеров в качестве вспомогательной. В [9] используются циклические конвейеры, но задействуют классические способы управления и остановки конвейеров.

В работах [3] и [10] рассматриваются классы задач, реализуемых на FPGA и ASIC, а также способы решения таких задач. Работы [11] и [12] близки нашему исследованию и посвящены реализации алгоритма вычисления хеш-функции SHA-3. Более подробный обзор упомянутых работ и их сравнение с нашим исследованием будут проведены в четвертой главе «Близкие работы».

Дальнейшее изложение будет построено следующем образом. В первой главе проведен обзор способов построения конвейера и методов управления конвейерами. Во второй главе будет описан предлагаемый подход к построению неостанавливаемых циклических конвейеров. В третьей главе будет представлена практическая ценность подхода на основе неостанавливаемых циклических конвейеров. Четвертая глава посвящена обзору работ близких к нашей. И наконец заключение завершает статью.

1. Построение конвейеров

1.1. Пример программы

Для сравнения различных подходов к архитектуре конвейера рассмотрим простую задачу, программа которой на псевдокоде изображена на листинге 1.

```
для всех a ∈ A:
    b := a
    выполнить N раз:
        b := f(b)
    положить b в B
```

Листинг 1. Простой алгоритм-пример

Программа представляет собой двойной цикл, который обрабатывает элементы из множества A. Для каждого элемента a из множества A выполняются следующие действия:

(1) Переменная b инициализируется значением a.

(2) Вложенный цикл выполняется N раз, на каждой итерации вычисляется функция $f(b)$ и результат помещается снова в переменную b.

(3) После завершения вложенного цикла финальное значение из переменной b помещается в множество B.

В результате программа применяет N раз функцию f к каждому элементу множества A, а результаты сохраняет во множестве B.

Алгоритмы близкого вида встречаются в различных областях: численная математика, криптографические вычисления и другие.

Обычно реализация такого алгоритма на процессорах общего назначения предполагает, что множества A и B хранятся во внешней по отношению к процессору памяти (например в ОЗУ).

1.2. Реализации в ASIC/FPGA

1.2.1. *«Наивная» реализация*

Выполним перенос описанной выше программы на FPGA или ASIC. Сначала используем «наивный» подход, который заключается в том, что структура схемы будет максимально похожа на структуру программы. На верхнем архитектурном уровне «наивный» вариант схемы можно представить так:

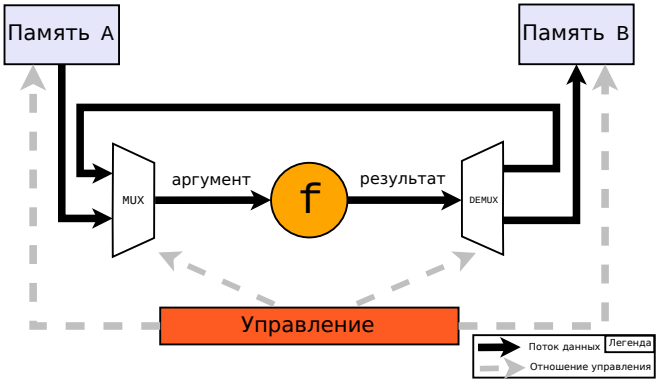


Рисунок 1. Архитектура «наивной» схемы

На рисунке 1 толстыми линиями показаны шины данных. Пунктирными серыми линиями условно показано управление блоками, реальные сигналы управления для простоты опущены.

Отметим, что шины данных содержат сигналы, для передачи как самих данных, так и дополнительного управляющего сигнала `valid`. Это обычная практика, позволяющая отличить валидные данные от невалидных без понимания природы самих данных. Данный сигнал широко используется в интерфейсах различных блоков и протоколах (например, AXI [13] или Avalon [14]). В данном кратком описании используемых подходов мы не будем заострять внимание на использовании сигнала `valid`, так как оно естественно и широко известно.

Данные из памяти `A`, в которой хранятся элементы множества A, подаются на вход блока `f`, который реализует функцию f. Данные с выхода блока `f` снова подаются ему на вход, чтобы выполнить N вызовов функции f. Финальный результат с выхода блока `f` записывается в память `B`.

Чтобы решить конфликты, возникшие на входе и выходе блока `f`, используются мультиплексор `MUX` и демультиплексор `DEMUX`. По сигналу с блока управления мультиплексор `MUX` передает в блок `f` либо данные из памяти `A`, либо выход блока `f`. А демультиплексор `DEMUX` по сигналу от блока управления направляет результат вычисления блока `f` либо снова на вход `f` (через мультиплексор `MUX`), либо в память `B`.

В рамках данной статьи будем считать, что реализация блока `f` вычисляет результат за 1 такт и только на основе своих входов (без зависимости от состояния блока `f` на прошлом такте). Возможны различные варианты реализации блока `f`, которые не будут рассматриваться в данной статье. Мы акцентируем внимание на конвейерных реализациях внутреннего цикла.

Такт — это логическая единица времени, реальное значение которой определяется частотой тактового сигнала схемы. Чем выше тактовая частота — тем меньше времени занимает 1 такт и тем быстрее (в реальных единицах времени) схема выдает результат. Однако это значит, что меньше вычислений можно выполнить за 1 такт, именно поэтому так важно сократить накладные расходы в рамках такта.

Отметим, что в качестве блоков памяти `A` и `B` можно рассматривать не только внутреннюю блочную память ASIC или FPGA, но и любые блоки, выдающие исходные данные и потребляющие результаты: внешнюю память, каналы связи и другие.

Представленная на рисунке 1 «наивная» схема обладает существенным недостатком: она обычно работает медленней процессора общего назначения. Основная причина медленной работы «наивной» схемы заключается в том, что обработка элементов множества A выполняется последовательно: одновременно схемой может обрабатываться только один элемент множества A. Частота процессора общего назначения существенно выше частоты FPGA, поэтому без параллельной обработки данных затруднительно спроектировать схему, которая будет работать быстрее процессора.

1.2.2. *Конвейерная реализация*

Для повышения производительности схемы рассмотрим другой вариант реализации программы, обеспечивающий высокое быстродействие. В отличие от процессора общего назначения, в котором параллельная обработка реализуется либо за счет независимой работы нескольких ядер, либо за счет использования векторных инструкций, в ASIC или FPGA основным методом распараллеливания является конвейер. Это, однако, не исключает возможности реализации независимых блоков для параллельной обработки, подобных ядрам процессора.

На рисунке 2 изображена схема, которую принято называть *конвейерной*. В ней отсутствует цикл (а также мультиплексор и демультиплексор), в котором данные с выхода блока `f` передаются на вход этому же блоку. Вместо циклической структуры, выбран другой вариант — данные с одного

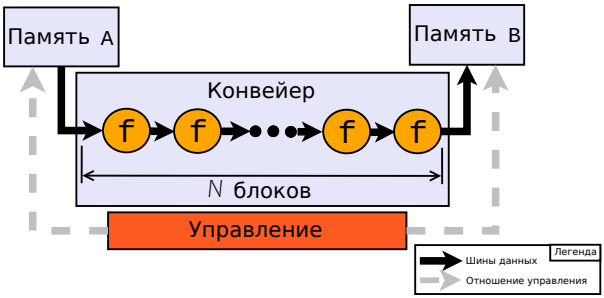


Рисунок 2. Архитектура конвейерной схемы

блока `f` передаются на вход другому идентичному блоку `f`. Каждый отдельный блок `f` принято называть *стадией* конвейера.

Конвейерная структура позволяет обрабатывать несколько вычислений внутреннего цикла параллельно. Каждый элемент множества A из памяти `A` в порядке очереди подается на вход конвейера. Когда первый элемент a_1 входит в конвейер, он попадает на первую стадию. После этого, в следующем такте результат вычислений первой стадии конвейера $f(a_1)$ будет передан на вторую стадию, а на первую стадию поступит следующий элемент a_2. Затем, в следующем такте результат второй стадии конвейера $f(f(a_1))$ будет передан на третью стадию, а на первую стадию будет подан следующий элемент a_3 и так далее. В результате на выходе с последней стадии конвейера для каждого элемента a будет получено $f^N(a)$, что и требуется вычислить.

Отметим, что после заполнения конвейера, при его полной загрузке, обрабатываются N элементов одновременно: по одному элементу на каждой стадии.

Конвейерный вариант устройства схемы имеет ряд преимуществ над «наивным»:

- Если число элементов в множестве A гораздо больше длины конвейера N, то конвейерный вариант в N раз производительней «наивного» за счет возможности параллельной обработки элементов из A.
- Отсутствуют блоки `MUX` и `DEMUX`, что уменьшает вычисления в рамках одного такта и повышает тактовую частоту схемы.
- Возможна взаимная оптимизация соседних блоков `f`, так как они связаны напрямую.

Нужно отметить, что рассматриваемая конвейерная архитектура в некоторых случаях обладает существенным недостатком: если число N велико, то размеры конвейера получаются слишком большими и конвейер может не поместиться в заданной площади микросхемы ASIC или FPGA. Или иногда бывает не нужна такая высокая производительность микросхемы, а важнее использовать меньше ресурсов микросхемы. В таких случаях обычно строят меньший конвейер, прогоняя данные через него в несколько заходов, сохраняя промежуточные результаты в памяти.

Применительно к нашей задаче размер конвейера получался больше, чем вся имеющаяся FPGA, поэтому необходимо было сократить конвейер. При этом и использование промежуточной памяти было невозможно.

1.3. Управление конвейером

При использовании конвейеров разработчику необходимо решить проблему, что делать, если память `B` не может принять данные из конвейера. Рассмотрим наиболее часто используемые подходы.

1.3.1. *Управление сигналом `enable`*

Одним из очевидных способов является использование сигнала `enable`, включающего или выключающего весь конвейер целиком (см. рисунок 3).

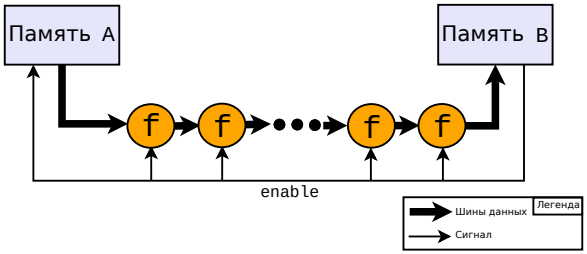


Рисунок 3. Управление сигналом enable

Несмотря на его простоту, у этого метода есть несколько недостатков:

- Длинный сигнал `enable` и зависимость всех блоков от него может значительно снижать тактовую частоту такой схемы.
- Если память `B` не готова принимать данные, то останавливается весь конвейер, хотя какие-то стадии конвейера могли бы вычислять данные, если следующие за ними стадии пусты.

1.3.2. *Управление сигналами `ready`*

Другим часто используемым способом [4] является использование между всеми стадиями сигнала `ready`, направленного против потока данных, для индикации, что следующая стадия или память `B` готовы принимать данные (см. рисунок 4).

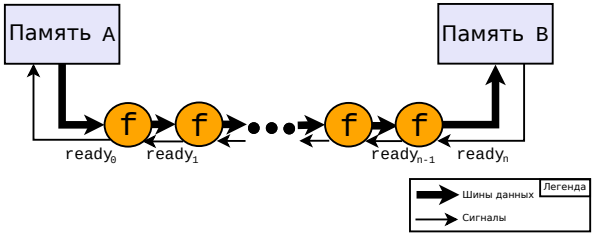


Рисунок 4. Управление сигналами ready

Такого рода сигналы широко используются в различных интерфейсах, например AXI [13], Avalon [14] и другие.

В зависимости от способа [4] реализации сигнала `ready` схема может иметь разные недостатки:

- Если сигнал $\texttt{ready}_i$ комбинационно зависит от сигнала $\texttt{ready}_{i+1}$, то фактически получается длинный сигнал `enable`, который может снижать тактовую частоту схемы.
- Если реализовывать промежуточную буферизацию на каждой стадии, чтобы разорвать комбинационную связь между соседними сигналами `ready`, то потребуются дополнительные ресурсы для реализации такой буферизации.

Тем не менее данный способ широко используется, поскольку позволяет без усилий соединять стадии конвейера и другие блоки, написанные разными разработчиками.

1.3.3. *Управление на основе кредитов*

Альтернативным вариантом, применяемым в промышленности, является кредитный механизм или просто кредиты [6, с. 57; 7]. Кредиты широко используются в различных сетевых протоколах (например, TCP/IP или PCIe), и могут быть использованы для управление конвейером.

На рисунке 5 показана схема конвейера, с управлением на основе кредитного счетчика. По сравнению с предыдущими вариантами, все

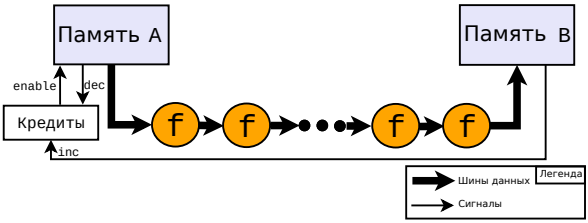


Рисунок 5. Управление кредитами

управление вынесено в отдельный блок, а конвейер реализуется без возможности остановки (неостанавливаемый конвейер). Наличие кредита у памяти `A` гарантирует, что отправленные в конвейер данные будут записаны в память `B`, не случится переполнение памяти. Если кредита на отправку данных нет, то память `A` не должна посылать новые данные в конвейер.

В результате разделяются задачи построения эффективного конвейера и недопущения переполнения памяти `B`. Отметим, что если в конкретной схеме в качестве получателя результатов конвейера используется не память, а какой-то блок с непредсказуемым заранее временем работы, то в таких случаях после конвейера устанавливают очередь (FIFO), наличие места в которой определяет значение кредитного счетчика, и такая очередь играет роль буфера между блоками, готовыми отправлять и получать данные на разных тактах, но работающих с одинаковой средней скоростью.

2. Циклический конвейер

2.1. Общая архитектура

В данном разделе будет описана предлагаемая нами схема реализации конвейера. Для того, чтобы уменьшить используемые конвейером ресурсы, надо зациклить конвейер (см. рисунок 6).

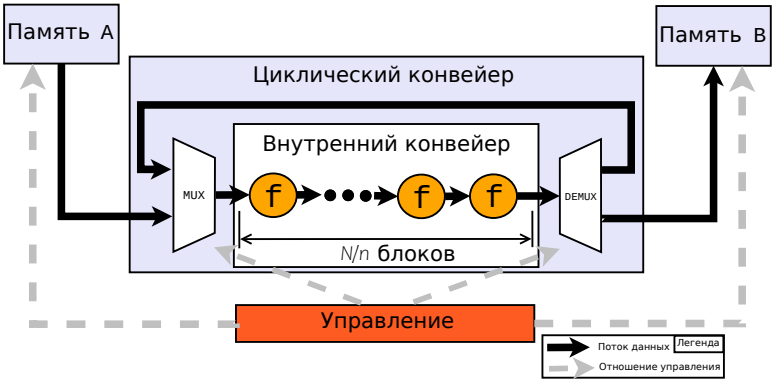


Рисунок 6. Архитектура циклического конвейера

Суть схемы циклического конвейера заключается в том, что длинный конвейер укорачивается в n раз, то есть длина внутреннего линейного конвейера становится N/n. При этом в схеме появляется цикличность: после

того как элемент данных обработан последней из N/n стадий внутреннего конвейера он снова подается на вход первой стадии внутреннего конвейера. Такой цикл выполняется n раз, в результате каждый элемент проходит $N/n \cdot n = N$ стадий внутреннего конвейера и представляет из себя искомый результат вычислений.

Циклический вариант обрабатывает элементы параллельно: в нем одновременно может обрабатываться N/n элементов множества A. И он занимает ресурсов в n раз меньше конвейерной схемы, поэтому подобрав n, его можно вместить в требуемые размеры ASIC или FPGA.

Циклический конвейер является комбинацией двух описанных выше — «наивной» и конвейерной. Как и у «наивной» реализации в нем имеются два места возможных конфликтов: блоки мультиплексор и демультиплексор, а также запись из конвейера в память `B`, которая может быть заполнена.

Эти проблемы можно попытаться решить, управляя стадиями конвейера одним из описанных ранее способов или их комбинацией, однако это оказалось проблематично. Во-первых, это управление достаточно трудоемко реализовать. И во-вторых, сигналы остановки конвейера значительно снижают рабочую тактовую частоту.

Другим важным требованием было то, что нам надо было реализовать несколько связанных между собой конвейеров: результат одного должен был передаваться на вход другому. Поэтому мы не могли останавливать поток данных. Мы хотели сделать решение, которое позволяло бы комбинировать такие конвейеры так же, как и обычные конвейеры без циклов.

Поэтому мы решили реализовать неостанавливаемые циклические конвейеры с использованием кредитного счетчика, который разрешает конфликты при доступе в память `B`. Теперь необходимо разработать решение по управлению мультиплексором `MUX` и демультиплексором `DEMUX`.

2.2. Управление мультиплексором и демультиплексором

Мы предлагаем решение, в котором мультиплексор на входе циклического конвейера управляется без необходимости понимать, какие данные сейчас обрабатывает конвейер, и других нюансов. Это существенно повышает максимально возможную тактовую частоты схемы, а значит и производительность в целом.

Снова рассмотрим циклический неостанавливаемый конвейер (рисунок 7). Мы хотим, чтобы данные, пришедшие из памяти `A` прошли n

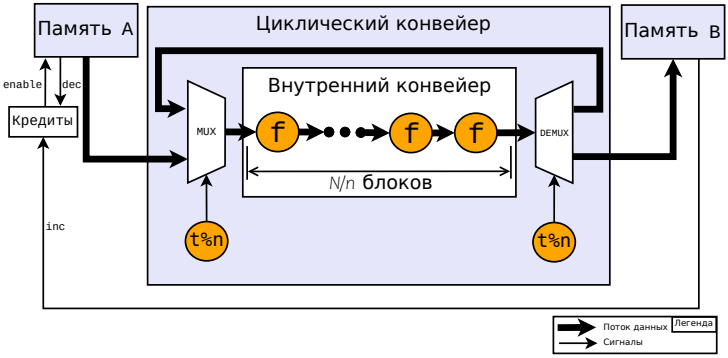


Рисунок 7. Циклический конвейер со скважностью n

кругов по конвейеру из N/n стадий. Обозначим $L := N/n$ количество стадий во внутреннем линейном конвейере. Для простоты будем считать, что каждая стадия выполняется 1 такт. И предположим, что L будет взаимопростым с n.

Отметим, что данные на шине данных снабжены дополнительным сигналом `valid`. Для полного описания предлагаемой схемы нам потребуется явно зафиксировать правила работы с этим сигналом, которые были опущены ранее для простоты изложения.

Пусть блоки конвейера работают по следующим правилам:

- Данные из памяти `A` разрешается подавать на вход циклического конвейера только в такты, номер которых делится на n. Если память `A` подает данные, то она обязана выставить сигнал `valid` в 1, иначе — в 0.
- Мультиплексор `MUX` в такты, номер которых делится на n, пропускает данные от памяти `A` на вход внутреннего конвейера. В остальные такты он пропускает данные с выхода демультиплексора `DEMUX`.
- Демультиплексор `DEMUX` в такты, номер которых делится на n, пропускает данные от внутреннего конвейера в память `B`. В остальные такты он пропускает данные на вход мультиплексора `MUX`.
- Данные записываются в память `B` только на тактах, номера которых делятся на n, и сигнал `valid` равен 1.

Отметим, что мультиплексор `MUX` и демультиплексор `DEMUX` работают синхронно: либо они оба направляют данные по циклу, либо мультиплексор направляет новое значение из памяти `A` на вход конвейера, а демультиплексор направляет вычисленное значение в память `B`.

Докажем следующую теорему:

Теорема. *Пусть длина L внутреннего конвейера взаимно проста с количеством кругов n и все блоки работают согласно описанному выше алгоритму. Тогда в память* `B` *будут записаны результаты вычислений $f^N(a)$ для всех a из памяти* `A`*.*

Доказательство. Данные a_i из памяти `A` передаются на вход циклическому конвейеру в такты, чей номер делится на n, т.е. в такты вида $i \cdot n$ $(i \geqslant 0)$.

Данные перемещаются с входа внутреннего линейного конвейера до его выхода за L тактов, т.е. они окажутся на выходе в такты вида $i \cdot n + j \cdot L$ $(j > 0)$.

Если $i \cdot n + j \cdot L$ не будет делиться на n, то данные будут поданы на вход внутреннего линейного конвейера. А если будет делиться, то записаны в память `B`.

Так как L и n взаимопростые, то $i \cdot n + j \cdot L$ будет делиться на n только тогда, когда j будет делиться на n, то есть первый раз это произойдет при $j = n$. Значит данное сделает n проходов по внутреннему линейному циклу длины L и на такте $i \cdot n + n \cdot L$ будет записано в память `B`. Что и требовалось доказать.

□

Рассмотрим построенный циклический конвейер. В отличие от обычного конвейера, который готов принимать данные каждый такт, наш циклический конвейер готов принимать данные только каждые n тактов, n называется *скважностью* (интервалом инициализации или периодичностью подачи данных) конвейера [3].

Реализовать скважность в схеме достаточно просто: поставим счетчик от 0 до $n - 1$, который увеличивается на 1 каждый такт и вместо достижении значения n сбрасывается на 0. Такой счетчик не зависит от каких-либо внешних сигналов, кроме тактового и сигнала сброса всей схемы, поэтому на реализацию скважности почти не тратится ресурсов, как и площади кристалла. При необходимости счетчики скважности можно дублировать, что приводит к меньшим задержкам, а значит повышает тактовую частоту, на которой может работать схема.

Итак, если дополнить предлагаемое решение кредитным счетчиком и FIFO на выходе, то важным преимуществом является отсутствие механизма остановки конвейера: новые данные всегда можно подавать раз в n тактов независимо от состояния системы. И результат всегда будет на выходе раз в n тактов.

Это, с одной стороны, уменьшает сложность конвейера: в результате упрощается реализация стадий конвейера, а отсутствие сигналов управления или остановки приводит к более высокой частоте схемы. С другой стороны, такие конвейеры требуют внешнего механизма защиты от переполнения памяти B или другого потребителя данных. Впрочем, это аналогично обычным неостанавливаемым конвейерам.

3. Практическая ценность

Неуправляемые конвейеры возникают в реальных задачах. Описанный в настоящей статье подход достаточно легко применить, если дан конвейер длины N без циклов, который на каждой отдельной стадии совершает одинаковые преобразования над потоком данных. Такой конвейер можно уменьшить приблизительно в n раз зациклив.

Единственным условием зацикливания является взаимная простота результирующей длины конвейера $L = N/n + k$ и количества итераций n. Число k означает количество пустых стадий — при необходимости, конвейер, полученный в результате зацикливания, можно дополнить несколькими пустыми стадиями для соблюдения условия взаимной простоты n и L.

При зацикливании конвейера количество стадий уменьшается в n раз (в идеальном случае) и, как результат, его производительность, определяемая количеством обработанных данных в единицу времени, также уменьшается в n раз. При этом одновременно с уменьшением производительности площадь на кристалле, занимаемая конвейером, также уменьшается (приблизительно в n раз).

Требуется отметить, что из-за потенциального наличия пустых стадий, конвейер может уменьшиться не ровно в n раз, а чуть меньше. Однако, если длины конвейеров достаточно большие, число пустых стадий практически не влияет на размеры.

Таким образом, предлагаемый метод зацикливания разумно использовать, когда уменьшение производительности допустимо и существует необходимость сэкономить занимаемое конвейером место в микросхеме.

Дополнительно отметим, что такого рода неостанавливаемые циклические конвейеры со скважностью больше 1 могут комбинироваться ровно так же, как и привычные конвейеры со скважностью, равной 1. Единственное, что требуется, – следить за согласованием конвейеров как по скважности, так и по фазе, и при необходимости вставлять дополнительные стадии для согласования.

В качестве примера класса задач, при решении которых возникают циклические конвейеры, можно привести реализацию различных численных, криптографических и других алгоритмов.

Для подтверждения практической ценности было проведено сравнение нового подхода управления циклическими конвейерами и традиционного на основе явного контроля потока данных с возможностью остановки различных частей конвейера. Для сравнения была выбрана задача вычисления криптографического хэша. Чтобы схема поместилась в доступную нам FPGA нам пришлось реализовать несколько циклов. Но в результате при полной нагрузке все стадии конвейера работают, что обеспечивает максимальную производительность при использовании данного количества ресурсов.

На рисунке 8 изображена схема, выбранная для эксперимента. Центральную часть занимает основной конвейер, который является цикли-

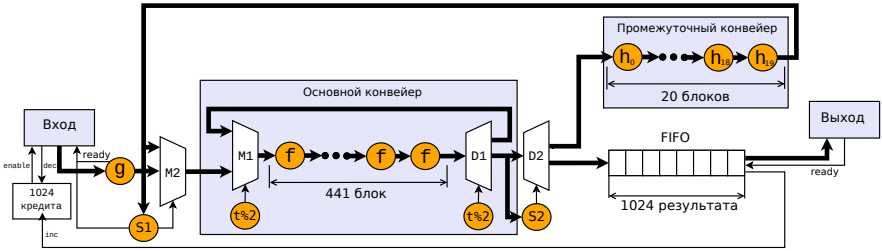


Рисунок 8. Архитектура схемы инициализации хеша

ческим. В нем 441 стадия, и он сжат в 2 раза по отношению к потенциально полному конвейеру. Таким сжатием мы смогли уложиться в отведенные для данной схемы ресурсы, сохранив приемлемую производительность. Итак, сжатие n равно 2, также как и скважность.

Мультиплексор `M1` и демультиплексор `D1` входят в основной циклический конвейер и их управление не зависит от данных. Мультиплексор `M1` каждый четный такт подает во внутренний конвейер данные от внешнего мультиплексора `M2`, а каждый нечетный – данные с выхода основного конвейера, замыкая цикл. Дополняя `M1`, демультиплексор `D1` каждый нечетный такт подает данные на вход циклическому конвейеру, а каждый четный такт – далее по схеме для последующей обработки другими блоками.

Мультиплексор `M2` является приоритетным. Он управляется блоком `S1`. Блок `S1` смотрит на данные из промежуточного конвейера и если они валидны, то мультиплексор `M2` передает их на вход циклическому конвейеру, иначе передаются данные из блока `g`.

Демультиплексор `D2` управляется блоком `S2`. `S2` вынужденно заглядывает в поступающие в него данные и на основе этого решает, куда их передать – в промежуточный конвейер или в выходное FIFO. Это свойство алгоритма формирования хэша.

Данные, прошедшие два круга в циклическом конвейере, по решению демультиплексора D2 могут поступать в промежуточный конвейер. Последний состоит из двадцати различных стадий, после преобразования в которых данные передаются опять на вход циклическому конвейеру. При этом нет необходимости задерживать промежуточный конвейер, поскольку его длина подобрана так, что данные с выхода промежуточного конвейера поступают на вход основному конвейеру только в четные такты.

Также схема содержит FIFO и кредитный счетчик, которые защищают конвейер от переполнения.

Как показано в таблице 1, предлагаемый новый способ оказался более простым в реализации (требует меньше строк кода), требует меньше ресурсов и получаемая схема работает на большей частоте.

Таблица 1. Сравнение параметров циклических конвейеров

	Частота	Логич. ячейки	Регистры	Строки кода
Традиционный подход	195 МГц	28446	19499	1031
Неостанавливаемый конвейер	233 МГц	26559	14874	912
Отношение	0.837	1.07	1.31	1.13

Для проведения экспериментов использовалась FPGA фирмы Xilinx семейства xc7a200tfbg484-2.

В эксперименте сравнивались схемы:

(*1*) с рисунка 8;

(*2*) схема, в вычислительной части такая же, как на рисунке 8, но с традиционным управлением конвейерами с помощью enable.

4. Близкие работы

Сравним наше исследование с другими близкими работами.

В классическом труде [4] в главе 23 рассматривается конвейерная организация вычислений. Показано, что при достаточно большом объеме данных (существенно больше глубины конвейера) ускорение от использования конвейера равно его длине.

Также в [4] рассматривается проблема переполнения принимающей памяти и необходимость остановки конвейера. Рассматривается два подхода к построению управляемых конвейеров: без дополнительной буферизации и с дополнительной буферизацией. В первом случае от сигнала о готовности принимающей памяти ready комбинационно зависят все стадии конвейера. Во втором случае используются очереди на регистрах глубины два (также называемые skid-buffer), позволяющий разрывать

длинную комбинационную связь ready ценой дополнительных регистров на каждом этапе конвейера.

В отличие от нашей работы, в [4] рассматриваются традиционные, базовые варианты конвейеров. Такие конвейеры являются полными, поскольку в них отсутствуют циклы. Схемы, основанные на принципах из [4], могут получаться очень большими. Наш метод позволяет сжимать рассматриваемые полные конвейеры в циклические меньшего размера, позволяя вписаться в площадь микросхемы за счет допустимого понижения производительности. Также в [4] обсуждаются механизмы остановки конвейеров, а мы предлагаем делать их неостанавливаемыми (и излагаем способ), что повышает производительность и позволяет экономить ресурсы FPGA или ASIC в случае циклических систем. Суммируя, можно утверждать, что работа [4] является источником информации по традиционным методам создания конвейеров, с которыми мы сравниваем свои варианты архитектур.

Конвейеры активно используются в промышленности для организации вычислений как в FPGA, так и в заказных микросхемах ASIC. Указанные проблемы с управлением конвейера во многих случаях решаются другим путем: используется неуправляемый конвейер, но вокруг которого строится схема на основе кредитов, не допускающая переполнения выходной памяти [6, 7].

Предлагаемая нами архитектура требует защиты от переполнения, для который изложенный, например, в [6, с. 57; 7] кредитный механизм подходит очень хорошо.

Конвейеры активно используются при построении различных процессоров. В книге [5] в главе 7 рассматриваются различные микроархитектуры процессора RISC-V, в том числе и традиционная конвейерная микроахритектура из 5 шагов: выборка инструкции (Fetch), декодирование инструкции (Decode), выполнение инструкции (Execute), операции с памятью (Memory), запись результата (WriteBack).

Процессоры могут использовать результаты вычисления одной из прошлых команд для вычисления новых, тем самым имея некоторый аналог цикла. Это роднит процессорные конвейеры с нашими. Однако стадии в процессоре обычно уникальны и не повторяются, поэтому их не сжать в циклический конвейер нашим способом. Также процессор часто требует остановки конвейера в силу наличия многотактных стадий: это препятствует построению неостанавливаемой архитектуры, на которой основывается наша работа. Тем не менее процессорные микроархитектуры являются классическими примерами конвейеризации.

Для выполнения арифметических операций с плавающей точкой также используются конвейеры. Библиотека программного IP-блока[1] [8] и другие аналогичные реализуют неуправляемую конвейерную схему для вычислений умножения. При этом блоки имеют простой интерфейс ввода и вывода нужных данных.

Так же как и у нас, в умножителе [8] используется неостанавливаемый подход, но он не имеет циклов. Все это позволяет, например, использовать конвейеризованное умножение в качестве нескольких стадий в циклических конвейерах нашей архитектуры (или даже в качестве отдельного промежуточного конвейера).

Есть и более универсальные библиотеки с большим набором функций. Например, библиотека IP-блока [9] реализует блоки различных операций с числами с плавающей точкой. Используется AXI [13] протокол (с сигналом ready для остановки потока данных при неготовности) для ввода данных и получения результата. Важной особенностью этой библиотеки является наличие параметра «Cycles per Operation», который задает скважность. Например, при значении 1 получается полностью конвейерная схема, готовая на каждом такте принимать новые данные. При значении 2 получается схема, готовая принимать данные только каждые два такта, снижает пропускную способность блока, но и примерно в два раза снижает используемые ресурсы. Именно для того, чтобы приостанавливать входные данные, когда блок не готов принимать входные данные или получатель результата не готов принимать результат, используется AXI протокол.

Так же как и у нас, в библиотеке программного IP-блока [9] используется скважность и зацикливание. В итоге [9] является примером циклической схемы, которая использует управляемый конвейер, в то время как мы предлагаем более производительный вариант – неостанавливаемую циклическую архитектуру на основе принципа взаимной простоты чисел. В общем случае наш вариант позволяет добиться большей тактовой частоты и использует меньше ресурсов микросхемы.

Конвейеры широко применяются для организации вычислений. В работе [3] исследуется какие задачи (и каким образом) могут решаться на суперкомпьютере с FPGA. Анализируется конвейерная организация вычислений и рассматриваются классы задач, для которых такая организация вычислений возможна. В работе [10] рассматриваются языки и подходы к организации вычислений в виде конвейера и встающие на этом пути проблемы.

[1] Intellectual Property block – сложный функциональный блок, который лицензируется для использования другим компаниям

Работы [3] и [10] описывает общие принципы создания конвейерных ускорителей на FPGA, которые могут использовать и нашу циклическую архитектуру.

Циклические конвейерные структуры возникают при реализации криптографических алгоритмов. Так в работе [11] рассматривается архитектура системы, состоящей из нескольких ядер, выстроенных в конвейер. При этом каждое ядро имеет циклическую «наивную» реализацию, подобную описанной нами в разделе 1. Фактически ядро в [11] является циклическим конвейером длины 1. Наш метод построения конвейеров более общий, поскольку ограничивает длину циклического конвейера только взаимной простотой чисел.

В работе [12] рассматривается конвейерная реализация алгоритма вычисления хэша SHA-3. Архитектура системы в [12] основывается на циклическом конвейере. Однако возможное переполнение конвейера не рассматривается, считается, что система снаружи сама заботится о заполнении конвейера. Также работа [12] рассматривает не общий способ построения циклических конвейеров, а только для алгоритма SHA-3. Мы в своей работе исследовали более общий случай, не зависящий от конкретного алгоритма. Также мы допускаем наличие нескольких взаимосвязанных конвейеров, тогда как в [12] конвейер единственный.

Заключение

Авторы настоящей статьи в своих исследованиях столкнулись с необходимостью реализации производительной конвейерной схемы в ограниченных ресурсами рамках. В качестве решения был разработан способ построения системы неуправляемых циклических конвейеров на основе принципа взаимной простоты чисел. Предложенное решение является компромиссом и сочетает в себе компактность с производительностью.

Преимущество циклического конвейера в том, что после введения цикла полный конвейер сжимается в n раз, в результате чего занимаемая конвейером площадь также уменьшается приблизительно в n раз. Такое сжатие позволяет уместить схему в фиксированные рамки FPGA или ASIC. Нужно упомянуть, что создание циклического конвейера также сокращает производительность в n раз по отношению к полному конвейеру. Однако часто производительность полного конвейера является избыточной, а вот ресурсов FPGA или ASIC не хватает.

Предложенный способ построения неостанавливаемых циклических конвейеров зачастую позволяет упростить разработку, сэкономить ресурсы и существенно повысить тактовую частоту схемы по сравнению с традиционными вариантами.

Список литературы

[1] Taraate V. *Logic Synthesis and SOC Prototyping.*– Singapore: Springer.– 2020.– ISBN 978-981-15-1313-8.– xix+251 pp. ↑68

[2] Kilts S. *Advanced FPGA Design: Architecture, Implementation, and Optimization.*– Wiley-IEEE Press.– 2007.– ISBN 9780470127896.– 352 pp. ↑68, 69

[3] Андреев С. С., Дбар С. А., Лацис А. О, Плоткина Е. А. *Как и почему могут быть использованы на практике суперкомпьютеры на базе FPGA.*– М.: РАН.– 2017.– ISBN 978-5-906906-61-8.– 40 с. ↑68, 70, 79, 84, 85

[4] Dally W. J., Harting R. C. *Digital Design: A Systems Approach.*– Cambridge University Press.– 2012.– ISBN 978-0-521-19950-6.– 636 pp. ↑69, 74, 75, 82, 83

[5] Harris S. L., Harris D. *Digital Design and Computer Architecture, RISC-V Edition.*– Elseiver Inc.– 2022.– ISBN 978-0-12-820064-3.– 592 pp. ↑69, 83

[6] *Intel® Hyperflex™ Architecture High-Performance Design Handbook*, ID: 683353, Version: 2021.10.04.– Intel Corporation.– 2021.– 147 pp. ↑69, 75, 83

[7] Emas M. N., Baylis A., Stitt G. *High-frequency absorption-FIFO pipelining for Stratix 10 HyperFlex*, 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM) (Boulder, CO, USA, 2018).– 2018.– Pp. 97–100. ↑69, 75, 83

[8] *LogiCORE IP Multiplier v11.2*, DS255 March 1, 2011.– Xilinx, Inc.– 2011.– 13 pp. ↑69, 84

[9] *LogiCORE IP Floating-Point Operator v6.0*, DS816 January 18, 2012.– Xilinx, Inc.– 2012.– 41 pp. ↑69, 84

[10] Андреев С. С., Дбар С. А., Лацис А. О., Плоткина Е. А. *О применении технологий высокоуровневого синтеза к схемной реализации вычислений* // Препринты ИПМ им. М.В. Келдыша.– 2021.– ид. 34.– 19 с. ↑70, 84, 85

[11] Ioannou L., Michail H. E., Voyiatzis A. G. *High performance pipelined FPGA implementation of the SHA-3 hash algorithm*, 2015 4th Mediterranean Conference on Embedded Computing (MECO) (Budva, Montenegro, 2015).– 2015.– Pp. 68-71. ↑70, 85

[12] Wong M. M., Haj-Yahya J., Sau S. Chattopadhyay A. *A new high throughput and area efficient SHA-3 implementation*, 2018 IEEE International Symposium on Circuits and Systems (ISCAS) (Florence, Italy, 2018).– 2018.– Pp. 1-5. ↑70, 85

[13] *Vivado Design Suite: AXI Reference Guide*, UG1037 (v4.0) July 15, 2017.– Xilinx, Inc.– 2017.– 175 pp. ↑71, 75, 84

[14] *Avalon® Interface Specifications*, ID: 683091, Version: 2022.01.24.– Intel Corporation.– 2022.– 71 pp. ↑71, 75

Поступила в редакцию 08.11.2023;
одобрена после рецензирования 29.11.2023;
принята к публикации 29.11.2023;
опубликована онлайн 13.12.2023.

Рекомендовал к публикации *к.ф.-м.н. С. А. Романенко*

Информация об авторах:

Игорь Алексеевич Адамович

Научный сотрудник Института программных систем имени А. К. Айламазяна РАН. Научные интересы: метавычисления, суперкомпиляция, частичные вычисления, верификация программ, разработка на FPGA и ASIC. Принимал активное участие в разработке интерконнектов для коммуникационных сетей «Паутина» и «3D-тор» суперкомпьютера «СКИФ-Аврора»

0000-0001-9728-3024
e-mail: *i.a.adamovich@gmail.com*

Юрий Андреевич Климов

Старший научный сотрудник ИПМ им. М.В. Келдыша РАН, к.ф.-м.н. Разработчик метода специализации на основе частичных вычислений, принимал активное участие в разработке коммуникационного программного обеспечения для сетей SCI, 3D-тор суперкомпьютера «СКИФ-Аврора» и «МВС-Экспресс» суперкомпьютера К-100.

0000-0001-5081-1547
e-mail: *yuklimov@keldysh.ru*

Вклад авторов: *И. А. Адамович* – 50% (идея, методология, программное обеспечение, расследование, сбор материала, написание черновой версии, доработка и редактирование, визуализация); *Ю. А. Климов* – 50% (идея, методология, расследование, сбор материала, курирование данных, написание черновой версии, доработка и редактирование, визуализация, администрирование).

Авторы заявляют об отсутствии конфликта интересов.

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article HARDWARE AND SOFTWARE FOR DISTRIBUTED AND SUPERCOMPUTER SYSTEMS

UDC 004.312.4
10.25209/2079-3316-2023-14-4-67-89

Cyclic pipeline systems

Igor Alekseevich **Adamovich**[1], Yuri Andreevich **Klimov**[2]

[1] Ailamazyan Program Systems Institute of RAS, Ves'kovo, Russia
[2] Keldysh Institute of Applied Mathematics of RAS, Moscow, Russia
[1]✉*i.a.adamovich@gmail.com*

Abstract. One of the most efficient ways to organize calculations on ASIC or FPGA is the creation of non-stallable pipelines. However, for some computing circuits, the resulting pipeline may be too large for available ASIC or FPGA resources. The authors propose a method for constructing cyclic pipelines, in which data flow control is based on counters and does not depend on the data being transmitting along the pipeline. We proposed the method makes it possible to build more compact non-stallable pipelines. One of the main details of method is to use cycle ratio equal to the number of times the data must go through the loop, after which the pipeline converts the data into the desired result. (*In Russian*).

Key words and phrases: pipeline, ASIC, FPGA, integrated circuit, periodicity, queue, credit

2020 *Mathematics Subject Classification*: 94-05; 94C30

For citation: Igor A. Adamovich, Yuri A. Klimov. *Cyclic pipeline systems*. Program Systems: Theory and Applications, 2023, **14**:4(59), pp. 67–89. (*In Russ.*). https://psta.psiras.ru/read/psta2023_4_67-89.pdf

References

[1] V. Taraate. *Logic Synthesis and SOC Prototyping*, Springer, Singapore, 2020, ISBN 978-981-15-1313-8, xix+251 pp.

[2] S. Kilts. *Advanced FPGA Design: Architecture, Implementation, and Optimization*, Wiley-IEEE Press, 2007, ISBN 9780470127896, 352 pp.

[3] S. S. Andreev, S. A. Dbar, A. O Lacis, E. A. Plotkina. *How and why FPGA-based Supercomputers can be used in Practice*, RAN, M., 2017, ISBN 978-5-906906-61-8 (in Russian), 40 pp.

[4] W. J. Dally, R. C. Harting. *Digital Design: A Systems Approach*, Cambridge University Press, 2012, ISBN 978-0-521-19950-6, 636 pp.

[5] S. L. Harris, D. Harris. *Digital Design and Computer Architecture, RISC-V Edition*, Elseiver Inc, 2022, ISBN 978-0-12-820064-3, 592 pp.

[6] *Intel® Hyperflex™ Architecture High-Performance Design Handbook*, ID: 683353, Version: 2021.10.04, Intel Corporation, 2021, 147 pp.

[7] M. N. Emas, A. Baylis, G. Stitt. "High-frequency absorption-FIFO pipelining for Stratix 10 HyperFlex", 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM) (Boulder, CO, USA, 2018), 2018, pp. 97–100.

[8] *LogiCORE IP Multiplier v11.2*, DS255 March 1, 2011, Xilinx, Inc, 2011, 13 pp.

[9] *LogiCORE IP Floating-Point Operator v6.0*, DS816 January 18, 2012, Xilinx, Inc, 2012, 41 pp.

[10] S. S. Andreev, S. A. Dbar, A. O. Lacis, E. A. Plotkina, *Preprinty IPM im. M.V. Keldysha*, 2021, id. 34 (in Russian), 19 pp.

[11] L. Ioannou, H. E. Michail, A. G. Voyiatzis. "High performance pipelined FPGA implementation of the SHA-3 hash algorithm", 2015 4th Mediterranean Conference on Embedded Computing (MECO) (Budva, Montenegro, 2015), 2015, pp. 68-71.

[12] M. M. Wong, J. Haj-Yahya, S. Chattopadhyay A. Sau. "A new high throughput and area efficient SHA-3 implementation", 2018 IEEE International Symposium on Circuits and Systems (ISCAS) (Florence, Italy, 2018), 2018, pp. 1-5.

[13] *Vivado Design Suite: AXI Reference Guide*, UG1037 (v4.0) July 15, 2017, Xilinx, Inc, 2017, 175 pp.

[14] *Avalon® Interface Specifications*, ID: 683091, Version: 2022.01.24, Intel Corporation, 2022, 71 pp.

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ПРОГРАММНОЕ И АППАРАТНОЕ ОБЕСПЕЧЕНИЕ РАСПРЕДЕЛЕННЫХ И СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМ

УДК 004.432.4
10.25209/2079-3316-2023-14-4-91-122

Цветные сети Петри и язык распределенного программирования UPL: их сравнение и перевод

Аркадий Валентинович **Климов**✉
Институт проблем проектирования в микроэлектронике РАН
✉*arkady.klimov@gmail.com*

Аннотация. Сети Петри широко используются как средство моделирования распределенных мультиагентых систем. Существуют инструменты работы с расширенными сетями Петри, в которых токены нагружены произвольными данными. В частности, CPN Tools позволяет описывать, проигрывать и исследовать цветные сети Петри (Coloured Petri Nets, CPN). Ставится вопрос о возможности использовать этот инструмент для разработки, прототипирования и исследования параллельных распределенных вычислительных алгоритмов, в идеале — превращения их в работающие эффективные параллельные программы. У нас есть опыт экспериментального программирования разных алгоритмов в нашем графическом языке UPL, который пока существует как бы «на бумаге». Его сравнение с CPN показывает, что в их семантиках много общего. В статье оба языка определяются, сравниваются на примерах и через правила перевода из одного в другой. Также описываются средства управления распределением вычислений для UPL. Интересен вопрос об их переносе в CPN, где им пока аналога нет.

Ключевые слова и фразы: сети Петри, цветные сети Петри, параллельное программирование, потоковая модель вычислений, граф алгоритма, графическое программирование, язык UPL, функция распределения

Благодарности: Работа поддержана ИППМ РАН

Для цитирования: Климов Ар. В. *Цветные сети Петри и язык распределенного программирования UPL: их сравнение и перевод* // Программные системы: теория и приложения. 2023. **Т. 14**. № 4(59). С. 91–122. https://psta.psiras.ru/read/psta2023_4_91-122.pdf

Введение

Сети Петри [1, 2] широко используются как средство моделирования распределенных мультиагентых систем самых разных видов: от технических установок [3] до больших интегральных схем [4] и нейронных сетей [5]. Существуют инструменты работы с расширенными сетями Петри, в которых токены нагружены произвольными данными. В частности, *CPN Tools* позволяет описывать, проигрывать и исследовать цветные сети Петри (Coloured Petri Nets, CPN)[6].

В Цветных Сетях Петри (ЦСП) каждый переход может сопровождаться вычислением новых выходных значений из старых входных. Это приводит к возможности описания на языке CPN не только моделей распределенных систем, но вычислительных алгоритмов широкого класса.

У нас накоплен большой опыт экспериментального программирования в языке потоков данных UPL [7, 8]. Процесс выполнения CPN во многом похож на вычислительный процесс в языке UPL. В последнем так же, как в CPN переходы, имеются узлы-шаблоны, определяющие правила локальных трансформаций множеств токенов. Это множество, как и разметка в CPN, представляет собой состояние вычислительного процесса. Процесс смены состояний состоит из срабатываний узлов, как и переходов в CPN, образуя виртуальный вычислительный граф, переводящий состояние с исходными данными в состояние с результатом вычислений. И как в CPN, для срабатывания узла в текущем состоянии должны найтись несколько токенов с согласованными компонентами.

Название «UPL» означает универсальный язык параллельного программирования. Мы полагаем, что в нем адекватным образом может быть выражен любой вид алгоритмического параллелизма, имеющийся в различных существующих языках параллельного программирования. И хотя Цветные Сети Петри обычно не позиционируются как «язык программирования», они порождают свой особый алгоритмический параллелизм, который хотелось бы проверить на предмет его выразимости в языке UPL и наоборот.

При попытке перевода из CPN в UPL возникают трудности, показывающие недостаточную универсальность и выразительность UPL. Мы отмечаем эти трудности и задаем ограничения на CPN, при которых они не возникают. С другой стороны, эти трудности указывает на желательность введения некоторых расширений UPL для их преодоления. Рассмотрению этих расширений будет посвящена другая статья.

Перевод из UPL в CPN также в некоторых случаях, причем весьма существенных, наталкивается на трудности, показывающие недостаток

определенных возможностей CPN. Их подробное обсуждение также войдет в будущую статью.

Статья написана по следующему плану. В разделе 1 даются формальные определения сетей Петри – сначала стандартных (с «пустыми фишками»), потом цветных (где токены несут данные – «цвета»). Для обоих вариантов описывается точная семантика, причем для цветных двумя способами: непосредственно, как это принято в литературе, и путем сведения к стандартной сети Петри, возможно бесконечной. Завершается данный раздел работающим примером параллельного алгоритма сортировки. В разделе 2 язык UPL вводится сначала как результат структурных преобразований подкласса сетей CPN с указанием необходимых ограничений на них. Затем он же определяется независимо со своими правилами выполнения. В разделе 3 показаны примеры типовых фрагментов одновременно на CPN и UPL, а в разделе 4 дан более развернутый пример алгоритма на обоих языках. Сходства и особенности двух языков подробно обсуждаются в разделе 5, показывая, что в каждом имеются элементы, не выразимые в другом. В разделе 6 кратко описаны средства управления распределенным выполнением программ UPL, которые едва ли могут быть применены к CPN (оставляем это как задачу дальнейших исследований). Сравнение с близкой работой приводится в разделе 7. В заключительном разделе 9 сведены основные тезисы статьи.

1. Язык CPN и его семантика

Здесь дадим не совсем полное, но достаточное для наших целей полуформальное описание. Сначала напомним определение стандартных сетей Петри, а затем дадим его расширение до цветных сетей Петри.

1.1. Стандартные сети Петри

1.1.1. *Структура*

(*Стандартная*) *сеть Петри* (SPN) определяется как двудольный ориентированный граф. *Вершины* (*nodes*) графа называются *переходами* (*transitions*) $t \in T$ и *местами* (*places*) $p \in P$, а связи – *дугами* $a \in A$. (Множества T, P и A попарно не пересекаются). Еще задана функция вершин $\mathcal{N} : A \to (P \times T) \bigcup (T \times P)$, которая каждой дуге приписывает элемент либо из $P \times T$ (входные дуги), либо из $T \times P$ (выходные дуги). Множество входных (выходных) дуг перехода t обозначается $\bullet t$ (соответственно, $t\bullet$). Такое определение допускает наличие любого числа дуг из места p в переход t (а также из перехода t в место p) при том, что формально это различные элементы множества дуг A.

Текущее состояние стандартной сети Петри, называемое *разметкой* (*marking*), задается как некоторое мультимножество[1] мест, $M \in P_{MS}$. Другими словами, разметка задает для каждого места p некоторое количество $M(p)$ *токенов* (*tokens*), находящихся в этом месте. Обозначения $\bullet t$ и $t\bullet$ будем также толковать как мультимножества мест, соответственно, входных или выходных для перехода t.

Для сети Петри должна быть задана начальная разметка M_0. Таким образом, Стандартная Сеть Петри с разметкой задается как набор из пяти компонентов: $SPN = (P, T, A, N, M_0)$.

1.1.2. *Семантика*

Теперь определим поведение SPN. Будем говорить, что разметка M *допускает* (*enables*) переход t, если $\bullet t \leqslant M$. В этом случае возможно *срабатывание* (*occurrence*) перехода t, которое приводит к новой разметке $M' = M - (\bullet t) + (t\bullet)$. В общем случае *шаг* (*step*) есть непустое конечное мультимножество переходов $Y \in T_{MS}$. Шаг Y *допустим* (при разметке M), если $\sum_{t\in Y}(\bullet t) \leqslant M$ (суммирование проводится с учетом кратностей элементов Y). В этом случае возможно *одновременное срабатывание* всех переходов шага Y, и оно приводит к разметке $M' = M - \sum_{t\in Y}(\bullet t) + \sum_{t\in Y}(t\bullet)$. Тогда говорят, что разметка M' *непосредственно достижима* из разметки M, и записывают это как $M[Y\rangle M'$. Конечную последовательность шагов и разметок $M_1[Y_1\rangle M_2[Y_2\rangle \ldots M_n[Y_n\rangle M_{n+1}$, в которой $M_i[Y_i\rangle M_{i+1}$ для всех $i \in \{1, 2, \ldots, n\}$, будем называть *конечной последовательностью срабатываний* с начальной разметкой M_1, конечной разметкой M_{n+1} и длиной $n \geqslant 0$. В этом случае говорят, что разметка M_{n+1} *достижима из разметки* M_1. (Аналогично можно определить бесконечную последовательность срабатываний). Множество разметок, достижимых из разметки M_1, обозначают как $[M_1\rangle$. Разметка *достижима*, если она входит в $[M_0\rangle$.

---

[1] *Мультимножество* (*multiset*) типа X есть функция $h : X \to \mathbb{N}$, которая каждому элементу X ставит в соответствие его вес – неотрицательное целое число. $|h| = \sum_{x\in X} h(x)$˜– размер мультимножества h. Обычно используют конечные мультимножества, $|h| < \infty$, даже если тип X бесконечный. Сложение (вычитание), умножение на число и отношение вложения мультимножеств определяют как расширение операций $+$ $(-)$,$\cdot$ и $\leqslant$ в $\mathbb{N}$ (последней в смысле *forall*). Множество (тип) всех (конечных) мультимножеств над X обозначают X_{MS}. Его элементы записывают как формальные суммы простых мультимножеств вида $k'x$, где k˜– кратность (вес) элемента x. Пустое мультимножество обозначают $\varnothing_X$, или просто $\varnothing$.

1.2. Цветные сети Петри

1.2.1. *Структура*

Теперь расширим стандартные сети Петри до цветных (Coloured) сетей Петри (CPN)[6]. Понадобится некоторый язык, в котором можно определять типы данных и записывать выражения над значениями этих типов. Авторы CPN используют слегка модифицированный функциональный язык Standard ML[9] (SML), называя его CPN ML. Типы в нем называются наборами цветов (colour sets) и включают целые (int), логические (bool), интервалы целых, конечные перечисления, а также структурные типы – кортежи (products), записи (records) и списки (lists) ограниченной или неограниченной длины.

Чтобы перейти от стандартной сети Петри к цветной, надо вначале задать совокупность типов данных (наборов цветов) Σ и функцию цветности $C : P \to \Sigma$, которая каждому месту p приписывает некоторый тип $C(p)$. Тип места указывает тип значений, которые может содержать токен, находящийся в этом месте. А какие именно токены и в каком количестве находятся в каждом месте задает разметка. То есть, разметка M это функция, которая с каждым местом связывает мультимножество $M(p) \in C(p)_{MS}$. Далее, каждой дуге $a \in A$ припишем выражение $E(a)$ типа $C(p)_{MS}$, где p – место, связанное с дугой a. (Можно использовать также выражение типа $C(p)$, тогда оно автоматически приводится к типу $C(p)_{MS}$ путем приписывания префикса $1'$). Для каждого перехода t зададим выражение $G(t)$ типа bool – охрану (по умолчанию – True).

Все выражения могут содержать в качестве свободных переменных только переменные из заранее заданного списка переменных V, причем для каждой переменной $v \in V$ должен быть зафиксирован ее тип $Type(v) \in \Sigma$. Если задано *связывание* (*binding*) $b : V \to \bigcup\Sigma$ всех переменных $v \in V$ со значениями $b(v)$ нужного типа $Type(v)$, то всякое выражение $E(a)$ или $G(t)$ может быть вычислено до значения типа $C(p)_{MS}$ или bool, соответственно. Для результата вычисления будем использовать обозначение $E(a)\langle b\rangle$, соответственно $G(t)\langle b\rangle$. При этом связывание b должно содержать как минимум все (свободные) переменные данного выражения. Наконец, в качестве начальной разметки M_0 следует задать для каждого места p замкнутое (не содержащее свободных переменных) выражение $I(p)$, которое вычисляется до некоторого конечного мультимножества типа $C(p)_{MS}$. Таким образом, цветная сеть Петри задается как набор из десяти компонентов: $CPN = (P, T, A, N, \Sigma, V, C, E, G, I)$.

1.2.2. *Семантика*

Теперь определим поведение CPN. Сделаем это двумя различными способами, и затем покажем их эквивалентность.

Первый способ – непосредственный

Этот способ используется в статьях и руководствах по CPN [6]. Нам понадобятся следующие понятия и обозначения:

Разметка это функция M, которая каждому месту $p{\in}P$ приписывает мультимножество токенов: $M(p){\in}C(p)_{MS}$.

Начальная разметка M_0 задается как результат вычисления замкнутых выражений $I(p)$.

Внешний токен (token element) – это пара (p, v), где $v{\in}C(p)$. Множество всех возможных внешних токенов данной CPN обозначим TE.

Множество переменных перехода t, обозначаемое $Var(t)$ состоит из всех свободных переменных всех выражений, стоящих на дугах, связанных с t, а также в $G(t)$.

Связывание перехода t это функция b, которая каждой переменной $v{\in}\,Var(t)$ приписывает значение $b(v)$ типа $Type(v)$ так, что выполняется охрана $G(t)\langle b\rangle$. Множество всех связываний перехода t обозначается $B(t)$.

Внешнее связывание (binding element) это пара (t, b), где $t{\in}T$ и $b{\in}B(t)$. Множество всех внешних связываний перехода t определим как $BE(t) = \{(b, t)|b{\in}B(t)\}$. Множество всех внешних связываний данной CPN обозначим BE.

Шаг $Y{\in}BE_{MS}$ это непустое конечное мультимножество внешних связываний. Далее прилагательное «внешнее» (оно лишь как бы уточняет связывание для «внешнего» наблюдателя) будем опускать, когда из контекста понятно, о чем речь.

Определим $A(p, t)$ как множество всех дуг из p в t.

$$A(p, t) = \{a|N(a) = (p, t)\}$$

. Аналогично $A(t, p)$.

Определим $E(p, t)$ как формальную сумму всех выражений на дугах из $A(p, t)$. Это допустимо, поскольку типы значение всех этих выражений одинаковые. Если нет дуг из p в t, то $E(p, t) = \varnothing$. Аналогично $E(t, p)$.

Связывание (t, b) **допускается** разметкой M_1, если для всякого места p выполняется $E(p, t)\langle b\rangle{\leqslant}M_1(p)$. Тогда переход t (со связыванием b) может *сработать*, породив новую разметку $M_2 = M_1 - \sum_{p\in P} E(p, t)\langle b\rangle + \sum_{p\in P} E(t, p)\langle b\rangle$. Иначе говоря, значения выражений $E(p, t)\langle b\rangle$ на входных дугах показывают мультимножества токенов, которые необходимы

для срабатывания перехода t со связыванием b и которые при этом срабатывании будут из разметки удалены, а взамен будет добавлено мультимножество токенов, заданное выражением $E(t,p)\langle b\rangle$.

Шаг Y **допускается** разметкой M_1, если для всякого места p справедливо $\sum_{(t,b)\in Y} E(p,t) \leqslant M_1(p)$, где суммирование производится с учетом кратностей элементов Y. Тогда могут *сработать одновременно* все элементы шага Y, порождая новую разметку M_2, такую что для всякого места p выполняется $M_2(p) = M_1(p) - \sum_{(t,b)\in Y} E(p,t) + \sum_{(t,b)\in Y} E(t,p)$. И тогда говорят, что разметка M_2 *непосредственно достижима* из разметки M_1, и записывают это как $M_1[Y\rangle M_2$.

Конечная или бесконечная последовательность шагов, а также отношение достижимости определяются и обозначаются так же, как и для стандартных сетей Петри (см. выше).

Замечание по реализации. Надо признать, что поиск кандидатов связываний для срабатывания является в общем случае неразрешимой задачей: выражения могут содержать общерекурсивные функции, для которых требуется найти аргументы по заданным значениям. Но есть хорошо определенный класс выражений, для которых эта задача разрешима: это образцы. Переменная или константа это образец, образцом является структура из образцов, а также список образцов. Важно (это проверяется автоматически), чтобы на части входных дуг стояли образцы, и чтобы в них встречалась каждая переменная из $Var(t)$, не считая лишь те переменные, которые встречаются только на выходных дугах. Для последних при срабатывании порождается случайное значение из ее типа, число элементов которого должно быть невелико.

Второй способ — через сведение к стандартной сети Петри

Для заданной CPN определим $SPN = (\acute{P}, \acute{T}, \acute{A}, \acute{N}, \acute{I})$ следующим образом.

Множество мест $\acute{P} = \{(p,v)|p{\in}P, v{\in}C(p)\}$. Иначе говоря, каждое место p превращается в семейство мест, индексированное типом $C(p)$.

Множество переходов $\acute{T} = \{(t,b)|t{\in}T, b{\in}B(t)\}$. Каждый переход t превращается в семейство переходов, индексированное всевозможными связываниями $b{\in}B(t)$ (не забудем, что в понятие связывания уже включено условие $G(t)\langle b\rangle = true$).

Пусть a — входная/выходная дуга, $N(a) = (p,t)/(t,p)$. Она превращается в семейство входных/выходных дуг, индексированных всевозможными связываниями $b{\in}B(t)$. При этом $\acute{N}((a,b)) = (\acute{p},\acute{t})/(\acute{t},\acute{p})$, где $\acute{p} = (p, Expr(a)\langle b\rangle)$, $\acute{t} = (t,b)$.

Начальная разметка: $\acute{I}(\acute{p}) = \acute{I}(p,v) = (I(p)\langle\rangle)(v)$, где $v{\in}C(p)$, а $\langle\rangle$ – пустое связывание.

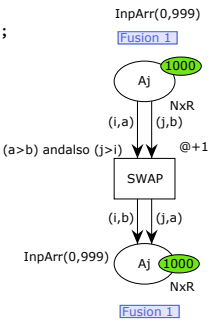
Для всякой разметки $M(p)$ для CPN соответствующая ей разметка стандартной сети есть $\acute{M}(\acute{p}) = \acute{M}(p,v) = M(p)(v)$ для всех $p{\in}P$ и $v{\in}C(p)$. Легко видеть, что это изоморфизм разметок, $\mu : M \leftrightarrow \acute{M}$ (который есть не что иное, как «закарривание» и «раскарривание» функций). Понятия шага для CPN и SPN попросту совпадают. Одинаковость преобразований разметок, производимых шагом, вытекает непосредственно из определений. Поэтому совпадают и отношения достижимости. Этим фактически доказана

Теорема 1. *Два указанных выше способа определения семантики цветных сетей Петри эквивалентны.*

1.3. Пример

Приведем простой пример, иллюстрирующий богатые возможности CPN как средства описания алгоритмов. На рисунке 1 показана CPN-сеть, реализующая своеобразный вариант параллельной сортировки.

```
colset NxR = product INT*REAL timed;
fun InpArr(k,n)=if n<k then []
  else 1`(k,InpElem(k,n))++InpArr(k+1,n);
fun InpElem(k,n)=uniform(1.0,10.0);
var a,b:REAL;
var i,j:INT;
```




```
1`(0,1.01412269381)@3+++
1`(1,1.0440885341)@10+++
1`(2,1.05297652402)@10+++
1`(3,1.06968683706)@9+++
1`(4,1.08057352951)@10+++
1`(5,1.08209373433)@10+++
1`(6,1.08632284453)@9+++
1`(7,1.09025600231)@10+++
1`(8,1.09530804776)@14+++
1`(9,1.10128428186)@15+++
1`(10,1.11296671867)@16+++
1`(11,1.11577442992)@16+++
1`(12,1.12042167136)@15+++
1`(13,1.12383640005)@16+++
1`(14,1.13396820991)@16+++
1`(15,1.13412490121)@17+++
1`(16,1.13752250147)@17+++
1`(17,1.14745061851)@17+++
1`(18,1.15276270413)@18+++
1`(19,1.15760359688)@18+++
1`(20,1.17313336915)@19+++
1`(21,1.18506303019)@19+++
1`(22,1.19233149811)@19+++
```

Рисунок 1. Пример: CPN-сеть, реализующая алгоритм сортировки

Это вполне работающий параллельный алгоритм, выполняемый за $O(\log N)$ шагов[2]. Он состоит из одного места Aj и одного перехода SWAP. На рисунке мы видим два места, но это просто два избражения одного и того же места, о чем свидетельствует подпись `Fusion 1` (для

[2]Это установлено эмпирически: с каждым удвоением размера массива число шагов увеличивается на 3-5 единиц (в прямой зависимости от начальной упорядоченности). В данном прогоне было выполнено 12420 переходов за 33 временных шага.

удобства). Слева вверху – сопутствующие определения на CPN ML. Место `Aj` имеет тип (colset) `NxR`, являющийся произведением `INT` на `REAL`. Массив вещественных чисел представляется в виде мультимножества таких пар, у которых компонент `INT` кодирует номер (индекс) в массиве, а `REAL` – собственно значение. Две функции, `InpArr` и `InpElem`, служат для генерации случайного начального массива заданного размера. Соответствующий вызов `InpArr(0,999)`, порождающий 1000 элементов, приписан месту `Aj`. Сортировка заключается в подмене номеров в соответствии с порядком значений. На рисунке 1 справа показан начальный отрезок массива после завершения работы.

Переход `SWAP` соединен с местом `Aj` двумя входными и двумя выходными дугами. Согласно заданным на дугах выражениям переход срабатывает, выдергивая из места какие-либо два элемента, и тут же возвращает их, переставляя индексы. При переходе указана охрана: `(a>b)andalso(j>i)`, распознающая нарушение порядка. За один шаг могут одновременно сработать сразу несколько экземпляров перехода, при условии, что они не используют общих входных токенов. Поскольку каждый переход выполняется за время 1, на каждом такте выполняется шаг, состоящий из максимального множества непересекающихся готовых пар.

Все используемые переменные (`a`,`b`,`i`,`j`) и их типы заранее объявлены. Тип места задан как `timed`, чтобы моделировалось время работы. При этом переходу приписана задержка `@+1`, то есть 1 такт. С учетом всяких случайностей алгоритм сортирует 1000 элементов за 30–40 модельных тактов, выполняя в общей сложности около 13000 элементарных (попарных) обменов. Реальное же время работы интерпретатора – около 30 секунд, причем, чем ближе к концу, тем дольше идет подбор пар для очередного такта.

Существенный и почти неустранимый недостаток данного решения – отсутствие признака завершения. Второй дефект – оно нереализуемо «в железе». То есть непонятно, моделью какого реального процесса оно может быть. Здесь оно использовано просто для иллюстрации возможностей CPN. Как видим, они заметно шире, чем это нужно для описания (моделей) «реальных» систем.

2. Графический язык UPL

2.1. UPL как результат перевода из CPN

UPL – графический язык описания параллельных алгоритмов в парадигме потока данных (dataflow). Он имеет много общего с CPN. Вначале покажем, как UPL может быть получен в результате небольших трансформаций CPN при соблюдении некоторых ограничений возможностей CPN. Налагаемые ограничения будут помечаться буквами и **выделяться**. В подразделе 2.2 будет дано независимое определение графического языка UPL.

Первое и самое существенное ограничение – в исходной CPN-схеме **(A) из каждого места исходит ровно одна дуга**.

С точки зрения теории сетей Петри это очень сильное ограничение. Оно запрещает конфликты (между разными переходами), которые составляют основу сетей Петри. Правда, сохраняются конфликты между разными связываниями одного перехода (которые переходят в разные переходы стандартной сети Петри при описанном выше сведении). В дальнейшем это ограничение частично будет ослаблено через механизм ветвей языка UPL [11], что будет подробно рассматриваться в другой (будущей) статье.

Теперь мы можем избавиться от входных дуг на схеме, просто «приклеив» входные места к переходам, которые теперь будем называть *узлами*, а их входные места – *входами* узлов. Дуги из переходов в места остаются в UPL и ведут из узла на вход другого (или своего же) узла. Теперь будем называть их *стрелками*.

Входу в UPL приписывается локальное (для узла) имя и тип. Предполагается, что в исходной CPN-схеме **(B) выражение на входной дуге было просто переменной или структурой из переменных**. Будем называть такие выражения *форматными*. В частности, они должны быть одноэлементными (как мультимножества). Они являются образцами частного вида, которые не накладывают ограничения на тип, а только обеспечивают удобный доступ к его частям.

На выходных дугах выражения могут быть любыми, но они могут подвергаться преобразованиям. Условное выражение распадается на две стрелки, с соответствующими условиями на них. Пустая альтернатива ликвидируется. Выражения-мультимножества также распадаются на несколько стрелок, но одиночное кратное значение переходит в одну стрелку с кратным токеном. Новые переменные (не использованные на входной дуге) заменяются явным вызовом генератора случайных значений.

Следующее важное ограничение — **(C) допускаются только охраны типа равенства между входными переменными на разных дугах.** Некоторые входные переменные могут встречаться повторно на разных дугах, что равносильно использованию охранного равенства между переменными. Заметим, что пример на рисунке 1 не может быть переведен в UPL из-за несоблюдения ограничения (A) и (C).

Переменные, использованные в охранных равенствах (или повторно), в UPL составляют индекс и выносятся в отдельную структуру, заключаемую в угловые скобки (на графической схеме – в шестиугольники). Все остальные переменные считаются элементами данных и должны быть уникальными среди выражений на входных дугах данного перехода. Структура индекса $\langle i_1, \ldots, i_k \rangle$ формируется из повторных переменных выражений на входных дугах перехода N и является общей как для узла N в целом, так и

для всех его входов. Пусть входное место p перехода N в CPN стало входом p узла N в UPL. Тогда каждая дуга, идущая в место p в CPN, превращается в стрелку, идущую на вход p в UPL, а стоящее на ней выражение E распадается на выражение-значение e и выражения-индексы $\langle e_1, \ldots, e_k \rangle$. Компоненты выделяются путем сопоставления выражения E с форматным выражением F на дуге из места p в переход N. Компоненты, сопоставившиеся индексным переменным i_j, переходят в поля индекса e_j, оставшиеся – в значение входа e. Но не все индексные переменные i_j могут содержаться в F. Тогда на позиции отсутствующих в F переменных индекса ставится знак «*», который понимается как «любой».

Теперь вместо поиска связывания перехода N, которое делает переход в CPN активным, в UPL будем искать набор токенов, по одному на каждом входе, имеющим ровно один общий индекс. В результате срабатывания узла выбранные токены с его входов снимаются (токен, имеющий кратность, сохраняется с уменьшенной на 1 кратностью), а по всем его выходным стрелкам формируются и передаются токены на входы других узлов.

На рисунке 2*а* представлен условный фрагмент CPN-сети, содержащий допустимый переход N общего вида, и на рисунке 2*б* – его преобразование в язык UPL согласно описанным выше правилам.

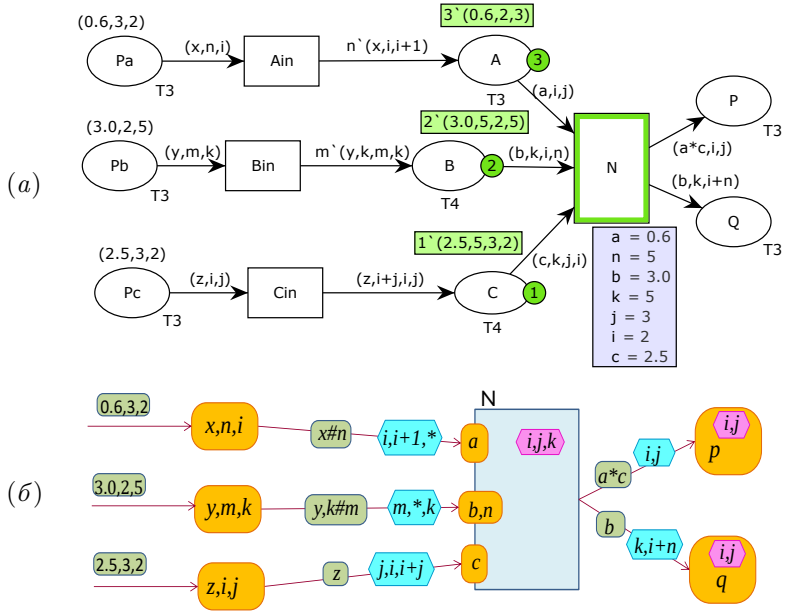


Рисунок 2. Преобразование типового перехода (N) сети CPN (*а*) в узел N схемы UPL (*б*)

Переход N имеет три входных места A, B, C типов T3, T4, T4 (записей из 3 или 4 элементов, соответственно). Среди переменных перехода N три переменные: i, j, k – используются повторно, поэтому они станут индексом. Остальные переменные: a,b,n,c – в схеме на UPL будут входными данными трех входов. Подающие переходы Ain и Bin порождают кратные токены, Cin – одиночный. На схеме изображено состояние непосредственно перед срабатыванием перехода N. Под ним изображено связывание, с которым переход N готов к срабатыванию. После срабатывания место C станет пустым, а на местах A и B останутся токены с кратностями 2 и 1 соответственно.

Результирующий узел N изображен в виде прямоугольного блока. В нем имеется индекс $\langle i, j, k\rangle$, показанный в розовом шестиугольнике, и три входа: a, "b,n", и c. Подающие переходы Ain, Bin, Cin превратились в три одновходовых узла без индексов, которые изображаются без прямоугольников. В качестве имен (входов) фигурируют списки входных переменных.

Первые два подающих узла выдают токены с кратностями, стоящими после знака #. Входные данные на каждой дуге превратились в данные с индексами. Между индексом токена на стрелке и индексом узла имеется соответствие согласно позициям. На места недостающих во входном выражении переменных ставится знак «*», который является «джокером». При сопоставлении он может принять любое значение. С учетом заданных значений будут поданы токены:

$0.6\#3 \to \mathsf{N.a}\langle 2, 3, *\rangle$

$3.0\#2 \to \mathsf{N.(b, n)}\langle 2, *, 5\rangle$

$2.5 \to \mathsf{N.a}\langle 2, 3, 5\rangle$

Данная тройка образует активную комбинацию с единственным общим индексом $\langle 2, 3, 5\rangle$. Джокер в первом токене примет значение 5, во втором – 3. В результате срабатывания узла $N\langle 2, 3, 5\rangle$ на двух первых входах останутся токены с кратностями 2 и 1, на третьем – ничего.

Примечание. Выходные места P и Q здесь превращены в одновходовые узлы с индексами, хотя вводить индексы здесь не было необходимости, поскольку это разные узлы, а не входы одного узла.

Использованные здесь обозначения и правила поясняются в следующем разделе. Забегая вперед, укажем на еще одно необходимое ограничение CPN, которое в данном примере соблюдено: в результате перевода в UPL **(D) хотя бы на одном входе должен быть токен с кратностью 1**.

В противном случае может произойти семантическая путаница (см. пояснение к примеру 2 ниже). Пока отметим только, что (в отличие от CPN) в UPL токен с кратностью, вообще говоря, *не* равносилен соответствующему количеству одинаковых экземпляров однократного токена.

Рассмотренные преобразования с сопутствующими им ограничениями собраны в таблице 1.

Таблица 1. Сводная таблица соответствия (преобразований) CPN→UPL с сопутствующими ограничениями

Преобразование CPN → UPL	Ограничения CPN	Ограничения UPL
Входные места «приклеиваются» к переходу в виде входного порта.	(A) Из каждого места исходит лишь одна дуга, (B) на ней выражение форматное.	
Условие в выражении на выходной дуге переходит в условие выдачи токена. Альтернативы и суммы распадаются на разные стрелки. Новые переменные на выходных дугах заменяются явным вызовом генератора случайных значений.	После предварительных преобразований выходные дуги перехода несут одноэлементные (как множества) выражения, возможно с кратностью.	
Отождествляемые (через охраны или повторные переменные) части цвета собираются в индекс.	(D) Охраны: только равенства переменных (включая повторные переменные) разных дуг.	
В выходном токене на место недостающей переменной индекса ставится «*».		
Кратность значения на выходной дуге переходит в кратность токена	(D) Хотя бы на одно входное место перехода должны приходить только токены кратности 1.	Хотя бы на один вход узла приходят только токены кратности 1.

2.2. Независимое определение UPL

В этом подразделе дается независимое (от CPN) определение языка UPL и его семантики.

Программа на языке UPL раскладывается на отдельные узлы, которые передают друг другу сообщения – токены.

Узел имеет *имя*, один или несколько именованных *входов* и *индекс*, которым различаются экземпляры узлов. Индекс обычно имеет тип

структуры с целочисленными полями. Входные токены приходят на вход узла и могут на нем накапливаться. Узел в программе изображается блоком, входы – небольшими кружками на его стороне. Узел определяет класс *конкретных* узлов, характеризующихся конкретным значением индекса. Конкретный узел срабатывает, когда на каждом входе для него имеется токен. Этот набор токенов называется *активной комбинацией* (АК), а срабатывание – *активацией*.

При срабатывании участвующие в АК токены обычно одномоментно снимаются с входов или, если у токена есть кратность, остаются на нем с уменьшенной на 1 кратностью. Оставшийся токен затем может участвовать в создании других АК с другими токенами. Но: никакая АК не может сработать повторно. Это принцип *однократной активации* (ПОА). Он имеет значение, когда все токены АК имели кратность. Это допустимо в UPL и полезно для организации взаимодействий по типу «каждый с каждым».

Обычно требуется, чтобы по приходу каждого токена на один из входов срабатывали (в некотором порядке) все новые АК. Мы называем это принципом *последовательной активации* (ППА). Но это не исключает параллелизм. Если две (или более) АК не пересекаются (нет общих токенов), они могут сработать одновременно. Это позволяет разносить изолированные подмножества токенов на разные процессоры (ядра), которые внутри работают последовательно, а между собой параллельно.

Узел-класс также имеет *программу*, которая выполняется (возможно, с некоторой задержкой) после срабатывания АК.

Программа узла пишется на некотором процедурном языке программирования (C, Паскаль и т.п.). В ней в качестве аргументов могут использоваться только входные данные (из участвующих в АК токенов) и поля индекса. Наряду с обычными операторами языка программа узла может содержать операторы посылки токена вида:

$$D[\#m] \to N.p\langle I\rangle\,,$$

где D – выражение, вырабатывающее значение, кратность m – целочисленное выражение или «#», N – имя узла, p – имя входа, индекс I – список целочисленных выражений или символов «*» (через запятую). Квадратные скобки здесь это метасимволы, указывающие на возможность отсутствия (когда кратность равна 1). Запись «##» понимается как бесконечная кратность. Оператор посылки токена читается: послать значение D в виде токена кратности m на вход p узла N с индексом I.

Эту же запись (с вычисленными выражениями) можно рассматривать как формат самого токена.

В графической форме программа записывается внутри блока (обычно это набор присваиваний локальным переменным, завершаемый выражением, вырабатывающим посылаемое значение) а каждый оператор посылки изображается стрелкой от границы блока к входу узла. Формат индекса задается в (розовом) шестиугольнике внутри блока. На стрелке записываются: у основания – условие посылки, в середине в округленном прямоугольнике – посылаемое значение как выражение, ближе к концу в шестиугольнике – целевой индекс (структура из выражений или «*»). Все выражения отрабатываются в контексте узла-отправителя в терминах его имен входов, полей индекса и локальных переменных.

Результатом работы программы узла являются сформированные в ней токены. Весь внешний эффект программы узла состоит только в порождении некоторого числа новых токенов. По завершении своей программы конкретный узел прекращает свою активность, но может быть вновь активирован новым набором токенов.

В общем случае индекс в токене может содержать «*» на месте некоторых полей, и тогда его адресатом считается не один конкретный узел, а сразу множество конкретных узлов, индекс которых в этой позиции имеет любое значение. Пусть узел N имеет s входов: $p_1, \ldots, p_s$. Тогда набор из s токенов, направленных на все различные входы узла N будет активной комбинацией (АК), если их индексы $I_1, \ldots, I_s$ как множества имеют непустое, и притом одноэлементное пересечение I_0. Иначе говоря, когда имеется единственный конкретный узел $N\langle I_0\rangle$, входящий в множества адресатов каждого токена из данной АК. (Ситуация, когда такой узел не единственный, считается ошибкой, которая вызывает аварийное прекращение работы.) Поэтому активироваться может только конкретный узел, а не их множество. Мы называем это принципом конкретности активации (ПКА).

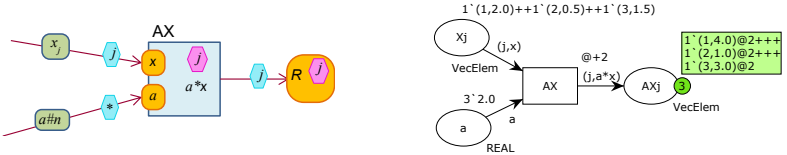
Все выражения в программе узла могут использовать в качестве своих аргументов только локальные переменные, имена входов и имена полей индекса. Есть еще доступ к глобальным константам, которые принимают значение перед началом работы пограммы и в процессе ее не изменяются. Таким образом, множество и содержимое исходящих (из узла с конкретным индексом) токенов может зависеть только от информации, содержащейся во входящих (в данный конкретный узел) токенах.

3. Примеры простых фрагментов программ на UPL и CPN

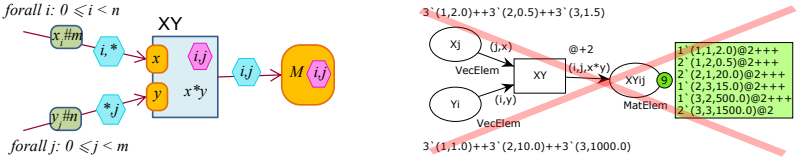
На рисунке 3 приведены несколько примеров простых операций линейной алгебры над скалярами и векторами. Слева показана схема

UPL CPN

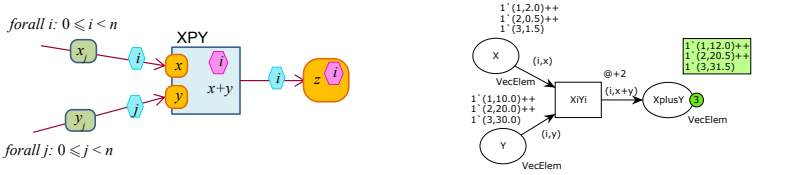
1. Умножение ▫ × ▭ = ▭ скаляра a на вектор X



2. Внешнее умножение ▯ × ▭ = ▭ вектора X на вектор Y



3. Сложение ▯ + ▯ = ▯ плотных векторов



4. Скалярное произведение ▭ · ▯ = ▫ плотных векторов

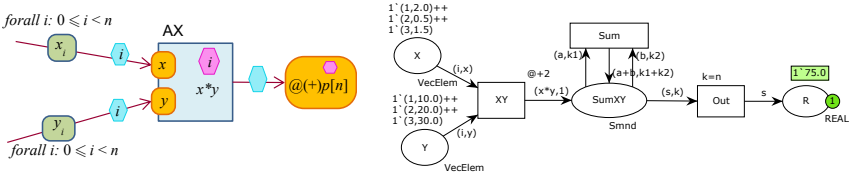


Рисунок 3. Примеры простых программ на UPL (слева) и их «перевод» в CPN (справа)

на UPL, справа – ее перевод на CPN. В обоих вариантах токены содержат только скалярные данные, поэтому векторы и матрицы представляются как множества токенов, содержащих значения и индексы.

Все схемы являются не законченными программами, а их фрагментами. Поэтому в схемах на UPL входные стрелки имеют свободное начало, а выражения на них условны. По некоторым из них придут множество токенов, что отражено в комментарии forall. (Вообще, все надписи вне объектов в UPL считаются комментариями в UPL, включая имена узлов). А выходные стрелки фрагментов упираются в условные одновходовые транзитные узлы, изображаемые как «вход без узла». В CPN этой «проблемы» нет, поскольку дуги всегда идут либо из места в переход, либо из перехода в место, и все необходимые интерфейсные места на схемах присутствуют.

Далее приводятся пояснения по каждому фрагменту.

(*1*) **Умножение n-вектора X на число a**

UPL. На вход `x` узла `AX` приходят элементы вектора (в количестве n штук), а на вход `a` – один токен с общим множителем, индексом $\langle$`*`$\rangle$ и кратностью n. (Здесь и далее `n` – константа, определенная глобально). В узле выполняется операция умножения множителя `a` на каждый отдельный элемент `x`. Произведение посылается как значение на выход (здесь оно подразумевается по умолчанию) с индексом `j`. (Здесь индекс на выходной стрелке тоже можно было бы не писать, поскольку он совпадает текстуально с индексом целевого узла `R`). Это простой групповой узел.

CPN. Входные места `Xj` и `a` принимают, соответственно, набор пар `(j,x)` и множитель с кратностью, равной количеству этих пар. Каждая пара создает срабатывание, от которого на выход идет пара `(j, a*x)`.

(*2*) **Умножение (внешнее) n-вектора X на n-вектор Y**

UPL. На оба входа поступают элементы векторов, причем кратность каждого элемента равна (или больше) числу элементов на другом входе. Согласно правилам происходит взаимодействие по типу «каждый с каждым». Обратите внимание на «*» в индексах. В узле аргументы `x` и `y` перемножаются, и выдаются произведения с «приклеенными» двумя индексами. Это двойной групповой узел. Вектора могут быть разреженными, важно только, чтобы кратности соответствовали числу токенов на противоположном входе.

CPN. Если выполнить перевод по обычным правилам, ничего хорошего не получится. Поскольку на обоих входных местах присутствуют кратные элементы, то ничто не мешает срабатывание

с ними повторить несколько раз, в пределах меньшей из кратностей. В этом проявляется различие семантик кратности: в CPN она равносильна соответствующему количеству копий токена, а в UPL токен с кратностью это один токен, но с атрибутом кратности. Правила взаимодействия в UPL запрещают повторную активацию одной и той же комбинации токенов. В логике CPN такого нет и быть не может, поскольку кратность есть просто сокращенное обозначение для нескольких одинаковых токенов. Приходится признать, что в CPN отсутствует непосредственный аналог двойным групповым узлам, имеющим большое прикладное значение (умножением матриц, задача «N тел» и т.п.). Проблему их моделирования в CPN будем рассматривать в другой статье.

(*3*) **Сложение плотных векторов** UPL. Здесь так же на оба входа приходят вектора, но, во-первых, их размеры и множества индексов должны совпадать, а, во-вторых, срабатывают только пары с равными индексами. И не важно, что на стрелке индекс токена `y` назван `j` (это просто выражение, вычисляемое в контексте узла-отправителя), важно, что он стоит в нужной позиции, которая в целевом узле имеет имя `i`. Важно также, что оба вектора плотные: если какой-то элемент в одном векторе будет отсутствовать, то соответствующий элемент «зависнет» на другом входе.

CPN. Перевод прямолинейный и адекватный. Обращаем внимание на одинаковые индексы `i` на обоих входных стрелках.

(*4*) **Скалярное произведение плотных векторов**

UPL. Здесь также на входах два плотных вектора с одинаковым числом элементов, но их соответственные элементы теперь перемножаются и произведения передаются на суммирующий узел `p` без индекса. Префикс имени `@(+)` указывает на использование редукции сложением. Суффикс имени `[n]` говорит, что должно прийти ровно n слагаемых, после чего узел сработает. Здесь `n` – это глобальная константа (также может стоять число). Переменная в квадратных скобках также может быть именем другого входа – тогда число ожидаемых слагаемых определяется динамически. Ниже будет пример такого использования.

CPN. Здесь пришлось описать конкретный способ накопления суммы. Произведения (слагаемые) `s` посылаются на место `SumXY` в виде пар `(s,k)` с приклеенным счетчиком $k = 1$. В общем случае пара `(s,k)` означает, что `s` является суммой `k` слагаемых. Если в данном месте появляются два токена такого вида, то они покомпонентно складываются (переход `Sum`) и сумма возвращается обратно на то же

место. Когда счетчик станет равен n, сработает переход `Out`, который положит итоговую сумму на место `R`.

Все рассмотренные примеры схем на языке CPN были проиграны в *CPN Tools*. Начальные данные можно видеть непосредственно над входными местами, а результаты – в зеленых прямоугольниках возле выходных мест. К сожалению, непосредственной реализации языка UPL в настоящее время нет. (Но есть эмулятор предшествующего языка DFL с ограниченными возможностями, реализованный на модели мультипроцессора ППВС «БУРАН» [11]).

Интересно выразить пример 4 на UPL без использования суммирующего входа. К сожалению, сделать это по аналогии с вариантом на CPN не получится, поскольку там одно место `SumXY` используется двумя переходами, а переходом Sum даже дважды. К тому же второй переход `Out` имеет охрану общего вида `(k=n)`, а это сравнение переменной и константы, а не между переменными. Самый простой вариант изображен на рисунке 4. Здесь пришлось дополнительно посылать начальный токен

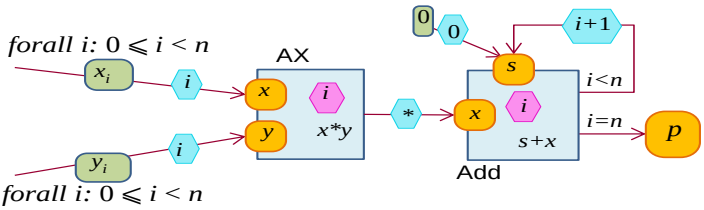


Рисунок 4. Вариант примера 4 без собирающих узлов

$0\langle 0\rangle$ на вход s, куда также подается каждая следующая сумма $(s+x)\langle i+1\rangle$ при $i < n$. Та же сумма при $i = n$ посылается на выходной порт `p` (без индекса). Заметим, что слагаемые подаются с индексом $\langle * \rangle$, что ведет к недетерминированному порядку суммирования.

4. Пример: решение треугольной СЛАУ

Теперь рассмотрим более цельный пример алгоритма, а именно – решатель треугольных СЛАУ с разреженной матрицей. Он может применяться как ядро итерационного решателя СЛАУ общего вида методом Гаусса-Зейделя. Алгоритм ядра решателя [12] показан на рисунке 5.

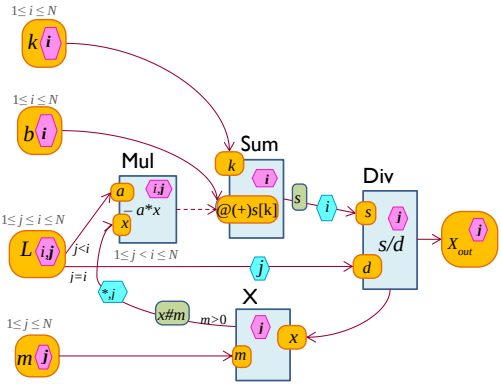


Рисунок 5. Алгоритм решения треугольной СЛАУ на UPL

Требуется вычислить вектор x из уравнения $Lx = b$, где матрица L – нижнетреугольная, то есть все ее элементы выше диагонали нулевые. При этом все диагональные элементы L_{ii} ненулевые. Решение вычисляется по формуле (1):

$$x_i = (b_i - \sum_{j=1}^{i-1} x_j L_{ij})/L_{ii} \qquad \text{для } i \text{ от 1 до } N \tag{1}$$

Согласно букве формулы (1) вектор X вычисляется через себя. Однако, благодаря треугольности матрицы L зависимости между отдельными элементами ациклические. Матрица L может быть разреженной, и даже не быть треугольной, но сводимой к треугольной при некоторой синхронной перестановке строк и столбцов.

Алгоритм, изображенный на рисунке 5, реализует формулу (1) один к одному, не считая замены вычитания сложением. Каждая операция (`+`, `*`, `/`) становится самостоятельным узлом. Еще один узел (`X`) принимает вычисленное значение x_i и присоединяет к нему нужную кратность. Входные данные:

- k_i – число ненулевых элементов в i-й строке;
- b_i – элемент вектора правой части (плотного);
- L_{ij} – ненулевой элемент матрицы;
- m_j – число ненулевых элементов в j-м столбце без диагонального.

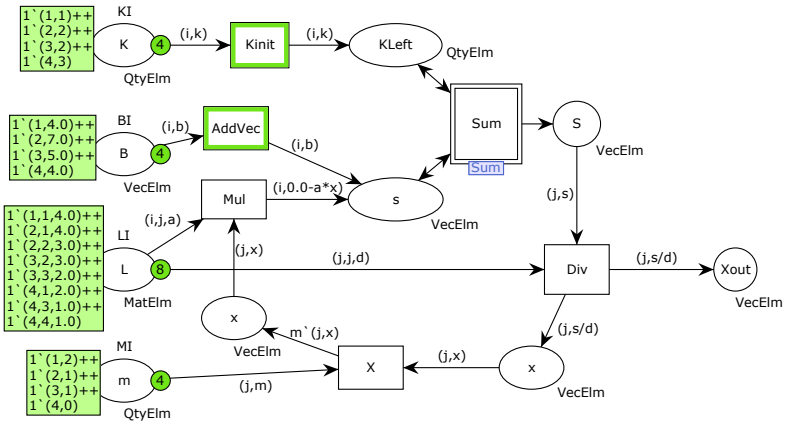
Суммирование производится собирающим входом `@(+)s[k]` узла `Sum`. Число слагаемых k приходит на другой вход этого же узла. Здесь оно всегда положительное, поскольку включает обязательный диагональный

элемент. (Если бы пришло нулевое k, то узел Sum сразу выдал бы нулевой результат, не дожидаясь и не используя слагаемых `s`).

После деления суммы на диагональный элемент узлом `Div` частное x_i посылается на узел `X`. Тот приклеивает к нему кратность $\#m_i$ и отправляет на умножитель `Mul` со звездочкой в позиции индекса i, то есть на весь столбец j. Если $m = 0$, токен не посылается.

Во всех узлах индекс i — номер строки — служит разделяющим контекстом, благодаря которому вычисления для разных строк происходят параллельно и не мешают друг другу.

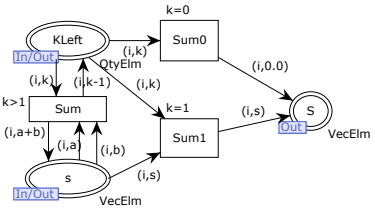
Перевод в CPN, показанный на рисунке 6, имеет свои особенности.



(*а*) верхний уровень

```
colset MatElm = product INT*INT*REAL;
colset QtyElm = product INT*INT;
colset VecElm = product INT*REAL;
var i,j,k,m:INT;
var a,b,d,x,s:REAL;
val BI=1'(1,4.0)++1'(2,7.0)++1'(3,5.0)++1'(4,4.0);
val KI=1'(1,1)++1'(2,2)++1'(3,2)++1'(4,3);
val LI=1'(1,1,4.0)++ 1'(2,1,4.0)++1'(2,2,3.0)++...
val MI=1'(1,2)++1'(2,1)++1'(3,1)++1'(4,0);
```

(*б*) декларации переменных, цветов и констант



(*в*) подсеть суммирования, задающая переход-подстановку Sum

Рисунок 6. Иерархическая CPN-сеть решения треугольной СЛАУ

Если в UPL выделение диагональных элементов реализуется условием `(j=i)` у основания стрелки как частью программы узла, то в CPN оно задается образцом `(j,j,d)` и реализуется как часть механизма

срабатывания перехода Div. На другой стрелке, исходящей из того же места L, образец (i,j,a) не накладывает условия $i \neq j$, но оно тут и не требуется, поскольку токен (j,x) в месте X (также входном для перехода Mul) появится только после удаления диагонального элемента (j, j, d).

Узел суммирования Sum переводится в CPN как переход-подстановка Sum. Он раскрывается статически в одноименную подсеть, описанную отдельно. Подсеть сообщается с основной сетью через свои интерфейсные места Kleft, s и S, из которых первые два входно-выходные, последнее – чисто выходное. Соответственно, в подсети имеются дуги, ведущие из первых двух мест, и дуги, ведущие во все три места. Подсеть реализует конкретный способ суммирования, примерно как в примере 4 выше. Отличие в том, что число слагаемых поступает динамически. Оно приходит на место Kleft, в котором всегда будет находиться число оставшихся слагаемых. Слагаемые приходят в место s. Всякий раз, когда в нем обнаруживаются два слагаемых (с общим i), они складываются переходом Sum и сумма возвращается туда же. Одновременно из Kleft вычитается 1. Когда Kleft достигнет 1, сработает переход Sum1, передающий результат из s на S. Также предусмотрено, что если придет $Kleft = 0$, то переход Sum0 отреагирует выдачей суммы 0.0. Строго говоря, чтобы передать точную семантику суммирующего узла UPL, переходы Sum1 и Sum0 должны иметь более высокий (по сравнению с Sum) приоритет (а такая возможность в CPN есть), но здесь, если все данные корректны, в этом нет необходимости. Перевод узлов Div и X прямолинеен: m также становится кратностью значения (j,x). И хорошо, что на другом входе узла Mul в UPL находятся токены (элементы матрицы L_{ij}) с полным индексом, и не возникает ситуации двойного группового узла.

5. Сходства и различия UPL и CPN

Узлы UPL соответствуют переходам в CPN, входы узлов – местам, токены (стрелки) – выходным дугам переходов. Как и в CPN, на входах токены могут накапливаться в виде мультимножеств. Конкретные узлы и активные комбинации соответствуют связываниям в CPN.

Есть сходство и в способах нахождения активных наборов. В обоих случаях имеется некоторое множество векторов вида $I^m = \langle i_1, \ldots, i_m \rangle$, из которых нужно отобрать набор из k векторов, причем условием для образования набора является равенство некоторых компонентов между собой. Тем самым для поиска наборов требуется сопоставление и сравнение по содержимому. Правда, есть и отличие. В CPN могут использоваться вектора разного типа, размера (в разных местах) и сравниваться могут компоненты разных позиций (как одного вектора, так и разных). Это задается выражениями-образцами, а равенство компонент определяют

одинаковые имена. А в UPL вектора (индексы) в одном наборе должны быть одного размера, и на равенство проверяются компоненты разных векторов, находящиеся в одной позиции. При этом в самих векторах указывается для каждой позиции, используется ли данный компонент в сравнении.

Есть и другие отличия. В CPN при переходах может быть охрана — условие произвольного вида, проверяемое в рамках операции срабатывания. В UPL такие проверки могут производиться только в рамках программы узла, выполняемой после срабатывания АК, но не в рамках самого срабатывания.

Есть довольно тонкое семантическое различие в подборе и срабатывании группы АК в UPL и шага в CPN. В CPN токены, образованные на предыдущем срабатывании считаются сразу записанными в целевые места. При наличии тайминга (задержек на дугах или на переходах) это откладывается на некоторое время. Вообще, наличие времени говорит о назначении CPN как средства моделирования поведения. В UPL времени нет, поскольку это средство записи вычислительных алгоритмов. О времени можно говорить только в рамках моделирования работы «реальной» UPL-машины. Семантика самого UPL утверждает, что никакой токен не может быть доставлен мгновенно, какое-то положительное время это обязательно займет. Поэтому семантика UPL рассматривает состояние не как одно, а как два (мульти) множества токенов: входной буфер (ВБ) и рабочее пространство (РП). ВБ есть абстракция токенов, находящихся в движении к своей цели. РП — это токены, достигшие своих мест назначения на входах узлов. Все токены АК, готовой к срабатыванию, должны быть в РП. В теории из ВБ токены по-одному передаются в РП. И после каждой такой передачи ищутся все возникшие АК и срабатывают. Перед приемом нового токена из ВБ в РП (по крайней мере в теории) не должно оставаться АК.

Выше (раздел 2.2) мы назвали это принципом последовательной активации (ППА). Именно он позволяет без проблем обеспечивать соблюдение принципа однократной активации (ПОА) в ситуации, когда все токены АК имеют кратности больше 1. Действительно, по приходу одного из таких токенов, машина проводит поиск АК только с участием нового токена, и в рамках этого поиска каждая найденная АК срабатывает только один раз, каждый раз уменьшая кратности на 1. Работа с новым токеном прекращается, если его кратность упадет до 0. И только после срабатывания всех найденных АК новый токен заносится в РП с кратностью, уменьшенной на число срабатываний. Понятно, что при таком механизме, АК из тех же токенов уже не может сработать вновь, что и обеспечивает принцип однократности почти «бесплатно». При этом

важно, что токен с уменьшенной кратностью считается «тем же самым» токеном. И это радикально отличает ситуацию токена с кратностью от ситуации с заданным числом копий однократного токена. В CPN этого семантического различия нет, и потому нам приходится накладывать ограничение, чтобы хотя бы на один из входов токены всегда приходили с кратностью 1. А это не позволяет проводить взаимодействия по принципу «каждый с каждым».

Принцип последовательной активации кажется противоречащим параллелизму и распределенности. Однако, это не так. ППА соблюдается как принцип теории. На практике же виртуальное адресное пространство (ВАП) конкретных узлов может быть разбито на независимые части (с некоторыми оговорками) так, что каждый токен направляется в свою часть РП, локализованную в своем физическом процессоре (ядре). И хотя внутри каждого ядра поступающие токены обрабатываются строго последовательно по одному, разные ядра работают независимо и параллельно. В следующем разделе подробно описывается, как это достигается.

6. Функция распределения

В UPL глобальное адресное пространство определяется программой. В нем виртуальным адресом является пара (имя узла, конкретный индекс). Для распределения вычислений по вычислительным ядрам пользователь задает функцию распределения (ФР), которая отображает виртуальные адреса в номера ядер. ФР однозначно указывает для каждого конкретного узла номер ядра, в котором должно произойти его срабатывание.

Распределение конкретных узлов индуцирует соответствующее распределение токенов. Поэтому каждый токен, будучи создан в каком-либо ядре, направляется через коммуникационную сеть непосредственно в то ядро, где будут производиться срабатывания с его участием. Номер ядра определяется функцией распределения, которая вычисляется непосредственно после создания токена перед выдачей его в сеть. Будем называть его целевым адресом (токена) и обозначать как $F(N, I)$ или $F_N(I)$, где N – имя узла, а I – индекс.

Чтобы целевой адрес токена $F_N(I)$ был всегда однозначным, следует избегать использования в качестве аргумента функции F тех компонент индекса, которые в некоторых токенах, направляемых на узел `N` заданы как «*». (В противном случае значением функции F для такого токена будет *множество* номеров ядер, представленное битовой строкой в алфавите $\{0, 1, *\}$ [**13**]. Так, для решателя треугольной СЛАУ на рисунке 5 будет разумным использовать для всех узлов индекс `j` (а если его нет, как

у узла `Sum`, то `i`): на схеме он всюду выделен жирным шрифтом. Это будет разбиение «по столбцам».

В работе [12] эта задача рассматривается в специальной постановке, когда как номер уравнения i, так и номер переменной j являются трехмерными координатами элементов сетки (x, y, z). При этом коэффициенты матрицы L ненулевые только для тех пар (i, j), которые соответствуют соседним ячейкам сетки (отличающимся не более чем на 1 по каждой координате). Там была предложена следующая функция распределения $F : (x, y, z) \to (p_1, p_2)$:

```
F(x,y,z) =
  let
    val X =(x+y+2*z)/d
    val Y = (y+z)/d
    val Z = z/d
  in
    ( (K1*d*(X-Y-Z)/nx) mod K1,
      (K2*d*(Y-Z)/ny) mod K2 )
```

где `K1`, `K2`, `d`, `nx`, `ny` – константы, а `X`, `Y`, `Z` – блочные координаты. Она порождает нетривиальное разбиение адресного пространства на блоки, являющиеся косыми параллелепипедами, и последующее распределение блоков между ядрами. Оно оказалось выигрышным с точки зрения длины критического пути (цепочки зависимых передач между ядрами) и объема коммуникаций по сравнению с более простым распределением на прямоугольные блоки.

Здесь мы привели этот пример лишь как иллюстрацию того, что распределение вычислений между ядрами можно изменять в широких пределах, просто варьируя формулы функций распределения. Важно, что ФР задается отдельно от основного тела программы на UPL и не может «испортить» ее результат (если, конечно, программа правильная, и ФР корректна [13]), а может повлиять только на время ее работы. Выбор ФР, как правило, имеет целью улучшить следующие показатели работы программы:

(*1*) Равномерность вычислительной нагрузки на ядра.

(*2*) Объем коммуникационной нагрузки на сеть (на разных уровнях ее иерархии) и ее равномерность во времени.

(*3*) Длина критического пути (задержка на цепочках передач между ядрами, где каждая следующая передача зависит от предыдущей).

Улучшение этих показателей может способствовать существенному повышению эффективности работы программы на распределенной системе. Что касается возможности распределенного выполнения CPN, пока эта проблематика не исследована. Есть много работ по проблеме

распределенной реализации обычных (не цветных) сетей Петри, причем почти всегда они либо очень сложны, либо не полны (не для любых сетей). Есть много статей о применении CPN для моделирования распределенных систем (фактически таковы почти все применения), но о распределенной реализации самого CPN речи нигде нет. Это вообще вряд ли возможно, если не накладывать какие-либо ограничения. Фактически в данной статье мы и накладываем ограничения (A) – (D), говоря о преобразовании из CPN в UPL. И, как показывает наш опыт использования UPL, они не создают проблем (кроме D) при описании вычислительных алгоритмов. Возможно, некоторые можно и не накладывать. Можно надеяться, что техника распределения UPL посредством функций распределения может быть адаптирована и к CPN при каких-то, возможно более слабых, ограничениях. Это требует дальнейшего изучения.

7. Близкие работы

Есть очень много работ по применению CPN для моделирования и анализа различных распределенных конкурентных (concurrent) систем. Однако, среди них почти не видно работ по применению CPN к анализу алгоритмов, тем более вычислительного типа. Редкое исключение – работа [14]. В ней CPN используется как инструмент моделирования параллельных алгоритмов, написанных в парадигме нескольких последовательных процессов, выполняемых на общей памяти с использованием средств обеспечения взаимного исключения. Из алгоритма извлекается CPN-сеть, моделирующая как последовательное выполнение каждого процесса, так и разделяемый доступ к общим ресурсам. Возможны разные степени абстракции, вплоть до полной симуляции вычислений. Исследуется пространство состояний моделей для обнаружения нежелательных состояний.

Наша работа отличается использованием языка описания алгоритмов, который по духу близок к CPN, и поэтому мы используем CPN не столько как средство анализа с целью верификации на разных уровнях абстракции (хотя это тоже возможно и полезно), сколько как средство симуляции и имплементации самого UPL. Также нам интересно сопоставление выразительных возможностей этих двух языков как средств описания вычислительных алгоритмов.

Заключение

Были даны определения двух систем, CPN и UPL, достаточно точные для их сопоставления и сравнения как средств описания параллельных вычислительных алгоритмов. По-видимому, эта область применения,

в отличие от моделирования распределенных систем, налагает свои особые требования. В частности, полезна особая семантика кратности, допускающая взаимодействие множеств токенов по типу «каждый с каждым», а также специальные собирающие (в частности, суммирующие) места или входы. С другой стороны, некоторые черты CPN востребованы слабо: охраны общего вида, возможность иметь несколько конфликтующих стрелок из одного места.

Для обоих языков является удобным и наглядным графическое представление в виде графа, в котором блоки представляют действия (в CPN переходы, в UPL узлы), овалы – данные, а стрелки передачу данных. При этом в UPL входные места (порты) «приклеены» к узлам–действиям, что, на наш взгляд, отражает нужды языка вычислений, улучшая наглядность и снижая запутанность. Это также согласуется с тем, что, в отличие от CPN, в UPL каждое входное место принадлежит своему узлу-действию.

Семантика обоих языков основана на актах срабатывания действий, обусловленных готовностью для них входных данных и ничем иным. Одни действия создают данные, потребляемые другими действиями. Так возникает информационный граф между действиями. В литературе по CPN его называют occurrence graph. Ему близко понятие информационного графа алгоритма, введенного в [**15**]. В UPL это трасса вычислительного процесса, которая может быть визуализирована в виде графа (для отладки и анализа).

Особенностью UPL является разбиение входных данных узла на собственно данные и индекс, который присутствует, целиком или частично, в каждом входном токене. Только индекс используется при сопоставлении токенов с разных входов, и именно по индексу осуществляется распределение вычислений по «пространству». Для этой цели в распределенную (многопроцессорную) вычислительную систему (РВС), исполняющую UPL-программу, вводится возможность задавать, отдельно от программы, функцию распределения, зависящую от пары (имя узла, индекс), и вырабатывающую номер процессорного ядра РВС. Можно так же организовать и распределение по времени, способствующее уменьшению заполнения сопоставляющей памяти токенов. Вряд ли подобное можно ввести в CPN без наложения ограничений типа тех, что отличают UPL от CPN.

К сожалению, язык UPL пока не реализован. Имеются только реализации первоначального языка DFL, из которого UPL вырос как его обобщение. Все примеры пока просто набирались в PowerPoint, а для выполнения переводились вручную в DFL. Хотелось бы иметь аналог CPN Tools с графической формой входного языка для разработки и выполнения алгоритмов на UPL или близком к нему языке. Возможно

появление обобщенной рамочной системы графического программирования, частными случаями которой были бы как CPN, так и UPL.

В данной статье были рассмотрены только базовые средства языка UPL. Следующие элементы были только упомянуты и будут подробнее освещены в будущей статье (статьях):

(*1*) Узлы с ветвями, реализующие конфликтные входы в рамках одного узла [**7**, **10**].
(*2*) Распределенные групповые узлы [**14**] в UPL.
(*3*) Моделирование двойных групповых узлов в CPN.
(*4*) Рекурсия и мемоизация.
(*5*) Распознавание «тишины».

Список литературы

[1] Petri C.A. *Kommunikation mit Automaten*, PhD thesis.– University of Bonn.– 1962.– 128 pp. URL ↑92
[2] Котов В. Е. *Сети Петри.*– М.: Наука.– 1984.– 160 с. ↑92
[3] Orlov S. P., Susarev S. V., Uchaikin R. A. *Application of hierarchical colored Petri nets for technological facilities' maintenance process evaluation* // Applied Sciences.– 2021.– Vol. **11**.– No. 11.– id. 5100.– 26 pp. doi ↑92
[4] Shapiro R. M. *Validation of a VLSI chip using hierarchical coloured Petri nets* // Microelectronics Reliability.– 1991.– Vol. **31**.– No. 4.– Pp. 607–625. doi ↑92
[5] Jitmit C., Vatanawood W. *Simulating artificial neural network using hierarchical coloured Petri nets* // *Proceedings of the 2021 6th International Conference on Machine Learning Technologies*, ICMLT 2021 (Jeju Island Republic of Korea, April 23–25, 2021), New York: ACM.– 2021.– ISBN 978-1-4503-8940-2.– Pp. 127–131. doi ↑92
[6] K. Jensen *Coloured Petri nets: A high level language for system design and analysis* // *Advances in Petri Nets 1990*, ICATPN 1989, Lecture Notes in Computer Science.– vol. **483**, Berlin–Heidelberg: Springer.– 1991.– ISBN 978-3-540-53863-9.– Pp. 342–416. doi ↑92, 95, 96
[7] Климов А. В., Окунев А. С. *Графический потоковый метаязык для асинхронного распределенного программирования*, МЭС-2016 (Россия, Москва, октябрь 2016), Проблемы разработки перспективных микро- и наноэлектронных систем.– №2, М.: ИППМ РАН.– 2016.– С. 151–158. URL ↑92, 118
[8] Климов А. В. *О парадигме универсального языка параллельного программирования* // *Языки программирования и компиляторы-2017*, PLC-2017 (Южный федеральный университет, Институт математики, механики и компьютерных наук им. И. И. Воровича, 3–5 апреля 2017), Ростов-на-Дону: ЮФУ.– 2017.– ISBN 978-5-9275-2349-8.– С. 141–146. URL ↑92
[9] Harper R. *Programming in Standard ML.*– Carnegie Mellon University.– 2011.– 297 pp. URL ↑95

[10] Климов А. В., Левченко Н. Н. *Механизм ветвей в потоковом метаязыке UPL (METAL) и методы его реализации в ППВС «БУРАН»*, МЭС-2018 (Россия, Москва, октябрь 2018), Проблемы разработки перспективных микро- и наноэлектронных систем.– №3, М.: ИППМ РАН.– 2018.– С. 31–37. ↑118

[11] Климов А. В., Левченко Н. Н., Окунев А. С., Стемпковский А. Л. *Вопросы применения и реализации потоковой модели вычислений*, МЭС-2016 (Россия, Москва, октябрь 2016), Проблемы разработки перспективных микро- и наноэлектронных систем.– №2, М.: ИППМ РАН.– 2016.– С. 100–106. ↑100, 109

[12] Змеев Д. Н., Климов А. В., Окунев А. С., Левченко Н. Н. *Особенности реализации теста HPCG для ППВС «БУРАН»* // *XXII Харитоновские тематические научные чтения* (онлайн, 24-27 мая 2021), Саров: РФЯЦ–ВНИИЭФ.– 2022.– ISBN 978-5-9515-0507-1.– С. 193–205. ↑109, 115

[13] Климов А. В. *Средства верификации распределения вычислений в потоковой архитектуре ППВС «Буран»*, МЭС-2020 (Россия, Москва, октябрь 2020), Проблемы разработки перспективных микро- и наноэлектронных систем.– №4, М.: ИППМ РАН.– 2020.– С. 236–243. ↑114, 115

[14] Westergaard M. *Towards verifying parallel algorithms and programs using coloured Petri nets*, PNSE'2011 (Newcastle upon Tyne, UK, June 20-21, 2011), CEUR Workshop Proceedings.– vol. **723**.– 2011.– Pp. 57–71. ↑116, 118

[15] Воеводин В. В., Воеводин Вл. В. *Параллельные вычисления.*– СПб: БХВ-Петербург.– 2004.– ISBN 5-94157-160-7.– 599 с. ↑117

Поступила в редакцию 24.10.2023;
одобрена после рецензирования 26.11.2023;
принята к публикации 26.11.2023;
опубликована онлайн 14.12.2023.

Рекомендовал к публикации *д.ф.-м.н. Н. Н. Непейвода*

Информация об авторе:

Аркадий Валентинович Климов

ст. научн. сотр. Института проблем проектирования в микроэлектронике РАН. Научные интересы: нетрадиционные модели параллельных вычислений

0000-0002-7030-1517

e-mail: *arkady.klimov@gmail.com*

Автор заявляет об отсутствии конфликта интересов.

PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article HARDWARE AND SOFTWARE FOR DISTRIBUTED AND SUPERCOMPUTER SYSTEMS

UDC 004.432.4
10.25209/2079-3316-2023-14-4-91-122

Coloured petri nets and the language for distributed programming UPL: their comparison and translation

Arkady Valentinovich **Klimov**
Institute for Design Problems in Microelectronics
✉*arkady.klimov@gmail.com*

Abstract. Petri nets are widely used as a means of modeling distributed multi-agent systems. There are tools for working with extended Petri nets, in which tokens are loaded with arbitrary data. For example, CPN Tools allows you to describe, play and explore Colored Petri Nets (CPN). The question is raised about the possibility of using this tool for the development, prototyping and research of parallel distributed computing algorithms, ideally turning them into working efficient parallel programs. We have experience in experimental programming of various algorithms in the graphical data flow language UPL, which currently exists "on paper". Its comparison with CPN shows that their semantics have a lot in common. In the article, both languages are defined, compared with examples and through the rules of translation from one to another. The means for defining distribution of computations (distribution functions) are also described. An interesting question is about their transfer to CPN, where they have no analogues yet. (*In Russian*).

Key words and phrases: Petri nets, Coloured Petri Nets, parallel programming, dataflow computation model, algorithm graph, visual programming, UPL language, distribution function

2020 *Mathematics Subject Classification*: 68N15, 68N19; 97P40

Acknowledgments: The work is supported by Institute for Design Problems in Microelectronics

For citation: Arkady V. Klimov. *Coloured petri nets and the language for distributed programming UPL: their comparison and translation*. Program Systems: Theory and Applications, 2023, **14**:4(59), pp. 91–122. (*In Russ.*). https://psta.psiras.ru/read/psta2023_4_91-122.pdf

References

[1] C.A. Petri. *Kommunikation mit Automaten*, PhD thesis, University of Bonn, 1962, 128 pp.

[2] V. E. Kotov. *Petri Nets*, Nauka, M., 1984 (in Russian), 160 pp.

[3] S. P. Orlov, S. V. Susarev, R. A. Uchaikin. "Application of hierarchical colored Petri nets for technological facilities' maintenance process evaluation", *Applied Sciences*, **11**:11 (2021), id. 5100, 26 pp.

[4] R. M. Shapiro. "Validation of a VLSI chip using hierarchical coloured Petri nets", *Microelectronics Reliability*, **31**:4 (1991), pp. 607–625.

[5] C. Jitmit, W. Vatanawood. "Simulating artificial neural network using hierarchical coloured Petri nets", *Proceedings of the 2021 6th International Conference on Machine Learning Technologies*, ICMLT 2021 (Jeju Island Republic of Korea, April 23–25, 2021), ACM, New York, 2021, ISBN 978-1-4503-8940-2, pp. 127–131.

[6] Jensen K.. "Coloured Petri nets: A high level language for system design and analysis", *Advances in Petri Nets 1990*, ICATPN 1989, Lecture Notes in Computer Science, vol. **483**, Springer, Berlin–Heidelberg, 1991, ISBN 978-3-540-53863-9, pp. 342–416.

[7] A. V. Klimov, A. S. Okunev. "A graphical dataflow meta-language for asynchronous distributed programming", MES-2016 (Rossiya, Moskva, oktyabr' 2016), Problemy razrabotki perspektivnyx mikro- i nanoelektronnyx sistem, 2, IPPM RAN, M., 2016, pp. 151–158 (in Russian).

[8] A. V. Klimov. "On the paradigm of a universal parallel programing language", *Yazyki programmirovaniya i kompilyatory-2017*, PLC-2017 (Yuzhnyj federal'nyj universitet, Institut matematiki, mexaniki i komp'yuternyx nauk im. I. I. Vorovicha, 3–5 aprelya 2017), YuFU, Rostov-na-Donu, 2017, ISBN 978-5-9275-2349-8, pp. 141–146 (in Russian).

[9] R. Harper. *Programming in Standard ML*, Carnegie Mellon University, 2011, 297 pp.

[10] A. V. Klimov, N. N. Levchenko. "Branches in the Dataflow Metalanguage UPL (METAL) and Methods of their Implementation in the PDCS 'Buran'", MES-2018 (Rossiya, Moskva, oktyabr' 2018), Problemy razrabotki perspektivnyx mikro- i nanoelektronnyx sistem, 3, IPPM RAN, M., 2018, pp. 31–37 (in Russian).

[11] A. V. Klimov, N. N. Levchenko, A. S. Okunev, A. L. Stempkovskij. "The application and implementation issues of dataflow computing system", MES-2016 (Rossiya, Moskva, oktyabr' 2016), Problemy razrabotki perspektivnyx mikro- i nanoelektronnyx sistem, 2, IPPM RAN, M., 2016, pp. 100–106 (in Russian).

[12] D. N. Zmeev, A. V. Klimov, A. S. Okunev, N. N. Levchenko. "Features of HPCG benchmark implementation for the "BURAN" PDCS", *XXII Xaritonovskie tematicheskie nauchnye chteniya* (onlajn, 24-27 maya 2021), RFYaCz–VNIIEF, Sarov, 2022, ISBN 978-5-9515-0507-1, pp. 193–205 (in Russian).

[13] A. V. Klimov. "Tools for Verification of Computation Distribution in the Parallel Dataflow Computing System (PDCS) 'Buran'", MES-2020 (Rossiya, Moskva, oktyabr' 2020), Problemy razrabotki perspektivnyx mikro- i nanoelektronnyx sistem, 4, IPPM RAN, M., 2020, pp. 236–243 (in Russian). doi URL

[14] M. Westergaard. "Towards verifying parallel algorithms and programs using coloured Petri nets", PNSE'2011 (Newcastle upon Tyne, UK, June 20-21, 2011), CEUR Workshop Proceedings, vol. **723**, 2011, pp. 57–71. URL

[15] V. V. Voevodin, Vl. V. Voevodin. *Parallel Computations*, BXV-Peterburg, SPb, 2004, ISBN 5-94157-160-7 (in Russian), 599 pp.

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ПРИКЛАДНЫЕ ПРОГРАММНЫЕ СИСТЕМЫ

УДК 004.946:793.7
10.25209/2079-3316-2023-14-4-123-140

Генерация сюжетной составляющей для настольной ролевой игры

Диана Антоновна **Челышева**[1]✉, Аркадий Викторович **Ключиков**[2]
[1] СГТУ им. Ю.А. Гагарина, Саратов, Россия
[2] ФГБОУ ВО Вавиловский университет, Саратов, Россия
[1]✉*diana.chelysheva.98@mail.ru*

Аннотация. Проведён анализ предметной области и актуальности настольных ролевых игр, а также их влияние на сферы деятельности человека. Изучена механика игрового процесса настольной ролевой игры Подземелья и драконы. Рассмотрены программные решения для создания сюжетного описания локаций, в том числе графическое представление карт и одностраничные приключения. Реализованы дизайн и адаптивный пользовательский интерфейс. Сформирована архитектура базы данных, а также реализована в формате JSON-файлов. Разработана архитектура модели, а также произведена её настройка. Описан блочный способ построения сюжета. Разработан алгоритм работы приложения по генерации сюжетной составляющей. Сформированы списки сюжетных составляющих, а также реализована логика случайного выбора участков описания с проверкой на адекватность и сочетаемость описания.

Ключевые слова и фразы: генерация сюжета, настольная ролевая игра, «Подземелья и Драконы», мобильное приложение, пользовательский интерфейс, блочный способ построения сюжета, база данных

Для цитирования: Челышева Д. А., Ключиков А. В. *Генерация сюжетной составляющей для настольной ролевой игры* // Программные системы: теория и приложения. 2023. **Т. 14**. № 4(59). С. 123–140. https://psta.psiras.ru/read/psta2023_4_123-140.pdf

Введение

Популярность настольных игр на сегодняшний день очень высока [1]. По данным игрового сайта Roll20, в разные настольные ролевые игры (НРИ) [2] играет 70 процентов пользователей. В НРИ участники устно описывают действия своих персонажей, основываясь на их особенностях. Самой распространённой среди таких игр является Dungeons Dragons (DND). По данным сайта Dungeon Vault, DND является популярной и востребованной, её сообщество насчитывает 13.7 млн. игроков по всему миру. В области НРИ ведутся разработки по графической интерпретации локаций и одностраничные приключения [3].

При написании игровых сценариев возникает ряд трудностей, замедляющих процесс повествования. Среди них — проблема выбора места, времени и условия действия, поскольку игра является вариативной в сюжетной составляющей, а порядок действий случаен. Эти аспекты усложняют задачу игрового мастера.

Рассмотрим сценарный подход с графической интерпретацией карт. Мастер игры описывает свой сценарий с использованием нарисованной карты, где изображены ключевые точки сюжетных событий.

Из видов карт выделяют разделенные на:

- этажи;
- комнаты (рисунок 1).

Одностраничные приключения создаются для личного использования мастером игры. Такое представление информации позволяет повысить вариативность сюжетных линий и визуализировать сюжет для быстрой ориентации в нем.

Чтобы упростить процесс генерации сюжетных аспектов, разработчики в сообществе любителей НРИ создали инструменты автоматической генерации. В настоящий момент существуют приложения и веб-ресурсы генерирующие сюжетные элементы для игр [4], чтобы сократить время на подготовку для игрового мастера.

Современные решения не способны генерировать описание локации совместно с функциями настройки сеттинга, местности, ключевых персонажей и предметов. В связи с этим предлагается разработать приложение для создания сюжетной составляющей игры. Среди доступной информации пользователю предоставляется описание: персонажей (героев), квестов и добычи, а также изображения описываемой территории, где текст генерируется блоками.

В липких лапах М'куса

Прохладная завязка:

На обычно безопасной дороге жарким солнечным утром 1к6 гоблинов из клана "Сопливые носы" пытаются украсть у героев медикаменты. Они явно чем-то больны.

Игра начинается в момент, когда персонажи игроков замечают воришек. Гоблины будут сражаться, если возникнет угроза их жизни, и убегают после первого раунда боя, их можно уговорить или подкупить, чтобы узнать некоторые детали происходящего.

Проследуйте в город.

Городок Нус:

Шахтерский город, все жители которого увлечены местным фестивалем на рыночной площади. В течение дня почти все люди заболевают хворью, вызывающей чрезмерное выделение слизи.

Герои делают спасбросок средней сложности против болезни каждый раз, когда они пьют воду.

Все эти массовые заболевания, похоже, никого не беспокоят.

Вечером все жители направляются в заброшенную шахту (храм), чтобы местные жрецы "излечили" их.

Заброшенная шахта – святилище носоподобного бога М'куса (Комната 1). Жрецы бесплатно раздают зелья, выводящие слизь из носов людей и "исцеляющие" их. Вся слизь стекается в желоб и в виде слизняков уползает внутрь статуи большого носа, чтобы затем проникнуть дальше в святилище, где служители культа превращают всю собранную субстанцию в живую Слизь (Ooze), для продажи заинтересованным лицам.

Все это повторяется ежедневно.

Если герои останутся в городе на другой день, то снова будет риск заболеть, т.к. больные люди-рыбы все еще загрязняют воду. Их нужно остановить (см. описание подземелья – комната 8).

Сокровища и вещи на продажу (1к10)

1-5: вещи или монеты стоимостью 1-5 золотых (ЗМ).

6-7: Бутылка для чистки носа (50 ЗМ)

Всасывает слизь в пределах 1 фута (30 см), исцеляя больных на 1к6 ходов. через 3+1к6 дней из нее выходит малый Золотистый студень (Ochre jelly).

8: Щекочущая стрела (50 ЗМ)

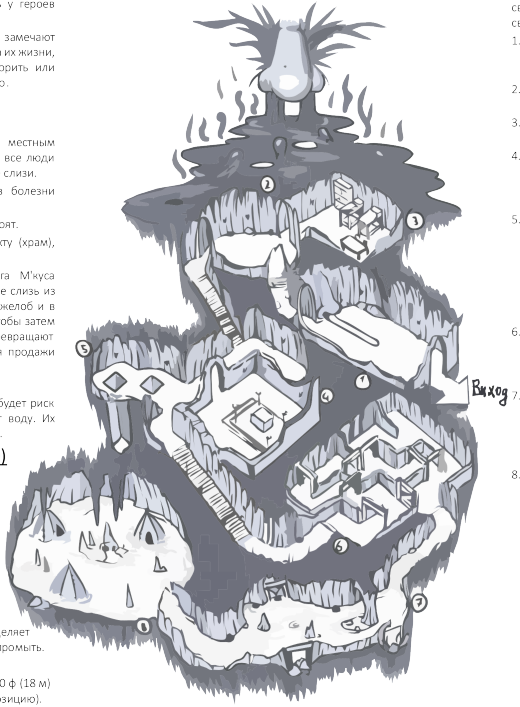
Если стрела пролетит в пределах 15 футов (4,5 м) от существа, оно чихнет, выдав свое местоположение.

9: Исцеляющее полотенце (100 ЗМ)

Можно использовать во время короткого отдыха . Исцеляет любую хворь. После использования надо тщательно промыть.

10: Жезл кашля (150 ЗМ), 3 заряда

Один заряд заставляет каждое существо в пределах 60 ф (18 м) сильно кашлять (получая помеху и раскрывая свою позицию).



Подземелье: Шахты М'куса

Дверей нет. В зонах 1,2,3 и 4 есть факелы, дающие слабый свет, области 7 и 8 покрыты светящимся мхом (тусклый свет).

1. Вход / Статуя Носа – 1к4 служителей культа, крошечное существо может пройти через ноздрю и попасть в центр комнаты 4.
2. Главный зал – пара ящиков с липкими мантиями или одним из сокровищ.
3. Жилые помещения – 2к4 служителей культа, спят или читают, один из них – маг.
4. Лаборатория слизи – 1к4 служителей культа (один маг), пытающихся превратить липкую субстанцию в Черную слизь (Black pudding). Через 1к4 хода у них это получится, если им не помешать.
5. Ямы для экспериментов – здесь магическим образом заперты 2 человека. Их можно использовать для замены погибших героев или же в виде монстров:

   Хозяин слизи (слабый гуманоид, клинически безумен). 1 из 8 шанс/ход извергнуть малый Золотистый студень (Ochre jelly). Ямы открываются, если маг убит или если магия рассеяна соответствующим заклинанием.
6. Лабиринт. Желатиновый куб движется всегда вдоль правой стены и не может использовать лестницы, занимает весь коридор по ширине, но сверху есть зазор в 3 фута (1 метр) и торчат несколько потолочных балок.
7. Затопленная шахта – заброшена в спешке, драгоценные камни и минералы стоимостью 5к4*10 золотых можно добыть при помощи кирки: легкая проверка Интеллекта + бонус за наличие инструментов, при провале проверки произойдет обрушение (1к10 дробящего урона).
8. Пещера колонии – 2к6 зараженных людей-рыб (один из них – жрец) поклоняются М'кусу. Если победить их или опрокинуть статую в центре комнаты, то заражение прекратится, и слизь (и сам М'кус) исчезнет.

   Жрец может призвать (сложная магическая проверка) *Аватара М'куса* – парящую в воздухе Черную слизь (Black pudding) в форме носа, атакующую выстрелами из ноздрей.

Рисунок 1. Пример одностраничного приключения

1. Разработка пользовательского интерфейса

1.1. Постановка задачи

В связи с выявленной проблематикой вариативности игры, практическая цель исследования заключается в сокращении затрачиваемого времени написания сюжетной составляющей настольной ролевой игры.

Для достижения цели поставлены следующие задачи:

- разработать дизайн пользовательского интерфейса программы;
- создать дизайн взаимодействия с пользователем (UX – User Experience)— создание диаграммы пользовательского пути (User flow);
- реализовать графическую часть интерфейса (UI – User Interface);
- разработать концепт интерфейса в онлайн-сервисе Figma;
- реализовать клиентскую часть приложения (frontend) на игровом движке Unity;
- разработать алгоритм блочной генерации сюжета.

Объектом исследования является процесс разработки интерфейса мобильного приложения для генерации сюжетной составляющей настольной ролевой игры. Предмет исследования — программа для генерации сюжетной составляющей настольной ролевой игры.

1.2. Описание решения

Перед реализацией визуальной части приложения требуется разработать дизайн взаимодействия с пользователем, где UX — это впечатление пользователя от работы с интерфейсом приложения. Для UX важны: дизайн, архитектура приложения, отзывчивость интерфейса и сложности формулировок и терминологии.

Основной задачей на этом этапе являлось создание диаграммы пути пользователя. Flow–диаграммы, отображающие полный путь, движения пользователя при использовании продукта.

После создания UX разрабатывается графическая часть интерфейса.

Графическая часть интерфейса работает в связке с UX-основой приложения, удовлетворяя при этом эстетические желания пользователя. Рассмотрим ключевые этапы создания UI.

На этапе разработки реализован стилистический аспект приложения. Принято решение сфокусироваться на ассоциации игроков DND со старыми компьютерными RPG. Это отражено в выборе шрифтов, цветов и стиля пиксель-арт, используемых в приложении.

Создано несколько коллажей с текстурами, элементами типографики, цветовой палитрой и цитатами (мудборд).

Для реализации текстовых блоков использовался шрифт Fifaks 1.0 dev1.

После разработки дизайна UX/UI реализована клиентская часть с использованием игрового движка Unity. На этом этапе выполнена верстка пользовательского интерфейса (рисунок 2).

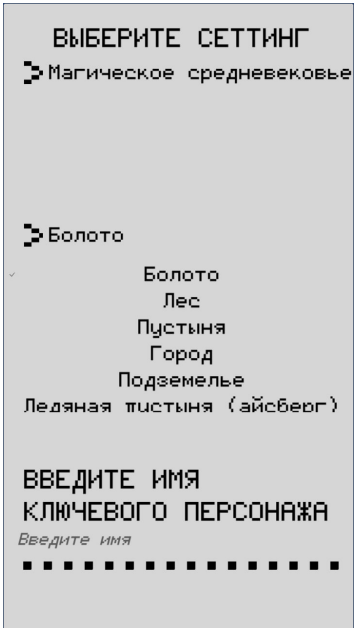


Рисунок 2. Выбор сеттинга

Unity предоставляет инструменты для настройки пользовательского интерфейса и возможность отображения 2D графики с использованием компонента Sprite Renderer. Для адаптивной верстки интерфейса приложения в движке можно установить опорное разрешение, выбрать ориентацию экрана и указать пропорции для изменения ширины и/или

высоты дочерних элементов (скалирование — «scalling», масштабирование). Применяемые инструменты и процесс верстки показаны на рисунке 3.

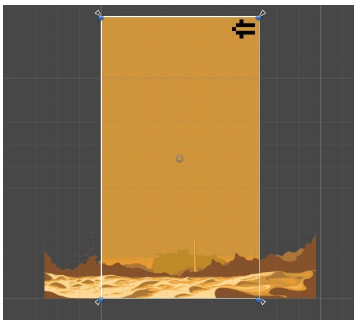


Рисунок 3. Верстка интерфейса в игровом движке Unity

Интерфейс адаптивный (рисунок 4). Для изменения размеров элемен-

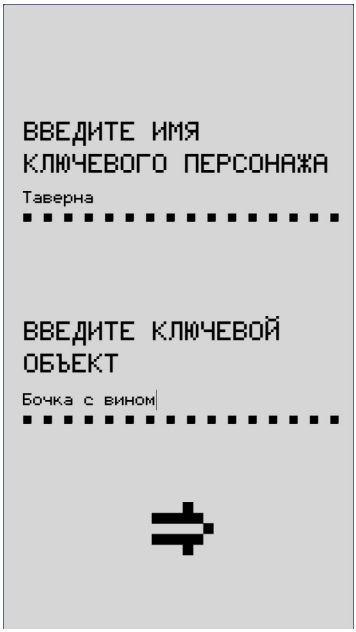


Рисунок 4. Поля ввода входных параметров

тов, а также расположения элементов в зависимости от разрешения экрана устройства используются якорные точки и инструменты скалирования.

Результат работы визуализируется во вкладке Game, где возможно

варьировать разрешение тестируемой сцены или устройства. Финальный вариант интерфейса приложения с учетом дизайна и верстки отображен на рисунке 5.

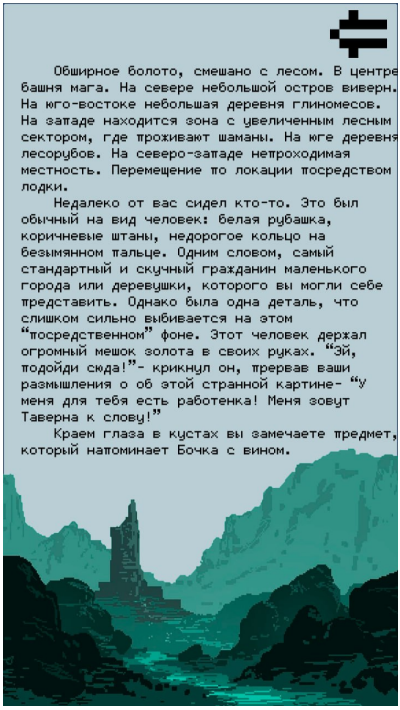


Рисунок 5. Финальный вариант интерфейса

Интерфейс создан с использованием программного обеспечения Photoshop, Figma и Unity.

Далее разработан и описан алгоритм работы приложения по генерации сюжетной составляющей.

В процессе исследования разработан алгоритм [5] работы приложения по генерации сюжетной составляющей [6–8]. На стартовом экране пользователю необходимо ввести входные характеристики для создания истории (генерации сюжета).

Реализован функционал приложения. Переключение между экранами, выбор сеттинга, выбор локации, ввод информации о предмете и персонаже.

Сформированы списки сюжетных составляющих, а также реализована логика случайного выбора участков описания с проверкой на адекватность

и сочетаемость описания.

2. Разработка программной составляющей приложения

Unity — игровой движок, в котором скрипты функционирования игровых объектов интерпретируются на языках высокого уровня C# и JavaScript (JS). Однако для обработки кода на JS требуется дополнительный интерпретатор, поэтому основным языком проекта выбран C#.

Архитектура Unity основана на компонентно-ориентированной системе, в которой игровые объекты объединены в сущности с различными компонентами. C# является объектно-ориентированным и жестко типизированным языком.

2.1. Анализ существующих технических решений

Выбор метода зависит от конкретных задач и требований генерации историй и сюжетов. В таблице 1 приведён сравнительный анализ под-

Таблица 1. Анализ существующих технических решений

№	Способ генерации сюжета	Описание метода	Языковая модель и архитектура	Набор данных (Dataset)
1	Event2event	Разбиение набора событий на последовательности и генерация этих событий с использованием языковой модели	Рекуррентные нейронные сети (RNN)	Общедоступный
2	event2sentence	Преобразование структурированных данных в естественные предложения	RNN	Общедоступный
3	УРНГ	Управляемая и редактируемая нейронная генерация сюжета	Control-and-Edit Transformer (CET)	Сюжетное описание фильмов из Википедии
4	Desc2Story	Автоматическое создание и развитие истории, основываясь на предоставленной информации или инструкциях	RNN	Общедоступный набор данных VIST (Visual Storytelling Dataset)
5	Блочный способ	Блок — часть сюжета	transformer — GPT2 от OpenAI	Возможна тренировка на WritingPrompts dataset, или на авторской БД

ходов [9, 10]: метода event2event, event2sentence, УРНГ, Desc2Story и предложенного способа блочной генерации сюжета.

(*1*) Event2event. Метод основан на генерации событий и их последовательностей. Модель способна прогнозировать возможные последствия каждого события в контексте истории.

Преимущества: формирует последовательности событий для автоматической генерации историй и сюжета с логической структурой и связностью между ними, моделирует и предсказывает следующее событие на основе предыдущих.

Ограничения: метод менее информативный в сравнении с Event2-sentence, т.к. генерирует только последовательность без подробных описаний. Модель не всегда гарантирует логичный и связный сюжет для всех входных данных.

(*2*) Event2sentence. Предложения, описывающие отдельные события, генерируются с использованием модели, обрабатывающей входные последовательности событий и создающей связные и правильные предложения, соответствующие контексту.

Преимущества: создание более информативных и содержательных описаний событий в сравнении с Event2event, превосходящих простую последовательность, например, при необходимости генерации детальных описаний. Генерирует истории с сохранением сюжетной структуры.

Ограничения: для успешной работы требуется модель, способная понимать семантику и взаимосвязи между событиями. Уделяет внимание логической структуре и связности меньше, чем блочный способ. Имеет ограничения по переводу типов событий в предложение. Требуются большие объемы подходящих данных (больше, чем у блочного способа или метода event2event).

(*3*) УРНГ – управляемая и редактируемая нейронная генерация сюжета.

Преимущества: позволяет вмешиваться в процесс генерации текста, искать подходящие части текста и редактировать их по своему усмотрению. Контролируемая генерация текстовых описаний.

Недостатки: ограниченность вариативности генерации приводит к повторению или предсказуемости. Сложность обучения и моделирования связей между историями. Имеет ограниченную способность генерировать сложные истории с нелинейной структурой.

(*4*) Desc2Story. Построен на генерации сюжетов на основе описания или аннотации. Модель получает входные данные в виде описания сюжета или набора ключевых слов, и генерирует историю или сюжет.

Преимущества: генерация истории из коротких описаний создаёт разные сценарии и сюжетные линии. Короткие описания в исходной последовательности могут быть независимыми, что обеспечивает гибкость в создании истории.

Ограничения: недостаток информации в наборе данных. Ограничения контекста могут привести к поверхностному представлению сюжета. Сложность в создании закономерностей.

(5) Блочный способ построения сюжета. Предложенный подход основан на методе генерации сюжетов с использованием блоков, где история разделяется на отдельные блоки или сцены, а затем генерируется содержимое каждого блока в рамках происходящих в нем событий. Каждый блок представляет отдельное действие, сцену или последовательность событий, и генерация сюжета осуществляется блок за блоком. Блочный подход позволяет определить ключевые моменты и переходы между элементами, что создаёт более качественные и интересные истории.

Преимущества: создаёт структурированные сюжеты, разбитые на блоки или сцены. Уделяет внимание разделению и категоризации информации по этапам развития сюжета. Обеспечивает контролируемый процесс генерации, т.к. основана на предварительно созданных блоках. Такой подход применяется при необходимости управления развитием сюжета.

Ограничения: требует предварительного разбиения истории на блоки или сцены. Ограничение гибкости и творчества, т.к. имеет строго описанную структуру. Создание авторской базы данных потребует поиска, обработку и структурирования информации, провоцируя временные и материальные издержки.

В результате проведенного анализа принято решение выбрать блочный способ построения сюжета. Одним из главных преимуществ блочного подхода является его структурированность. Каждый блок имеет определенные функции и связи с другими, что обеспечивает логическую последовательность и качественное развитие сюжета.

2.2. Разработка алгоритма функционирования приложения

Началом работы в приложении является стартовый экран, на котором пользователю предлагается указать характеристики для создания сюжета. Пользователь может выбрать сеттинг и локацию из списка предложенных значений. Затем требуется ввести имя ключевого персонажа и название объекта.

В соответствии с выбранным сеттингом и локацией определяется стиль текстового описания. В зависимости от выбранной локации, приложение подбирает описание местности из предварительно разработанного списка элементов и фоновое изображение для следующего экрана. Затем, пользователь может дополнить описание, вводя информацию о персонаже и объекте, при этом значения также зависят от выбранного сеттинга и локации.

Результатом является текстовое описание, которое появляется на экране №2, и которое может смещаться вверх или вниз в зависимости от его длины.

2.3. Проектирование базы данных на основе анализа предметной области

Информация об одностраничных приключениях хранится в JSON-файлах и применяется в дальнейшем для создания базы данных (БД). В файлах содержится следующая информация: NPC, Actions, Locations, Quests и Items (рисунок 6).

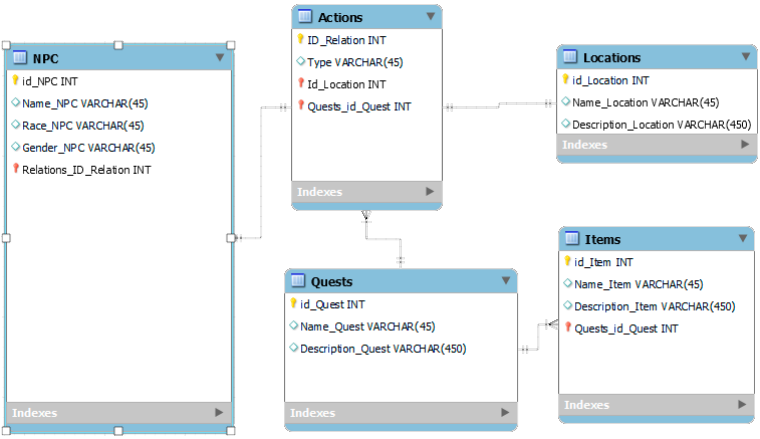


Рисунок 6. ER-диаграмма

БД состоит из различных сюжетных элементов, оцифрованных и сохраненных в формате JSON. Каждый JSON-файл содержит описание и список ключевых моментов, характеризующих события и персонажей в составе сюжета, отделяя события от менее важных, чтобы обеспечить более качественный результат. БД является авторской, а ее архитектура основана на одностраничных приключениях. Это позволяет обеспечить более легкий и удобный доступ к сюжетным элементам и ключевым моментам, улучшая понимание структуры и логики повествования. Формат JSON также обеспечивает простоту и удобство работы с БД для дальнейшего использования, хранения и обработки ее содержимого.

2.4. Описание используемого технического решения

Блочный способ построения сюжета состоит из четырех полей: персонажи, действия, описание локации и квесты. Каждое из этих полей

содержит текстовое описание, комбинирующееся вместе, чтобы создать единую историю.

Персонажи – главные герои или участники истории, имеющие свои уникальные характеристики, определяющие роль в сюжете.

Действия – последовательность событий, происходящих в истории и развивающих сюжет, физические действия персонажей, их мысли и эмоции.

Описание локации – текстовое описание места, где происходят события истории. Состоит из информации о физической обстановке, атмосфере, архитектуре локации. Создаёт образное представление о месте действия.

Квесты – цели и задачи, поставленные перед персонажами. Развивают сюжет, достигая определенных результатов.

Блочный способ создаёт связные истории, основываясь на заранее определенных блоках информации и предварительно обученной модели. Fine-tuning – точная настройка, подход к переносу обучения в deep learning, при котором веса предварительно обученной модели обучаются на новых данных[1]. После определения архитектуры (рисунок 7) модель обучалась

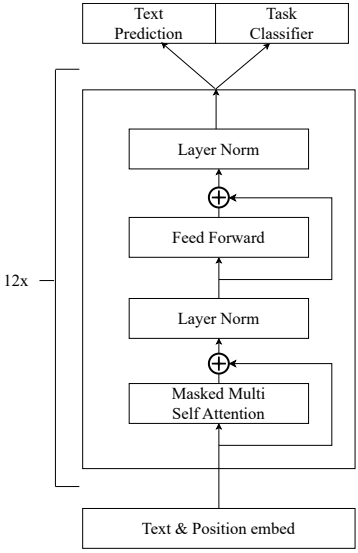


Рисунок 7. Архитектура модели

[1]См. https://www.datacamp.com/tutorial/fine-tuning-gpt-3-using-the-open-ai-api-and-python

на тренировочном наборе данных, затем оценивалась на тестовом, где для достижения результата потребовалась дополнительная настройка модели.

Использование fine-tuning модели позволяет снизить затраты на вычисления и использовать современные языковые модели без необходимости обучения их с нуля. Transformers предоставляют доступ к моделям для различных задач.

При настройке модели указываются параметры:

(1) Output_dir, определяет директорию, в которой хранятся результаты тренировки модели. В этой директории сохранены: словарь используемого токенизатора, конфигурация модели, контрольные точки, значения метрик и состояние обучения.

(2) Num_train_epochs, устанавливает общее количество эпох для обучения модели. Чем больше эпох, тем больше времени потребуется для тренировки модели.

(3) Per_device_train_batch_size, определяет размер пакета данных, использующийся для обучения модели. Этот параметр влияет на скорость обучения и использование памяти.

(4) Per_device_eval_batch_size, устанавливает размер пакетов данных, использующийся для оценки модели, что влияет на скорость и использование памяти.

(5) Warmup_steps – количество шагов разминки для установления темпов обучения. Используется низкая скорость обучения в начале тренировки модели, а затем скорость постепенно увеличивается.

(6) Weight_decay, определяет силу снижения веса в сети и влияет на регуляризацию модели. Более высокое значение этого параметра приводит к большей регуляризации.

(7) Logging_dir – директория, хранящая логи тренировки модели. Логи содержат информацию о потерях, точности и других метриках, а также обновляюся через определенное количество шагов.

(8) Logging_steps, устанавливает количество шагов обновления между записью логов, что позволяет контролировать частоту обновления логов и сохранять информацию о прогрессе тренировки.

Произведена настройка модели. Fine-tune сокращает ресурсы, необходимые для обучения новой модели с нуля, т.к. он использует уже существующую модель в качестве отправной точки. Время генерации составило 12 секунд.

2.5. Реализация функционала мобильного приложения

Start и Update являются основными методами, используемыми в игровом движке Unity. Start отвечает за создание больших списков, состоящих из описаний местоположений и настроек. Update проверяет недопустимые символы в полях персонажа и объекта и вычисляет длину выражений, используя методы CheckCharacters и CheckObjects (рисунок 8).

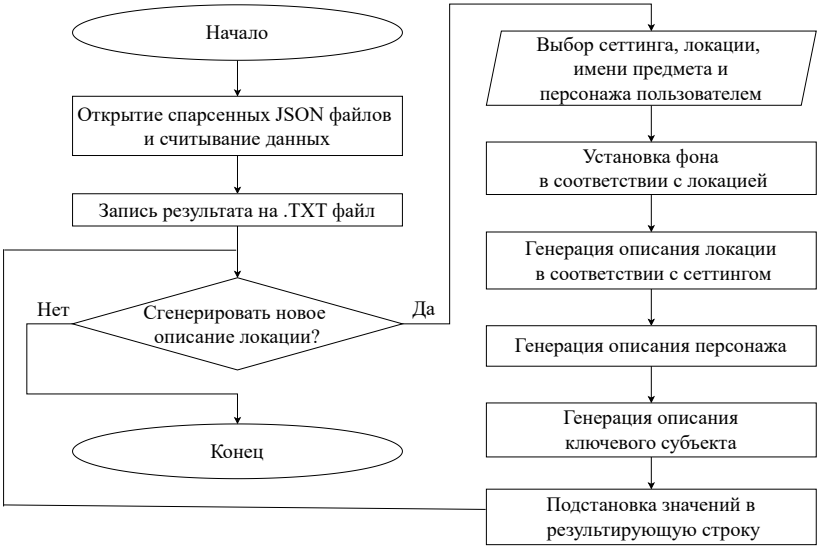


Рисунок 8. Блок-схема решения

Метод SummarySetActive отображает описание на странице и переключает на второй экран. Он привязан к кнопке в пользовательском интерфейсе и срабатывает при нажатии. Кроме того, SummarySetActive выбирает значение для отображения на экране на основе случайно сгенерированного ввода.

При генерации нового сюжета, выбирается случайный набор элементов из всех доступных JSON файлов, чтобы создать цельный и логичный сюжет. Для извлечения фрагментов сюжетов из различных JSON файлов используется парсинг, комбинирующий их в новые уникальные истории.

Заключение

В ходе исследования проведена работа по созданию сюжетно-генеративного приложения для настольной ролевой игры Dungeons&Dragons, а именно разработаны:

- UX/UI приложения;
- БД одностраничных приключений;
- алгоритм блочной генерации сюжета.

Разработанное приложение способно генерировать сюжет на основе выбранных пользователем сеттинга и локации за 12 секунд.

Для подтверждения полученных результатов приложение протестировано Саратовским сообществом игровых мастеров DND.

Согласно результатам опроса фокус группы, большинство отзывов людей, принимавших участие в тестировании, были положительными. Время, необходимое для подготовки игры, сократилось в среднем с одной недели до нескольких часов.

Список литературы

[1] Писаревская Д. Б. *Ролевые игры: пример «социализации» субъекта* // Этнографическое обозрение.– 2008.– № 1.– С. 8–18. ↑124

[2] Седых И. А. *Индустрия компьютерных игр-2020.*– Высшая школа экономики. Центр развития.– 2020.– 74 с. ↑124

[3] Малыхина А. С. *Понятие о сюжетно-ролевой игре. Структурные элементы игры* // Молодой ученый.– 2020.– № 22(312).– С. 539–541. ↑124

[4] Сахибгареева Г. Ф., Кугуракова В. В. *Редактор интерактивной структуры для инструмента генерации сценарных прототипов* // Электронные библиотеки.– 2022.– Т. **24**.– № 6.– С. 1184–1202. ↑124

[5] Каюмов Б. И. *Проблемы визуализации разветвленных сюжетов компьютерных игр.*– Казанский (Приволжский) федеральный университет.– 2021.– 79 с. ↑129

[6] Сахибгареева Г. Ф., Бедрин О. А., Кугуракова В. В. *Раскадровка как одно из представлений сценарного прототипа компьютерных игр* // Электронные библиотеки.– 2021.– Т. **24**.– № 2.– С. 408—444. ↑129

[7] Сахибгареева Г. Ф., Кугуракова В. В. *Концепт инструмента автоматического создания сценарного прототипа компьютерной игры* // Электронные библиотеки.– 2018.– Т. **21**.– № 3–4.– С. 235—249. ↑129

[8] Сахибгареева Г. Ф. *Применимость разветвленных структур для генерации сценарных прототипов видеоигр* // *65-я Международная научная конференция Астраханского государственного технического университета* (Астрахань, 26–30 апреля 2021 г.), Астрахань: Изд-во АГТУ.– 2021.– С. 596–600. ↑129

[9] Martin L., Ammanabrolu P., Wang X., Hancock W., Singh S., Harrison B., Riedl M. *Event representations for automated story generation with deep neural nets*, Thirty-Second AAAI Conference on Artificial Intelligence (February 2–7, 2018, New Orleans, Lousiana, USA) // Proceedings of the AAAI Conference on Artificial Intelligence.– 2018.– Vol. **32**.– No. 1. ↑130

[10] Das A., Verma R. M. *Can machines tell stories? A comparative study of deep neural language models and metrics* // IEEE Access.– 2020.– Vol. **8**.– Pp. 181258–181292. ↑130

Поступила в редакцию 12.05.2023;
одобрена после рецензирования 06.11.2023;
принята к публикации 02.12.2023;
опубликована онлайн 17.12.2023.

Рекомендовал к публикации *д.ф.-м.н. А. М. Елизаров*

Информация об авторах:

Диана Антоновна Челышева

Foto by D. A. Chelysheva

Студент, является автором трёх публикаций, участником четырёх международных молодёжных научно-практических конференций, победитель регионального конкурса "Студенческий стартап"(III очередь), работает над созданием проектов в Студенческом конструкторском бюро "Робототехника и интеллектуальные системы" , сфера научных интересов - искусственный интеллект, дистанционное управление 3D печатью, программирование, робототехника, компьютерная графика.

0009-0002-4941-0767
e-mail: *diana.chelysheva.98@mail.ru*

Аркадий Викторович Ключиков

К.т.н., зав. кафедрой "Цифровое управление процессами в АПК" ФГБОУ ВО Вавиловский университет, директор Студенческого конструкторского бюро "Робототехника и интеллектуальные системы"СГТУ имени Гагарина Ю.А., сфера научных интересов - 3D дисплеи, робототехника, имитационное моделирование, программирование.

0000-0002-8657-6486
e-mail: *krok9407@mail.ru*

Вклад авторов: *Д. А. Челышева* – 50.00% (идея, методология, создание стилевого файла, сбор материала); *А. В. Ключиков* – 50.00% (написание черновой версии, доработка и редактирование, визуализация, администрирование).

Авторы заявляют об отсутствии конфликта интересов.

PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article APPLIED SOFTWARE SYSTEMS

UDC 004.946:793.7
10.25209/2079-3316-2023-14-4-123-140

Generation of a plot component for a tabletop role-playing game

Diana Antonovna **Chelysheva**[1], Arkady Viktorovich **Klyuchikov**[2]

[1] Saratov State Technical University, Saratov, Russia
[2] Vavilov University, Saratov, Russia
[1]✉*diana.chelysheva.98@mail.ru*

Abstract. The analysis of the subject area and relevance of tabletop role-playing games, as well as their impact on the spheres of human activity, is carried out. The mechanics of the gameplay of the tabletop role-playing game Dungeons and Dragons have been studied. Software solutions for creating a plot description of locations, including a graphical representation of maps and one-page adventures, are considered. Implemented design and adaptive user interface. The database architecture has been formed, as well as implemented in JSON file format. The architecture of the model has been developed, as well as its configuration. The block method of plot construction is described. The algorithm of the application for generating the plot component has been developed. Lists of plot components are formed, as well as the logic of random selection of sections of the description with a check on the adequacy and compatibility of the description is implemented. (*In Russian*).

Key words and phrases: plot generation, tabletop role-playing game, "Dungeons and Dragons", mobile application, user interface, block method of plot construction, database

2020 *Mathematics Subject Classification*: 97P30; 68Q87

For citation: Diana A. Chelysheva, Arkady V. Klyuchikov. *Generation of a plot component for a tabletop role-playing game*. Program Systems: Theory and Applications, 2023, **14**:4(59), pp. 123–140. (*In Russ.*).
https://psta.psiras.ru/read/psta2023_4_123-140.pdf

References

[1] D. B. Pisarevskaya. “Role-playing games: a case of the "subculture" socialization”, *Etnograficheskoe obozrenie*, 2008, no. 1, pp. 8–18 (in Russian).

[2] I. A. Sedyx. *Computer Games Industry-2020*, Vysshaya shkola ekonomiki. Centr razvitiya, 2020 (in Russian), 74 pp.

[3] A. S. Malyxina. “The concept of a plot-role-playing game. Structural elements of the game”, *Molodoj uchenyj*, 2020, no. 22(312), pp. 539–541 (in Russian).

[4] Saxibgareeva G. F., Kugurakova V. V.. “Interactive structure editor for scenario prototyping tool”, *Elektronnye biblioteki*, **24**:6 (2022), pp. 1184–1202 (in Russian).

[5] Kayumov B. I.. *Problems of visualization of branched plots of computer games*, Kazanskij (Privolzhskij) federal’nyj universitet, 2021 (in Russian), 79 pp.

[6] G. F. Saxibgareeva, O. A. Bedrin, V. V. Kugurakova. “Storyboard as one of the representations of the scenario prototype ofcomputer games”, *Elektronnye biblioteki*, **24**:2 (2021), pp. 408—444 (in Russian).

[7] G. F. Saxibgareeva, V. V. Kugurakova. “The concept of automatic creation tool for computer game scenario prototype”, *Elektronnye biblioteki*, **21**:3–4 (2018), pp. 235—249 (in Russian).

[8] G. F. Saxibgareeva. “Applicability of branched structures for generating scenario prototypes of video games”, *65-ya Mezhdunarodnaya nauchnaya konferenciya Astraxanskogo gosudarstvennogo texnicheskogo universiteta* (Astraxan’, 26–30 aprelya 2021 g.), Izd-vo AGTU, Astraxan’, 2021, pp. 596–600 (in Russian).

[9] L. Martin, P. Ammanabrolu, X. Wang, W. Hancock, S. Singh, B. Harrison, M. Riedl. “Event representations for automated story generation with deep neural nets”, Thirty-Second AAAI Conference on Artificial Intelligence (February 2–7, 2018, New Orleans, Lousiana, USA), *Proceedings of the AAAI Conference on Artificial Intelligence*, **32**:1 (2018).

[10] A. Das, R. M. Verma. “Can machines tell stories? A comparative study of deep neural language models and metrics”, *IEEE Access*, **8** (2020), pp. 181258–181292.

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В МЕДИЦИНЕ

УДК 004.89:616.1
10.25209/2079-3316-2023-14-4-141-188

Облачный сервис дифференциальной диагностики и назначения персонифицированного лечения воспалительных заболеваний сердца

Валерия Викторовна **Грибова**[1], Елена Арефьевна **Шалфеева**[2], Маргарита Вячеславовна **Петряева**[3], Дмитрий Борисович **Окунь**[4], Леонид Александрович **Федорищев**[5], Роман Игоревич **Ковалев**[6]✉
Институт автоматики и процессов управления ДВО РАН, Владивосток, Россия
✉*koval-995@mail.ru*

Аннотация. Исследуется автоматизация проведения полного цикла диагностики и дифференциальной диагностики воспалительных заболеваний сердца и назначения персонифицированного лечения. Методом исследования является формирование декларативных баз знаний и объясняющего свои результаты решателя по единой онтологии. Онтологический решатель интерпретирует формализованные знания при получении сведений о новом медицинском случае (пациенте).

Описаны общие принципы разработки и концептуальная архитектура интеллектуального сервиса с декларативными знаниями. Сформированы информационные компоненты для воспалительных заболеваний сердца и программные компоненты для полного цикла диагностики и дифференциальной диагностики, персонифицированного лечения. Указаны источники знаний и проведено тестирование на случаях из практики, описанных в литературе. Приведено обоснование выбора технологий и алгоритмов, выявлены и сформулированы требования к программному комплексному сервису, шаги по разработке всех компонентов.

Сервис поддержки принятия диагностических решений со свойствами объяснимого искусственного интеллекта в кардиологии реализован на медицинском портале облачной платформы IACPaaS. Платформа позволяет масштабировать предложенное решение и обеспечивает доступ практикующих врачей со свободной регистрацией для экспериментов в реальных ситуациях.

Ключевые слова и фразы: система поддержки принятия врачебных решений, интеллектуальный сервис, воспалительные заболевания сердца, база знаний, онтологический подход, специализированный решатель

Благодарности: Работа выполнена при финансовой поддержке государственного задания Минобрнауки: 0202-2021-0004

Для цитирования: Грибова В. В., Шалфеева Е. А., Петряева М. В., Окунь Д. Б., Федорищев Л. А., Ковалев Р. И. *Облачный сервис дифференциальной диагностики и назначения персонифицированного лечения воспалительных заболеваний сердца* // Программные системы: теория и приложения. 2023. **Т. 14**. № 4(59). С. 141–188. https://psta.psiras.ru/read/psta2023_4_141-188.pdf

Введение

Заболевания сердечно-сосудистой системы (ССЗ) являются основной причиной летальности у всех категорий взрослого населения: по данным РосСтата отмечается неуклонный рост числа заболеваемости, особенно за последние годы [1, 2]. Диагностика и лечение воспалительных заболеваний сердца (миокардитов, перикардитов, эндокардитов и др.) остается одним из наиболее сложных разделов работы терапевтов и кардиологов из-за неоднородности и неспецифичности клинических проявлений. Латентное, хроническое или нетипичное течение заболеваний дополнительно повышает трудность дифференциальной диагностики. Для оценки значимости признаков, на комплексе которых строится диагноз, предложен целый ряд диагностических алгоритмов. Практики и опыта конкретного врача случается недостаточно для того, чтобы своевременно выявить любую проблему, возникшую в организме человека.

Быстро классифицировать трудный случай, предложить и обосновать решения на любом этапа взаимодействия с пациентом (профилактики, диагностики, лечения или реабилитации) могли бы помочь системы поддержки принятия решений, обладающие доступом к огромному объему данных, передовой научной литературе и миллионам историй болезней. Польза от внедрения систем поддержки принимаемых решений в медицинской сфере в первую очередь ожидается в раннем подозрении начавшегося заболевания, увеличении точности диагностики, персонификации лечения установленного заболевания. Поэтому разработка систем поддержки принятий врачебных решений (СППВР), использующих современные методы искусственного интеллекта, сегодня является приоритетной задачей для многих стран мира.

В последние годы с выбором систем поддержки принимаемых решений для внедрения в медицинской сфере вырос интерес к объяснимому ИИ. Врачи, несущие медицинскую ответственность, вряд ли могут доверять результатам системы без объяснения, лежащего в ее основе процесса принятия решений.

Отсюда *цель нашего исследования*: разработка интеллектуального сервиса для диагностики и лечения воспалительных заболеваний сердца, дифференциальной диагностики воспалительных с другими заболеваниями сердца, объясняющего свои гипотезы, поскольку именно объясняющий сервис может стать полезным электронным консультантом для современного врача-кардиолога.

Для достижения поставленной цели необходимо решить следующие задачи:

- выбрать источники получения знаний по воспалительным заболеваниям сердца,
- выбрать методы формирования декларативных баз знаний (БЗ),
- разработать БЗ для диагностики, дифференциальной диагностики воспалительных заболеваний сердца, для назначения лечения и его коррекции по необходимости в процессе динамического наблюдения за процессом лечения пациента.
- выбрать метод реализации решателя и архитектурного фреймворка с решателем для конструирования сервиса, основанного на декларативной БЗ,
- разработать пользовательский интерфейс интеллектуального сервиса (GUI),
- провести конструирование программного продукта (интеллектуального сервиса) для реализации выбранного решения,
- определить материалы для тестирования БЗ и сервиса в целом,
- выбрать методология проведения тестирования.
- провести апробацию разработанных методов в реальной СППВР (или тестирование БЗ и СППВР в целом).

Близкие исследования

Автоматизация поддержки обнаружения или диагностики некоторых воспалительных заболеваний сердца (миокардит, перикардит) предложена в многочисленных интернет-средствах проверки симптомов [1][2][3][4], но знания по некоторым другим воспалительным заболеваниям сердца (коронарит и лимфоцитарный эндокардит) в такие средства редко бывают заложены. Средства проверки симптомов не обеспечивают возможность сочетанной диагностики таких болезней и поддержки их лечения. Это связано с упрощенным представлением правил клинических решений, а также с представлением клинической картины заболеваний и диагностических правил без учета временного аспекта. Подобные программные продукты

[1]Программа *Symptom Checker* разработанная группой профессиональных врачей различных специальностей МЦАО «Поликлиника святого Антипы» и программистами.

[2]*Healthdirect. Symptom cheker* – сервис, помогающий австралийцам, найти подходящего врача.

[3]«Искусственный интеллект *Киберис* - индивидуальная медицина» – медицинский ассистент на основе искусственного интеллекта для диагностики и персонифицированной терапии, подбора аналогов лекарств, проверки безопасности назначений и автозаполнения медкарты.

[4]*«ASCVD Risk Estimator Plus»* – сервис Американского колледжа кадиологии.

хорошо себя зарекомендовали при расчете и анализе определенных рисков в кардиологии [3–5]. Такие представители как [6], используют алгоритмы машинного обучения для выявления признаков и предсказания сердечно-сосудистых событий.

В настоящее время разработаны СППВР для диагностики и лечения отдельных заболеваний сердца таких как: ишемическая болезнь сердца [7–9] как компонент пациенториентированной модели кардиологической реабилитации [10], в диагностике и лечению артериальной гипертензии [11][5].

Что касается воспалительных заболеваний сердца, то можно остановиться на таких системах как: «Инфекционный эндокардит», которая использует базу данных о клинических случаях эндокардита, а также алгоритмы машинного обучения для прогнозирования риска осложнений и выбора наиболее эффективного лечения; «Кардио-ЭКО» производит оценку степени повреждения клапанов сердца при эндокардите используя данные эхокардиографии. Прототип СППВР для ведения пациентов с инфекционным эндокардитом [12], основанный на интеграции гипертекста и знаний, где качественные данные были проанализированы методом постоянного сравнения. Использование системы, показало, что подобные решения способны обеспечить поддержку конкретного пациента для подтверждения клинических решений и управление терапии на более высоком уровне.

Однако, для интеллектуальной поддержки дифференциальной диагностики и назначения персонифицированного лечения воспалительных заболеваний сердца требуются сложные модели знаний, соответствующие системе понятий специалистов-кардиологов. С развитием графовых представлений знаний рассуждения на основе графов знаний стали широко использоваться для поддержки и автоматического принятия решений [13, 14]. Помимо графового представления знаний, важно иметь возможность интеграции их обработчиков (интерпретаторов) с медицинскими документами: электронной медицинской картой (ЭМК), электронной историй болезни (ЭИБ); средства проверки симптомов такую возможность не дают.

Системы, основанные на машинном обучении, не выдают объяснений; в некоторых случаях имеется некоторая интерпретация полученного результата. Некоторые средства проверки симптомов формируют краткое объяснение в соответствии со своей упрощенной моделью знаний.

[5]см. также *Витакор*[URL] – система поддержки принятия решений для формирования назначений терапии больным с артериальной гипертензией.

Требования к СППВР

Современным электронным консультантом для врача в процессе полного цикла диагностики и назначения персонифицированного лечения воспалительных заболеваний сердца сможет стать такой программный сервис, который будет:

- использовать данные из ЭИБ, ЭМК или подобного документа о пациенте,
- использовать заслуживающую доверия БЗ,
- предлагать обоснованные гипотезы о предварительном диагнозе заболевания,
- после предварительной диагностики предлагать запрос (перечень) дополнительных лабораторных и инструментальных исследований,
- проводить дифференциальную диагностику воспалительных заболеваний сердца между собой, а также с другими заболеваниями сердца,
- предлагать варианты персонифицированного лечения с максимально благоприятным прогнозом и объяснить такие гипотезы о решении,
- формировать обоснованное определение основного диагноза заболевания, либо запрос дополнительных наблюдений для дообследования или уточнения

1. Методы формирования баз знаний

Для формирования декларативных БЗ был использован онтологический подход, который обеспечивает:

- повторное использование программных решателей задач медицинской диагностики и лечения,
- многовариантную (т.е. разными путями) и коллективную работу над новыми версиями БЗ,
- «объективизацию» проверки правильности и точности БЗ,
- «понятность» представленных в БЗ знаний.

Для формирования БЗ для диагностики и дифференциальной диагностики, персонифицированного лечения требуется либо выбрать готовые онтологии и убедиться в их достаточности либо создать новые для своих потребностей. Обычно в проработанной области есть единая ее онтологическая модель или набор онтологий для разных задач. На медицинском портале платформы IACPaaS [15] более 10 лет эксплуатируются онтология

для диагностики заболеваний и онтология персонифицированного лечения. Онтологическая база наполняется в соответствии с онтологической структурой (моделью представления) и предписанными ограничениями целостности.

Онтологическая база знаний наполняется разными способами. Описанные в литературе знания (из документов, стандартов, учебников, научных статей) могут извлекаться автоматически. Это могут делать и вручную эксперты или инженеры, применяя специализированные (онтологические) средства редактирования. Правильность, полнота и другие свойства качества БЗ зависят от выбранных источников знаний, а также от методов и инструментов формирования. В частности, ручное наполнение сопряжено с человеческим фактором (с одной стороны, оно более осознанно, ответственно: попутно выполняется анализ актуальности каждого элемента, с другой стороны ожидаются ошибки).

Автоматическое извлечение закономерностей и ассоциативных связей из наборов данных, имеющих табличный вид, технически привлекательно, но есть ограничения:

(*а*) объем извлекаемых знаний в разы меньше содержащихся в учебниках (даже для простых перечисляемых признаков),

(*б*) представление наборов данных не охватывает описание процессов развития, течения заболеваний, вариантов контролируемого выздоровления.

IACPaaS-БЗ для диагностики может наполняться автоматически из текстов, либо индуктивно на основе наборов данных. IACPaaS-БЗ для лечения может производиться автоматически из текстов клинических рекомендаций или экспертами предметной области с помощью редакторов знаний, управляемых онтологиями. Отметим, что наборов данных для задачи персонифицированного лечения ССЗ в настоящее время не найдено.

Так как ни один из способов в отдельности не обеспечивает полноту знаний, важно соединять не менее двух вариантов формирования БЗ. Цикл построения и обновления БЗ обязательно включает этапы тестирования, проверки правильности на регулярно расширяемом эталонном наборе прецедентов.

Базы знаний могут быть наполнены по потребности их пользователей. Распространены два варианта создания новой базы для нового сервиса: формируют знания по конкретному набору заболеваний либо комплексируют БЗ из ранее описанных заболеваний и вновь создаваемых.

На медицинском портале платформы IACPaaS по онтологиям диагностики и лечения предусмотрены соответствующие программные рассуждатели, интерпретирующие формализованные знания в соответствии с соглашениями о правилах решения этих задач в медицине. При этом в случае диагностики «выгодно» работать с комплексированными БЗ: уже накопленные формализованные описания диагностических знаний ряда заболеваний расширяют возможности дифференциальной диагностики «новых» заболеваний с накопленными ранее. Во всех случаях БЗ подлежат тестированию (например, на наборе эталонных случаев).

2. Разработка баз знаний

2.1. Источники знаний по воспалительным заболеваниям сердца

В качестве источника знаний использовались официально утвержденные медицинские документы в виде клинических рекомендаций и методических руководств Министерства здравоохранения РФ, поскольку клинические кардиологи используют их в своей работе и доверие к этим источникам 100%-ное, а также учебная медицинская литература и научные статьи [16,17]. Так как распознавание таких текстов с полным извлечением диагностических признаков и методов лечения автоматически пока не обеспечивает полного извлечения имеющегося в них «медицинского смысла», особенно динамики процесса развития, течения заболеваний, были привлечены эксперты по формализации медицинских знаний.

Последующие версия БЗ будут созданы при появлении заслуживающих доверия наборов данных, при обновлении Минздравом клинических рекомендаций, а также при взаимодействии с экспертами из клинической практики, заинтересованных в формализации их опыта.

2.2. Наполнение баз знаний

Информационные ресурсы сервиса представлены следующими компонентами: База медицинской терминологии и наблюдений, База знаний диагностики воспалительных заболеваний сердца, Фармакологический справочник, База МКБ-10, База знаний о лечении заболеваний, Архив электронных историй болезней. Каждый информационный ресурс является источником либо терминов, либо непосредственно знаний для решения вопросов в диагностике или лечения воспалительных заболеваний сердца и ресурсы объяснений процесса диагностического поиска и рекомендованной терапии. БЗ имеют семантическое представление, сформированы с помощью редактора знаний, это позволяет обеспечить их формирование и

модифицирование непосредственно экспертами знаний. Инфоресурс «База медицинской терминологии и наблюдений» был расширен признаками и наблюдениями, необходимыми для описания новых воспалительных заболеваний сердца.

Для разработки БЗ был использован комплекс онтологий IACPaaS: «Онтология знаний о диагностике заболеваний», «Онтология лечения заболеваний».

Сформированная «База знаний о диагностике воспалительных заболеваний сердца» включает структурированные формализованные знания о следующих заболеваниях: инфекционный миокардит, инфекционный перикардит, острый и подострый бактериальный эндокардит, острый панкардит (4506 понятий). Для диагностики воспалительных заболеваний сердца описаны: формы заболевания возможные степени тяжести, варианты этиологии. Дополнительные симптомокомплексы с описанием специфических признаков сформированы для возможности детализации диагноза заболеваний в ходе лечебно-диагностического процесса и формирования развернутого клинического диагноза. Использованная IACPaaS-онтология знаний о диагностике дает возможность внесения особых (сложных) случаев, с которым врачи встречаются в клинической практике (и такие симптомокомплексы были внесены в базу знаний).

Для возможности дифференциальной диагностики воспалительных заболеваний с другими болезнями сердца, БЗ была дополнена и расширена информационным ресурсом «База знаний заболеваний сердечно-сосудистой системы», который дополнительно включает описания таких заболеваний как: острый инфаркт миокарда, стенокардия напряжения, нестабильная стенокардия, гипертоническая болезнь и др. Этот ресурс используется ранее разработанным на платформе сервисом «Диагностика заболеваний сердечно-сосудистой системы» [14]. Описание базы знаний по диагностике заболеваний включает более 5000 понятий.

Для генерации рекомендаций о лечении сформирован ресурс [18] «База знаний о лечении воспалительных заболеваний сердца» (1311 понятий). В ней скомбинированы фрагменты клинической картины заболевания, особенностей динамики заболевания и методы лечения на основе данных о клинической эффективности лекарственных средств. Такой принцип позволит сочетать стандарт лечения болезни и индивидуальную тактику лечения каждого пациента, а также вносить изменения в процесс лечения при появлении новых клинически значимых показателей. Назначение персонифицированного лечения и его коррекция возможна благодаря

описанию в БЗ различных схем терапии данной группы заболеваний с учетом показаний и противопоказаний к назначению лекарственных препаратов, базирующийся на учете индивидуальных особенностей каждого человека, основанное на клинических, генетических, геномных и средовых факторах.

3. Метод реализации решателя и средств конструирования сервиса

Для автоматизации решения интеллектуальных задач с получением объяснений гипотез (о решении) используются методы инженерии знаний. Для задач дифференциальной диагностики и персонифицированного лечения, онтология, во-первых, отражает структуру соответствующих знаний из предметной области, а во-вторых, устанавливает стратегии поиска решения или формирования результата. Они могут зависеть от сути задачи, роли систем с БЗ и предпочтений пользователей (например, предлагать и обосновывать лучшее решение, предлагать все возможные решения, критиковать каждое пользовательское решение и т.п.).

Конкретная стратегия выдвижения гипотез для диагностируемого аномального процесса требует полностью подтвердить некоторый *вариант развития процесса* либо требует найти все неотвергнутые *варианты развития* всех аномальных процессов. Стратегии поиска являются одним из видов *онтологических соглашений* по применению явных знаний для конкретных задач. Поддержка принятия решений для одной задачи (или генерация решения) производится соответствующим программным рассуждателем (reasoner), специализированным на онтологии знаний для этой задачи и онтологических соглашениях. Такой специализированный *решатель* интерпретирует знания: производит обход онтологической БЗ, в соответствии с соглашениями, выдвигая, либо опровергая гипотезы, рассуждает и собирает аргументы в пользу одних версий, гипотез (и против других), формируя практически полезное *объяснение*.

Содержание одного из онтологических соглашений для лечения: «возможно устранение симптома (*вида проблемы*), подтверждаемого текущими входными данными, если в БЗ среди множества периодов развития (вида *последствий проблемы*), ожидаемых после *текущего периода*, есть период, в котором симптом будет в норме (компонент *нормального состояния*), и в БЗ *для такого последствия* указаны виды *воздействий*, и возможность этих *воздействий* не отвергается текущими входными данными.

Специализированный рассуждатель для поддержки решения задачи относительно текущей ситуации (документ ЭМК или ЭИБ) генерирует объяснение в соответствии с содержимым доступной онтологической БЗ и с соглашениями. Квалифицированному пользователю полезно объяснение с отсылкой к возможным и обнаруживаемым причинам, следствиям, влияющим факторам, свойствам, условиям для соединения частей в конструкцию и т.д. В связи с этим «наиболее понятным и наименее затратным» будет объяснение, структура которого близка к структуре знаний (ее аналог или ее инверсия) – и сути применяемых соглашений.

Специализированный по онтологии рассуждатель в процессе рассуждения и поиска гипотез применяет правила обработки: связывания *посылок* со *следствиями* и поиска информации для подтверждения посылок. Обычно обработка имеет несколько этапов. Это поиск в описании текущей ситуации (входных данных) фактов для подтверждения условий появления процессов (вывод в отчет-объяснение каждого подтверждаемого условия проверяемого процесса и всех фактов, соответствующих подтверждаемому условию); поиск в описании текущей ситуации фактов для подтверждения варианта течения процесса (вывод в отчет-объяснение каждого проверяемого варианта процесса течения и всех фактов, соответствующих ему); выдвижение гипотез на основе подтвержденных и неотвергнутых посылок.

При создании алгоритмов обработка информации распределяется на работу над БЗ, работу с фактами, работу по фиксации аргументов-объяснений в отчете-объяснении. Например, для создания алгоритма *проверки гипотезы о варианте развития и обоснования варианта* одна программная единица работает над причинно-следственной связью «вариант развития – признаки» и обращается к другим: *проверка наличия признака* (среди фактов), *проверка гипотезы о соответствии значения признака*, *установление статуса проверенного признака* в «объяснении», *установление статуса проверенной гипотезы* и т.д. Программная единица, обрабатывая свои виды связей между элементами информации, делает промежуточные заключения процесса логического вывода. В алгоритме решателя происходит поочередный вызов процедур вывода следствий из обработанных посылок, которые записывают результат проверки посылок в объяснение.

Созданный на облачной платформе IACPaaS специализированный (на онтологии знаний и онтологических соглашениях) *решатель* сформирован декларативно, интегрирован с пользовательским интерфейсом (рисунок 1).

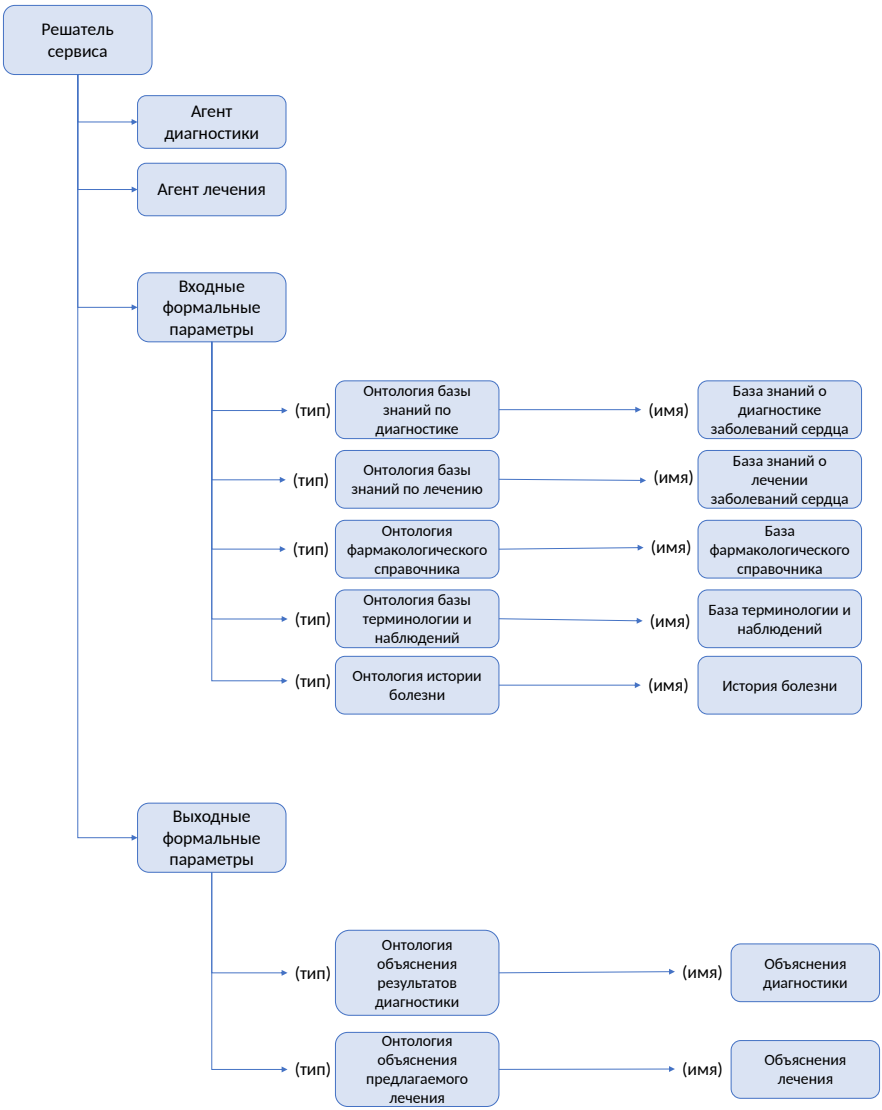


Рисунок 1. Схема специализированного решателя

Решатель разработан с применением агентного подхода: корневой агент обеспечивает функциональность интерфейса и взаимодействие пользователя с системой, а также отвечает за вызов агентов выполнения

подзадач диагностики и лечения. Формальными параметрами решателя являются метки онтологических Баз знаний и *входных* информационных ресурсов (наборов ЭИБ).

3.1. Пользовательский интерфейс интеллектуального сервиса

Пользовательский интерфейс сервиса состоит из трех основных частей:

- Интерфейс ввода историй болезни
- Интерфейс просмотра результатов диагностики
- Интерфейс просмотра рекомендованного лечения

Интерфейс ввода историй болезни предоставляет возможность ввести всю имеющуюся информацию по болезни (рисунок 2).

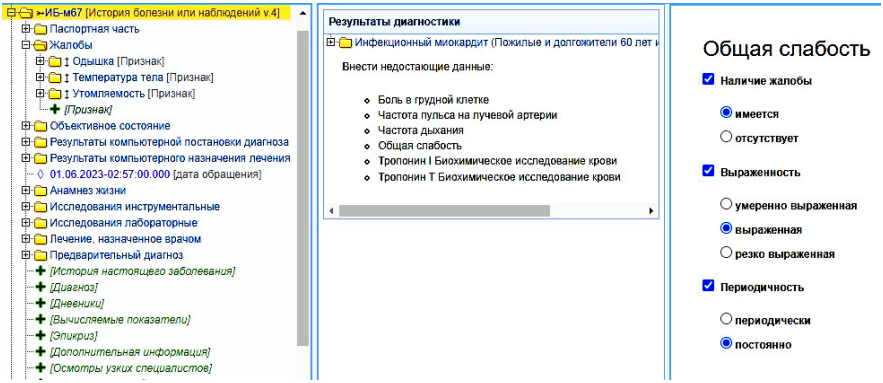


Рисунок 2. Интерфейс ввода истории болезни

Особенность интерфейса ввода историй болезни заключается в том, что он автоматически формирует и связывает между собой знания, построенные по разнородным онтологиям. Это обеспечивает высокую скорость ввода структурированных историй болезней на основе множества разных больших формализованных баз медицинских знаний. Рассмотрим подробнее, как это происходит на практике.

На рисунке 3 представлены (немного упрощенно) две онтологии, которые необходимо соотнести друг с другом для автоматизации ввода признака из истории болезни.

С одной стороны, эти структуры достаточно похожи, так как, по сути, представляют собой описание признака и, в частности, даже имеют

Рисунок 3. Разнородные структуры (онтологии)

полностью совпадающие элементы. Но, с другой стороны, эти структуры далеки от полного совпадения: так в онтологии справа можно увидеть:

(*1*) некоторые вспомогательные элементы («Тип возможных значений»),

(*2*) разделение признака на подтипы («Простой признак», «Составной признак»),

Эти структуры не были сделаны одинаковыми при онтологическом инжиниринге, хотя обе отвечают за сущность «признак», поскольку предназначены для разных целей: структура признака в ЭИБ нацелена на внесение признаков с *конкретными* значениями, в то время как в Базе медицинской терминологии и наблюдений – для описания *всех вариантов* значений признака, синонимии, нормальных и референсных диапазонов. Например, признак «Температура тела» в архиве истории болезни будет иметь какое-то конкретное значение, например, 37.0, а в Базе он будет иметь диапазоны значений (например, 35-42). Кроме того, идея автоматизации создания интерфейса заключается также и в том, чтобы постараться не зависеть от точного совпадения структур данных, а, наоборот, предоставить максимальную гибкость в возможности подключения произвольных баз знаний с их несовпадающими структурами. Таким образом, в данной задаче мы имеет две разнородные структуры

знаний, которые все же необходимо автоматически соотнести между собой. Для реализации этой задачи предложено ввести дополнительную структуру – таблицу, в которой будут описаны необходимые специальные связи для установления соответствия между разнородными онтологиями.

Каждый элемент в данной структуре описывает элемент *исходящей* онтологии, который должен быть преобразован для структуры *конечной* онтологии. Поле «Цель» описывает, как должен быть представлен «Элемент» исходящей онтологии в конечной. При этом возможны следующие варианты: если «Цель» содержит ненулевое строковое значение, то элемент исходящей онтологии переименовывается в соответствии с этим значением. Если же «Цель» не содержит никакого значения или содержит значение «null», то это значит, что элемент исходящей онтологии должен быть «вырезан» из структуры конечной онтологии. «*Вырезание*» элемента не означает удаление этого элемента, а лишь скрытие его заголовка, все дочерние элементы этого элемента становятся дочерними для его родительского элемента (меняется иерархическая структура дерева). Поле «Дочерний элемент», наоборот, служит для того, чтобы в конечной онтологии *появился* такой дочерний элемент, который есть у заданного элемента. Он появляется не как пустой новый элемент, а как промежуточный родительский узел между текущими элементами в конечной онтологии (т.е. опять меняется иерархическая структура дерева).

Для просмотра введенных историй болезней, всех введенных признаков и их характеристик в интерфейсе предусмотрено иерархическое разворачиваемое дерево. Некоторые ветки этого дерева интерфейсно преобразованы в одну, упрощая визуализацию структуры за счет скрытия не влияющих на результат длинных цепочек структур знаний. Для внесения новых знаний пользователю предоставляется интерфейс поиска. Интерфейс обеспечивает механизм конкретизации поиска по нескольким ключевым критериям-подстрокам, вводимым через пробел. Поиск может учитывать как части названий признаков, их характеристик или строкового содержимого, так и вспомогательную информацию о знаниях, например, синонимию.

Интерфейс просмотра результатов диагностики отображает результаты диагностики в разных формах, в зависимости от итогов диагностики. В случае нехватки каких-либо данных или знаний интерфейс отображает всю необходимую информацию и предоставляет возможность ввести дополнительные данные, автоматически выполняя за пользователя часть

работы по поиску. В случае достаточности информации в интерфейсе отображается полная картина диагностики с подробными объяснениями по каждому критерию. Интерфейс просмотра рекомендованного лечения отображает варианты лечения в форме иерархического дерева, в котором можно увидеть всю необходимую информацию по способам лечения, лекарствам, периодам их применения и т.д.

4. Конструирование интеллектуального сервиса с развиваемой БЗ

На медицинском портале платформы IACPaaS интеллектуальные сервисы строятся как «сборка» программных компонентов (специализированного решателя) с информационными (формализованными знаниями в виде БЗ). Их связывание основано на разработке по единой онтологии. Перед сборкой каждый компонент тестируется в своей тестовой среде со своими наборами тестов. Проверка БЗ для диагностики требует наборов электронных историй болезни (ЭИБ), где пациент был успешно диагностирован. Проверяется правильность и точность. Проверка БЗ для персонифицированного лечения требует наборов ЭИБ, где пациент был вылечен либо стабилизирован. Проверяется правильность и полнота. После сборки, процесс которой занимает считанные минуты, собранный сервис испытывается на существующем контрольном наборе случаев (ЭИБ) с решением всех требуемых подзадач: правильной постановки диагноза и успешного лечения (см. рисунок 4). Ответственность за качество несут все участники.

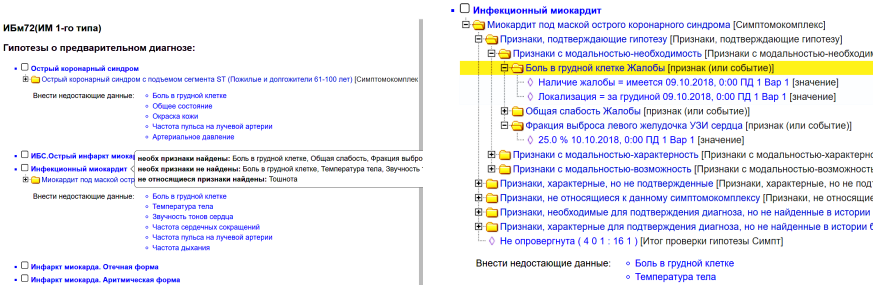


Рисунок 4. Фрагменты диалога с врачом при анализе возможного решения

Возможность дополнения и расширения БЗ обусловлена онтологическим подходом, позволяющем экспертам редактировать БЗ в привычных

терминах и оценивать адекватность зафиксированных знаний. Адаптация клинической системы и управление БЗ осуществляется непосредственно через перенос новой информации в структурированную БЗ с проверкой ее на существующем контрольном наборе ЭИБ (или наборов прецедентов). Управлять БЗ диагностики требуется в связи с обнаружением новых знаний: выявленных зависимостей развития заболеваний от категорий и характеристик пациентов, опубликованных и формализованных в соответствии с онтологией. Для управления БЗ применяется добавление варианта течения заболевания к уже ранее сформированным в той структурированной БЗ, которая будет проверяться на эталонах (применяются автоматизированные методы дальнейшей проверки на существующем контрольном наборе ЭИБ). После положительных результатов проверки на контрольном наборе ЭИБ, полученная БЗ с новым вариантом («веткой»в структурированной БЗ) поступит управляющему на замену прежней версии.

Управлять БЗ лечения требуется в связи с обнаружением новых знаний: новых методов лечения или сведений о новых влияниях лекарственных средств на пациентов с конкретными характеристиками. Для управления БЗ применяется добавление варианта лечения заболевания или условий для такового в сформированную структуру модели, схемы и вида лечения. Методы проверки предлагаемого лечения на эталонах (контрольном наборе ЭИБ) не позволяют делать вывод о том, что лечение неправильно, если ни в одном эталоне не применялся новый метод. Но требуется при проверке, чтобы обновленная БЗ позволила сформировать решение с вариантами, один из которых указан в эталоне.

5. Материалы для тестирования и методология проведения тестирования

На этапе аттестационного тестирования для проверки правильности и достаточности БЗ используется инфоресурс с эталонным набором реальных случаев (ЭИБ или ЭМК). Наиболее эффективно соединить в этом наборе случаи из архивов экспертов, разработчиков и пользователей-врачей.

На потоке таких случаев из практики ожидается, что истинный (задокументированный в реальной ЭИБ) диагноз будет либо подтвержден сервисом, либо не опровергнут (с преобладающим количеством подтвержденных необходимых признаков). Для случаев с подтвержденным

(не опровергнутым) диагнозом тестировщик ожидает, что сервисом будет предложен вариант лечения, соответствующий прописанному этому пациенту в реальности (назначение всех таких же лекарственных средств либо аналогов из такой же фармацевтической группы). В связи с обнаружением случая из практики (прецедента), не соответствующего (или противоречащего) явно описанным экспертным знаниям в конкретной БЗ (или совокупности ЭИБ, демонстрирующих развитие болезни отличное от предлагаемого сервисом на основе БЗ), экспертом принимается решение либо о переносе ее/их в *базу особых случаев* (для использования в поиске «методом близости»), либо – в обучающую выборку (для применения методов индуктивного (до)обучения). В некоторых случаях эксперт может принять решение о дополнении БЗ новыми формализованными знаниями.

Для проверки работы сервиса ССЗ создан архив ЭИБ с эталонным набором реальных случаев (рисунок 5), взятых из открытых публикаций из интернет-пространства (на таких ресурсах как: `www.lvrach.ru`, `https://cyberleninka.ru`, `http://heart-master.com`, `https://www.heartj.asia` и др.). 10 случаев из составленной коллекции выгружены как набор данных (в json-формате) на `https://disk.yandex.ru/d/BF0mSxMufjn0Rg`).

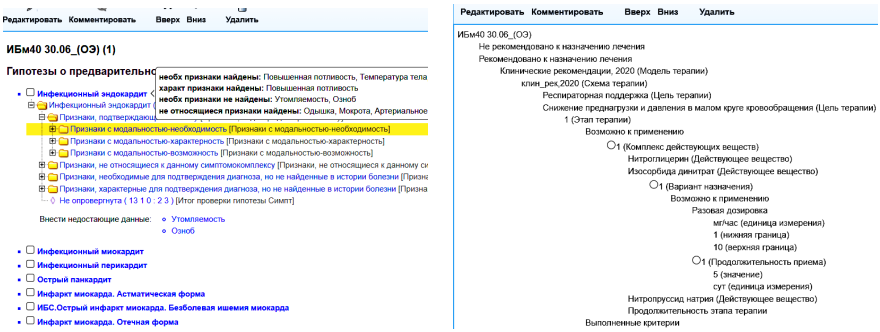


Рисунок 5. Фрагменты диалога с врачом на разных этапах принятия решения

Пример 1. *В литературе обнаружен случай из практика «Острый миокардит под маской инфаркта миокарда с подъемом сермента ST»* [19], *для такого варианта течения заболевания был сформирован новый симптомокомплекс с описанием добавочных значений признаков (жалоб, объективного обследования, данных лабораторных и инструментальных исследований). Затем история болезни переведена в ЭИБ и прошла*

проверку на сервисе. Система предложила две гипотезы о предварительном диагнозе, после введения дополнительных данных система провела дифференциальную диагностику с объяснением, тестирование постановки диагноза прошло успешно. Следующий этап тестирования связан с рекомендуемой терапией. При клиническом соответствии знаний и данных истории болезни предлагается схема терапии, включающая название лекарственного средства, его дозировку с описанием режима приема и продолжительности его использования.

Пример 2. *На медицинском научно-практическом портале* Lvrach.ru URL *описан вариант атипичного течения миокардита «Трудный диагноз. Острый инфаркт миокарда или миоперикардит?», после тестирования на сервисе этот вариант был добавлен в* базу *особых случаев.*

На найденных в открытых публикациях реальных случаях с пятью миокардитами и пятью другими ССЗ (размещенных в json-наборе данных) с помощью сервиса получены такие результаты: 4 диагноза подтверждены, 5 случаев получили несколько гипотез, включая истинный диагноз (они имели часть подтвержденных признаков, остальные признаки были запрошены), один случай оказался с нетипичной клинической картиной (и был размещен в базе особых случаев, также используемой для поддержки решений врача); для шести было предложено лечение, аналогичное контрольному – совпали фармакологические группы лекарственных средств, а для трех описанных реальных случаев было недостаточно данных для персонификации лечения.

Апробированные случаи были также использованы для сравнения возможностей других пяти доступных сервисов помощи врачебных решений. Почти ни в одном из них нельзя было провести полноценную диагностику, так как их словарь признаков минимален (и слабо расширяется, как следует из повторяемых экспериментов с интервалами 3-6 месяцев). Нет возможности обращаться к документу, приходится тратить время на новый повторный ввод. В истории болезни с числом признаков от 12 до 35 вводить удавалось только их часть (от 25 до 80%), а истинный диагноз вошел в топ-5 только в шести случаях. Объяснения либо нет, либо им можно считать вопросы с ответом «да», в лучшем случае выводится перечень тех указанных отклонений он нормы, которые присущи гипотезе. Помощь в лечении заболеваний сердца генерирует лишь один из пяти сервисов. Результат отличается от предложенного в эталонном прецеденте.

Объяснения нет. Таким образом сделан вывод, что доступные сервисы не интегрированы с медицинскими документами, Не дают подробных объяснений, а верные гипотезы выдают для случаев, которые очевидны или хорошо описаны в рекомендациях. Адаптация к новым знаниям имеет место в двух сервисах, расширение словаря признаков отсутствует.

6. Оценки трудозатрат

Онтология знаний о диагностике заболеваний является результатом работы специалистов в течение нескольких лет, а после успешной апробации в исследовательских коллективах стала основой для производства СППВР по различным нозологиям. На формирование по ней базы знаний по воспалительным заболеваниям сердца (миокардитов, перикардитов, эндокардитов и др.), включая поиск в литературе особых, сложных случаев, а также создание эталонных ЭИБ для проверки качества знаний потребовалось 5 человеко-месяцев. Поддержка данной БЗ должна осуществляться по мере обновления клинических рекомендаций Минздрава и появления авторитетных источников о методах диагностики и лечения.

Параллельно на основе данной онтологии создаются базы знаний для других заболеваний, многие рекомендовано объединять для расширения возможностей дифференциальной диагностики. В частности, ранее был сформирован блок знаний по инфарктам и стенокардиям.

Регулярно (не реже 2 раз в месяц) 2 специалиста участвуют в развитии (поддержании актуальности и усовершенствовании) базы знаний: поиск обновления для знаний (например, дополнительных вариантов проявления или течения болезни), поиск эталонных случаев из практики, проверка качества решений на найденных эталонах, добавление новых знаний, проведение процедуры уточнения знаний. Разработанные онтологии не зависят от нозологий. На их основе к настоящему времени разработано большое количество баз знаний, например, по группе вирусных заболеваний, желудочно-кишечных, орфанных и др. Принципиально важно, что решатель и интерфейс также не зависят от нозологий, а это означает, что не требуется модификация программного кода при дополнении/изменении базы знаний, что значительно затраты как на их создание, так и сопровождение.

В случае автоматизации деятельности медицинских коллективов через «облачные» системы врачам придется вводить все данные в предложенную

СППВР вручную. Однако этот процесс поддержан, все требуемые термины для описания состояния пациента поддерживаются строкой быстрого ввода по соответствующим разделам ЭИБ; для интеграции Медицинских информационных систем с электронными медкартами нужна государственная поддержка.

В облачной парадигме ожидается экономия в разы за счет единого центра обновления знаний и ее регулярных проверок на единой накапливаемой *базе эталонов*. Эксперты могут управлять качеством баз знаний посредством «облачно» доступных инструментов.

Заключение

В работе описан метод реализации и основные принципы создания сервиса диагностики, включая дифференциальную диагностику, и планирования лечения пациентам с воспалительными заболеваниями сердца. Особенностью сервиса является генерация детализированных объяснений.

В основе реализации сервиса лежит онтологический подход. Онтологические базы знаний по диагностике и лечению реализованы на основе соответствующих онтологий, созданных ранее коллективом и прошедших апробацию при реализации баз знаний для других нозологий. Выбор метода реализации (основанный на онтологических базах знаний) обусловлен, во-первых, их свойством быть интерпретируемыми для выбора аргументов выдаваемым гипотезам, во-вторых, необходимостью их оперативного изменения (без изменения кода решателя) для приведения в соответствие новым версиям клинических рекомендаций Минздрава. Более того, на основе соответствующих онтологий апробирована возможность интеграции IACPaaS сервисов с различными МИС (информационными системами): делался конвертор данных из онтологической истории болезни для передачи данных.

Сформированные базы знаний могут быть модернизированы, например, расширены экспертами. Инфраструктура платформы предлагает инструментарий для этого и поддерживает замену компонентов-знаний в уже эксплуатируемой СППВР. Хотя такая замена технически на платформе IACPaaS осуществляется «мгновенно», перед передачей в эксплуатацию обновляемый сервис с БЗ проходит тестирование на наборе эталонных случаев. Таким образом реализация сервиса на платформе IACPaaS обеспечивает возможность его непрерывного развития.

В настоящее время сервис размещен на платформе https://iacpaas.dvo.ru, логин – medicine-services@mail.ru (пароль высылается по запросу автору статьи или администратору IACPaaS.

Список литературы

[1] Иванов Д. О., Орел В. И., Александрович Ю. С., Пшениснов К. В., Ломовцева Р. Х. *Заболевания сердечно сосудистой системы как причина смертности в Российской Федерации: пути решения проблемы* // Медицина и организация здравоохранения.– 2019.– Т. **4**.– № 2.– С. 4–12. ↑142

[2] Karatzia L., Aung N., Aksentijevic D. *Artificial intelligence in cardiology: Hope for the future and power for the present* // Frontiers in Cardiovascular Medicine.– 2022.– Vol. **9**.– id. 945726. ↑142

[3] Шляхто Е. В., Звартау Н. Э., Виллевальде С. В., Яковлев А. Н., Соловьева А. Е., Алиева А. С., Авдонина Н. Г., Медведева Е. А., Федоренко А. А., Кулаков В. В., Карлина В. А., Ендубаева Г. В., Зайцев В. В., Соловьев А. Е. *Система управления сердечно-сосудистыми рисками: предпосылки к созданию, принципы организации, таргетные группы* // Российский кардиологический журнал.– 2019.– Т. **24**.– № 11.– С. 69–82. ↑144

[4] Kang S.-H., Baek H., Cho J., Kim S., Hwang H., Lee W., Park J. J., Yoon Y. E., Yoon C.-H., Cho Y.-S., Youn T.-J., Cho G.-Y., Chae I.-H., Choi D.-J., Yoo S., Suh J.-W. *Management of cardiovascular disease using an mHealth tool: a randomized clinical trial* // npj Digit. Med.– 2021.– Vol. **4**.– id. 165.– 7 pp. ↑144

[5] Алиева А. С., Павлюк Е. И., Алборова Э. М., Звартау Н. Э., Конради А. О., Катапано А. Л., Шляхто Е. В. *Системы поддержки принятия решений при нарушениях липидного обмена: актуальность, перспективы* // Российский кардиологический журнал.– 2021.– Т. **26**.– № 6.– С. 124–127. ↑144

[6] Lin Z., Cheng Y. T., Cheung B. M. Y. *Machine learning algorithms identify hypokalaemia risk in people with hypertension in the United States National Health and Nutrition Examination Survey 1999–2018* // Ann. Med.– 2023.– Vol. **55**.– No. 1.– id. 2209336.– 13 pp. ↑144

[7] Choi D. J., Park J. J., Ali T., Lee S. *Artificial intelligence for the diagnosis of heart failure* // npj Digital Medicine.– 2020.– Vol. **3**.– No. 1.– id. 54.– 6 pp. ↑144

[8] Assadi A., Laussen P. C., Freire G., Ghassemi M., Trbovich P. *Decision-centered design of a clinical decision support system for acute management of pediatric congenital heart disease* // Frontiers in Digital Health.– 2022.– Vol. **4**.– id. 1016522.– 11 pp. ↑144

[9] Лямина Н. П., Котельникова Е. В. *Система поддержки принятия решений как компонент пациент-ориентированной модели кардиологической реабилитации* // Доктор.Ру.– 2017.– № 5 (134).– С. 42–46. ↑144

[10] Groenhof T. K. J., Rittersma Z. H., Bots M. L., Brandjes M., Jacobs J. J. L., Grobbee D. E., van Solinge W. W., Visseren F. L. J., Haitjema S., Asselbergs F. W., et Members of the UCC-CVRM Study Group *A computerised decision support system for cardiovascular risk management 'live' in the electronic health record environment: Development, validation and implementation — the Utrecht Cardiovascular Cohort Initiative* // Netherlands Heart Journal.– 2019.– Vol. **27**.– Pp. 435–442. doi ↑144

[11] Авдонина Н. Г., Болгова Е. В., Ионов М. В., Звартау Н. Э., Конради А. О. *Результаты применения системы поддержки принятия решений в лечении артериальной гипертензии контроль корректности ввода данных в электронную историю болезни* // Артериальная гипертензия.– 2018.– Т. **24**.– № 6.– С. 704–709. doi ↑144

[12] Pieszko K., Hiczkiewicz J., Budzianowski J., Musielak B., Hiczkiewicz D., Faron W., J. Rzeźniczak, Burchardt P. *Clinical applications of artificial intelligence in cardiology on the verge of the decade* // Cardiology Journal.– 2021.– Vol. **28**.– No. 3.– Pp. 460–472. doi ↑144

[13] Chen X., Jia Sh., Xiang Y. *A review: Knowledge reasoning over knowledge graph* // Expert Syst. Appl.– 2020.– Vol. **141**.– id. 112948.– 21 pp. doi ↑144

[14] Грибова В. В., Петряева М. В., Окунь Д. Б., Шалфеева Е. А. *Онтология медицинской диагностики для интеллектуальных систем поддержки принятия решений* // Онтология проектирования.– 2018.– Т. **8**.– № 1(27).– С. 58–73. doi ↑144, 148

[15] Грибова В. В., Окунь Д. Б., Петряева М. В., Шалфеева Е. А. *Инфраструктура IACPaaS для формирования интерпретируемых баз диагностических знаний по заболеваниям произвольной направленности* // *Семнадцатая Национальная конференция по искусственному интеллекту с международным участием*, Сборник научных трудов.– Т. 2, КИИ-2019 (21–25 октября 2019 г., г. Ульяновск, Россия), Ульяновск: УлГТУ.– 2019.– ISBN 978-5-9795-1940-1.– С. 81–89. ↑145

[16] Сергеева В. А., Липатова Т. Е. *Миокардит при инфекции COVID-19: патогенетические механизмы, сложности диагностики (обзор)* // Саратовский научно-медицинский журнал.– 2021.– Т. **17**.– № 3.– С. 571–577. URL ↑147

[17] Котова Е. О., Писарюк А. С., Кобалава Ж. Д., Тимофеева Ю. А., Чипигина Н. С., Караулова Ю. Л., Ежова Л. Г. *Инфекционный эндокардит и COVID-19: анализ влияния инфицирования SARS-CoV-2 на особенности диагностики, течения, прогноз* // Российский кардиологический журнал.– 2023.– Т. **28**.– № 1.– С. 28–42. doi ↑147

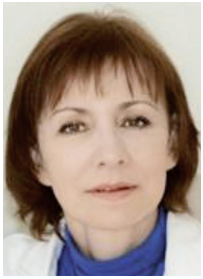
[18] Окунь Д. Б., Ковалев Р. И. *База знаний лечения миокардита: представление знаний для дифференцированной этиотропной терапии // Материалы XV международной научной конференции «Системный анализ в медицине»*, САМ 2021, ред. В. П. Колосов, Благовещенск: ДНЦ ФПД.– 2021.– ISBN 978-5-905864-24-7.– С. 53–56. ↑148

[19] Андреев Д. А., Фирсакова В. Ю., Дорохова О. В., Масленникова О. М. *Острый миокардит под маской инфаркта миокарда с подъемом сегмента ST* // Вестник Ивановской медицинской академии.– 2014.– Т. **19**.– № 4.– С. 64–68. ↑157

Поступила в редакцию 25.09.2023;
одобрена после рецензирования 27.10.2023;
принята к публикации 27.11.2023;
опубликована онлайн 31.12.2023.

Рекомендовал к публикации *к.т.н Я. И. Гулиев*

Информация об авторах:



Валерия Викторовна Грибова

Заместитель директора по научной работе, научный руководитель лаборатории интеллектуальных систем Института автоматики и процессов управления Дальневосточного отделения РАН, д.т.н., чл.-корр. РАН. Научные интересы: онтологии и базы знаний, прикладные и проблемно-ориентированные системы, основанные на знаниях, управление базами знаний

0000-0001-9393-351X

e-mail: gribova@iacp.dvo.ru

Елена Арефьевна Шалфеева

Ведущий научный сотрудник лаборатории интеллектуальных систем Института автоматики и процессов управления Дальневосточного отделения РАН, д.т.н. Научные интересы: онтологический инжиниринг, интерпретируемые клинические руководства, технология создания систем с декларативными знаниями, объяснительный искусственный интеллект, управление базами знаний

0000-0001-5536-2875

e-mail: shalf@dvo.ru

Маргарита Вячеславовна Петряева

к.м.н. (2001), научный сотрудник лаборатории интеллектуальных систем Института автоматики и процессов управления Дальневосточного отделения РАН. Научные интересы: онтологии и базы знаний, медицинские интеллектуальные системы. Является автором 92 научных работ

0000-0002-1693-4508

e-mail: margaret@iacp.dvo.ru

Дмитрий Борисович Окунь

к.м.н., Научный сотрудник лаборатории интеллектуальных систем Института автоматики и процессов управления Дальневосточного отделения РАН. Окончил Владивостокский государственный медицинский университет по специальности «Лечебное дело»

0000-0002-6300-846X

e-mail: okdm@iacp.dvo.ru

Леонид Александрович Федорищев

к.т.н. (2013). Старший научный сотрудник лаборатории интеллектуальных систем Института автоматики и процессов управления ДВО РАН, доцент ВГУЭС. В списке научных трудов более 40 работ

0000-0002-2049-2570

e-mail: fleo1987@mail.ru

Роман Игоревич Ковалев

Научный сотрудник лаборатории интеллектуальных систем Института автоматики и процессов управления Дальневосточного отделения РАН. Научные интересы: онтологии и базы знаний, интеллектуальные системы

0000-0002-1704-2675

e-mail: koval-995@mail.ru

Все авторы сделали эквивалентный вклад в подготовку публикации.

Авторы заявляют об отсутствии конфликта интересов.

PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article INFORMATION TECHNOLOGY IN MEDICINE

UDC 004.89:616.1
10.25209/2079-3316-2023-14-4-141-188

Cloud service for differential diagnosis and personalized treatment of inflammatory heart diseases

Valeria Viktorovna **Gribova**[1], Elena Arefyevna **Shalfeeva**[2], Margarita Vyacheslavovna **Petryaeva**[3], Dmitry Borisovich **Okun**[4], Leonid Aleksandrovich **Fedorishchev**[5], Roman Igorevich **Kovalev**[6]
Institute of Automation and Control Processes of the Far Eastern Branch of the RAS, Vladivostok, Russia
koval-995@mail.ru

Abstract. The aim of this research is to automate the process of differential diagnosis and prescription of personalised treatment of inflammatory heart diseases. The main method of research is the formation of declarative knowledge bases and a solver explaining its results. The ontology solver interprets the formalized knowledge. Main principles of development and the conceptual architecture of an intelligent service with declarative knowledge are described. Information components for inflammatory heart diseases and software components for the differential diagnostics and personalised treatment are formed. The clinical decision support system for cardiology with the properties of explainable artificial intelligence has been developed. It is hosted on the medical portal of the cloud platform IACPaaS. The practical significance of the results lies in the fact that the system is hosted on a platform with free registration. For the confidence of practitioners, knowledge sources are indicated and testing is performed on data described in the literature. The service is available for experimentation on the role of declarative knowledge base in real-life situations where a third opinion is required. The method used is also practically significant - the rationale for the choice of technologies and algorithms is given, the requirements for the software complex service are identified and formulated, and the steps for the development of all components are outlined. This allows us to reproduce the proposed solution on the same principles of explainable artificial intelligence, but with other knowledge and on another platform.

Key words and phrases: clinical decision support system, intelligent service, inflammatory heart diseases, knowledge base, ontological approach, specialized solver

2020 *Mathematics Subject Classification*: 68T30; 92C50

Acknowledgments: The work was carried out with financial support from the state assignment of the Ministry of Education and Science: 0202-2021-0004

For citation: Valeria V. Gribova, Elena A. Shalfeeva, Margarita V. Petryaeva, Dmitry B. Okun, Leonid A. Fedorishchev, Roman I. Kovalev. *Cloud service for differential diagnosis and personalized treatment of inflammatory heart diseases*. Program Systems: Theory and Applications, 2023, **14**:4(59), pp. 141–188. https://psta.psiras.ru/read/psta2023_4_141-188.pdf

Introduction

Diseases of the cardiovascular system are the main cause of mortality in all categories of the adult population: according to ROSSTAT, there has been a steady increase in the number of cases, especially in recent years [1, 2]. Diagnosis and treatment of inflammatory heart diseases (myocarditis, pericarditis, endocarditis, etc.) remains one of the most difficult sections of the work of therapists and cardiologists due to the heterogeneity and non-specificity of clinical manifestations. Latent, chronic or atypical course of diseases further increases the difficulty of differential diagnosis. A number of diagnostic algorithms have been proposed to assess the significance of the signs on the basis of which the diagnosis is based. The practice and experience of a particular doctor may not be enough to timely identify any problem that has arisen in the human body.

Decision support systems with access to a huge amount of data, advanced scientific literature and millions of case histories could help to quickly classify a difficult case, propose and justify solutions at any stage of interaction with the patient (prevention, diagnosis, treatment or rehabilitation). The benefits of implementing decision support systems in the medical area are primarily expected in the early suspicion of an incipient disease, an increase in the accuracy of diagnosis, and personification of treatment of an established disease. Therefore, the development of clinical decision support systems (CDSS) using modern artificial intelligence methods is a priority task for many countries around the world today.

In recent years, with the choice of decision support systems for implementation in the medical field, interest in explainable AI has grown. Doctors with medical responsibility can hardly trust the system without an explanation of the underlying decision-making process.

The aim of this research is to develop an intelligent service for the diagnosis and treatment of inflammatory heart diseases, differential diagnosis of inflammatory with other heart diseases, explaining their hypotheses, because it is service that provides explanations that can become a useful electronic consultant for a modern cardiologist.

To achieve this goal, we have to solve the following tasks:

- to choose sources of knowledge on inflammatory heart diseases,
- to choose methods of forming declarative knowledge bases,
- to develop knowledge bases for the diagnosis, differential diagnosis of inflammatory heart diseases, for the appointment of treatment and its correction, if necessary, in the process of dynamic monitoring of the patient's treatment process.
- to choose a method for implementing a solver and architecture of a framework with a solver for constructing a service based on declarative knowledge base,
- to develop the user interface of an intelligent service (GUI),
- to implement a software (intelligent service),
- to form the dataset for testing the knowledge base and the service as a whole,
- to choose a methodology for testing.

Close research

Automation of support for the detection or diagnosis of certain inflammatory heart diseases (myocarditis, pericarditis) is offered in numerous online symptom checkers[1] [2] [3] [4], knowledge on some other inflammatory heart diseases (coronary and lymphocytic endocarditis) is rarely embedded in such checkers. Also, checkers do not provide the possibility of combined diagnosis of such diseases and support for their treatment. It is due to the simplified representation of the rules of clinical decisions, as well as the presentation of the clinical picture of diseases and diagnostic rules without taking into account the temporal aspect. Such software products have proven themselves well in the calculation and

[1] *Symptom Checker* is developed by a group of professional doctors of various specialties of the Saint Antipas Polyclinic and programmers

[2] *Healthdirect. Symptom cheker* helps Australians to find proper healthcare provider

[3] "Artificial intelligence *Kiberis* - individual medicine" is a medical assistant based on artificial intelligence for diagnostics and personalized therapy, selection of drug analogs, checking the safety of prescriptions and auto-filling a medical record.

[4] *«ASCVD Risk Estimator Plus»* of American cardiology college.

analysis of certain risks in cardiology [3–5]. Representatives such as [6] use machine learning algorithms to identify signs and predict cardiovascular events.

At present, CDDS have been developed for the diagnosis and treatment of certain heart diseases such: as coronary heart disease [7–9], as a component of a patient-oriented model of cardiological rehabilitation [10], as the diagnosis and treatment of arterial hypertension [11]. As for inflammatory heart diseases, we can focus on such systems as: "Infectious endocarditis", which uses a database of clinical cases of endocarditis, as well as machine learning algorithms to predict the risk of complications and choose the most effective treatment; Cardio-ECO evaluates the degree of damage to the heart valves in endocarditis using echocardiography data. A prototype of the DDS for the management of patients with infectious endocarditis [12], based on the integration of hypertext and knowledge, where qualitative data were analyzed by constant comparison. The use of the system has shown that such solutions are able to provide support for a specific patient to confirm clinical decisions and manage therapy at a higher level.

However, complex knowledge models corresponding to the system of concepts of cardiologists are required for the intellectual support of differential diagnosis and the appointment of personalized treatment of inflammatory heart diseases. Reasoning based on knowledge graphs has become widely used to support and automatically make decisions [13, 14] with the development of graph representations of knowledge. In addition to graph representation of knowledge, it is important to integrate their interpreters with medical documents such as personal health records or electronic health records (EHR). Symptom checkers do not provide this opportunity.

Machine learning-based systems do not provide explanations, however there is some interpretation of the result obtained in some cases. Some symptom checkers form a brief explanation according to their simplified knowledge model.

Requirements for the CDSS

A modern electronic consultant for a doctor will be able to become a software service for process of a full diagnosis and appointment of personalized treatment of inflammatory heart diseases that will to:

- use data from the EHR or similar document about the patient,
- use a trustworthy knowledge base,
- offer reasonable hypotheses about the preliminary diagnosis of the disease,
- offer after the preliminary diagnosis a request (list) of additional laboratory and instrumental examinations,
- carry out differential diagnostics of inflammatory heart diseases among themselves, as well as with other heart diseases,
- offer personalized treatment options with the most favorable prognosis and explain such hypotheses about the solution,
- form an explanation of the main diagnosis of the disease, or request additional observations for further examination or clarification.

1. Methods of developing knowledge bases

For the formation of declarative knowledge bases the ontological approach was used, which provides:

- reuse of software solvers for medical diagnostics and treatment,
- multivariant (i.e. different ways) and collective work on new versions of knowledge bases,
- «objectification» of verification of correctness and accuracy of a knowledge base,
- «comprehensibility» of knowledge represented in the knowledge base.

To form knowledgebase for diagnosis and differential diagnosis, personalized treatment requires either to choose the ready-made ontologies and make sure they are sufficient or create the new ones. Usually in a given domain there is a single ontological model or set of ontologies for different tasks. The medical portal of the IACPaaS [15] platform has been operating

ontology for diagnosis of diseases and ontology of personalized treatment for more than 10 years. The ontological base is filled according to the ontological structure (representation model) and prescribed integrity limitations.

The ontological knowledge base is filled up in various ways. The knowledge described in the literature (from documents, standards, textbooks, scientific articles) can be extracted automatically. This can also be done manually by experts or engineers using specialized (ontological) editing tools. Correctness, completeness and other properties of the knowledge base quality depend on selected sources of knowledge, as well as on methods and tools of formation. In particular, manual filling involves a human factor (on the one hand, it is more conscious, responsible: at the same time an analysis of the relevance of each element is performed, on the other hand errors are expected).

The automatic extraction of patterns and associative relationships from datasets having a tabular appearance is technically attractive, but there are limitations:

(*a*) the amount of knowledge extracted is many times less than contained in textbooks (even for simple enumerated features)

(*b*) datasets do not cover the description of the process of disease development, options for controlled recovery.

Since none of the methods individually provides the completeness of knowledge, it is important to combine at least two options for the formation of the knowledge base. The cycle of forming and updating the knowledge base includes the stages of testing, verifying correctness on a reference set of use cases, that regularly expanded.

join nowledge bases can be filled according to the needs of their users. Two options for creating a new base for a new service are common: formation of knowledge on a specific set of diseases or combination knowledge base from previously described diseases and newly created ones.

The IACPaaS platform's medical portal provides appropriate interpreting of formalized knowledge in accordance with agreements on the rules for solving these problems in medicine. At the same time, in the case of

diagnosis, it is "advantageous" to work with complex knowledge base: already accumulated formalized descriptions of diagnostic knowledge of a number of diseases expand the possibilities of differential diagnosis of "new" diseases with previously accumulated ones. In all cases, knowledge interpretation is subject to testing (for example, on a set of reference cases).

2. Developing of knowledge base

2.1. Sources of knowledge of inflammatory heart diseases

Officially approved clinical recommendations and methodological guidelines of the Ministry of Health of the Russian Federation, as well as educational medical literature and scientific articles were used as a source of knowledge, since clinical cardiologists use it in their work [16, 17]. Recognition of such texts with full extraction of diagnostic signs and treatment methods automatically does not yet provide full extraction of the "medical meaning" available in them, especially the dynamics of the process, the course of diseases, experts on the formalization of medical knowledge were involved. Subsequent versions of the knowledge base will be created when trustworthy datasets appear, when the Ministry of Health updates clinical recommendations, as well as when interacting with experts from clinical practice interested in formalizing their experience.

2.2. Content of knowledge bases

Information resources of the service are represented by the following components: the Database of medical terminology and observations, the Knowledge base of diagnosis of inflammatory heart diseases, the Database of Pharmacological Reference, the Database of the International classifier of diseases (ICD-10), the Knowledge base on treatment of diseases, the Archive of health records. Each information resource is a source of either terms or direct knowledge to diagnosis or treatment of inflammatory heart diseases. Knowledge bases have the semantic representation, formed with the help of the knowledge editor, it allows us to ensure their formation and modification directly by experts of knowledge. The database of medical terminology and observations has been expanded with signs and observations necessary for description of new inflammatory heart diseases.

For the development of knowledge bases the complex of ontologies IACPaaS was used: «Ontology of knowledge about diagnosis of diseases», «Ontology of treatment of diseases».

Formed the Knowledge base on the diagnosis of inflammatory heart diseases includes structured formalized knowledge about the following diseases: infectious myocarditis, infectious pericarditis, acute and subacute bacterial endocarditis, acute pancarditis (4506 concepts). For its diagnosis the following are described: the forms of the disease, possible degrees of severity, the variants of etiology. Additional symptom complexes with the description of specific features are formed for the possibility of detailing the diagnosis of diseases during the medical and diagnostic process and the formation of a developed clinical diagnosis. The IACPaaS ontology of diagnostic knowledge makes it possible to apply special (complex) cases with which doctors meet in clinical practice (and such symptom complexes have been formed into the knowledge base).

Knowledge base has been supplemented and extended by the information resource Knowledge base of diseases of the cardiovascular system for the possibility of differential diagnosis of inflammatory diseases with other heart diseases. It additionally includes descriptions of such diseases as: acute myocardial infarction, angina tension, unstable angina, hypertension, etc. This resource is used by earlier developed service «Diagnosis of diseases of the cardiovascular system» [**14**]. The description of the knowledge base for the diagnosis of diseases includes more than 5000 concepts.

For generation of recommendations on treatment the resource [**18**] Knowledge base on treatment of inflammatory heart diseases (1311 concepts) was formed. It combines fragments of the clinical picture of the disease, features of disease dynamics and treatment methods based on data on the clinical effectiveness of drugs.

This principle will allow us to combine the standard of disease treatment and individual treatment tactics of each patient, as well as to make changes in the treatment process when new clinically significant indicators appear. Prescription of personalized treatment and its correction is possible due to the description in the knowledge base of various therapy schemes of this group of diseases, taking into account the indications and contraindications

to the prescription of medicines, based on the individual characteristics of each person according to clinical, genetic, genomic and environmental factors.

3. The method of implementing the solver and the means of constructing the service

join nowledge engineering methods are used to automate the solution of intellectual problems with obtaining explanations of hypotheses. For the tasks of differential diagnosis and personalized treatment, ontology, firstly, reflects the structure of the relevant domain knowledge, and secondly, establishes strategies for finding a solution or forming a result. They may depend on the essence of the task, the role of systems with knowledge bases and user preferences (for example, to propose and justify the best solution, to propose all possible solutions, to criticize each user solution, etc.). A specific strategy for hypothesizing the diagnosed abnormal process requires fully confirming some variant of the development of the process or requires finding all non-rejected variants of the development of all abnormal processes. Search strategies are a type of ontological conventions for applying explicit knowledge to specific tasks. Decision-making support for a single task (or solution generation) is performed by an appropriate software reasoner, specialized in the ontology of knowledge for this task and ontological conventions. Such a specialized Solver interprets knowledge: bypasses the ontological knowledge base, in accordance with the conventions, putting forward or refuting hypotheses, argues and collects arguments in favor of some versions, hypotheses (and against others), forming a practically useful explanation. The content of one of the ontological agreements for treatment: "it is possible to eliminate the symptom (type of problem), confirmed by the current input data, if in the knowledge base among the many periods of development (type of consequences of the problem) expected after the current period there is a period in which the symptom will be normal (a component of the normal state), and types of impacts for such consequence are indicated in the knowledge base, and the possibility of these impacts is not rejected by the current input data.

The specialized reasoner to support the solution of the problem regarding the current situation generates an explanation in accordance with

the contents of the available ontological database and with agreements. An explanation with reference to possible and detectable causes, consequences, influencing factors, properties, conditions for connecting parts into a structure, etc. is useful to a qualified user. In this regard, the "most understandable" explanation will be one whose structure is close to the structure of knowledge (its analogue or its inversion) – and the essence of the used agreements. The reasoner specialized by ontology applies processing rules in the process of reasoning and searching for hypotheses: linking premises with consequences and searching for information to confirm the premises. Usually the processing has several stages. This is a search in the description of the current situation (input data) for facts to confirm the conditions for the appearance of processes (output to the report is an explanation of each confirmed condition of the process being checked and all the facts corresponding to the confirmed condition); search in the description of the current situation for facts to confirm the process flow variant (output to the report is an explanation of each verifiable flow process variant and all the facts corresponding to it); hypotheses based on confirmed and unverified premises.

When creating algorithms, information processing is distributed to work on the knowledge base, work with facts, work on fixing arguments-explanations in the explanation report. For example, to create an algorithm for testing a hypothesis about a variant of development and substantiating a variant, one software unit works on the causal relationship "variant of development - signs" and turns to others: checking the presence of a sign (among the facts), checking the hypothesis of the correspondence of the sign value, establishing the status of the verified sign in the "explanation", establishing the status of the verified hypotheses, etc. The program unit, processing its own types of connections between information elements, makes intermediate conclusions of the logical inference process. In the solver algorithm, the procedures for deriving consequences from the processed parcels are alternately called, which record the result of checking the parcels in the explanation. The specialized reasoner was created on the IACPaaS cloud platform, (based on the ontology of knowledge and ontological conventions) and formed declaratively, integrated with the user interface (Figure 1).

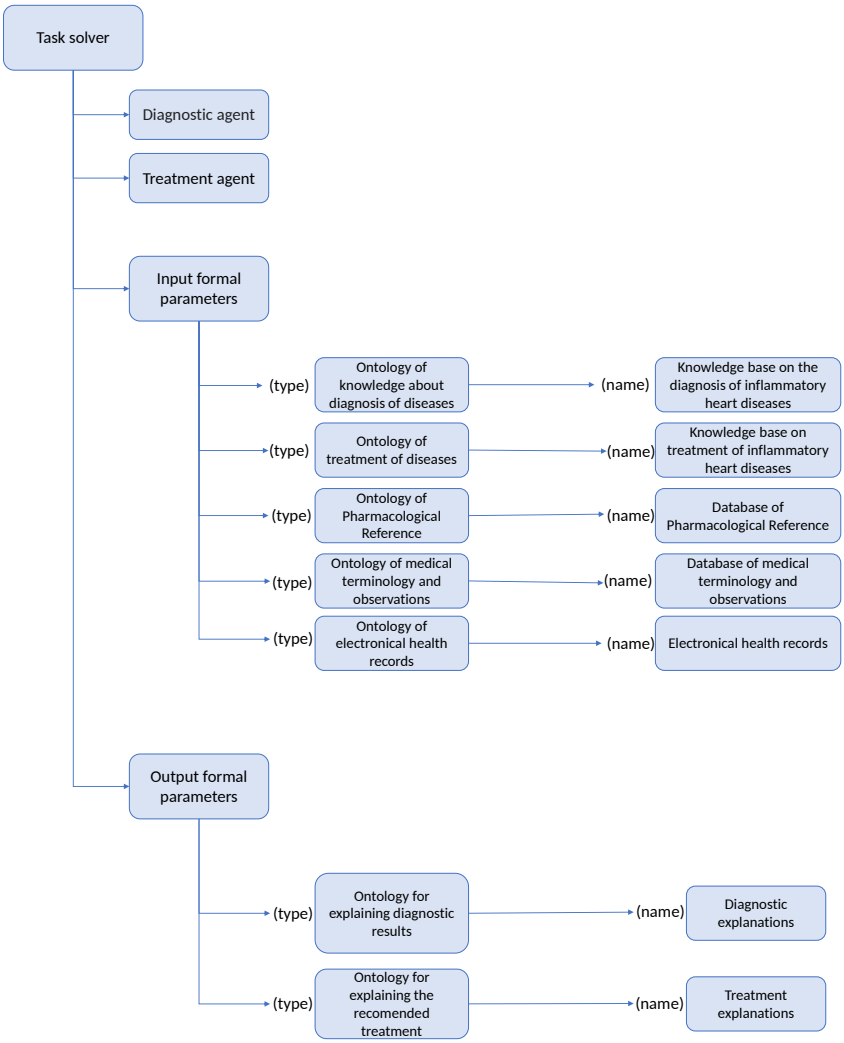


FIGURE 1. Scheme of a specialized reasoner

The reasoner was developed using an agent-based approach: the root agent provides interface functionality and user interaction with the system, and is also responsible for calling agents to perform diagnostic and treatment subtasks. The formal parameters of the solver are the labels of

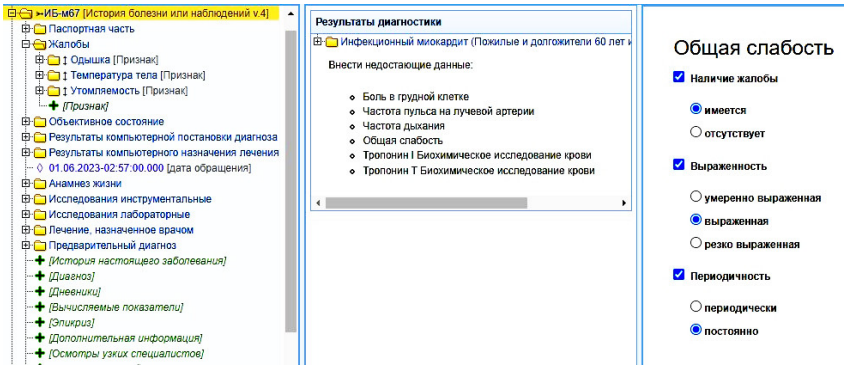


Figure 2. EHR input interface

the ontological knowledge bases and input information resources (sets of EHR).

3.1. GUI of Intelligent Service

The GUI of the service consists of three main parts:

- Interface for entering EHR
- Interface for viewing diagnostic results
- Interface for viewing recommended treatment

The interface for entering medical histories provides an opportunity to enter all available information on the disease (Figure 2).

The peculiarity of the interface for EHR is that it automatically forms and connects knowledge based on heterogeneous ontologies. This ensures a high speed of entering structured case histories based on many different large formalized databases of medical knowledge. Let's take a closer look at how this happens in practice.

Figure 3 shows (in a slightly simplified way) two ontologies that need to be correlated with each other to automate the entry of a sign from the dataset of EHR. On the one hand, these structures are quite similar, because, in fact, they are a description of a feature and, in particular, even have completely identical elements. But, on the other hand, these

FIGURE 3. Difference of structure (ontology)

structures are far from a complete coincidence: so in the ontology on the right you can see:

(*1*) some auxiliary elements ("Type of possible values"),

(*2*) division of the attribute into subtypes ("Simple attribute", "Composite attribute").

These structures were not designed the same during ontological engineering, although both are responsible for the essence of "sign", since they are intended for different purposes: the structure of a sign in the EHR is aimed at introducing signs with specific meanings, while in the Database of medical terminology and observations – to describe all variants of the values of the sign, synonymy, normal and reference ranges. For example, the sign "Body temperature" in the EHR will have some specific value, for example, 37.0, and in the Database of medical terminology and observations it will have ranges of values (for example, 35-42). In addition, the idea of automating the creation of the interface is also to try not to depend on the exact coincidence of data structures, but, on the contrary, to provide maximum flexibility in the ability to connect arbitrary knowledge bases with their mismatched structures. Thus, in this task we have two heterogeneous knowledge structures that still need to

be automatically correlated with each other. To implement this task, it is proposed to introduce an additional structure – a table in which the necessary special relationships will be described to establish correspondence between heterogeneous ontologies.

Each Element in this structure describes an element of the initial ontology that must be transformed for the structure of the final ontology. The "Goal" field describes how the "Element" of the initial ontology should be represented in the final one. The following options are possible: if the "Goal" contains a non-null string value, then the element of the initial ontology is renamed according to this value. If the "Goal" does not contain any value or contains the value "null", then this means that the Element of the initial ontology must be "cut out" from the structure of the final ontology. At the same time, "cut out" an element does not mean deleting this element, but only hiding its title, all successor elements of this Element become successor of its parent element (the hierarchical structure of the tree changes). The "Succerssor element" field, on the contrary, serves to ensure that the child element that the specified Element has appears in the final ontology. At the same time, it appeared not as an empty new element, but as an intermediate parent node between the current elements in the final ontology (i.e., the hierarchical structure of the tree is changing again).

To view in the EHR all the entered signs and their characteristics, a hierarchical unfolding tree is provided in the interface. Some branches of this tree are interface-transformed into one, simplifying the visualization of the structure by hiding long chains of knowledge structures that do not affect the result. To introduce new knowledge, the user is provided with a search interface. The interface provides a mechanism for specifying the search by several key criteria-substrings entered through a space. The search can take into account both parts of feature names, their characteristics or string content, and auxiliary information about knowledge, for example, synonymy.

The interface for viewing diagnostic results displays diagnostic results in different forms, depending on the diagnostic results. In case of lack of any data or knowledge, the interface displays all the necessary information and provides an opportunity to enter additional data, automatically performing part of the search work for the user. If the information is

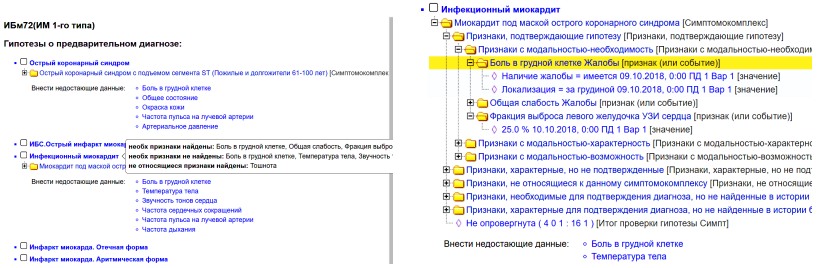


FIGURE 4. The fragments of a dialogue with a doctor when analyzing a possible solution

sufficient, the interface displays a complete picture of the diagnosis with detailed explanations for each criterion. The recommended treatment viewing interface displays treatment options in the form of a hierarchical tree in which you can see all the necessary information on treatment methods, medications, periods of their use, etc.

4. Construction an intelligent service with an evolving knowledge base

On the medical portal of the IACPaaS platform, intelligent services are built as an "assembly" of software components (a specialized solvers) with information (formalized knowledge in the form of knowledge base). Such assembly is based on the development from single ontology. Before linking, each component is tested in the IACPaaS portal with its own test suites. Checking diagnosis' knowledge base requires sets of EHR, where the patient has been successfully diagnosed. Correctness and accuracy are checked. The knowledge base check for personalized treatment requires such EHR kits where the patient has been cured or stabilized. Correctness and completeness are checked. After the assembly, the process of which takes a few minutes, the assembled service is tested on an existing control set of cases (EHR) with the solution of both required subtasks: correct diagnosis and successful treatment (see Figure 4). All participants are responsible for quality.

The possibility of supplementing and expanding knowledge bases, assessing their adequacy and relevance is due to the ontological approach,

which allows you to read and edit the knowledge base in familiar terms. It is required to manage diagnostics knowledge base in connection with the discovery of new knowledge: the revealed dependencies of the development of diseases on the categories and characteristics of patients, published (and formalized in accordance with the ontology). The adaptation of clinical system and the management of knowledge base is carried out directly through the transfer of new information into a structured knowledge base with its verification on the existing control set of EHR (or sets of precedents). The addition of a variant of a course of some disease to those already previously formed structured knowledge that will be tested on control set of EHRs is used (automated methods of further verification are used). After positive test results on the EHR control set, the received knowledge base with a new version ("branch" in the structured knowledge base) will be sent to the manager to replace the previous version.

It is necessary to manage treatment' knowledge base in connection with the discovery of new methods of treatment or information about new effects of drugs on patients with specific characteristics. To manage the knowledge base, the addition of a disease treatment option or conditions is used in the formed structure of the model, scheme and type of treatment. The methods of testing the proposed treatment on the EHR control set do not allow us to conclude that the treatment is incorrect if no new method was used in any EHR. But it is required during verification that the updated knowledge base allows you to form a solution with options, one of which is specified in the control case.

5. Testing materials and testing methodology

At the stage of certification testing an information resource "reference set of real cases" is used to verify the correctness and sufficiency of the knowledge base.

In this set, it is most effective to combine cases (EHR) from the archives of experts, and from developers and from medical users. In this set, it is most effective to combine cases – EHR from the archives of experts, and from developers and from medical users. Based on the flow of such cases (from practice), it is expected that the true diagnosis (documented in

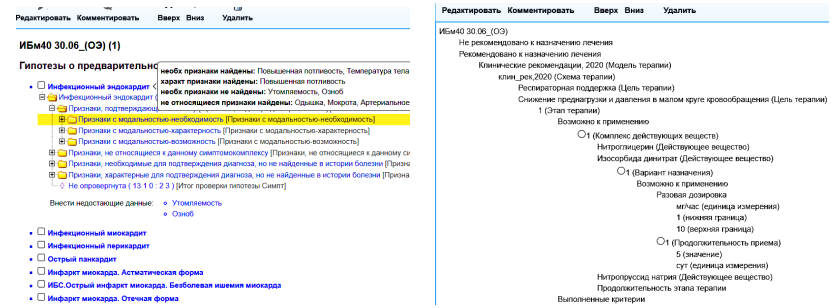


Figure 5. Fragments of dialogue with a doctor at different stages of decision-making

EHR) in every case will either be confirmed by the service or not refuted (with the predominant number of confirmed necessary signs). For the cases with a confirmed diagnosis, the tester expects that the service will prescribe the treatment option that corresponds to real case (prescribing all the same medicines or analogues from the same pharmaceutical group). In connection with revealing case from practice (precedent) that does not correspond (or contradict) the explicitly described expert knowledge in a particular knowledge base (or revealing such precedents set), the expert decides either to transfer it (they) to the database of special cases (for application in the search "by the proximity method"), or in training sample (for application in inductive methods). In some cases, the expert may decide to supplement the knowledge base with new formalized knowledge.

The next stage of testing is related to recommended therapy. With clinical compliance of knowledge and medical history data, a therapy scheme is proposed, including drug name, its dosage with a description of intake regimen and duration of its use.

To check the operation of the developed AI service, an EHR archive was created with a reference set of real cases (Figure 5) taken from open publications from the Internet space (on such resources as: www.lvrach.ru, https://cyberleninka.ru , http://heart-master.com , https://www.heartj.asia and others). 10 cases from the compiled collection are uploaded as a dataset (in json format) to https://disk.yandex.ru/d/BF0mSxMufjn0Rg).

The examples of real cases that influenced the improvement of service knowledge.

Example 1. *A case study "Acute myocarditis under the guise of myocardial infarction with ST segment elevation" was found in the literature* [**19**]. *The service recognized it as a case of a heart attack. Therefore, for this variant of disease course, a new symptom complex was been formed: with a description of additional values of signs (complaints, objective examination, laboratory and instrumental data). Then EHR was once again submitted to the service input. The system proposed two hypotheses about the preliminary diagnosis, after the introduction of additional data, the system performed a differential diagnosis with an explanation, testing of the diagnosis was successful.*

Example 2. *On the medical scientific and practical portal Lvrach.ru, a variant of the atypical course of myocarditis was found «Difficult diagnosis. Acute myocardial infarction or myopericarditis?». After using the case as a test case for the service with the knowledge base, it was decided to add this manifestation option to the database of special cases.*

Based on real cases with five myocarditis and five other cases of heart diseases found in open publications (posted in the json dataset), the following results were obtained using the service with knowledge base including different cardiovascular diseases, the following results were obtained. 4 true: 4 diagnoses were confirmed, 5 cases received several hypotheses, including the true diagnosis (they had some confirmed signs, the remaining signs were requested), one case turned out to have atypical clinical the picture (and was placed in the special cases database, also used to support the doctor's decisions); for six, treatment similar to the control cases was offered – pharmacological groups of drugs coincided, and for three of the described real cases there was insufficient data in EHR for personification of treatment.

The tested cases were also used to compare the capabilities of other five available medical assistance internet-services. Almost none of them could be fully diagnosed, because their vocabulary of observations is minimal (and not beeing expanded, as follows from repeated experiments at intervals of 3-6 months). There is no way to access EHR, you have to spend time on a new re-entry. For the cases with many observations (from 12 to 35), only a part of them (from 25 to 80%) could be inputed, and the true diagnosis was in the top 5 only in six cases. There are either no explanations, or they are questions with a "yes" answer; at best, a list of

those inputed deviations from the norm that are inherent in the hypothesis is displayed. Only one in five such services treats heart disease, but the result differs from that proposed in control cases. There is no explanation.

Thus, it was concluded that the available services are not integrated with medical records and do not provide detailed explanations, and the correct hypotheses are given for cases that are obvious or well described in the recommendations. Adaptation to new knowledge there is only in two services; there are no opportunities to expand the vocabulary.

6. Some labor cost estimates

The Ontology of knowledge about diagnosis of diseases was the result of specialists activity over several years, and after successful approbation in research teams, it became the basis for production of CDSS for various nosologies. It took five man-months to form a knowledge base on inflammatory heart diseases (myocarditis, pericarditis, endocarditis, etc.), including a search in the literature for special, complex cases, as well as the creation of reference EHRs to check knowledge quality. The knowledge base will change as clinical recommendations of the Russian Ministry of Health are updated. It can be supplemented as authoritative sources on new diagnostic and treatment methods become available.

In parallel, knowledge bases for other diseases are being created on the basis of this ontology. It is recommended to combine bases or blocks of knowledge on different diseases for the sake of differential diagnosis. The previously formed block of knowledge on heart attacks and angina pectoris was added to the knowledge on inflammatory heart diseases for this purpose.

Regularly (at least 2 times a month), 2 specialists participate in maintaining relevance and improving of the knowledge base: searching new knowledge (for example, additional variants of disease course manifestation), searching for control cases from practice, checking with their application the quality of solutions, conducting a knowledge refinement procedure.

Because solver and interface do not depend on nosologies, then when adding/changing the knowledge base, no modifications of the program code are required. Which is essential when systems operating.

In the case of automation of medical activities through the "cloud" systems, doctors will have to enter all the data into the proposed DSS. However, this process is supported, all the required terms for describing the patient's condition are supported by a quick entry line for the corresponding sections of the EHR. (For integration with electronic medical records of Medical information systems, state support is needed.)

In the cloud paradigm, savings are expected at times due to a single knowledge update center and its regular checks on a single accumulated cases base. The experts can manage the quality of knowledge bases through cloud-accessible tools.

Conclusion

The paper describes basic principles of creating and method of implementation of clinical DSS for differential diagnosis and treatment planning for patients with inflammatory heart diseases. The ability of the service to generate detailed explanations is ensured by its implementation based on an ontological approach.

Ontological knowledge bases for diagnosis and treatment have formed on the basis of corresponding ontologies previously created by the team and tested for other nosologies. The choice of this implementation method is due, also to the need for knowledge bases evolve or change (to bring them into line with the new versions of the clinical recommendations of the Ministry of Health) without changing the solver code. Moreover, on the basis of the corresponding ontologies, the possibility of integrating IACPaaS services with various medical information systems has been tested: a data converter was made from the ontological medical history for data transmission. The formed knowledge bases can be upgraded, for example, expanded by experts. The platform's infrastructure offers tools for this and supports the replacement of knowledge components in an already operational DSS. Although such a replacement is technically carried out "instantly" on the IACPaaS platform, before being put into operation, updated service with knowledge base is tested on set of control and reference cases.

Thus, the implementation of the service on the IACPaaS platform provides the possibility of its continuous development.

Currently, the service is hosted on the platform https://iacpaas.dvo.ru , login – medicine-services@mail.ru (the password is sent on request to the author of the article or the IACPaaS administrator).

References

[1] D. O. Ivanov, V. I. Orel, Yu. S. Aleksandrovich, K. V. Pshenisnov, R. X. Lomovceva. "Diseases of the cardiovascular system as the leading cause of death in Russian Federation: Ways of problem solution", *Medicina i organizaciya zdravooxraneniya*, **4**:2 (2019), pp. 4–12 (in Russian). ↑166

[2] L. Karatzia, N. Aung, D. Aksentijevic. "Artificial intelligence in cardiology: Hope for the future and power for the present", *Frontiers in Cardiovascular Medicine*, **9** (2022), id. 945726. ↑166

[3] E. V. Shlyaxto, N. E. Zvartau, S. V. Villeval'de, A. N. Yakovlev, A. E. Solov'eva, A. S. Alieva, N. G. Avdonina, E. A. Medvedeva, A. A. Fedorenko, V. V. Kulakov, V. A. Karlina, G. V. Endubaeva, V. V. Zajcev, A. E. Solov'ev. "Cardiovascular risk management system: prerequisites for developing, organization principles, target groups", *Rossijskij kardiologicheskij zhurnal*, **24**:11 (2019), pp. 69–82 (in Russian). ↑168

[4] S.-H. Kang, H. Baek, J. Cho, S. Kim, H. Hwang, W. Lee, J. J. Park, Y. E. Yoon, C.-H. Yoon, Y.-S. Cho, T.-J. Youn, G.-Y. Cho, I.-H. Chae, D.-J. Choi, S. Yoo, J.-W. Suh. "Management of cardiovascular disease using an mHealth tool: a randomized clinical trial", *npj Digit. Med.*, **4** (2021), id. 165, 7 pp. ↑168

[5] A. S. Alieva, E. I. Pavlyuk, E. M. Alborova, N. E. Zvartau, A. O. Konradi, A. L. Katapano, E. V. Shlyaxto. "Clinical decision support system for lipid metabolism disorders: Relevance and potential", *Rossijskij kardiologicheskij zhurnal*, **26**:6 (2021), pp. 124–127 (in Russian). ↑168

[6] Z. Lin, Y. T. Cheng, B. M. Y. Cheung. "Machine learning algorithms identify hypokalaemia risk in people with hypertension in the United States National Health and Nutrition Examination Survey 1999–2018", *Ann. Med.*, **55**:1 (2023), id. 2209336, 13 pp. ↑168

[7] D. J. Choi, J. J. Park, T. Ali, S. Lee. "Artificial intelligence for the diagnosis of heart failure", *npj Digital Medicine*, **3**:1 (2020), id. 54, 6 pp. ↑168

[8] A. Assadi, P. C. Laussen, G. Freire, M. Ghassemi, P. Trbovich. "Decision-centered design of a clinical decision support system for acute management of pediatric congenital heart disease", *Frontiers in Digital Health*, **4** (2022), id. 1016522, 11 pp. ↑168

[9] N. P. Lyamina, E. V. Kotel'nikova. "A decision support system as a component of a patient-centered model of cardiac rehabilitation", *Doktor.Ru*, 2017, no. 5 (134), pp. 42–46 (in Russian). ↑168

[10] T. K. J. Groenhof, Z. H. Rittersma, M. L. Bots, M. Brandjes, J. J. L. Jacobs, D. E. Grobbee, van Solinge W. W., F. L. J. Visseren, S. Haitjema, F. W. Asselbergs, Members of the UCC-CVRM Study Group et. "A computerised decision support system for cardiovascular risk management 'live' in the electronic health record environment: Development, validation and implementation — the Utrecht Cardiovascular Cohort Initiative", *Netherlands Heart Journal*, **27** (2019), pp. 435–442. ↑168

[11] N. G. Avdonina, E. V. Bolgova, M. V. Ionov, N. E. Zvartau, A. O. Konradi. "The decision support system in the treatment of arterial hypertension — Control of the data entry into the electronic chart", *Arterial'naya gipertenziya*, **24**:6 (2018), pp. 704–709 (in Russian). ↑168

[12] K. Pieszko, J. Hiczkiewicz, J. Budzianowski, B. Musielak, D. Hiczkiewicz, W. Faron, J. Rzeźniczak, P. Burchardt. "Clinical applications of artificial intelligence in cardiology on the verge of the decade", *Cardiology Journal*, **28**:3 (2021), pp. 460–472. ↑168

[13] X. Chen, Sh. Jia, Y. Xiang. "A review: Knowledge reasoning over knowledge graph", *Expert Syst. Appl.*, **141** (2020), id. 112948, 21 pp. ↑168

[14] V. V. Gribova, M. V. Petryaeva, D. B. Okun', E. A. Shalfeeva. "Medical diagnosis ontology for intelligent decision support systems", *Ontologiya proektirovaniya*, **8**:1(27) (2018), pp. 58–73 (in Russian). ↑168, 172

[15] V. V. Gribova, D. B. Okun', M. V. Petryaeva, E. A. Shalfeeva. "IACPaaS infrastructure for generating interpretable diagnostic knowledge bases for diseases of interest", *Semnadczataya Nacional'naya konferenciya po iskusstvennomu intellektu s mezhdunarodnym uchastiem*, Sbornik nauchnyx trudov. V. 2, KII-2019 (21–25 oktyabrya 2019 g., g. Ul'yanovsk, Rossiya), UlGTU, Ul'yanovsk, 2019, ISBN 978-5-9795-1940-1, pp. 81–89 (in Russian). ↑169

[16] V. A. Sergeeva, T. E. Lipatova. "COVID-19 associated myocarditis: mechanisms of pathogenesis, problems of diagnostics (review)", *Saratovskij nauchno-medicinskij zhurnal*, **17**:3 (2021), pp. 571–577 (in Russian). ↑171

[17] E. O. Kotova, A. S. Pisaryuk, Zh. D. Kobalava, Yu. A. Timofeeva, N. S. Chipigina, Yu. L. Karaulova, L. G. Ezhova. "Infective endocarditis and COVID-19: the impact of SARS-CoV-2 infection on diagnostics, course, and prognosis", *Rossijskij kardiologicheskij zhurnal*, **28**:1 (2023), pp. 28–42 (in Russian). ↑171

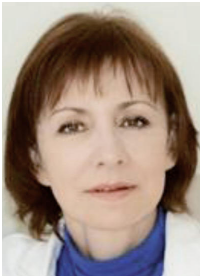
[18] D. B. Okun', R. I. Kovalev. "Myocarditis management knowledge base: Presenting knowledge for differentiated etiotropic therapy", *Materialy XV mezhdunarodnoj nauchnoj konferencii "Sistemnyj analiz v medicine"*, SAM 2021, ed. V. P. Kolosov, DNCz FPD, Blagoveshhensk, 2021, ISBN 978-5-905864-24-7, pp. 53–56 (in Russian). ↑172

[19] D. A. Andreev, V. Yu. Firsakova, O. V. Doroxova, O. M. Maslennikova. "Acute myocarditis in camouflage of myocardium infarction with st segment ascent", *Vestnik Ivanovskoj medicinskoj akademii*, **19**:4 (2014), pp. 64–68 (in Russian). ↑182

Received 25.09.2023;
approved after reviewing 27.10.2023;
accepted for publication 27.11.2023;
published online 31.12.2023.

Information about the authors:

Valeria Viktorovna Gribova



Deputy director for scientific work, scientific director of the Laboratory of intelligent systems of the Institute of Automation and Control Processes of the Far Eastern Branch of the Russian Academy of Sciences, Doctor of Technical Sciences, Corresponding Member of the Russian Academy of Sciences. Scientific interests: ontologies and knowledge bases, applied and problem-oriented knowledge-based systems, knowledge base management. The list of scientific papers includes more than 350 works

0000-0001-9393-351X
e-mail: gribova@iacp.dvo.ru

Elena Arefyevna Shalfeeva

Leading Researcher at the Laboratory of Intelligent Systems of the Institute of Automation and Control Processes of the Far Eastern Branch of the Russian Academy of Sciences, Doctor of Technical Sciences Scientific interests: ontological engineering, interpreted clinical guidelines, technology for creating systems with declarative knowledge, explanatory artificial intelligence, knowledge base management. The list of scientific papers includes more than 170 works

0000-0001-5536-2875
e-mail: shalf@dvo.ru

Margarita Vyacheslavovna Petryaeva

PhD, researcher at the Laboratory of Intelligent Systems of the Institute of Automation and Control Processes of the Far Eastern Branch of the Russian Academy of Sciences. Scientific interests: ontologies and knowledge bases, medical intelligent systems. She is the author of 92 scientific papers

0000-0002-1693-4508

e-mail: margaret@iacp.dvo.ru

Dmitry Borisovich Okun

PhD, researcher at the Laboratory of Intelligent Systems of the Institute of Automation and Control Processes of the Far Eastern Branch of the Russian Academy of Sciences

0000-0002-6300-846X

e-mail: okdm@iacp.dvo.ru

Leonid Aleksandrovich Fedorishchev

Ph.D, senior researcher at the Laboratory of Intelligent Systems at the Institute of Automation and Control Processes of the Far Eastern Branch of the Russian Academy of Sciences, Associate Professor at the VGUES. The list of scientific papers includes more than 40 works

0000-0002-2049-2570

e-mail: fleo1987@mail.ru

Roman Igorevich Kovalev

Researcher at the Laboratory of Intelligent Systems of the Institute of Automation and Control Processes of the Far Eastern Branch of the Russian Academy of Sciences. Scientific interests: ontologies and knowledge bases, intelligent systems

0000-0002-1704-2675

e-mail: koval-995@mail.ru

The authors declare no conflicts of interests.

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья МЕТОДЫ ОПТИМИЗАЦИИ И ТЕОРИЯ УПРАВЛЕНИЯ

УДК 519.68:007.52
10.25209/2079-3316-2023-14-4-189-206

Построение нелинейной обратной связи в задаче слежения для модели колесного робота, основанное на технике SDDRE

Юлия Сергеевна **Белинская**[1], Дмитрий Александрович **Макаров**[2✉]
Федеральный исследовательский центр «Информатика и управление» РАН, Москва, Россия
[2✉]*makarov@isa.ru*

Аннотация. В статье рассматривается задача построения нелинейной обратной связи в задаче слежения для колесной робототехнической системы. Особенностью работы является постановка задачи, в которой желаемые траектории системы известны заранее, а также модификация ранее известного алгоритма на основе техники State-Dependent Differential Riccati Equation. Численные эксперименты показывают, что предложенный подход позволяет обеспечить компромисс между качеством управления и скоростью работы.

Ключевые слова и фразы: Техника SDDRE, задача слежения, двухколесная робототехническая система

Благодарности: Исследование выполнено за счет гранта Российского научного фонда № 22-21-00716

Для цитирования: Белинская Ю. С., Макаров Д. А. *Построение нелинейной обратной связи в задаче слежения для модели колесного робота, основанное на технике SDDRE* // Программные системы: теория и приложения. 2023. **Т. 14**. № 4(59). С. 189–206. https://psta.psiras.ru/read/psta2023_4_189-206.pdf

Введение

Для построения оптимальных обратных связей большую роль играет уравнение Гамильтона-Якоби-Беллмана (ГЯБ). В общем случае его решение для нелинейных систем управления представляет собой трудоемкую задачу. Для ее упрощения была разработана техника SDRE. Ее основная идея заключается в представлении исходной нелинейной аффинной системы и критерия в псевдолинейном виде, при котором матрицы системы и критерия зависят от вектора состояния. Тогда при дополнительных предположениях уравнение ГЯБ сводится к алгебраическому матричному уравнению Риккати с коэффициентами, зависящими от состояния (State-Dependent Riccati Equation - SDRE), что дало название этой технике. Несмотря на эвристический характер полученного решения, такой подход получил достаточно широкое распространение при решении практических задач (см. обзоры [1–3] и литературу в них). Он также был распространен на оптимальные задачи на конечном времени. В последнем случае ищется решение начальной задачи для дифференциального матричного уравнения Риккати с зависящими от состояния коэффициентами (State-Dependent Differential Riccati Equation - SDDRE) [4, 5]. Техника SDDRE была применена для решения ряда практических задач (см, например, [6, 7]).

К преимуществам SDRE и SDDRE относится применимость к достаточно общему классу нелинейных систем, полный учет нелинейности системы (отсутствует линеаризация), некоторая субоптимальность и простота реализации. К недостаткам относятся эвристический характер и необходимость решать уравнения SDRE или SDDRE для каждого нового состояния системы в темпе функционирования объекта. Последнее может требовать значительных вычислительных ресурсов, например, в случае, если система обладает большой размерностью и/или имеет быструю динамику.

Для повышения вычислительной эффективности в работе [8] предложен модифицированный SDRE подход. Он заключается в использовании

или искусственном введении малого параметра при нелинейностях системы. Затем на основе асимптотического анализа получается численно-аналитический алгоритм, что повышает вычислительную эффективность. Метод был распространён на задачи слежения в разных постановках [9–11].

Техники SDRE и SDDRE крайне редко встречаются для задач управления колесными системами. Главной причиной этого является невыполнение основного условия – поточечной управляемости. Для преодоления этого ограничения в работе [12] предложен особый способ вывода динамики колесной системы (в пространство состояния включаются углы поворота колес и их скорости), а также вводится оригинальный искусственный выход системы. Далее используется известная техника SDDRE из [4, 5] для построения следящего управления. В работе [13] была предложена модификация этого решения на основе подхода из [9–11]. Показано, что несмотря на незначительную потерю в качестве управления, полученный алгоритм существенно превышает исходный с точки зрения вычислительной эффективности.

Отметим, что в постановке задачи, используемой и в [12], и в [13], считалось, что референсная траектория известна лишь в текущий момент. Однако, во многих практических задачах желаемые траектории являются известными функциями времени. Они могут быть построены на основе соответствующих планировщиков. Поэтому в данной работе рассматривается постановка задачи, когда референсные траектории известны заранее. Другой особенностью работы является модификация алгоритма из [13]: основная часть управления теперь учитывает конечный интервал времени управления, что положительно сказывается на качестве решения задач с небольшим интервалом регулирования. Для исследования эффективности предложенного подхода был проведен ряд численных экспериментов.

1. Модель системы и постановка задачи

На рисунке 1 приводится схема колесной робототехнической системы из [12], для которой в дальнейшем рассматривается задача управления на плоскости.

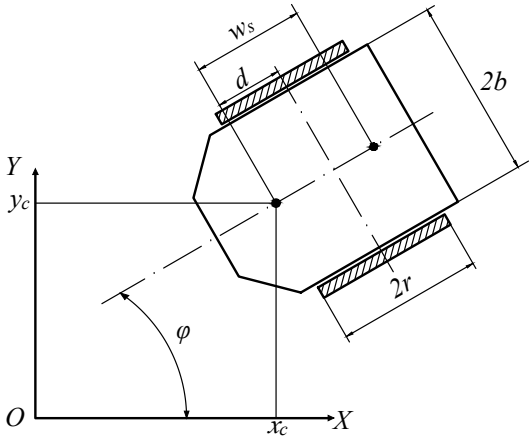


Рисунок 1. Схема двухколёсной тележки [12]

На схеме: XOY – земная декартова система координат; d – расстояние между центром масс робота и осью колес; x_c и y_c – координаты центра масс робота в XOY; φ – угол между продольной осью робота и осью OX (угол курса); r – радиус колес; b– половина ширины платформы робота; параметр w_s будет описан ниже.

Модель динамики робота имеет вид [12]

$$(1)\quad \dot{x} = \begin{bmatrix} \dot{q} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} Sv \\ (S^T M S)^{-1}(-S^T M \dot{S} v - S^T c) \end{bmatrix} + \begin{bmatrix} 0_{4\times 2} \\ \left(S^T M S\right)^{-1} \end{bmatrix} \tau,$$

где x – вектор состояния системы; точка означает дифференцирование по времени; $q = [x_c, y_c, \theta_r, \theta_l]^T$, $v = \left[\dot{\theta}_r, \dot{\theta}_l\right]^T$; θ_l и θ_r – углы поворота левого и правого колес; $\tau = [\tau_l, \tau_r]^T$ – вектор управления; τ_l и τ_r – моменты сил левого и правого колес соответственно; $0_{4\times 2}$ – нулевая матрица соответствующей размерности; T означает транспонирование. Матрицы S и M задаются как

$$S = \begin{bmatrix} \frac{r}{2}\cos(\phi) & \frac{r}{2}\cos(\phi) \\ \frac{r}{2}\sin(\phi) & \frac{r}{2}\sin(\phi) \\ 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$M = \begin{bmatrix} m_b + 2m_w & 0 & -\frac{m_w r d \sin(\phi)}{b} & \frac{m_w r d \sin(\phi)}{b} \\ 0 & m_b + 2m_w & \frac{m_w r d \cos(\phi)}{b} & -\frac{m_w r d \cos(\phi)}{b} \\ -\frac{m_w r d \sin(\phi)}{b} & \frac{m_w r d \cos(\phi)}{b} & m_{3,3} & m_{3,4} \\ \frac{m_w r d \sin(\phi)}{b} & -\frac{m_w r d \cos(\phi)}{b} & m_{3,4} & m_{4,4} \end{bmatrix},$$

где

$$m_{3,3} = \frac{r^2\left(2m_w\left(b^2 + d^2\right) + I_{zzb} + 2\,I_{zzw}\right)}{4b^2} + I_{yyw},$$

$$m_{3,4} = \frac{-r^2\left(2m_w\left(b^2 + d^2\right) + I_{zzb} + 2\,I_{zzw}\right)}{4b^2},$$

$$m_{4,4} = \frac{r^2\left(2m_w\left(b^2 + d^2\right) + I_{zzb} + 2\,I_{zzw}\right)}{4b^2} + I_{yyw}.$$

Здесь m_w – масса колеса, m_b – масса колесной системы, I_{zzb} – момент инерции колесной системы относительно вертикальной оси OZ, I_{yyw} – момент инерции колес относительно оси вращения, I_{zzw} – момент инерции колес относительно вертикальной оси OZ. Отметим, что угол крена может быть найден как $\phi = {}^{r}\!/_{2b}\,(\theta_r - \theta_l)$.

Задача ставится следующим образом: необходимо построить обратную связь τ, которая минимизирует норму ошибки слежения $\|e\|$, где $e =$

$[x_c, y_c]^T - [x_r, y_r]^T$, а $x_r(t)$, $y_r(t)$ – известные желаемые (референсные) траектории для координат x_c и y_c соответственно.

Подчеркнем, что в данной работе $x_r(t)$, $y_r(t)$ считаются известными гладкими функциями на всем временном интервале управления, тогда как в работах [12, 13] рассматривалась постановка, когда значения этих функций были известны лишь для текущего временного момента и не были известны производные $\dot{x}_r(t)$, $\dot{y}_r(t)$.

2. Управление на основе SDDRE

Кратко опишем исходный способ решения из [12], который имеет эвристический характер, т.к. строгие оценки субоптимальности в работе отсутствуют. Способ состоит из двух этапов. На первом этапе вводится замена переменных, на втором – с помощью известной техники SDDRE из [4, 5] приближенно решается задача перевода системы в новых переменных из начального состояния в заданное.

2.1. Замена переменных

Вводится следующий искусственный выход системы

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_c + w_s \cos(\phi) - x_r \\ y_c + w_s \sin(\phi) - y_r \end{bmatrix}. \tag{2}$$

В выходе (2) присутствует параметр $w_s > 0$. Он задает расстояние (см. рисунок 1) между центром масс робота и некоторой референсной подвижной точкой в XOY, в которую выбранное в дальнейшем управление будет приводить робота. Дважды продифференцировав по времени (2) считая $\dot{x}_r \equiv \dot{y}_r \equiv 0$, можно получить систему

$$\dot{z} = \frac{d}{dt}\begin{bmatrix} y \\ \dot{y} \end{bmatrix} = \tag{3}$$

$$= \begin{bmatrix} \dot{y} \\ \alpha \left(S^T M S\right)^{-1} \left(-S^T M \dot{S} v - S^T c\right) + \beta + \alpha \left(S^T M S\right)^{-1} \tau \end{bmatrix}.$$

Здесь $z = [y, \dot{y}]$, а матрицы α, β задаются как

$$\alpha = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} \\ \alpha_{2,1} & \alpha_{2,2} \end{bmatrix}, \quad \beta = \bar{\beta} \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{bmatrix}, \quad c = \bar{C} \begin{bmatrix} \dot{\theta}_r \\ \dot{\theta}_l \end{bmatrix}, \quad \bar{\beta} = \begin{bmatrix} \bar{\beta}_{1,1} & \bar{\beta}_{1,2} \\ \bar{\beta}_{2,1} & \bar{\beta}_{2,2} \end{bmatrix},$$

$$\bar{C} = \begin{bmatrix} -\dfrac{m_w r^2 d\left(\dot{\theta}_r - 2\dot{\theta}_l\right)\cos(\phi)}{2b^2} & -\dfrac{m_w r^2 d\dot{\theta}_l \cos(\phi)}{2b^2} \\ -\dfrac{m_w r^2 d\dot{\theta}_r \sin(\phi)}{2b^2} & -\dfrac{m_w r^2 d\left(\dot{\theta}_l - 2\dot{\theta}_r\right)\sin(\phi)}{2b^2} \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

где

$$\alpha_{1,1} = \frac{r\left(b\cos(\phi) - w_s \sin(\phi)\right)}{2b}, \quad \alpha_{1,2} = \frac{r\left(b\cos(\phi) + w_s \sin(\phi)\right)}{2b},$$

$$\alpha_{2,1} = \frac{r\left(b\sin(\phi) + w_s \cos(\phi)\right)}{2b}, \quad \alpha_{2,2} = \frac{r\left(b\sin(\phi) - w_s \cos(\phi)\right)}{2b},$$

$$\bar{\beta}_{1,1} = -\frac{r^2\left(\dot{\theta}_r + \dot{\theta}_l\right)\sin(\phi)}{4b} + \frac{r^2 w_s\left(\dot{\theta}_l - \dot{\theta}_r\right)\cos(\phi)}{4b^2},$$

$$\bar{\beta}_{1,2} = \frac{r^2\left(\dot{\theta}_r + \dot{\theta}_l\right)\sin(\phi)}{4b} - \frac{r^2 w_s\left(\dot{\theta}_l - \dot{\theta}_r\right)\cos(\phi)}{4b^2},$$

$$\bar{\beta}_{2,1} = \frac{r^2\left(\dot{\theta}_r + \dot{\theta}_l\right)\cos(\phi)}{4b} + \frac{r^2 w_s\left(\dot{\theta}_l - \dot{\theta}_r\right)\sin(\phi)}{4b^2},$$

$$\bar{\beta}_{2,2} = -\frac{r^2\left(\dot{\theta}_r + \dot{\theta}_l\right)\cos(\phi)}{4b} - \frac{r^2 w_s\left(\dot{\theta}_l - \dot{\theta}_r\right)\sin(\phi)}{4b^2}.$$

2.2. Алгоритм SDDRE

Пусть имеется аффинная система

$$\dot{x} = f(x) + g(x)u, \quad x(0) = x^0. \tag{4}$$

где $x \in \mathrm{R}^n$ – вектор состояния, $u \in \mathrm{R}^m$– управление, f и g – кусочно-непрерывные гладкие вектор-функции, удовлетворяющие условию Липшица, $f(0) = 0$, x^0 – заданное начальное состояние. Система (4) может быть представлена в виде

$$\dot{x} = A(x)x + B(x)u, \tag{5}$$

с помощью обозначения $B(x) = g(x)$ и факторизации $f(x) = A(x)x$. Причем при $n > 1$ существует бесчисленно число способов факторизации $f(x) = A(x)x$ [2].

Для системы (5) рассматривается задача оптимального управления

(6)
$$I(u) = \frac{1}{2}x^T(t_f)Fx(t_f) + \frac{1}{2}\int_0^{t_f}\left(x^TQ(x)x + u^TR(x)u\right)dt \to \min_u, \quad t_f > 0,$$

где F, $Q(x)$ и $R(x)$ – положительно полуопределенные и положительно определенная весовые матрицы для каждого допустимого значения x. Предполагается также, что тройка матриц $(A(x), B(x), H(x))$, где $H(x)^TH(x) = Q(x)$, управляема и стабилизируема для каждого x из допустимой области.

Согласно технике SDDRE [4, 5] управление, приближенно решающее задачу (5)–(6), ищется в виде

(7)
$$u(x,t) = -R^{-1}(x)B^T(x)P(x,t)x,$$

где P определяется как решение матричного дифференциального уравнения Риккати с зависящими от состояния коэффициентами вида

(8)
$$-\dot{P}(x,t) = P(x,t)A(x) + A^T(x)P(x,t)$$
$$-P(x,t)B(x)R^{-1}(x)B^T(x)P(x,t) + Q(x).$$

Для приближённого решения (8) для каждого нового значения вектора состояния x последовательно повторяются следующие шаги.

Шаг 1. Найти отрицательно определенное решение $P_{ss}^-(x)$ матричного алгебраического уравнения Риккати $0 = P_{ss}^-(x)A(x) + A^T(x)P_{ss}^-(x) - P_{ss}^-(x)B(x)R^{-1}(x)B^T(x)P_{ss}^-(x) + Q(x)$.

Шаг 2. Вычислить $A_{cl}(x) = A(x) - B(x)R^{-1}(x)B^T(x)P_{ss}^-(x)$.

Шаг 3. Найти решение D алгебраического уравнения Ляпунова $D(x)A_{cl}^T(x) + A_{cl}(x)D(x) - B(x)R^{-1}(x)B^T(x) = 0$.

Шаг 4. Найти решение дифференциального уравнения Ляпунова $\dot{K}(x,t) = K(x,t)A_{cl}^T(x) + A_{cl}(x)K(x,t) - B(x)R^{-1}(x)B^T(x)$ по формуле $K(x,t) = e^{A_{cl}(t-t_f)}(K(x,t_f) - D(x))e^{A_{cl}^T(t-t_f)} + D(x)$, $K(x,t_f) = \left(F - P_{ss}^-(x)\right)^{-1}$.

Шаг 5. Вычислить $P(x,t) = K^{-1}(x,t) + P_{ss}^-(x)$.

Шаг 6. Найти управление с помощью (7).

Для применения указанного алгоритма в задаче слежения нужно представить систему (3) в виде (5) с помощью следующих матриц с зависящими от состояния коэффициентами [12]

$$A(z) = \begin{bmatrix} 0_{2\times 2} & I_{2\times 2} \\ 0_{2\times 2} & \alpha \left(S^T M S\right)^{-1} \left(-S^T M \dot{S} - S^T \bar{C}\right) + \bar{\beta} \end{bmatrix}, \tag{9}$$

$$B(z) = \begin{bmatrix} 0_{2\times 2} \\ \alpha \left(S^T M S\right)^{-1} \end{bmatrix},$$

где $I_{2\times 2}$ – единичная матрица соответствующей размерности. Итоговое управление для системы (1) принимает вид

$$u(x,t) = -R^{-1}(z)B^T(z)P(z,t)z = -R^{-1}(z)B^T(z)P(z,t)[\ y \quad \dot{y}\ ]^T. \tag{10}$$

3. Модификация алгоритма SDDRE

Модификация алгоритма состоит в следующем. Во-первых, в управлении ниже учитываются производные $\dot{x}_r(t),\ \dot{y}_r(t)$. Поэтому в правой части (3) появляются ненулевые значения $\dot{x}_r(t),\ \dot{y}_r(t)$. Во-вторых, для приближенного решения уравнения (8) вместо метода из [12] используется следующая численно-аналитическая процедура.

Шаг 1. Вычислить матрицы

$$A_0 = A(z)|_{z=0},\ B_0 = B(z)|_{z=0},\ Q_0 = Q(z)|_{z=0},$$

$$A_1(z) = \frac{A(z) - A_0}{\varepsilon}, \quad B_1(z) = \frac{B(z) - B_0}{\varepsilon}, \quad Q_1(z) = \frac{Q(z) - Q_0}{\varepsilon}.$$

Здесь $\varepsilon > 0$ является параметром алгоритма. Весовая матрица R предполагается постоянной.

Шаг 2. Вычислить матричную функцию $P_0(t)$ как решение следующей задачи Коши

$$\dot{P}_0 + P_0 A_0 + A_0^T P_0\ - P_0 B_0 R_0^{-1} B_0^T P_0 + Q_0 = 0, \quad P_0(t_f) = F.$$

Шаг 3. Найти $P_1(z,t)$ с помощью

$$P_1(z,t) = e^{A^T_{Pcl,\,0}(t_f - t)} M_P e^{A_{Pcl,0}(t_f - t)} + \int_0^\infty e^{A^T_{Pcl,\,0}\sigma} D_P(z) e^{A_{Pcl,\,0}\sigma} d\sigma, \tag{11}$$

где $D_P(z) = P_0\left(A_1 - B_1R^{-1}B_0^TP_0\right) + \left(A_1 - B_1R^{-1}B_0^TP_0\right)^T P_0 + Q_1$,
$A_{Pcl,0}(t) = A_0 - B_0R^{-1}B_0^TP_0(t)$, а M_P находится как
$M_P = -\int_0^\infty e^{A_{Pcl,0}^T\sigma}\ D_P\left(z(t_f)\right)|_{z(t_f)=z(t)}\, e^{A_{Pcl,0}\sigma}d\sigma$.

Шаг 4. Вычислить управление (10), где $P(z,t) = P_0(t) + \varepsilon P_1(z,t)$, перейти к шагу 3 для вычисления нового значения управления.

Предложенный алгоритм применим при следующих условиях:

(*i*) Траектории замкнутой системы (1),(10) существуют, единственны и принадлежат Z на $[0,\ t_f]$ для любого непрерывного управления $u(t)$, где Z – некоторое ограниченное множество пространства состояний; элементы матриц $A_1(z),\ B_1(z), Q_1(z)$ ограниченные, непрерывные и достаточно гладкие при $z \in Z,\ \varepsilon \in (0,\ \varepsilon_0]$.

(*ii*) Тройка матриц $\{A_0, B_0, H_0\}$, $H_0^TH_0 = Q_0$, стабилизируема и наблюдаема.

(*iii*) Матрицы системы $A_0, A_1(z),\ B_0, B_1(z), Q_0$ и симметрические матрицы критерия $R > 0,\ \ Q_0 \geq 0,\ \ Q_1(z) \geq 0,\ \ F > 0$, а также $\varepsilon_0 > 0$ таковы, что $P_0(t) + \varepsilon P_1(z,t) > 0$ при $z \in Z,\ t \in [0, t_f),\ \varepsilon \in (0,\ \varepsilon_0]$.

Замечание 1. Алгоритм является модификацией алгоритма из [**11**], который был использован в работе [**13**]. Отличия заключаются в том, что ранее на шаге 1 искалась постоянная матрица P_0 как положительно определенное решение матричного уравнения Риккати

$$P_0A_0 + A_0^TP_0\ - P_0B_0R_0^{-1}B_0^TP_0 + Q_0 = 0,$$

а на шаге 3 матричная функция $P_1(z,t)$ определялась как

$$P_1(z,t) = e^{A_{Pcl,0}^T(t_f-t)}M_Pe^{A_{Pcl,0}(t_f-t)} + \int_0^\infty e^{A_{Pcl,0}^T\sigma}D_P(z)e^{A_{Pcl,0}\sigma}d\sigma,$$

где $D_P(z) = P_0\left(A_1 - B_1R^{-1}B_0^TP_0\right) + \left(A_1 - B_1R^{-1}B_0^TP_0\right)^T P_0 + Q_1$,
$A_{Pcl,0} = A_0 - B_0R^{-1}B_0^TP_0$,
$M_P = \frac{1}{\varepsilon}(F - P_0) - \int_0^\infty e^{A_{Pcl,0}^T\sigma}\ D_P\left(z(t_f)\right)|_{z(t_f)=z(t)}\, e^{A_{Pcl,0}\sigma}d\sigma$.

Таким образом, теперь нестационарность $P(z,t)$ учитывается с помощью $P_0(t)$, которая соответствует линейной части управления, а не с помощью $P_1(z,t)$, соответствующей нелинейной коррекции линейного управления, как ранее. Предполагается, что подобная модификация

положительным образом отразится на качестве решения задач с небольшим временем управления, в которых $P(z,t)$ не успевает сойтись к установившему решению.

Замечание 2. Условие (*iii*) в силу наличия условий (*i*)–(*ii*) всегда может быть выполнено при достаточно малом ε_0.

Замечание 3. Представленный алгоритм отличается от алгоритма из пункта 2.2 вычислительной эффективностью. В нем матрица $P_0(t)$ находится лишь один раз (шаг 2) и в процессе работы необходимо найти лишь $P_1(z,t)$ с помощью аналитического выражения (11). Алгоритм из 2.2 на каждой итерации своей работы требует решения алгебраических матричных уравнений Риккати (шаг 1) и Ляпунова (шаг 3) и обращение матрицы (шаг 5).

Отметим, что метод решения из [**11**], который составляет основу предложенного в данной работе алгоритма, был изначально разработан для слабо нелинейных систем. Поэтому для изучения качества его работы были проведены численные эксперименты.

4. Численные эксперименты

Решим ряд задач слежения для модели двухколёсной системы (1) с помощью описанных выше подходов. Обозначим управление из подраздела 2.2 как u_1, управление из подраздела 2.2, в котором используется $\dot{x}_r(t)$, $\dot{y}_r(t)$, как u_2, управления из [**13**] и из раздела 3 как u_3 и u_4 соответственно. Весовые матрицы критерия (6) и параметр зададим следующим образом

$$\varepsilon = 1, \quad R = I_{2\times 2}, \quad F = Q_0 = diag\{100, 100, 10, 10\}, \quad Q_1 = 0_{4\times 4}.$$

В качестве референсных траекторий определим решения следующих уравнений

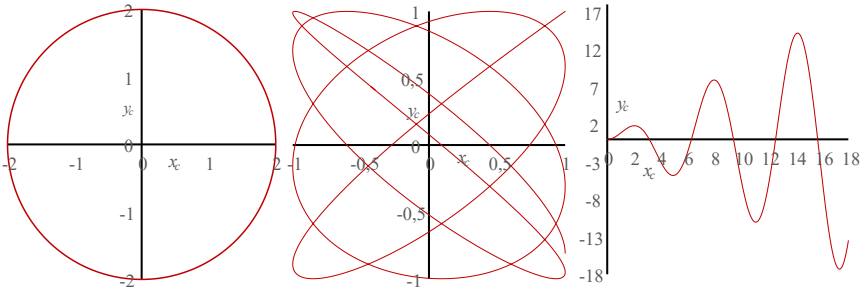
$$x_r = 2\cos\left(\frac{\pi t}{2}\right), y_r = 2\sin\left(\frac{\pi t}{2}\right), \quad t \in [0, 4], \tag{12}$$

$$x_r = \cos(\pi t), y_r = \cos(0.85\pi t), \quad t \in [0,\ 8], \tag{13}$$

$$x_r = 1.2t, y_r = 1.2t\sin(1.2t), \quad t \in [0,\ 15], \tag{14}$$

$$x_r = x_r^0, y_r = y_r^0, \quad t \in [0, 1]. \tag{15}$$

На рисунке 2 представлена визуализация референсных траекторий, кроме (15), которая представляет собой уравнение неподвижной точки и задаётся начальными условиями x_r^0, y_r^0.



(*а*) Референская траектория (12)

(*б*) Референская траектория (13)

(*в*) Референская траектория (14)

Рисунок 2. Графики для уравнений (12)–(14).

Параметры системы (1) обозначены в таблице 1. В силу этих параметров матрица $S^T M S$ является невырожденной.

Таблица 1. Значения параметров системы (1)

Параметр	Значение	Единицы измерения
b	0.145	м
d	0	м
r	0.08	м
m_w	0.32	м
m_b	6	м
w_s	0.1	м
I_{zzb}	0.06363	кг*м^2
I_{yyw}	0.6e-4	кг*м^2
I_{zzw}	0.81e-3	кг*м^2

В таблице 2. приводятся результаты численного моделирования для различных референсных траекторий и начального положения робота, заданного значениями $x_c(0)$ и $y_c(0)$. Во всех экспериментах начальная ориентация робота всегда соответствовала нулевому углу курса (робот направлен по направлению оси OX), скорость в начальный момент была

Таблица 2. Результаты численного моделирования

№	Реф. траектория	Начальное положение		$J(u_1)$	$J(u_2)$	$J(u_3)$	$J(u_4)$	$T(u_1)$	$T(u_2)$	$T(u_3)$	$T(u_4)$
		$x_c(0)$	$y_c(0)$								
1	(12)	0	0	23,3	13,49	21,72	14,55	46,7	36,2	1,782	21,47
2		1	0	12,74	8,065	13,58	9,02	40,8	33,02	1,859	21,5
3		0	-1	34	27,72	37,02	32,07	46,81	40,48	1,891	25,22
4	(13)	0	0	52,22	47,79	44,65	42,85	77,16	83,91	4,532	52,8
5		0	-1	60,56	52,6	54,54	48,13	79,53	82,28	4,516	55,55
6		0	1	45,91	40,3	37,72	36,24	74,17	78,38	4,203	49,59
7	(14)	0	0	250,9	152,6	149,1	140,4	119,2	123,5	6,907	85,31
8		0	-1	255,1	156,9	154,2	145,2	117	126,1	7,047	81,8
9		0	1	254,3	155,5	151,8	143,8	117,7	120	6,953	83,11
10	$x_r = -1$, $y_r = 0.5$	0	0	9,732	9,732	11,22	10,99	22,48	23,89	0,625	15,92
11	$x_r = -1$, $y_r = -1$	1	1	58,11	58,11	74,59	73,65	34,36	33,12	0,656	18,03
12	$x_r = -1$, $y_r = -1$	-1	0,5	10,39	10,39	11,92	11,71	26,3	22,78	0,594	15,23

равна нулю ($\theta_r(0) = \theta_l(0) = \dot{\theta}_r(0) = \dot{\theta}_l(0) = 0$). В этой таблице приводятся значения критерия (6), обозначенные как $J(*)$, и время численного счета в секундах, обозначенное через $T(*)$, для всех четырех управлений.

Таблица 3 содержит следующую информацию: $\%J(w)$ и $\%T(w)$ показывают, насколько в процентах $J(w)$ и $T(w)$ отличается от $J(u_1)$ и $T(u_1)$ соответственно для конкретного управления w, если результаты для u_1 взять за 100%.

Согласно таблице 3 в среднем самым эффективным по критерию (6) управлением оказалось u_2, оно же является одним из самых вычислительно затратных, незначительно уступая только u_1. Для экспериментов 10-12 с неподвижной точкой в качестве референса, эти управления показывают одинаковые результаты.

Следующим по эффективности относительно функционала (6) идет управление u_4. Оно незначительно уступает u_2 и в то же время существенно превосходит его по скорости работы.

Таблица 3. Отличия от управления u_1

№	Реф. траектория	Начальное положение		$\%J(u_2)$	$\%J(u_3)$	$\%J(u_4)$	$\%T(u_2)$	$\%T(u_3)$	$\%T(u_4)$
		$x_c(0)$	$y_c(0)$						
1	(12)	0	1	-42,10	-6,78	-37,55	-22,48	-96,18	-54,03
2		1	2	-36,70	6,59	-29,20	-19,07	-95,44	-47,30
3		0	3	-18,47	8,88	-5,68	-13,52	-95,96	-46,12
4	(13)	0	4	-8,48	-14,50	-17,94	8,75	-94,13	-31,57
5		0	5	-13,14	-9,94	-20,53	3,46	-94,32	-30,15
6		0	6	-12,22	-17,84	-21,06	5,68	-94,33	-33,14
7	(14)	0	7	-39,18	-40,57	-44,04	3,61	-94,21	-28,43
8		0	8	-38,49	-39,55	-43,08	7,78	-93,98	-30,09
9		0	9	-38,85	-40,31	-43,45	1,95	-94,09	-29,39
10	$x_r = -1$, $y_r = 0.5$	0	10	0,00	15,29	12,93	6,27	-97,22	-29,18
11	$x_r = -1$, $y_r = -1$	1	11	0,00	28,36	26,74	-3,61	-98,09	-47,53
12	$x_r = -1$, $y_r = -1$	-1	12	0,00	14,73	12,70	-13,38	-97,74	-42,09
Среднее значение				**-20,64**	**-7,97**	**-17,51**	**-2,88**	**-95,47**	**-37,42**

Далее следует u_3. Оно же показало самое малое время счета, существенно опередив все другие рассматриваемые управления.

Наконец, управление u_1 оказалось самым неэффективным, как по критерию (6), так и по времени счета.

Заключение

В работе предложена модификация ранее известных реализаций алгоритмов техники SDDRE. С ее помощью построена нелинейная обратная связь в задаче слежения для модели колесной системы. Особенностью постановки задачи является наличие известных гладких функций референсных траекторий, заданных на всем интервале времени регулирования. Учет этой информации при построении управления позволяет существенно улучшить качество решения задачи слежения, особенно в случае быстроменяющихся желаемых траекторий. Численные

эксперименты показали, что предложенная модификация улучшает исходный алгоритм примерно на 17% по критерию качества, при этом время счета уменьшается примерно на 37%. В то же время существуют аналоги, которые превосходят предложенную модификацию либо по качеству, либо по вычислительной эффективности. Таким образом, разработанный алгоритм может использоваться в тех задачах, в которых нужно обеспечить как приемлемое качество, так и скорость работы.

Список литературы

[1] Nekoo S. R. *Tutorial and review on the state-dependent Riccati equation* // Journal of Applied Nonlinear Dynamics.– 2019.– Vol. **8**.– No. 2.– Pp. 109–166. ↑190

[2] Çimen T. *Survey of state-dependent Riccati equation in nonlinear optimal feedback control synthesis* // Journal of Guidance, Control, and Dynamics.– 2012.– Vol. **35**.– No. 4.– Pp. 1025–1047. ↑190, 196

[3] Cloutier J. R. *State-dependent Riccati equation techniques: an overview* // *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*.– V. 2 (Albuquerque, NM, USA, 6 June 1997).– IEEE.– 1997.– ISBN 0-7803-3832-4.– Pp. 932–936. ↑190

[4] Heydari A., Balakrishnan S. N. *Path planning using a novel finite horizon suboptimal controller* // Journal of guidance, control, and dynamics.– 2013.– Vol. **36**.– No. 4.– Pp. 1210–1214. ↑190, 191, 194, 196

[5] Heydari A., Balakrishnan S. N. *Closed-form solution to finite-horizon suboptimal control of nonlinear systems* // International Journal of Robust and Nonlinear Control.– 2015.– Vol. **25**.– No. 15.– Pp. 2687–2704. ↑190, 191, 194, 196

[6] Naidu D. S., Paul S., Khamis A., Rieger C. R. *A simplified SDRE technique for finite horizon tracking problem in optimal control systems*, 2019 Sixth Indian Control Conference (ICC) (Hyderabad, India, 18–20 December 2019).– 2019.– Pp. 170–175. ↑190

[7] Khamis A., Naidu D. S. *Recent results on nonlinear, optimal regulation and tracking: theory and applications* // WSEAS Transactions on Systems.– 2016.– Vol. **15**.– Pp. 94–101.– id. 11. ↑190

[8] Дмитриев М. Г., Макаров Д. А. *Гладкий нелинейный регулятор в слабо нелинейной системе управления с коэффициентами, зависящими от состояния* // Труды института системного анализа Российской академии наук.– 2014.– Т. **64**.– № 4.– С. 53–58. ↑190

[9] Макаров Д. А. *Подход к построению нелинейного управления в задаче слежения с коэффициентами, зависящими от состояния. Часть I. Алгоритм* // ИТиВС.– 2017.– № 3.– С. 10–19. ↑191

[10] Макаров Д. А. *Построение управления и наблюдателя в слабо нелинейной задаче слежения с помощью дифференциальных матричных уравнений Риккати* // ИТиВС.– 2018.– № 4.– С. 63–71. ↑191

[11] Макаров Д. А., Хачумов М. В. *Синтез в слабо нелинейной задаче управления на основе SDRE техники на конечном интервале* // ИТиВС.– 2020.– № 4.– С. 17–25. ↑191, 198, 199

[12] Korayem M. H., Nekoo S. R., Korayem A. H. *Finite time SDRE control design for mobile robots with differential wheels* // Journal of Mechanical Science and Technology.– 2016.– Vol. **30**.– No. 9.– Pp. 4353–4361. ↑191, 192, 194, 197

[13] Макаров Д. А. *Приближенное решение задачи слежения для модели двухколесного робота, основанное на технике SDDRE* // *Управление развитием крупномасштабных систем (MLSD'2021)* (Москва, 27–29 сентября 2021 года), М.: Институт проблем управления им. В.А. Трапезникова РАН.– 2021.– ISBN 978-5-91450-256-7.– С. 665–672. ↑191, 194, 198, 199

Поступила в редакцию 11.11.2023;
одобрена после рецензирования 11.12.2023;
принята к публикации 12.12.2023;
опубликована онлайн 31.12.2023.

Рекомендовал к публикации *д.т.н. А. М. Цирлин*

Информация об авторах:

Юлия Сергеевна Белинская

Научный сотрудник ФИЦ ИУ РАН, кандидат физико-математических наук. Количество печатных работ: более 20. Область научных интересов: нелинейная теория управления, плоские системы, робототехника. E-mail: belinskaya.us@gmail.com

0000-0002-4136-7912
e-mail: belinskaya.us@gmail.com

Дмитрий Александрович Макаров

Старший научный сотрудник ФИЦ ИУ РАН, кандидат физико-математических наук. Количество печатных работ: более 70. Область научных интересов: методы синтеза нелинейных обратных связей, сингулярно возмущенные системы, искусственный интеллект, робототехника

0000-0001-8930-1288
e-mail: makarov@isa.ru

Вклад авторов: *Ю. С. Белинская* – 15% (проведение численных экспериментов, визуализация); *Д. А. Макаров* – 85% (идея, методология, программное обеспечение, валидация, написание черновой версии, доработка и редактирование).

Авторы заявляют об отсутствии конфликта интересов.

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article OPTIMIZATION METHODS AND CONTROL THEORY

UDC 519.68:007.52
10.25209/2079-3316-2023-14-4-189-206

SDDRE based nonlinear feedback construction in the tracking problem for a wheeled robot model

Yulia Sergeevna **Belinskaya**[1], Dmitry Alexandrovich **Makarov**[2✉]
Federal Research Center "Computer Science and Control" of RAS, Moscow, Russia
[2✉]*makarov@isa.ru*

Abstract. The article discusses the problem of constructing nonlinear feedback in the tracking problem for a wheeled robotic system. A special feature of the work is the formulation of a problem in which the reference trajectories of the system are known in advance, as well as a modification of a previously known algorithm based on the State-Dependent Differential Riccati Equation technique. Numerical experiments show that the proposed approach allows for a compromise between control quality and operating speed. (*In Russian*).

Key words and phrases: SDDRE technique, tracking task, two-wheeled robotic system

2020 *Mathematics Subject Classification*: 70B15; 93C85

Acknowledgments: The study was supported by the Russian Science Foundation grant No. 22-21-00716

For citation: Yulia S. Belinskaya, Dmitry A. Makarov. *SDDRE based nonlinear feedback construction in the tracking problem for a wheeled robot model*. Program Systems: Theory and Applications, 2023, **14**:4(59), pp. 189–206. (*In Russ.*).
https://psta.psiras.ru/read/psta2023_4_189-206.pdf

References

[1] S. R. Nekoo. "Tutorial and review on the state-dependent Riccati equation", *Journal of Applied Nonlinear Dynamics*, **8**:2 (2019), pp. 109–166.

[2] T. Çimen. "Survey of state-dependent Riccati equation in nonlinear optimal feedback control synthesis", *Journal of Guidance, Control, and Dynamics*, **35**:4 (2012), pp. 1025–1047.

[3] J. R. Cloutier. "State-dependent Riccati equation techniques: an overview", *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*. V. 2 (Albuquerque, NM, USA, 6 June 1997), IEEE, 1997, ISBN 0-7803-3832-4, pp. 932–936.

[4] A. Heydari, S. N. Balakrishnan. "Path planning using a novel finite horizon suboptimal controller", *Journal of guidance, control, and dynamics*, **36**:4 (2013), pp. 1210–1214.

[5] A. Heydari, S. N. Balakrishnan. "Closed-form solution to finite-horizon suboptimal control of nonlinear systems", *International Journal of Robust and Nonlinear Control*, **25**:15 (2015), pp. 2687–2704.

[6] D. S. Naidu, S. Paul, A. Khamis, C. R. Rieger. "A simplified SDRE technique for finite horizon tracking problem in optimal control systems", 2019 Sixth Indian Control Conference (ICC) (Hyderabad, India, 18–20 December 2019), 2019, pp. 170–175.

[7] A. Khamis, D. S. Naidu. "Recent results on nonlinear, optimal regulation and tracking: theory and applications", *WSEAS Transactions on Systems*, **15** (2016), pp. 94–101, id. 11.

[8] M. G. Dmitriev, D. A. Makarov. "Smooth nonlinear controller in a weakly nonlinear control system with state-dependent coefficients", *Proceedings of the Institute for System Analysis of the Russian Academy of Sciences*, **64**:4 (2014), pp. 53–58 (in Russian).

[9] D. A. Makarov. "A nonlinear approach to a feedback control design for a tracking state-dependent problem: Part I. An algorithm", *Information technology and computing systems*, 2017, no. 3, pp. 10–19 (in Russian).

[10] D. A. Makarov. "The design of observer based tracking control for weakly nonlinear systems using differential matrix equations Riccati", *Information technology and computing systems*, 2018, no. 4, pp. 63–71 (in Russian).

[11] D. A. Makarov, M. V. Khachumov. "SDRE-Based Synthesis in a Weakly Nonlinear Control Problem on a Finite Interval", *Information technology and computing systems*, 2020, no. 4, pp. 17–25 (in Russian).

[12] M. H. Korayem, S. R. Nekoo, A. H. Korayem. "Finite time SDRE control design for mobile robots with differential wheels", *Journal of Mechanical Science and Technology*, **30**:9 (2016), pp. 4353–4361.

[13] D. A. Makarov. "SDDRE based approximate solution in trajectory tracking control problem for a model of two-wheeled differentially driven mobile robot", 2021 14th International Conference Management of large-scale system development (MLSD) (Moscow, Russian Federation, 27–29 September 2021), pp. 1–5.