КОМПЬЮТЕРНЫЕ НАУКИ 2024. Т. 34. Вып. 2. С. 299–308.

УДК 004.04

© *А. П. Бельтюков, С. Г. Маслов, М. Джудакизаде*

ВЗАИМНОЕ МОДЕЛИРОВАНИЕ ПОСЛЕДОВАТЕЛЬНЫХ И ПАРАЛЛЕЛЬНЫХ СЛОВАРНЫХ ВЫЧИСЛЕНИЙ

Работа посвящена связи параллельных и последовательных вычислений. С одной стороны, рассматривается класс словарных предикатов, основанных на последовательных вычислениях, ограниченных по памяти константами и имеющих полиномиальную временную сложность. С другой стороны, рассматривается класс словарных предикатов, вычислимых на параллельных альтернирующих машинах за логарифмическое время. Доказано совпадение соответствующих классов. Предложено направление использования полученных результатов для взаимного преобразования и сочетания вычислений на молекулярных биоподобных последовательных машинах и параллельных вычислениях на векторно-матричных компьютерах. Предполагаемые области применения: обработка изображений в реальном масштабе времени для задач управления, анализ больших текстов и других больших данных.

Ключевые слова: словарные предикаты, параллельные вычисления, последовательные вычисления, большие данные, вычислительная сложность, биоподобные компьютеры, векторно-матричные компьютеры, альтернирование.

DOI: 10.35634/vm240208

Введение

Задачи взаимного моделирования вычислений с различными ограничениями разных мер сложности имеют важное значение. Для иллюстрации достаточно отметить, что широко известная проблема установления равенства или неравенства классов P и NP (предикатов, распознаваемых за детерминированное и недетерминированное полиномиальное время) может быть проинтерпретирована в этих терминах: равенство этих классов означало бы возможность быстрого решения полиномиально переборных задач. В частности, это следовало бы из равенства классов P и $PSPACE$ (предикатов, распознаваемых на полиномиальной зоне). Последнее означает возможность сократить время получения результата вычислительного процесса (попасть в класс P) за счёт небольшого увеличения объёма памяти (занятой исходным процессом в классе $PSPACE$, который включает в себя и класс NP), и применения радикально иного алгоритма. Такие задачи относятся к классу задач обмена между объёмом памяти и временем вычислений (*time-space trade-off*).

Более общая проблема в этом плане — согласование вычислительных процессов, выполняемых в различных средах. Здесь нас, в частности, интересует согласование вычислительных процессов в биоподобных последовательных компьютерах и массивно параллельных машинах. Те и другие вычисления при некоторых условиях можно охарактеризовать описанными далее классами предикатов.

В настоящей работе мы устанавливаем равенство описанных ниже двух классов вычислимых предикатов, подобно тому, как это сделано в работе П. Клоута [1]. Существенное отличие настоящей работы от работы [1] состоит в выборе классов, имеющих большее практическое значение: суммарная временная сложность рассматриваемых в настоящей работе процессов полиномиальна, то есть, ограничена многочленами от длины двоичной записи обрабатываемых данных, тогда как в работе [1] она экспоненциальна относительно этого параметра.

При обработке больших массивов данных многократное преобразование таких массивов часто имеет слишком большую сложность. Поэтому мы вводим в рассмотрение процессы побуквенной обработки таких цепочек. Кроме того, для одного из классов мы рассматриваем сильно ограниченные вычисления: объём памяти всех промежуточных результатов такого вычисления ограничен некоторой константой. Это, например, может понадобиться при реализации вычислительных процессов, на последовательных компьютерах, не имеющих большой управляемой памяти для передачи данных между этапами вычислений. Это могут быть компьютеры, использующие молекулярные структуры и процессы обработки информации подобно тому, как это происходит в живых клетках с наследственной информацией и информацией, поступающей от взаимодействия внешней и внутренней среды.

Другой определяемый здесь класс вычислений — параллельные (ветвящиеся) вычисления, в которых полное время работы каждой ветви процесса (запаздывание) ограничено логарифмом от размера обрабатываемых данных, число ветвей имеет порядок многочлена от размера входных данных, а результаты ветвей собираются логическими функциями в точках ветвления. Такие вычисления называются альтернирующими. В этих машинах предусматриваются нетривиальные средства, обеспечивающие при таких ограничениях доступ ко всем входным данным. Подобные вычисления могут быть реализованы, например, на векторных процессорах, состоящих из большого числа простых элементов, синхронно выполняющих простые действия по команде внешнего «дирижёра». В пределе это может быть, например, некое подобие компьютера-кристалла. При этом можно использовать физическое (например, оптическое) разделение и копирование сигнала.

В настоящей работе показано равенство вычисляемых такими способами классов предикатов. То есть показан метод, который позволяет в наших случаях осуществлять обмен между временной и емкостной сложностью вычисления. На практике это означает возможность взаимного преобразования биоподобных и «кристаллоподобных» вычислений.

§ 1. Краткий обзор связанных работ

Проблемы взаимосвязей вычислений с различными мерами сложности для различных моделей активно изучаются.

К исследованиям в области компьютеров молекулярной обработки относится работа по теории вычислительной сложности в контексте сплайсинговых систем, представляющих собой вычислительные модели, подобные системам обработки наследственной информации в живых организмах при участии нуклеиновых кислот. Р. Лус и М. Огихара в работе [2] исследовали возможности вычислений и классы сложности, связанные с такими системами.

Для ознакомления с современным взглядом на теорию вычислительной сложности читатель отсылается к работе [3] (О. Гольдрайх), в которой представлено введение в теорию вычислительной сложности.

В дополнение к этому в работе [4] (С. Таюр) дан обзор нетрадиционных вычислений. В статье рассмотрена мотивация для изучения альтернативных моделей вычислений для таких задач, как нелинейное целочисленное программирование, и выделены практические приложения в различных областях.

К. Тейшер в работе [5] исследовал нетрадиционные вычисления как новую область исследований, выходящую за рамки традиционных компьютерных технологий и парадигм. Автор упомянул различные нетрадиционные парадигмы вычислений такие, как квантовые вычисления, оптические вычисления, молекулярные вычисления и химические вычисления, и обсудил проблемы, с которыми сталкиваются нетрадиционные вычисления, включая необходимость превзойти традиционные подходы и понимание того, что до практических приложений ещё далеко. Автор пришел к выводу, что подходы к нетрадиционным вычислениям следует оценивать на основе их собственных достоинств, а не только в сравнении

с существующими технологиями.

Работа В. Хэндли [6] посвящена методам, близким к методам настоящей работы. Автор исследовал концепцию детерминированного суммирования по модулю полугруппы бинарных отношений. Эта концепция обобщает схемы ограниченного суммирования и мультиплицирования, используемые при построении элементарных функций. Она включает ограниченную квантификацию первого порядка по целым числам и основана на полугруппе бинарных отношений на множестве $\{0, 1, \ldots, n-1\}$. Автор показал, что детерминированное суммирование по модулю группы перестановок с ограниченной квантификацией и логическими операциями столь же эффективно, как и детерминированное суммирование по модулю моноида бинарных отношений, что имеет вычислительное значение для некоторых машин, где первое соответствует детерминизму, а второе — недетерминизму.

Другая работа, имеющая отношение к методам настоящей статьи — статья А. Эсбелена [7], в которой он исследовал замкнутость классов отношений относительно подсчёта по модулю или константно ограниченной рекурсии. Рассматриваемые отношения определяются арифметическими формулами первого порядка с ограниченными кванторами, использующими, в частности, такой подсчёт.

Ещё одна работа, связанная с методом настоящей статьи — работа [8] (А. Дюран и М. Мор). В ней исследуются различные расширения линейной иерархии времени, связанной с методами настоящей статьи.

§ 2. Словарные функции

Определим функции побуквенной обработки слов. Пусть задан некоторый конечный алфавит A. Элементы алфавита A принято называть буквами. В простейшем случае рассматриваем двоичный алфавит, состоящий из двух символов: цифры 0 и цифры 1. В данном контексте они также называются буквами.

Будем рассматривать класс многоместных словарных логических функций (предикатов), отображающих слова (конечные цепочки, тексты) в алфавите A в множество логических значений (0 и 1). Каждая такая функция имеет фиксированное число аргументов, каждый из которых может быть произвольным словом в заданном алфавите.

§ 3. Словарные последовательные вычисления с объёмом памяти, ограниченным константой

Далее приводим определение вычислительной модели словарных регистровых машин для вычисления функций рассматриваемого класса. Описываемые устройства состоят из следующих частей:

- конечный набор ячеек памяти для входных слов $w_1, \ldots, w_n$, для простоты здесь рассматриваем двоичный алфавит, содержимое этих ячеек не меняется в процессе работы;

- длины этих слов будем обозначать через $d_1, \ldots, d_n$ соответственно, они не меняются в процессе работы; для простоты считаем, что входные слова не пусты, то есть их длины положительны, позиции в этих словах нумеруются неотрицательными целыми числами (начиная с нуля); число D — наибольшее из чисел $d_1, \ldots, d_n$ — будем называть основанием машины; если считать, что входные слова — это файлы, поданные на вход машины, то основание этой машины — длина самого длинного из этих файлов;

- число K, называемое числом состояний машины;

- ячейка состояния машины, в которой в каждый момент времени находится номер состояния машины, — целое число от 1 до K; в начале работы (на нулевом шаге) в этой ячейке находится число 1 — номер начального состояния;

- допускающее состояние машины — номер состояния F, остановка в этом состоянии означает положительный результат работы машины (машина работает как распознаватель некоторого предиката), остановка в иных состояниях означает отрицательный результат;

- ячейки счётчика времени $t_0, \ldots, t_{m-1}$, на каждом шаге работы в этих ячейках хранится номер этого шага T в виде $T = t_0 + t_1 D + \ldots + t_m D^m$ (в представлении в позиционной системе счисления с основанием D), машина заканчивает работу после шага номер $D^m - 1$, последнего шага, номер которого представим в указанном виде; нетрудно видеть, что время работы машины полиномиально (то есть ограничено многочленом от длины записи исходных данных);

- программа, определяющая работу машины на одном шаге; работа этой программы описана ниже.

В качестве программы машины одного шага мы используем многоленточную машину Тьюринга с оракулами, работающую за время, ограниченное линейной функцией от длины двоичной записи длин её входных данных. Входными данными такой машины Тьюринга являются значения ячеек d_i и t_j. Таким образом, время работы одного шага — логарифмическое от длин входных данных всей словарной машины (так как размеры d_i и t_j ограничены логарифмами от суммы длин входных слов). Оракулы требуются для чтения отдельных позиций входных слов. Имеется по одному оракулу для каждого входного слова. Оракул вызывается при переходе машины в специальное для этого оракула состояние. Номер позиции входного слова кодируется в двоичном виде словом на специальной ленте машины одного шага. Результат работы оракула помещается на этой же ленте. Данная машина одного шага вычисляет номер следующего состояния всей словарной машины.

При практической необходимости можно рассматривать и машины с изменённым определением понятия программы одного шага. Так можно получить другие варианты основного результата настоящей работы.

Описанные словарные машины таким образом вычисляют логические функции (предикаты) на кортежах входных слов.

Заметим, что управляемая память таких машин между шагами сильно ограничена: она состоит только из одной ячейки, в которой может в каждый момент времени храниться только одно из K возможных значений. Значения ячеек t предопределены номером шага вычислительного процесса, программа на них не влияет.

Можно привести следующие примеры задач, решаемых описанными словарными машинами: распознавание конкатенаций ($x_1 x_2 = x_3$), палиндромов (перевёртышей), регулярных множеств, двоичных кодов протоколов работы машин Тьюринга.

§ 4. Словарные альтернирующие вычисления логарифмической временной сложности

В альтернирующих распознавателях вычислительный процесс может ветвиться, разделяясь на несколько параллельных процессов, результаты которых затем объединяются с помощью разных логических функций. Такое ветвление может происходить на каждом шаге вычисления. Здесь будем рассматривать простейший вариант, при котором процесс разделяется на каждом шаге на две ветви (далее — ветви с идентификаторами 0 и 1). По завершении

обеих ветвей их результаты объединяются с помощью отрицания их конъюнкции. Таким образом, внеся все отрицания вовнутрь всей получающейся формулы вычисления, общий результат можно равнозначно представить как получающийся чередованием конъюнкций и дизъюнкций результатов отдельных ветвей.

Опишем структуру и работу такой словарной альтернирующей машины. Машина кое в чём похожа по структуре на последовательную словарную машину из предшествующего параграфа и состоит из следующих частей:

- конечный набор ячеек памяти для входных слов $w_1, \ldots, w_n$ в двоичном алфавите, содержимое этих ячеек не меняется в процессе работы;
- длины этих слов будем обозначать через $d_1, \ldots, d_n$ соответственно, они не меняются в процессе работы; для простоты так же считаем, что эти слова не пусты, то есть их длины положительны;
- числа m — степени сложности работы машины;
- ячейки идентификатора ветви вычисления t, на последнем этапе ветви на шаге номер $m[\log_2(d_1 + \ldots + d_n + n - 1) + 1]$ (умноженная на m длина двоичной записи входных данных без учёта самих входных слов $w_1, \ldots, w_n$) в этой ячейке вычисления находится двоичный код этой ветви — последовательность идентификаторов ветвей (0 — «левая» ветвь, 1 — «правая» ветвь), записанная слева направо в порядке выполнения разветвлений;
- программы машины, работа которой описана ниже; программа работает на последнем шаге машины в каждой ветви.

В качестве программы мы используем машину Тьюринга с оракулами, работающую за время, ограниченное линейной функцией от длины двоичной записи длин входных данных. Входными данными такой машины являются значения ячеек d_i и t. Оракулы требуются для чтения отдельных позиций входных слов, для каждого входного слова имеется свой оракул, номер позиции входного слова кодируется в двоичном виде словом на специальной ленте этой машины Тьюринга. Данная машина Тьюринга выдаёт логическое значение — результат работы одной ветви. Время работы машины на каждой ветви («запаздывание») — логарифмическое (ограничено логарифмом от длины записи исходных данных — входных слов $w_1, \ldots, w_n$ — если считать, что входные данные имеют длину, бóльшую 1).

§ 5. Взаимное моделирование вычислений

В данном параграфе доказывается, что модели, приведённые в §§ 3, 4, вычисляют один и тот же класс предикатов.

Теорема 5.1 (О взаимном моделировании последовательных и параллельных словарных вычислений). *Словарные регистровые машины и словарные альтернирующие машины вычисляют один и тот же класс предикатов.*

§ 6. Доказательство теоремы 5.1

Доказательство, как и в работе [1], состоит из двух частей:

1) моделирование словарных регистровых машин словарными альтернирующими машинами,

2) моделирование словарных альтернирующих машин словарными регистровыми машинами.

Первая часть доказательства. Моделирование словарных регистровых машин словарными альтернирующими машинами выполняется довольно просто методом дихотомии. Указываются начальное и конечное состояния словарной регистровой машины и решается задача достижимости между ними. При этом весь процесс вычисления вычисления словарной регистровой машины делится пополам. Перебираются все возможные состояния машины в середине процесса. Затем рекурсивно моделируются первая и вторая части процесса. Полученные результаты соединяются дизъюнкцией. Рекурсия заканчивается при достижении длины процесса в 1 шаг, при этом альтернирующая машина может непосредственно выдать результат. Указанный процесс нетрудно запрограммировать для определённой выше модели альтернирующей машины.

Вторая часть доказательства. Моделирование словарных альтернирующих машин словарными регистровыми машинами производится тем же алгебраическим методом, которым и решается аналогичная задача в работе [1]. Так же, как в работе [1], можно ограничиться регистровыми машинами с 5 состояниями. Суть решения работы [1] состоит в том, что во множестве перестановок элементов $\{1, 2, 3, 4, 5\}$ следует выбрать два цикла (123) и (345). Обратные перестановки — (321) и (543) соответственно. Если мы хотим, например, соединить конъюнкцией результаты двух ветвей процесса: A и B, то с истинным значением ветви A нужно связать 2 процесса: первый (A_1) в случае истинности A выполняет перестановку (123), иначе он выполняет пустую перестановку, а второй (A_2) — аналогично выполняет перестановку (321) и пустую соответственно. Аналогично следует завести процессы B_1 и B_2 для B с перестановками (345) и (543) соответственно. Заметим, что эти циклы имеют общий элемент 3. Теперь запустим процессы в следующем порядке: $(A_1)(B_1)(A_2)(B_2)$. Если хотя бы одна из формул ложна, то в результате получится пустая перестановка. Иначе получим эволюцию: $12345, 31245, 31524, 15324, 15243$, то есть в итоге получим цикл (235). Переставля местами A_1 и A_2, а также B_1 и B_2, нетрудно получить циклы $(532), (134)$ и (431). То есть на следующем уровне моделирования альтернирующей машины опять имеем 2 цикла по 3 элемента в каждом, с одним общим элементом. Чередуя эти процессы, мы можем промоделировать весь процесс работы исходной альтернирующей машины. Нетрудно видеть, что связанные с логическим значением ветви перестановки вычисляются за линейное время по номеру шага работы регистровой машины. Это и даёт доказательство второй части теоремы.

Таким образом, доказательство теоремы завершено. □

§ 7. Возможные практические приложения

Теорему 5.1 с практической точки зрения можно интерпретировать следующим образом:

1) последовательные вычисления, подобные молекулярным биологическим процессам обработки генетической информации, могут быть ускорены экспоненциально на альтернирующем параллельном компьютере с не более чем полиномиальной потерей ресурса суммарного процессорного времени;

2) объём памяти вычислений на альтернирующем параллельном компьютере может быть уменьшен до константы с не более чем полиномиальной потерей ресурса процессорного времени при переносе на биоподобные молекулярные последовательные процессоры.

Следует отметить, что генетические вычисления традиционно используются для адаптации и обучения в результате взаимодействия с внешней средой и трансформации постановок задач. Так что первый пункт можно использовать для ускорения процессов обучения и адаптации.

Заключение: выводы и оставшиеся проблемы

По существу математически в работе решён новый более практичный вариант ранее решавшейся теоретической задачи, которая и сейчас остаётся чисто теоретической ввиду немыслимой вычислительной сложности, и на практике для достаточно больших аргументов такие вычисления не реализуемы (во всяком случае, на протяжении жизни человека). Тем не менее, вариант задачи, предложенный здесь нами, является более практичным с точки зрения сложности вычислений: эти вычисления требуют экспоненциально меньшего объёма ресурсов для тех же объёмов входных данных, то есть практически реализуемы. Оказалось, что задача в новой постановке решается способом, похожим на предложенный ранее для более ресурсоемкой задачи. Следует отметить, что в настоящей работе нами дано более компактное изложение решения, чем в исходной работе для исходной задачи, доступное поэтому для адаптации и дальнейшего использования.

Для дальнейших исследований остаётся проблема построения вычислительной модели, в которой эффективно сочетаются параллельные и последовательные вычисления. Например, слои последовательных и параллельных вычислений могут перемежаться. Эффективность при этом должна учитывать использование одних ресурсов для экономии других.

Имеется также проблема обеспечения надёжности и безопасности параллельных и последовательных процессов, а также сохранения надёжности и безопасности при преобразовании этих процессов.

Имеется проблема учёта постановки решаемой задачи программирования для более эффективного преобразования программ. Это важно, потому что часто при реальном программировании в самом начале уже стоят некоторые дилеммы, в частности, дилеммы обмена между емкостной и временной сложностью вычислений. Это также выбор наиболее эффективной задачи из решаемого семейства задач. Таким процессом можно управлять, в частности, с помощью возможно расширенного механизма типизации, учитывающего ресурсы в постановках задач, как физические, так и информационные. Сущность такого механизма может быть выражена средствами конструктивной логики в реализационной интерпретации.

Остаётся проблема связи между конструктивно-логическим типом значения и физическим его типом. Некоторые вычисления могут быть запрограммированы из размерностных соображений. Однако это никак не учитывается в конструктивно-логическом синтезе программ.

Близкими к этой тематике являются проблемы самоорганизации, самообучения и самоадаптации программно-информационных объектов.

Другие проблемы — проблемы организации памяти и доступности к использованию значений и процессов по операциям и преобразованиям. Это тоже относится к управлению ресурсами и процессами.

В конечном итоге решение этих проблем позволит создавать более эффективные, надёжные и безопасные вычислительные системы.

СПИСОК ЛИТЕРАТУРЫ

1. Clote P. Nondeterministic stack register machines // Theoretical Computer Science. 1997. Vol. 178. Issues 1–2. P. 37–76. https://doi.org/10.1016/S0304-3975(96)00051-5

2. Loos R., Ogihara M. Complexity theory for splicing systems // Theoretical Computer Science. 2007. Vol. 386. Issues 1–2. P. 132–150. https://doi.org/10.1016/j.tcs.2007.06.010
3. Goldreich O. Computational complexity: a conceptual perspective // ACM SIGACT News. 2008. Vol. 39. No. 3. P. 35–39. https://doi.org/10.1145/1412700.1412710
4. Tayur S. Unconventional computing: applications, hardware, algorithms // Quantum Computing. 2021. Vol. 48. No. 1. https://doi.org/10.1287/orms.2021.01.14
5. Teuscher C. Unconventional computing catechism // Frontiers in Robotics and AI. 2014. Vol. 1. Issue 10. https://doi.org/10.3389/frobt.2014.00010
6. Handley W. G. Deterministic summation modulo $\mathscr{B}_n$, the semigroup of binary relations on $\{0, 1, \ldots, n-1\}$ // Theoretical Computer Science. 1997. Vol. 172. Issues 1–2. P. 135–174. https://doi.org/10.1016/S0304-3975(95)00245-6
7. Esbelin H.-A. Counting modulo finite semigroups // Theoretical Computer Science. 2001. Vol. 257. Issues 1–2. P. 107–114. https://doi.org/10.1016/S0304-3975(00)00112-2
8. Durand A., More M. Nonerasing, counting, and majority over the linear time hierarchy // Information and Computation. 2002. Vol. 174. Issue 2. P. 132–142. https://doi.org/10.1006/inco.2001.3084

Поступила в редакцию 16.04.2024

Принята к публикации 20.05.2024

Бельтюков Анатолий Петрович, д. ф.-м. н., профессор, кафедра теоретических основ информатики, Удмуртский государственный университет, 426034, Россия, г. Ижевск, ул. Университетская, 1.
ORCID: https://orcid.org/0000-0002-3433-9067
E-mail: belt.udsu@mail.ru

Маслов Сергей Геннадьевич, к. т. н., доцент, кафедра теоретических основ информатики, Удмуртский государственный университет, 426034, Россия, г. Ижевск, ул. Университетская, 1.
ORCID: https://orcid.org/0000-0003-0902-6584
E-mail: msh.sci@mail.ru

Джудакизаде Милад, студент, кафедра теоретических основ информатики, Удмуртский государственный университет, 426034, Россия, г. Ижевск, ул. Университетская, 1.
ORCID: https://orcid.org/0000-0002-6167-6237
E-mail: dzhudakizade@udsu.ru

Цитирование: А. П. Бельтюков, С. Г. Маслов, М. Джудакизаде. Взаимное моделирование последовательных и параллельных словарных вычислений // Вестник Удмуртского университета. Математика. Механика. Компьютерные науки. 2024. Т. 34. Вып. 2. С. 299–308.

COMPUTER SCIENCE 2024. Vol. 34. Issue 2. Pp. 299–308.

A. P. Beltyukov, S. G. Maslov, M. Joudakizadeh

Mutual modeling of sequential and parallel word computations

Keywords: word predicates, parallel computing, sequential computing, big data, computational complexity, biosimilar computers, vector-matrix computers, alternation.

MSC2020: 03D15, 68Q05

DOI: 10.35634/vm240208

The work is devoted to the connection between parallel and sequential computing. On the one hand, we consider a class of word predicates based on sequential calculations, limited in memory by constants and having polynomial time complexity. On the other hand, we consider a class of word predicates that are computable on parallel alternating machines in logarithmic time. The coincidence of the corresponding classes is proven. The direction of using the obtained results for mutual transformation and combination of calculations on molecular biosimilar sequential machines and parallel calculations on vector-matrix computers is proposed. Intended applications: real-time image processing for control tasks, analysis of large texts and other big data.

REFERENCES

1. Clote P. Nondeterministic stack register machines, *Theoretical Computer Science*, 1997, vol. 178, issues 1–2, pp. 37–76. https://doi.org/10.1016/S0304-3975(96)00051-5
2. Loos R., Ogihara M. Complexity theory for splicing systems, *Theoretical Computer Science*, 2007, vol. 386, issues 1–2, pp. 132–150. https://doi.org/10.1016/j.tcs.2007.06.010
3. Goldreich O. Computational complexity: a conceptual perspective, *ACM SIGACT News*, 2008, vol. 39, no. 3, pp. 35–39. https://doi.org/10.1145/1412700.1412710
4. Tayur S. Unconventional computing: applications, hardware, algorithms, *Quantum Computing*, 2021, vol. 48, no. 1. https://doi.org/10.1287/orms.2021.01.14
5. Teuscher C. Unconventional computing catechism, *Frontiers in Robotics and AI*, 2014, vol. 1, issue 10. https://doi.org/10.3389/frobt.2014.00010
6. Handley W. G. Deterministic summation modulo $\mathscr{B}_n$, the semigroup of binary relations on $\{0, 1, \ldots, n-1\}$, *Theoretical Computer Science*, 1997, vol. 172, issues 1–2, pp. 135–174. https://doi.org/10.1016/S0304-3975(95)00245-6
7. Esbelin H.-A. Counting modulo finite semigroups, *Theoretical Computer Science*, 2001, vol. 257, issues 1–2, pp. 107–114. https://doi.org/10.1016/S0304-3975(00)00112-2
8. Durand A., More M. Nonerasing, counting, and majority over the linear time hierarchy, *Information and Computation*, 2002, vol. 174, issue 2, pp. 132–142. https://doi.org/10.1006/inco.2001.3084

Received 16.04.2024

Accepted 20.05.2024

Anatolii Petrovich Beltyukov, Doctor of Physics and Mathematics, Professor, Department of Theoretical Foundations of Computer Science, Udmurt State University, ul. Universitetskaya, 1, Izhevsk, 426034, Russia.
ORCID: https://orcid.org/0000-0002-3433-9067
E-mail: belt.udsu@mail.ru

Sergey Gennadievich Maslov, Candidate of Engineering, Associate Proftssor, Department of Theoretical Foundations of Computer Science, Udmurt State University, ul. Universitetskaya, 1, Izhevsk, 426034, Russia.
ORCID: https://orcid.org/0000-0003-0902-6584
E-mail: msh.sci@mail.ru

Milad Joudakizadeh, Student, Department of Theoretical Foundations of Computer Science, Udmurt State University, ul. Universitetskaya, 1, Izhevsk, 426034, Russia.
ORCID: https://orcid.org/0000-0002-6167-6237
E-mail: dzhudakizade@udsu.ru

Citation: A. P. Beltyukov, S. G. Maslov, M. Joudakizadeh. Mutual modeling of sequential and parallel word computations, *Vestnik Udmurtskogo Universiteta. Matematika. Mekhanika. Komp'yuternye Nauki*, 2024, vol. 34, issue 2, pp. 299–308.