ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ


Издается с сентября 2010 г.



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

# СОДЕРЖАНИЕ



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

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/pi.html E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

# SOFTWARE ENGINEERING

## PROGRAMMNAYA INGENERIA

№ 9
September
2015

**Published since September 2010**



## CONTENTS



**Information about the jornal is available online at:**
**http://novtex.ru/pi.html, e-mail:prin@novtex.ru**

УДК 004.4'416

**Н. И. Вьюкова**, ст. науч. сотр., e-mail: niva@niisi.msk.ru,
**В. А. Галатенко**, д-р физ.-мат. наук, ст. науч. сотр., e-mail: galat@niisi.msk.ru,
**С. В. Самборский**, ст. науч. сотр., e-mail: sambor@niisi.msk.ru,
Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук, г. Москва

# Совмещение выбора и планирования команд в программной конвейеризации циклов

*Представлен разработанный авторами метод программной конвейеризации циклов, в котором выбор и планирование команд реализуются совместно, как решение задачи целочисленного линейного программирования. Такой подход позволяет при выборе команд учитывать различные факторы, такие как возможности параллельного исполнения команд, межитерационные зависимости, дефицит регистров, наличие специализированных команд.*

***Ключевые слова:*** *генерация кода, программная конвейеризация циклов, планирование по модулю, выбор команд, целочисленное линейное программирование*

## Введение

Программная конвейеризация циклов — это способ оптимизации циклов, использующий параллелизм операций, относящихся к разным итерациям цикла. Наиболее широко в практике компиляции в настоящее время применяется метод программной конвейеризации, называемый планированием по модулю (*modulo scheduling*). Цель этого метода заключается в том, чтобы построить расписание, при котором итерации цикла запускались бы с некоторым постоянным интервалом, а также свести этот интервал к минимуму за счет совмещения операций из нескольких соседних итераций цикла.

В традиционной схеме компиляции программная конвейеризация циклов выполняется обычно на стадии генерации кода после выбора команд, но до распределения регистров. Выбор команд и конвейеризацию циклов реализуют, как правило, при помощи быстрых эвристических алгоритмов. Такой подход упрощает реализацию генератора кода, но обладает рядом недостатков с точки зрения оптимальности генерируемого кода. Во-первых, эвристические алгоритмы не всегда обеспечивают высокую эффективность кода для специализированных процессоров, которые используются во встроенных системах. Во-вторых, раздельное выполнение проходов генерации кода не позволяет учесть факторы их взаимного влияния. Например, как будет показано в подразд. 2.2, выбор команд может ограничить возможности эффективной конвейеризации циклов. Кроме того, прирост производительности, полученный за счет конвейеризации цикла, может быть потерян в результате вставки spill-кода при распределении регистров.

Идеальным с точки зрения оптимального характера генерируемого кода мог бы быть подход, представленный в докладе [1]. В нем задачи оптимизации и генерации кода рассматривали как единую задачу математического программирования. В настоящей работе предложен метод генерации кода, в котором совместно решаются задачи выбора и планирования команд, представленные как задача целочисленного линейного программирования (ЦЛП). Эффективность кода при таком подходе достигается за счет того, что при выборе команд учитывается множество факторов, включая возможность параллельного исполнения команд, межитерационные зависимости по данным, дефицит регистров, наличие общих подвыражений. При выборе и планировании команд учитываются ограничения по числу доступных регистров и при необходимости автоматически генерируется spill-код, так что на стадии распределения регистров его генерация не требуется [2, 3].

Актуальность совмещения проходов оптимизации и генерации кода, а также применения точных методов их реализации подтверждается результатами ряда исследовательских работ. Так, в работах [4—6] рассмотрено совместное решение задач выбора и планирования команд для линейных участков в генерации кода для разных типов процессоров с применением методов динамического программирования [4], методов ЦЛП [5] и методов программирования в ограничениях (*constraints programming*, CP) [6]. В работе [7] рассмотрена реализация совместного выполнения глобального планирования и распределения регистров на основе метода CP. В работе [8] представлена формулировка задачи ЦЛП трансформаций цикла (развертки и программной конвейеризации) с ми-

нимизацией энергопотребления при заданном ограничении по времени выполнения.

Наиболее близкий к предлагаемому в настоящей работе подход представлен в работе [9], где исследуется влияние совмещения проходов генерации кода на его эффективность. В работе предложена формулировка задачи ЦЛП выбора и планирования команд с распределением регистров для циклических участков кода в генерации кода для кластерной архитектуры. Преимущества подхода, предложенного в настоящей работе, по сравнению с подходом, предложенным в работе [9], заключаются в поддержке команд с несколькими выходными значениями; команд, переписывающих входные значения; механизмов автоматической генерации spill-кода или регенерации значений.

Дальнейшее содержание статьи построено по следующему плану. В разд. 1 введены основные понятия, связанные с программной конвейеризацией циклов, и рассмотрены особенности конвейеризации циклов с выбором команд. В разд. 2 рассмотрены примеры программной конвейеризации циклов с выбором команд. В разд. 3 обсуждены формулировки ЦЛП-задач программной конвейеризации циклов с выбором команд, сложность этих задач и солверы, применяемые для их решения. В заключении обобщен опыт проделанной работы и рассмотрены перспективы дальнейших исследований.

## 1. Планирование с выбором команд для циклических участков

В этом разделе представлены основные понятия, относящиеся к программной конвейеризации циклов. Также дано краткое описание метода программной конвейеризации, в котором совмещаются выбор и планирование команд.

### 1.1. Понятия, связанные с программной конвейеризацией циклов

Результатом программной конвейеризации является цикл, который выполняет те же вычисления, что и исходный. Этот цикл совмещает в своем теле команды, относящиеся к разным итерациям исходного цикла. За счет такого совмещения обеспечивается скрытие латентностей команд и достигается эффективное использование параллелизма на уровне команд, присущего современным микропроцессорным архитектурам. Интервал между запусками последовательных итераций конвейеризованного цикла (в тактах) называют основным интервалом, или интервалом запуска, и обозначают $II$ (*Initiation Interval*). Для того чтобы обеспечить эквивалентность конвейеризованного цикла исходному, перед входом в цикл и после выхода из него добавляют пролог и эпилог, содержащие некоторые команды, присутствующие соответственно в начальных и конечных итерациях исходного цикла. Кроме того, должно быть скорректировано число повторений цикла.

Размер диапазона соседних итераций исходного цикла, команды которых совмещаются в теле конвейеризованного цикла, называется глубиной конвейеризации. Например, если в первой итерации конвейеризованного цикла выполняются команды из первой, третьей и четвертой итераций исходного цикла, то значение глубины составляет 4. От глубины конвейеризации зависит размер пролога и эпилога конвейеризованного цикла. Она также косвенно влияет на общее число аппаратных регистров, которые необходимы для конвейеризованного цикла. Подробнее об этих и других аспектах генерации кода для конвейеризованных циклов см. обзор [10].

В большинстве современных компиляторов программную конвейеризацию циклов реализуют при помощи метода планирования по модулю (*modulo scheduling*). Суть этого метода состоит в том, что для значений $II$ из некоторого диапазона $[II_{\min}, II_{\max}]$ последовательно делают попытки построить расписание цикла, укладывающееся в $II$ тактов на итерацию. Расписание для минимального $II$, для которого это удалось сделать, и будет использовано при генерации кода.

Как правило, построение расписаний для фиксированного $II$ осуществляется при помощи быстрых эвристических алгоритмов, обзор которых можно найти в работе [10]. Для нахождения оптимальных расписаний могут применяться точные подходы, использующие методы ЦЛП [9, 11, 12].

### 1.2. Программная конвейеризация циклов с выбором команд

Процесс генерации кода включает следующие основные шаги.

**Шаг 1. Считывание и обработка входных данных**, состоящих из описания системы команд целевого микропроцессора и тела исходного цикла в виде набора поддеревьев, представленных в списочной ЛИСП-подобной форме (см. пример в подразд. 2.2).

**Шаг 2. Выделение множества уникальных поддеревьев** и отождествление общих подвыражений в древовидном представлении тела исходного цикла.

**Шаг 3. Первичный выбор команд**, т. е. формирование множества команд, которые в принципе могут участвовать в реализации заданного циклического вычисления. Для этого корень каждого уникального поддерева, сформированного на шаге 2, сопоставляется с древовидными шаблонами в описаниях команд процессора.

На этом шаге, так же как описано в работе [6], сохраняются все команды, для которых оказалось успешным сопоставление хотя бы одного из ее шаблонов с вершиной какого-либо поддерева из множества, сформированного на шаге 2. В отличие от традиционных декларативных методов выбора команд [13, 14] не делается попытка найти оптимальный набор команд, выбор которого откладывается до шага 4.

**Шаг 4. Формирование и решение ЦЛП-задач** выбора и планирования команд при фиксированных значениях основного интервала ($II$) и глубины конвейеризации ($DP$). Параметр $II$ пробегает значения 1, 2, ..., $II_{\max}$, где $II_{\max}$ — оценка сверху для величины $II$.

Для каждого *II* значение *DP* варьируется от 1 до $DP_{max}$, где $DP_{max}$ — максимальная глубина конвейеризации. Процесс завершается, когда для очередной пары значений *II*, *DP* удается найти решение ЦЛП-задачи.

**Шаг 5. Вывод результатов**. Решение ЦЛП-задачи, если оно было найдено, выводится в виде расписания выполнения конвейеризованного цикла.

## 2. Примеры генерации кода

В этом разделе рассмотрены примеры генерации кода для нескольких программных циклов, продемонстрированы основные преимущества предлагаемого метода программной конвейеризации. Предварительно приведено краткое описание гипотетической микропроцессорной архитектуры, для которой осуществляется генерация кода.

### 2.1. Описание целевой архитектуры

Описание целевой архитектуры состоит из нескольких разделов, содержание которых представлено далее.

**Ресурсы** (вычислительные устройства, регистры) процессора. Архитектура, описание которой использовано в примерах, включает два устройства выдачи команд на исполнение, два устройства сложения, устройство умножения, устройство деления, устройство деления и извлечения квадратного корня, а также 32 регистра общего назначения и один регистр кода условия.

**Типы хранилищ данных**. Результаты вычислений (значения) могут помещаться в разные типы хранилищ. В рассматриваемой архитектуре предусмотрены хранилища четырех видов: "`D`" — общие регистры в количестве 32; "`M`" — память (может использоваться без ограничений); "`V`" — фиктивные значения (типа void), порождаемые, в частности, командами записи в память; "`CC`" — регистр кода условия (один).

**Команды процессора** и правила оптимизации. В таблице представлена сводка команд целевого процессора и правил оптимизации, которые использовались в примерах. В первом столбце подчеркиванием выделены аргументы команд, являющиеся входными и выходными. В последнем столбце приведены латентности команд (**L**). Использование ресурсов командами определяется таблицами резервирования, содержание которых здесь не приведено. Каждая команда на первом такте использует одно устройство выдачи команд на исполнение. Вычислительные команды используют также устройства сложения, умножения, деления, деления и извлечения корня. Команды `save`, `restore` (являющиеся синонимами `store`, `load`) служат для реализации спиллинга — сохранения значения из регистра в память и последующего восстановления. Описания команд содержат также спецификации древовидных шаблонов сопоставления, при помощи которых осуществляется первичный выбор команд (см. шаг 3 в подразд. 1.2). Поддерживается представление команд, порождающих два (или более) результата.

**Команды процессора и правила оптимизации**

| Команда | Комментарий | L |
|---|---|---|
| **add** D0,D1,D2 | Сложение | 2 |
| **sub** D0,D1,D2 | Вычитание | 2 |
| **addsub** D0,D1 | Сложение и вычитание | 2 |
| **mul** D0,D1,D2 | Умножение | 3 |
| **madd** D0,D1,D2 | Умножение с накоплением | 3 |
| **msub** D0,D1,D2 | Умножение с вычитанием | 3 |
| **neg** D0,D1 | Смена знака | 2 |
| **div1** D0,D1,D2 | Деление на устройстве divsqrt | 5 |
| **div2** D0,D1,D2 | Деление на устройстве div | 4 |
| **sqrt** D0,D1 c_sqrt, 7 | Квадратный корень | 7 |
| **add** D0,D1,D1 | Оптимизация выражений 2*D1 и D1*2 | 2 |
| **save** D,M | Сохранение регистра | 1 |
| **restore** D,M | Восстановление регистра | 2 |
| **load** D,L | Чтение переменной | 2 |
| **store** D,L | Запись переменной | 3 |
| **load** D,C | Чтение константы | 2 |
| **move** D0,D1 | Копирование регистра | 2 |
| **test** D0 | Получить код условия | 1 |
| **move_cc** D0 c_simple, 2 | Копировать код условия в регистр | 2 |
| **if***U* D0,D2,D3 | Условное копирование | 2 |
| **if***U* D0,D1,D2,D3 | Условное копирование по CC, сохраненному в D1 | 2 |
| **istore** D0,D1 | Запись с преинкрементацией | 3 |
| **iload** D0,D1 | Чтение с преинкрементацией | 2 |
| **storei** D0,D1 | Запись с постинкрементацией | 3 |
| **loadi** D0,D1 | Чтение с постинкрементацией | 2 |

### 2.2. Цикл вычисления скалярного произведения векторов

Рассмотрим пример цикла, реализующего вычисление скалярного произведения векторов:

```
for (i = 0; i<N; i ++) {
a = *(++ pa)
b = *(++ pb)
c = c + a*b
}
```

Ниже приведена запись тела цикла на языке генератора кода. В комментариях показаны соответствующие вычисления на С-подобном псевдоязыке. Через pa′, pb′, c′ обозначены значения переменных pa, pb, c на предыдущей итерации.

```
Scal _ prod _ test = (
# c = c′ + a*b
("label", "c", ("out", ("D","M"),
  ("add", ("look", "c", 1),
      ("mul",("look", "a", 0),
            ("look", "b", 0)))),
# a = load(pa)
("label", "a",
  ("load", ("look", "pa", 0))),
# b = load(pb)
("label", "b",
  ("load", ("look", "pb", 0))),
# pa = pa′ + 4
("label", "pa", ("add",
  ("look", "pa", 1), ("const",4))),
# pb = pb′ + 4
("label", "pb", ("add",
  ("look", "pb", 1), ("const",4)))
)
```

Запись ("label", "*метка*", *дерево*) вводит метку, по которой можно сослаться на дерево при помощи конструкции ("look", "*метка*", *уровень*), где *уровень* указывает, к какой итерации относится ссылка: 0 — значение из той же итерации, 1 — значение из предыдущей и т. д.

Решение ЦЛП-задачи для данного цикла было получено при значениях $II = 2$, $DP = 4$, т. е. расписание конвейеризованного цикла требует два такта на итерацию, и в его теле совмещаются команды из четырех соседних итераций исходного цикла (листинг 1).

```
0: load Db,++(Dpb<<Dpb')   [4]
   add Dc,Dc',Da*b          [1]
1: mul Da*b,Da,Db           [3]
   load Da,++(Dpa<<Dpa')    [4]
```

**Листинг 1.** Тело конвейеризованного цикла для вычисления скалярного произведения векторов

В именах виртуальных регистров подстрочными индексами показаны программные переменные или выражения, которым соответствуют регистры. Штрих указывает на значение переменной от предыдущей итерации. Запись $D_{pb} << D_{pb}'$ указывает, что команда записывает новое значение регистра на место прежнего. В квадратных скобках показан номер итерации исходного цикла, к которой относится команда в первой итерации конвейеризованного цикла.

В листинге 1 не показаны пролог и эпилог конвейеризованного цикла, в которые выносятся "недостающие" команды из начальных и конечных итераций. При генерации выходного машинного кода потребуется также развертка цикла, поскольку время жизни некоторых значений на регистрах составляет более одной итерации. Подробное обсуждение этих аспектов генерации кода для конвейеризованных циклов можно найти в работе [10].

Можно заметить, что в представленном примере для вычисления значения c = c + a*b не была использована команда умножения с накоплением madd, хотя время ее выполнения (три такта) меньше, чем суммарное время выполнения команд mul (три такта) и add (два такта). Традиционный метод подбора команд сгенерировал бы вместо add и mul команду madd. Однако в данном примере использование madd создает циклическую (межитерационную) зависимость с латентностью 3, поэтому интервал запуска $II$ в этом случае не может быть меньше 3. Использование mul и add приводит к тому, что циклическая зависимость переносится на команду сложения, имеющую латентность 2, соответственно, ограничение снизу на $II$ также снижается до двух тактов.

### 2.3. Вычисление корней квадратных уравнений

Рассмотрим теперь более сложный цикл, реализующий вычисление корней квадратных уравнений, заданных массивами коэффициентов. Для вычисления корней применяются следующие формулы:

$$D = b^2 - 4ac;$$

$$y_1 = b + \sqrt{D};$$

$$y_2 = b - \sqrt{D};$$

$$y = \text{if } (b \geqslant 0)\ y_1 \text{ else } y_2;$$

$$x_1 = (-y)/2a;$$

$$x_2 = 2c/(-y).$$

Здесь выбор между $y_1$ и $y_2$ позволяет избежать потери точности. Предполагается, что $D \geqslant 0$. Формула для $x_2$ получается из стандартной формулы домножением числителя и знаменателя на $y_1$ или $y_2$ в зависимости от знака $b$.

Общее число изначально сгенерированных команд — 143. Результат конвейеризации с подбором команд — цикл с $II = 9$, $DP = 4$ (листинг 2).

Тело конвейеризованного цикла состоит из 18 команд, и на каждом такте выполняется по 2 команды. При этом в максимальной степени были использованы "составные" команды: чтение и запись с преинкрементацией, умножение с вычитанием msub, сложение с вычитанием addsub, что обеспечивает эффективное выполнение цикла.

```
 0:  load Da,++(Dpa<<Dpa',)          [4]
     load Dc,++(Dpc<<Dpc',)          [4]
 1:  ifGE,Dy = Dy1,Dy1               [2]
     sqrt Dsqrt(D),DD                [3]
 2:  mul Dac,Da,Dc                   [4]
     load Db,++(Dpb<<Dpb')           [4]
 3:  div2 Dx2,D2c,D-y                [1]
     neg D-y,Dy                      [2]
 4:  mul Db^2,Db,Db                  [4]
     add D2a,Da,Da                   [4]
 5:  add D2c,Dc,Dc                   [2]
     move D4,4                       [4]
 6:  test Db                         [3]
     store Dx1,++(Dpx1<<Dpx2')       [1]
 7:  msub DD<<Db^2,D4,Dac            [4]
     store Dx2,++(Dpx2<<Dpx1)        [1]
 8:  div2 Dx1,D-y,D2a                [2]
     addsub Dy1<<Db,Dy2<<Dsqrt(D)    [3]
```

**Листинг 2.** Конвейеризованный цикл вычисления корней квадратных уравнений

### 2.4. Вектор минимумов из элементов двух векторов

В этом подразделе продемонстрированы примеры автоматической генерации spill-кода. Рассмотрим цикл для вычисления вектора, содержащего минимумы элементов двух входных векторов:

```
for (i = 0; i<N; i ++) {
a = *(++ pa)
b = *(++ pb)
c = a > b ? b : a;
*(++ pc) = c;
}
```

При наличии шести или более регистров класса "D" дефицит регистров отсутствует и результат получается при $II = 3$, $DP = 3$ (листинг 3).

```
0: load Da,++(Dpa<<Dpa')          [3]
   load Db,++(Dpb<<Dpb')          [3]
  Da[2], Db[2], Dpa[2], Dpb[2], Dpc[0], Dc[1]
1: ifGE,Dc=Db,Da                  [2]
   store Dc,++(Dpc<<Dpc')         [1]
  Da[2], Db[2], Dpa[3], Dpb[3], Dpc[0],
  Voutc[0], CCa-b[2], Dc[1]
2: sub Da-b,Da,Db                 [3]
   Da[3], Db[3], Dpa[3], Dpb[3], Dpc[1],
```

**Листинг 3.** Результат конвейеризации (шесть регистров)

В листинге 3, помимо команд для каждого такта, показаны также списки занятых регистров. В квадратных скобках показано, к какой итерации относится значение (0 соответствует значению до входа в цикл). Нетрудно заметить, что на каждом такте занято не более шести регистров.

Если число регистров сократить до пяти, возникает дефицит регистров. Решение для этого случая получено при $II = 4$, $DP = 2$ (листинг 4).

```
0:  load Da,++(Dpa<<Dpa')  [2]
    load Db,++(Dpb<<Dpb')  [2]
  Db[1], Dpa[1], Dpb[1], Dpc[0], CCa-b[1]
1:  ifGE,Dc=Db,Da          [1]
  Da[1], Db[1], Dpa[2], Dpb[2], Dpc[0],
  CCa-b[1]
2:  sub Da-b,Da,Db         [2]
  Da[2], Db[2], Dpa[2], Dpb[2], Dpc[0],
  Voutc[0]
3:  load Da,Dpa            [2]
    store Dc,++(Dpc<<Dpc') [1]
  Db[2], Dpa[2], Dpb[2], Dpc[0], Dc[1]
```

**Листинг 4.** Результат конвейеризации (пять регистров)

В этом расписании присутствует спиллинг регистра $D_a$, который реализуется как повторная загрузка значения (команда load в такте 3). В листинге 4 команды загрузки значения a выделены полужирным шрифтом. Значение a используется дважды, а именно — в вычитании (такт 2) и в условном присваивании (такт 1). Повторная загрузка позволяет высвободить регистр в промежутке между двумя использованиями значения a.

При наличии четырех регистров типа D используется спиллинг регистра $D_{pa}$. Решение получено при $II = 5$, $DP = 2$ (листинг 5).

```
 0:  restore Dpa,Mpa       [1]
    Da[1], Db[1], Mpa[1], Dpb[1], Dpc[0]
 1:  ifGE,Dc=Db,Da         [1]
   Da[1], Db[1], Dpb[1], Dpc[0], Vcout[0],
   CCa-b[1]
 2:  load Da,++(Dpa<<Dpa')     [2]
     load Db,++(Dpb<<Dpb')     [2]
   Dpa[1], Dpb[1], Dpc[0]
 3:  save Dpa,Mpa          [2]
     store Dc,++(Dpc<<Dpc')    [1]
   Dpa[2], Dpb[2], Dpc[0]:1   Dc[1]
 4:  sub Da-b,Da,Db        [2]
   Da[2], Db[2], Mpa[2], Dpb[2], Dpc[1]
```

**Листинг 5.** Результат конвейеризации (четыре регистра)

При наличии трех или двух регистров решение получено при $II = 7$, $DP = 1$ и $II = 8$, $DP = 1$ соответственно. Программная конвейеризация увеличивает потребность в регистрах, и при их нехватке она оказывается невыгодной, поэтому она и не была выполнена при наличии трех и менее регистров ($DP = 1$ означает, что совмещения команд из разных итераций исходного цикла нет). Тем не менее для этих ограничений были получены оптимальные расписания цикла, автоматически выбраны регистры для спиллинга, сгенерированы и включены в расписание соответствующие команды save/restore.

## 3. Формулировки ЦЛП-задач и их сложность

Рассмотрим основные аспекты метода программной конвейеризации циклов с выбором команд и ограничения, используемые в формулиров-

ках ЦЛП-задач, на примере цикла из подразд. 2.2. На шаге 3 (см. подразд. 1.2) для него было сгенерировано 38 команд. Генерируются все возможные команды, которые в принципе могут быть использованы для вычисления значений. Например, для вычисления значения a генерируются следующие команды:

```
load Da,Dpa
iload Da,Dpa
save Da,Ma
restore Da,Ma
move Da,Da'
```

В этом наборе все команды, кроме save, вычисляют a в регистре $D_a$; save вычисляет a в памяти $M_a$. На практике в данном примере для вычисления a использована команда iload (см. листинг 1).

Построение ЦЛП-задач для конвейеризации циклов с подбором команд во многом схоже с построением ЦЛП-задач для планирования линейных участков с выбором команд [2]. Одна итерация исходного цикла рассматривается как линейный участок, и делается попытка спланировать его, используя число тактов, равное $II \cdot DP$.

На рис. 1 показан граф зависимостей для некоторого возможного набора команд, реализующего данный цикл. В скобках для каждой зависимости показаны ее латентность и дистанция.

На рис. 2 в столбце *t* показаны номера тактов. В последующих столбцах показаны расписания четырех последовательных итераций исходного цикла, удовлетворяющие зависимостям по данным. Полужирной рамкой выделено тело конвейеризованного цикла, полученное совмещением команд из черырех соседних итераций исходного цикла (см. листинг 1). Здесь показан только один из возможных наборов команд для этого цикла и одно возможное расписание. В действительности во время решения ЦЛП-задачи происходит перебор возможных наборов команд и расписаний для них.

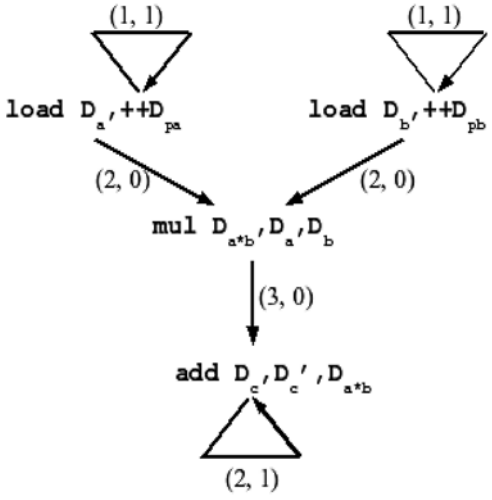


**Рис. 1. Граф зависимостей для цикла, вычисляющего скалярное произведение векторов**

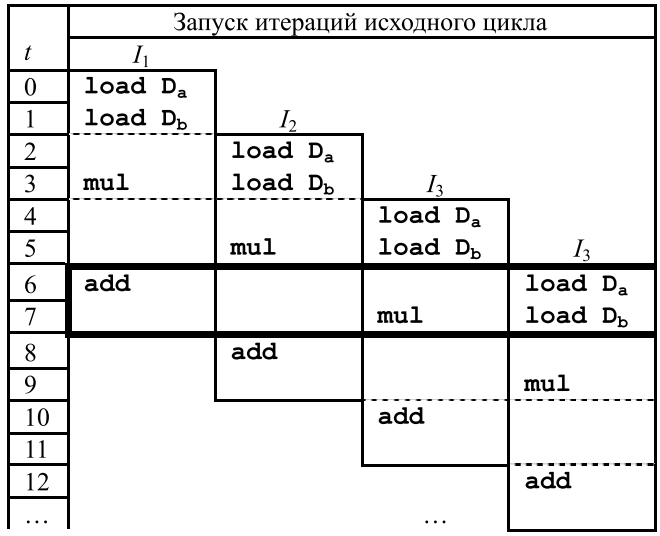


**Рис. 2. Планирование цикла вычисления скалярного произведения векторов**

Основное отличие используемой формулировки ЦЛП-задачи от формулировки, представленной в работе [9], заключается в том, что допускается многократное вычисление значений входного цикла. Такой подход обеспечивает автоматическую генерацию spill-кода или регенерацию значений при нехватке регистров. В формулировке учтена также возможность использования команд, перезаписывающих входные значения.

Сложность ЦЛП-задач зависит от размера входного цикла. Для небольших циклов, таких как примеры из подразд. 2.2, 2.4, были использованы солверы, свободно доступные в исходных текстах, а именно пакеты GLPK [15] и более эффективный CBC [16].

Для цикла из разд. 2.3 (решение квадратных уравнений) мощности этих солверов оказалось недостаточно. Сформированные для этого цикла задачи содержат 5000...7000 переменных, 7000...9000 ограничений с 30 000...40 000 ненулевых элементов. Для этого примера и других сложных циклов были использованы возможности сервера NEOS [17], предоставляющего сервис решения различных видов задач математического программирования, включая задачи ЦЛП, с использованием ряда коммерческих солверов. Наилучшие результаты продемонстрировали солверы Gurobi [18] и особенно XpressMP, являющийся частью пакета FICO™ Xpress Optimization Suite [19]. С их помощью для цикла из разд. 2.3 решение при $II = 9$, $DP = 4$ вычислялось 3...5 мин.

## Заключение

В работе рассмотрен метод программной конвейеризации циклов с выбором команд путем сведения этой задачи к ряду задач математического программирования. Разработана формулировка ЦЛП-задачи выбора и планирования команд с ограничениями по числу регистров для циклических участков кода при фиксированных значениях интервала запуска *II* и глубины конвейеризации *DP*. Решением ЦЛП-задачи, если оно существует, является набор команд, реализующих заданный программный цикл, и план их выполнения, обеспечивающий запуск последовательных итераций с интервалом *II* тактов, при соблюдении ограничений на число регистров.

Предложенный метод обеспечивает единый универсальный подход к решению ряда задач генерации кода: учет возможностей параллельного исполнения при выборе команд, оптимальное применение составных команд, оптимизация вычислений (например, реализация умножения на 2 при помощи сложения), учет числа доступных регистров и генерация spill-кода.

В силу сложности реализации рассмотренного метода и высоких затрат вычислительных ресурсов области его применимости ограничиваются разработкой генераторов высокопроизводительного кода для процессоров, используемых во встроенных системах [12], а также аппаратных ускорителей.

Развитием предлагаемого подхода может быть дополнение процесса генерации кода применением математических тождеств, включая законы коммутативности, ассоциативности и дистрибутивности, формулу разности квадратов и т. п. Как показали эксперименты с применением математических тождеств в генерации кода для линейных участков, это может дать значительный прирост производительности кода. Формулировка ЦЛП-задачи генерации кода может также быть дополнена ограничениями по энергопотреблению.

Перспективным направлением исследований представляется также применение программирования в ограничениях для спецификации и решения задач генерации кода [6, 7].

### Список литературы


1. **Galatenko V. A., Samborskij S. V., Vjukova N. I.** Code generation as a mathematical programming task // Advanced Mathematics, Computations and Applications — 2014. (AMCA—2014). URL: http://conf.nsc.ru/files/conferences/amca14/241897/AMCA_abstracts.pdf (дата обращения 11.03.2015).

2. **Вьюкова Н. И., Галатенко В. А., Самборский С. В.** Совместное решение задач выбора и планирования команд в условиях дефицита регистров // Программная инженерия. 2012. № 2. С. 35—41.

3. **Вьюкова Н. И., Галатенко В. А., Самборский С. В.** Генерация кода методом точного совместного решения задач выбора и планирования команд // Программная инженерия. 2014. № 6. С. 8—15.

4. **Bednarski A.** Integrated Optimal Code Generation for Digital Signal Processors // Linköping Studies in Science and Technology. 2006. Thesis No. 1021.

5. **Bednarski A., Kessler C.** Optimal Integrated VLIW Code Generation with Integer Linear Programming // In European Conf. on Parallel Computing (Euro-Par'06). Berlin: Springer, 2006. P. 461—472.

6. **Arslan M. A., Kuchcinski K.** Instruction Selection and Scheduling for DSP Kernels on Custom Architectures // Euromicro Conference on Digital System Design (DSD). 2013. P. 821—828.

7. **Lozano R. C.** Integrated Register Allocation and Instruction Scheduling with Constraint Programming. URL: https://www.sics.se/~rcas/publications/TRITA-ICT-ECS-AVH-14-13.pdf. (дата обращения 28.05.2015).

8. **Srinivasan K., Chatra K. S.** An ILP Formulation for System Level Througnput and Power Optimization in Multiprocessor SoC Architectures // Proceedings of the 17th International Conference on VLSI Design (VLSID'04). 2004. P. 255—260.

9. **Eriksson M., Kessler C.** Integrated Code Generation for Loops // ACM Transactions on Embedded Computing Systems. 2012. Vol. 11S, N 1. Art. 19. P. 1—25.

10. **Вьюкова Н. И., Галатенко В. А., Самборский С. В.** Программная конвейеризация циклов методом планирования по модулю // Программирование. 2007. № 6. С. 14—25.

11. **Stoutchinin A.** An Integer Linear Programming Model of Software Pipelining for the MIPS R8000 Processor // Proceedings of the 4th International Conference on Parallel Computing Technologies (PaCT '97). 1997. P. 121—135.

12. **Вьюкова Н. И., Галатенко В. А., Самборский С. В.** Программная конвейеризация циклов для ускорителя плавающей арифметики в составе процессора Комдив128-РИО // Программные продукты и системы. 2013. № 4. С. 35—43.

13. **Fraser C. W., Henry R. R., Proebsting T. A.** BURG — Fast optimal instruction selection and tree parsing // SIGPLAN Notices. 1992. N 4. P. 68—76.

14. **Hjort Blindell G. S.** Survey on Instruction Selection. Cornell University Library. URL: http://arxiv.org/pdf/1306.4898v2.pdf. (дата обращения 28.05.2015).

15. **GNU** Linear Programming Kit. URL: http://www.gnu.org/software/glpk (дата обращения 11.03.2015).

16. **CBC** User Guide. URL: http://www.coin-or.org/Cbc/cbcuserguide.html (дата обращения 11.03.2015).

17. **NEOS** Server: State-of-the-Art Solvers for Numerical Optimization. URL: http://www.neos-server.org/neos (дата обращения 11.03.2015).

18. **Gurobi** Optimization. URL: http://www.gurobi.com (дата обращения 11.03.2015).

19. **FICO™** Xpress Optimization Suite. URL: http://www.fico.com/en/products/fico-xpress-optimization-suite (дата обращения 11.03.2015).

**N. I. V'jukova,** Senior Research Fellow, e-mail: e-mail: niva@niisi.msk.ru,
**V. A. Galatenko,** Head of Department, e-mail: galat@niisi.msk.ru,
**S. V. Samborskij,** Senior Research Fellow, e-mail: sambor@niisi.msk.ru,
Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Science", Moscow

# Combining Instruction Selection and Scheduling in Software Pipelining

*The traditional code generation scheme involves three major passes: instruction selection, instruction scheduling, and register allocation. Separation of these tasks and use of heuristic algorithms for their implementation is quite acceptable in compilers for modern universal architectures. Still such an approach may provide insufficient code efficiency for peculiar architectures used in embedded systems.*

*The paper presents a method of software pipelining of loops by means of exact joint solution of the instruction selection and instruction scheduling tasks for cyclic blocks of program code. We propose an approach of treating the above tasks as a single Integer Linear Programming (ILP) task. This allows taking into account interdependences between the three code generation passes. The method being proposed provides a unified approach to various code generation and optimization problems including automatic use of complex instructions (with multiple results), common subexpression elimination, automatic generation and scheduling of spill code, instruction selection with respect to available instruction-level parallelism and data dependencies including interloop ones.*

*Complexity of implementation and heavy use of computing resources makes this approach hardly usable in commonly used compilers. Therefore the method is intended mostly for development of hardware accelerators and generation of high performance code for embedded systems.*

*Possible directions for further enhancement of the method include restrictions on energy consumption in formulations of ILP tasks and use of mathematical equalities in the process of code generation. One more perspective research direction is use of constraints programming for specification of code generation tasks.*

**Keywords:** *code generation, software pipelining, modulo scheduling, instruction selection, Integer Linear Programming (ILP)*

### References

1. **Galatenko V. A., Samborskij S. V., Vjukova N. I.** Code generation as a mathematical programming task, *Advanced Mathematics, Computations And Applications — 2014.* (*AMCA—2014*), available at: http://conf.nsc.ru/files/conferences/amca14/241897/AMCA_abstracts.pdf.

2. **V'jukova N. I., Galatenko V. A., Samborskij S. V.** Sovmestnoe reshenie zadach vybora i planirovaniya komand v usloviyakh defitsita registrov *Programmnaya ingeneria*, 2012, no. 2, pp. 35—41 (in Russian).

3. **V'jukova N. I., Galatenko V. A., Samborskij S. V.** Generatsiya koda metodom tochnogo sovmestnogo resheniya zadach vybora i planirovaniya komand. *Programmnaya ingeneria*, 2014, no 6, pp. 8—15. (in Russian).

4. **Bednarski A.** Integrated Optimal Code Generation for Digital Signal Processors. *Linkoping Studies in Science and Technology*, 2006, Thesis no. 1021.

5. **Bednarski A., Kessler C.** Optimal Integrated VLIW Code Generation with Integer Linear Programming. *In European Conf. on Parallel Computing (Euro-Par'06).* Berlin, Springer, 2006, pp. 461—472.

6. **Arslan M. A., Kuchcinski K.** Instruction Selection and Scheduling for DSP Kernels on Custom Architectures, *17th Euromicro Conference on Digital System Design*, 2013, pp. 821—828.

7. **Lozano R. C.** Integrated Register Allocation and Instruction Scheduling with Constraint Programming, available at: https://www.sics.se/~rcas/publications/TRITA-ICT-ECS-AVH-14-13.pdf.

8. **Srinivasan K., Chatra K. S.** An ILP Formulation for System Level Througnput and Power Optimization in Multiprocessor SoC Architectures. *Proceedings of the 17th International Conference on VLSI Design (VLSID'04)*, 2004, pp. 255—260.

9. **Eriksson M., Kessler C.** Integrated Code Generation for Loops. *ACM Transactions on Embedded Computing Systems*, 2012, vol. 11S, no. 1, art. 19, pp. 1—25.

10. **V'jukova N. I., Galatenko V. A., Samborskij S. V.** Programmnaya konveierizatsiya tsiklov metodom planirovaniya po modulyu. *Programmirovanie*, 2007, no. 6, pp. 14—25 (in Russian).

11. **Stoutchinin A.** An Integer Linear Programming Model of Software Pipelining for the MIPS R8000 Processor, *Proceedings of the 4th International Conference on Parallel Computing Technologies* (*PaCT '97*), 1997, pp. 121—135.

12. **V'jukova N. I., Galatenko V. A., Samborskij S. V.** Programmnaya konveierizatsiya tsiklov dlya uskoritelya plavayushchei arifmetiki v sostave protsessora Komdiv128-RIO. *Programmnye produkty i sistemy*, 2013, no. 4, pp. 35—43 (in Russian).

13. **Fraser C. W., Henry R. R., Proebsting T. A.** BURG — Fast optimal instruction selection and tree parsing, *SIGPLAN Notices,* 1992, no. 4, pp. 68—76.

14. **Hjort Blindell G. S.** Survey on Instruction Selection. Cornell University Library, available at: http://arxiv.org/pdf/1306.4898v2.pdf.

15. **GNU** Linear Programming Kit, available at: http://www.gnu.org/software/glpk

16. **CBC** User Guide, available at: http://www.coin-or.org/Cbc/cbcuserguide.html

17. **NEOS** Server: State-of-the-Art Solvers for Numerical Optimization, available at: http://www.neos-server.org/neos

18. **Gurobi** Optimization, available at: http://www.gurobi.com

19. **FICO™** Xpress Optimization Suit, available at: http://www.fico.com/en/products/fico-xpress-optimization-suite

УДК 004

**В. А. Васенин**[1], д-р физ.-мат. наук, проф., e-mail: vasenin@msu.ru,
**А. А. Иткес**[1], канд. физ.-мат. наук, науч. сотр., e-mail: itkes@imec.msu.ru,
**К. А. Шапченко**[1], канд. физ.-мат. наук, ст. науч. сотр., e-mail: shapchenko@iisi.msu.ru,
**В. Ю. Бухонов**[2], аспирант, e-mail: bukhonovvyu@gmail.com,
[1] НИИ механики МГУ имени М. В. Ломоносова
[2] МГУ имени М. В. Ломоносова

# Реляционная модель логического разграничения доступа на основе цепочек отношений

*Предложена новая модель логического разграничения доступа для многопользовательских информационно-аналитических систем, правила доступа в которой формулируются с использованием заданных в целевой системе отношений между объектами — пользователями и ресурсами. Механизмы на основе такой модели могут эффективным образом использовать задействованные в системе реляционные СУБД, в которых хранится информация об отношениях и других атрибутах объектов системы. Изложено формальное описание модели, рассмотрены ее важные свойства. Представлены оценки, полученные в ходе первых тестовых испытаний программной реализации предложенной модели применительно к многопользовательской информационно-аналитической системе "Наука-МГУ" ("ИСТИНА"), выполняющей функции по управлению информацией о результатах научной деятельности в МГУ им. М. В. Ломоносова и использующей сложные правила разграничения доступа.*

***Ключевые слова:*** *разграничение доступа, правила разграничения доступа, модель логического разграничения доступа, отношение, реляционная СУБД, реляционная модель логического разграничения доступа*

## Введение

В современных многопользовательских информационных системах зачастую появляется необходимость ограничить доступ к определенным действиям с данными на основе задаваемых правил. В системах с большим числом пользователей и в условиях частого изменения атрибутов и связей между сущностями в системе сам процесс формулирования и внедрения таких правил становится сложной задачей. К подобным системам могут быть отнесены, например, те, которые обладают теми или иными характеристиками социальных сетей, в том числе отдельные классы многопользовательских систем управления контентом.

Для выполнения задач по разграничению доступа к ресурсам информационных систем традиционно используют механизмы, в основу которых положены дискреционные и ролевые модели логического разграничения доступа (ЛРД), реже — мандатные многоуровневые модели. Такие механизмы, несмотря на свою эффективность в ряде сценариев использования, как правило, ограничены в своих выразительных возможностях по формулировке необходимых для их реализации правил. Они могут потребовать большого количества административных действий, необходимых для настройки в случае определенных изменений в целевой системе. При администрировании таких механизмов обычно не используют специализированные языковые средства, а настройка правил осуществляется последовательностью ограниченных административных действий.

Расширенные возможности по заданию правил доступа на основе атрибутов предоставляются в рамках так называемых атрибутных моделей ЛРД. Такие модели, как правило, требуют уже при своей настройке использования специализированных языковых средств. Примерами таких средств являются универсальные языки XACML и SAML [1, 2]. В силу универсального характера подобных языков механизмы разграничения доступа на их основе обычно не учитывают должным образом особенности целевой системы и могут обладать низкой производительностью.

Тем не менее условия предоставления доступа используют те или иные свойства или атрибуты сущностей в целевой системе, включая пользователей, ресурсы, вспомогательные объекты. Зачастую атрибуты таких сущностей связаны с отношениями между ними, например, иерархиями пользователей

или ресурсов, актуальным в социальных сетях симметричным отношением "дружбы" или другими отношениями.

В настоящей статье рассмотрена возможность использования таких отношений как основы для языковых средств задания правил разграничения доступа. С этих позиций статья является продолжением публикации [3], в которой авторы представили результаты анализа известных моделей ЛРД для социальных сетей на предмет возможности их использования в одном классе многопользовательских систем управления контентом и сформировали требования к более подходящей модели ЛРД на примере одной из целевых систем. В частности, такие требования указали на необходимость более полного учета отношений в системе по сравнению с известными моделями.

Отметим, что зачастую для хранения свойств и связей между объектами в многопользовательской информационной системе применяют реляционные СУБД. В таких условиях модели ЛРД, использующие явным образом понятие отношения, которое легко транслируется в термины реляционных баз данных, обладают рядом преимуществ. Во-первых, языковые средства модели ЛРД могут быть расширены для использования других данных в реляционной СУБД в рамках единого языка с широкими выразительными возможностями. При этом можно ожидать, что такой язык будет оперировать схожим набором понятий вместе с другими программными компонентами целевой системы, которые связаны с той же базой данных. Во-вторых, механизм ЛРД работает над теми же данными, что и другие компоненты системы, и может позволить получить доступ к дополнительным оптимизациям выполнения запросов к СУБД. Заметим, что использование информации в реляционных СУБД в качестве основы для правил разграничения доступа поддерживается, например, многими типовыми наборами библиотек для веб-приложений, в которых такая информация может применяться явным образом либо в рамках объектно-реляционного отображения.

Также в работе рассмотрены предпосылки к решению задачи разработки целевой модели ЛРД, изложены основные положения предлагаемой модели. Проанализированы особенности ее программной реализации и результаты первых проведенных на ее основе экспериментов. Объектом приложения в экспериментах является многопользовательская информационно-аналитическая система (ИАС) "Наука-МГУ" ("ИСТИНА") [4]. Эта система позволяет выполнять действия по управлению информацией о результатах научной деятельности, причем такая информация вносится в систему и корректируется преимущественно ее пользователями.

Примем следующие соглашения о терминологии в рамках настоящей работы. Ресурсами системы будем называть все ее объекты, не являющиеся пользователями. Усеченным социальным графом будем называть нагруженный граф, вершинами которого являются пользователи системы, а ребрами — пары пользователей, связанных каким-либо отношением. Каждому ребру при этом соответствует набор отношений, связывающих соответствующих пользователей. Полным социальным графом будем называть нагруженный граф, вершинами которого являются объекты (пользователи и ресурсы) системы, а ребрами — пары объектов, связанных каким-либо отношением. При этом каждому ребру соответствует набор отношений, связывающих соответствующие объекты. В силу того, что в целевой системе "Наука-МГУ" ("ИСТИНА") важен учет отношений между ресурсами системы, а не только между ее пользователями, в дальнейшем под социальным графом будем подразумевать полный социальный граф, если явно не указано обратное.

## 1. Предпосылки для разработки реляционной модели ЛРД

К подсистеме разграничения доступа к объектам ИАС "Наука-МГУ" ("ИСТИНА") предъявляются требования, в числе которых перечисленные далее.

- Модель должна учитывать отношения не только между пользователями, но и между ресурсами системы.

Данное требование следует из того факта, что некоторые объекты ИАС "Наука-МГУ" ("ИСТИНА") не связаны ни с каким пользователем непосредственно. Эта особенность системы в значительной степени обусловлена тем обстоятельством, что многие из сотрудников, информация о которых хранится в системе, в силу различных причин не зарегистрированы в ней как пользователи. Как следствие, зарегистрированные в системе публикации могут быть связаны отношением авторства не с объектом "пользователь", а со вспомогательным объектом "сотрудник". "Сотрудник", в свою очередь, может быть связан с "пользователем" отношением "является". Таким образом, учет отношений между ресурсами системы (в рассмотренном примере — между публикацией и вспомогательным объектом "сотрудник") необходим для того, чтобы в рассматриваемом моделью подмножестве социального графа существовал путь между пользователем и публикациями, автором которых он является.

- Модель должна допускать возможность разграничения доступа пользователей к объектам, учитывая не только топологические свойства социального графа, но и дополнительные свойства объектов и отношений между ними.

Рассмотрим пример. Пусть пользователь $u_1$, назначенный ответственным за сопровождение данных в системе по некоторому подразделению $s_1$, имеет право изменять атрибуты публикаций, выполненных в этом подразделении. Пусть сотрудник $a$, являющийся автором публикации $p$, работает в подразделении $s_1$, однако на момент подготовки публикации $p$ он работал в подразделении $s_2$, ответственным по которому является пользователь $u_2$. В этом случае, поскольку

публикация подготовлена в подразделении $s_2$, пользователь $u_2$ должен иметь право изменять ее атрибуты, а пользователь $u_1$ такого права иметь не должен. Вместе с тем пользователи $u_1$ и $u_2$ оба соединены с публикацией $p$ цепочкой отношений "ответственный по подразделению" — "в подразделении работает сотрудник" — "сотрудник является автором статьи". Таким образом, модель, предполагающая разграничение доступа к объектам системы на основании только топологических свойств социального графа, не может дать пользователям $u_1$ и $u_2$ разные права доступа к публикации $p$. В связи с этим обстоятельством необходимо использовать модель, предполагающую учет дополнительных свойств вершин и ребер социального графа. В представленном примере ребрам, связывающим сотрудника с подразделением отношением "работает в", можно поставить в соответствие дату поступления на работу и увольнения, а ребрам, связывающим сотрудника с публикацией отношением авторства, — дату сдачи публикации в печать. При этом публикация должна считаться выполненной в определенном подразделении в том и только том случае, если один из ее авторов работал в этом подразделением на момент сдачи публикации в печать.

Как отмечено в работе [3], многие из моделей ЛРД в социальных сетях предназначены для управления доступом к объектам подконтрольной системы на основании свойств усеченного социального графа. При этом каждому ресурсу в системе ставится в соответствие пользователь-владелец, а права доступа других пользователей к этому объекту определяют на основании последовательностей отношений, связывающих этого пользователя с владельцем объекта. По причинам, отмеченным ранее, такой подход не может применяться для управления доступом к объектам ИАС "Наука-МГУ" ("ИСТИНА").

Единственной из моделей ЛРД, которые проанализированы в работе [3], в полной мере поддерживающей управление доступом на основании свойств полного социального графа, является одна из моделей, разработанных Ченгом, Парком и Сандху, а именно — модель *User-to-Resource Relationship-based Access Control* (URRAC) [5]. Однако следует заметить, что данная модель ЛРД предполагает разграничение доступа пользователей к объектам системы на основании топологических свойств социального графа, но без учета дополнительных свойств объектов и отношений между ними.

В связи с изложенными выше соображениями необходимо отметить, что модели URRAC [5] и $\text{UURAC}_\text{A}$ [6] имеют также и общие недостатки, которые сильно затрудняют их применение для управления доступом к объектам ИАС "Наука-МГУ" ("ИСТИНА"). Один из них заключается в том, что для задания путей в социальном графе, определяющих пользователей, которые имеют доступ к заданному объекту, в этих моделях применяется сложный язык, по синтаксису во многом похожий на регулярные выражения. Однако этот язык также имеет операции, не предусмотренные в регулярных выражениях. Задание корректных ограничений на доступ к "своим" объектам может оказаться сложной задачей для пользователей, не знакомых с математической логикой, теорией графов и не имеющих навыков программирования. Более того, специальным образом сформированные правила, проверка которых потребует значительных вычислительных затрат, могут привести к увеличению времени отклика целевой системы на запрос о доступе и к отказам в обслуживании.

## 2. Определение реляционной модели ЛРД на основе цепочек отношений

Предлагаемая реляционная модель ЛРД учитывает топологические свойства полного социального графа, включающего как пользователей системы, так и ее ресурсы и все отношения между ними. При этом в модели жестко фиксируется множество цепочек отношений, которые могут связывать субъекты и объекты доступа. Благодаря этому свойству реляционная модель ЛРД, в отличие, например, от модели URRAC, не требует от пользователей системы написания сложных правил для проверки существования пути в социальном графе. Кроме того, такая модель позволяет ограничить использование вычислительных ресурсов для получения решения о запрашиваемом доступе, а также избежать отказов в обслуживании системы через исчерпание таких ресурсов.

Некоторые из отношений в системе можно отметить как транзитивные. Примером такого отношения является отношение вложенности подразделений друг в друга. В частности, транзитивность отношений может упростить запись правил предоставления пользователю, ответственному за сопровождение системы по определенному подразделению, возможность изменять атрибуты всех сотрудников этого подразделения и его дочерних подразделений всех уровней вложенности. Данное правило может быть сформулировано при условии, что отношение "одно подразделение является дочерним для другого" отмечено как транзитивное.

Объекты и их отношения могут иметь дополнительные атрибуты, влияющие на права доступа пользователей к объектам. Например, для отношения "сотрудник работает в подразделении" множеством атрибутов может быть множество пар (дата поступления, дата увольнения), которые задают период работы сотрудника в данном подразделении. Одновременно с этим отношение "сотрудник является автором публикации" может иметь дополнительный атрибут — дату подачи публикации в печать. При этом цепочка отношений "в подразделении работает сотрудник", "сотрудник является автором статьи" порождает отношение "публикация написана в данном подразделении" только в том случае, если дата завершения работы над публикацией входит в период работы автора этой публикации в заданном подразделении.

### 2.1. Базовые множества, отношения и отображения в модели

Будем считать, что заданы следующие конечные множества, отношения и отображения:

*Objects* — множество объектов;

*Users* $\subset$ *Objects* — множество пользователей;

*Relations* — множество отношений;

*PrimitiveRelations* $\subset$ *Relations* — множество примитивных отношений;

*InducedRelations* = *Relations* $\setminus$ *PrimitiveRelations* — множество порожденных отношений;

*TransitiveRelations* $\subset$ *PrimitiveRelations* — множество транзитивных отношений;

*Classes* — множество классов;

*Types* — множество типов;

*Roles* — множество ролей;

*Privileges* — множество привилегий;

*Actions* — множество операций над объектами;

*Chains* $\subset$ *Relations** — множество допустимых цепочек отношений, где *Relations** — множество всех конечных последовательностей элементов из множества *Relations*;

*ObjectRelations* $\subset$ *Objects* $\times$ *PrimitiveRelations* $\times$ *Objects* — отношение, определяющее, какие из объектов системы связаны тем или иным примитивным отношением;

$\leqslant_C \subset$ *Classes* $\times$ *Classes* — отношение, определяющее иерархию классов, при этом если $c_1 \leqslant_C c_2$, то будем считать, что $c_1$ является потомком класса $c_2$, а $c_2$ — предком $c_1$;

*UserRoles* $\subset$ *Users* $\times$ *Roles* — отношение, определяющее, какие пользователи обладают той или иной ролью;

*RolePrivileges* $\subset$ *Roles* $\times$ *Privileges* — отношение, связывающее роли системы с привилегиями;

*RelationTypeAccessGranted* $\subset$ *Relations* $\times$ *Types* $\times$ *Actions* — отношение, определяющее, какие действия с объектом определенного типа может совершать пользователь, связанный с ним определенным отношением;

*RelationTypeAccessDenied* $\subset$ *Relations* $\times$ *Types* $\times$ *Actions* — отношение, определяющее, какие действия пользователь не может совершать с объектом в зависимости от отношения этого пользователя к этому объекту;

*RoleTypeAccessGranted* $\subset$ *Roles* $\times$ *Types* $\times$ *Actions* — отношение, определяющее, какие действия с объектами определенного типа может совершать пользователь, обладающий определенной ролью;

*RoleTypeAccessDenied* $\subset$ *Roles* $\times$ *Types* $\times$ *Actions* — отношение, определяющее, какие действия с объектами определенного типа пользователь, обладающий определенной ролью, совершать не может;

*ClassActions* $\subset$ *Classes* $\times$ *Actions* — отношение, определяющее, какие виды доступа допустимы для объектов каждого класса;

*ObjectClass*: *Objects* $\rightarrow$ *Classes* — отображение, определяющее класс каждого объекта;

*RelationLeftClass*: *Relations* $\rightarrow$ *Classes* — отображение, определяющее класс объектов, которые могут быть связаны с другими объектами определенным отношением;

*RelationRightClass*: *Relations* $\rightarrow$ *Classes* — отображение, определяющее класс объектов, которые могут быть связаны с другими объектами определенным отношением;

*ChainInduces*: *Chains* $\rightarrow$ *InducedRelations* — отображение, определяющее отношение, порожденное каждой из цепочек.

Отображения *RelationLeftClass* и *RelationRightClass* используются следующим образом: если $(o_1,r,o_2) \in$ $\in$ *ObjectRelations*, то $o_1$ является объектом класса *RelationLeftClass(r)* или одного из его потомков, а $o_2$ — объектом класса *RelationRightClass(r)* или одного из его потомков.

Пусть при этом выполнены следующие условия:

$\exists$ *user* $\in$ *Classes*: $\forall$ *u* $\in$ *Users*: *ObjectClass*(*u*) $\leqslant_C$ *user*;

$\forall$ $(r_1, \ldots, r_n) \in$ *Chains*: $\forall$ $i \in \{1, \ldots, n-1\}$: *RelationLeftClass*($r_{i+1}$) $\leqslant_C$ *RelationRightClass*($r_i$) $\vee$ $\vee$ *RelationLeftClass*($r_i$) $\leqslant_C$ *RelationRightClass*($r_{i+1}$).

Пусть также для каждого примитивного отношения *r* задано множество дополнительных атрибутов *RelationAttributes*(*r*), а для каждого класса *c* — множество дополнительных атрибутов *ClassAttributes*(*c*). При этом, если $c_1 \leqslant_C c_2$, то *ClassAttributes*($c_2$) $\subset$ *ClassAttributes*($c_1$). Пусть каждое из множеств *RelationAttributes*(*r*) и *ClassAttributes*(*c*) является непустым и содержит по крайней мере фиктивный "пустой" атрибут *empty_attr*. Пусть для каждого объекта *o* задано значение *ObjectAttribute*(*o*) $\in$ $\in$ *ClassAttributes*(*ObjectClass*(*o*)). Пусть для каждой пары объектов, связанных примитивным отношением ($o_1$, *r*, $o_2$), задано значение *ObjectRelationAttribute*($o_1$, *r*, $o_2$) $\in$ *RelationAttributes*(*r*). Пусть с каждой цепочкой отношений ($r_1$, ..., $r_n$) ассоциирован предикат *ChainCondition*[$r_1$, ..., $r_n$]: *RelationAttributes*($r_1$) $\times$ $\times$ *RelationAttributes*($r_2$) $\times$ ... $\times$ *RelationAttributes*($r_n$) $\times$ $\times$ *ClassAttributes*($c_0$) $\times$ ... $\times$ *ClassAttributes*($c_n$) $\rightarrow$ $\rightarrow$ {true, false}, где $c_0$ = *RelationLeftClass*($r_1$), $c_k$ = = min(*RelationLeftClass*($r_k$), *RelationRightClass*($r_{k+1}$)) для $k = 1, \ldots, n-1$, $c_n$ = *RelationRightClass*($r_n$).

### 2.2. Расширенные множества цепочек отношений и связанных пар объектов

Одной из особенностей реляционной модели ЛРД является то обстоятельство, что, кроме явно указанных в модели цепочек отношений, в системе могут использоваться также неявные цепочки, которые получают из других заменой одного из входящих в цепочку отношений на подцепочку, порождающую это отношение. Расширенное множество цепочек *Chains**, содержащее как явные цепочки, так и все полученные из них указанным способом, определяется следующим образом:

$$Chains_0 = Chains;$$

$$Chains_{k+1} = Chains_k \cup \{(r_1, \ldots, r_{i-1}, r_1', \ldots, r_m', r_{i+1}, \ldots, r_n):$$

$$(r_1, \ldots, r_n) \in Chains_k,\ (r_1', \ldots, r_m') \in Chains_k,$$

$$ChainInduces(r_1', \ldots, r_m') = r_i\};$$

$$Chains^* = \cup_{k=0,1,\ldots} Chains_k.$$

При этом $ChainInduces(r_1, ..., r_{i-1}, r'_1, ..., r'_m, r_{i+1}, ..., r_n) = ChainInduces(r_1, ..., r_n)$ при $r_i = ChainInduces(r_1, ..., r_{i-1}, r'_1, ..., r'_m, r_{i+1}, ..., r_n)$.

Заметим, что на расширенном множестве цепочек функция *ChainInduces* может стать многозначной. В дальнейшем запись $ChainInduces(r_1, ..., r_n) = r$ означает, что $r$ — одно из значений, принимаемых $ChainInduces(r_1, ..., r_n)$.

Расширенным множеством пар объектов, связанных отношениями, назовем такое множество $ObjectRelations^* \subset Objects \times Relations \times Objects$, для которого выполняются следующие условия:

1) $ObjectRelations_1 = ObjectRelations \cup \{(o_{first}, r, o_{last}): r \in TransitiveRelations, \exists n, o_0 = o_{first}, o_1, ..., o_{n-1}, o_n = o_{last}: \forall i \in \{1, ..., n\}: (o_{i-1}, r, o_i) \in ObjectRelations\}$;

2) $ObjectRelations^* = ObjectRelations \cup \{(o_s, r, o_f) \in Objects \times InducedRelations \times Objects: \exists (r_1, ..., r_n) \in Chains^*: \exists o_0, ..., o_n \in Objects: o_s = o_0, o_f = o_n, \forall i \in \{1, ..., n\}: (o_{i-1}, r, o_i) \in ObjectRelations \wedge ChainCondition [r_1, ..., r_n] (ObjectRelationAttribute(o_0, r_1, o_1), ..., ObjectRelationAttribute(o_{n-1}, r_n, o_n)\}$.

Условие 1 означает, что некоторые отношения в системе (перечисленные явно в множестве *TransitiveRelations*) являются транзитивными. Этот факт означает, что если объект $a$ связан с объектом $b$ отношением $r$, а объект $b$ связан отношением $r$ с объектом $c$, то объект $a$ считается связанным с объектом $c$ отношением $r$, даже если это не указано в явном виде.

Условие 2 означает следующее: два объекта системы $o_s$ и $o_f$ связаны порожденным отношением $r$, если существует такая цепочка отношений, порождающая отношение $r$, а также такая последовательность объектов, начинающаяся с $o_s$ и заканчивающаяся $o_f$, что каждый последующий объект в ней связан с предыдущим соответствующим отношением из цепочки, порождающей $r$. Например, если цепочка отношений "в подразделении работает сотрудник", "сотрудник является автором публикации" порождает отношение "публикация выполнена в подразделении", то публикация $p$ связывается отношением "публикация выполнена в подразделении" с подразделением $s$, если существует сотрудник $a$, работающий в подразделении $s$ и являющийся автором публикации $p$.

### 2.3. Условия предоставления доступа

При использовании реляционной модели ЛРД предоставление пользователю $u \in Users$ права доступа $a \in Actions$ к объекту $o \in Objects$ осуществляется в том и только том случае, если выполнено одно из следующих условий:

$\exists r \in Relations \exists t \in Types: (u, r, o) \in ObjectRelations^* \wedge \wedge (r, t, a) \in RelationTypeAccessGranted \wedge (o, t) \in ObjectTypes$;

$\exists r \in Roles \exists t \in Types: (u, r) \in UserRoles \wedge (r, t, a) \in \in RelationTypeAccessGranted \wedge (o, t) \in ObjectTypes;$

и при этом не выполнено ни одно из следующих условий:

$\exists r \in Relations \exists t \in Types: (u, r, o) \in ObjectRelations^* \wedge \wedge (r, t, a) \in RelationTypeAccessDenied \wedge (o, t) \in ObjectTypes$;

$\exists r \in Roles \exists t \in Types: (u, r) \in UserRoles \wedge (r, t, a) \in \in RelationTypeAccessDenied \wedge (o, t) \in ObjectTypes.$

Таким образом, представленные выше множества, отношения, отображения и условия предоставления доступа определяют модель ЛРД.

Заметим, что множества *Roles* и *Privileges*, а также отношения *UserRoles*, *RolePrivileges*, *RoleTypeAccessGranted* и *RoleTypeAccessDenied* необходимы для использования в системе традиционной ролевой модели $RBAC_0$ одновременно с моделью, использующей отношения между объектами.

### 2.4. Свойство корректности реляционной модели ЛРД

Необходимо отметить, что при использовании представленного выше определения расширенного множества цепочек отношений данное множество при некоторых условиях может оказаться потенциально бесконечным. В этом случае все цепочки объектов, связанные цепочками отношений, все равно будут иметь длину, не превосходящую общего количества объектов в системе. Однако проверка существования цепочек настолько большой длины может потребовать недопустимо больших затрат процессорного времени. По этой причине предлагаемая реляционная модель ЛРД может применяться на практике только при наличии некоторых ограничений на число элементов расширенного множества цепочек и максимальную длину этих цепочек. Например, максимальное число и максимальная длина цепочек могут быть вычислены на основании тех значений атрибутов модели ЛРД, которые могут быть изменены только администратором системы при обновлении политики безопасности. И они не зависят от данных, введенных в систему ее пользователями.

Исходя из указанных соображений, назовем реляционную модель ЛРД корректной, если расширенное множество цепочек *Chains** конечно. При этом верна следующая теорема.

**Теорема о корректности реляционной модели ЛРД.** Пусть для любой цепочки $(r_1, ..., r_n) \in Chains$ и любого $i \in \{1, ..., n\}$ выполнено условие $ChainInduces(r_1, ..., r_n) \neq \neq r_i$. Тогда множество *Chains** конечно.

**Доказательство.** Назовем примитивные отношения отношениями уровня 0. Назовем порожденное отношение $r$ отношением первого уровня, если существует такая цепочка $c = (r_1, ..., r_l) \in Chains$, что $ChainInduces(c) = r$, и все отношения $r_i$ являются примитивными для $i \in \{1, ..., l\}$. Если для некоторого $n$ уже определены отношения уровня $n$, то назовем отношение $r$ отношением уровня $n + 1$, если существует такая цепочка $c = (r_1, ..., r_l) \in Chains$, что $ChainInduces(c) = r$, и все отношения $r_i$ имеют уровень не более $n$.

Если в системе ни одна из цепочек не содержит отношение, которое порождает, то при этом каждое из отношений в системе получит уровень, не превосходящий общего числа отношений в системе. Отмеченный факт следует из того, что для каждого $k$ выполнено следующее свойство: если существует

хотя бы одно отношение, не относящееся к уровню меньшему, чем $k$, то существует хотя бы одно отношение, которому будет присвоен уровень $k$. Пусть для некоторого $k$ это не так, т. е. все отношения $r_1$, ..., $r_m$, которым не присвоен уровень меньший, чем $k$, не относятся к уровню $k$. Это означает, что для любого $i = 1, ..., m$ и любой цепочки $c$, порождающей отношение $r_i$, существует такое отношение $r_{i,j}$, входящее в цепочку $c$, что отношению $r_{i,j}$ не присвоен уровень, не превосходящий $k - 1$.

Пусть $r_1$ — это одно из отношений, которым не присвоен уровень меньший, чем $k$. Пусть $r_2$ — это отношение, входящее в одну из цепочек, порождающих $r_1$, и также не относящееся к уровню меньшему, чем $k$. Пусть $r_{l+1}$ — это отношение, входящее в одну из цепочек, порождающих $r_l$, и также не относящееся к уровню меньшему, чем $k$. Последовательность отношений $r_l$ при этом будет бесконечной, так как по предположению для каждого $r_l$ возможно выбрать $r_{l+1}$. При этом, однако, общее число отношений, которым не присвоен уровень, не превосходящий $k - 1$, конечно, поэтому среди элементов последовательности $r_l$ есть одинаковые $r' = r_i = r_j$ для $i < j$.

В этом случае существует такая цепочка отношений $c$, содержащая $r'$, что $ChainInduces(c) = r'$. В самом деле, пусть $c_i$ — это цепочка, порождающая отношение $r_i$ и содержащая отношение $r_{i+1}$. Пусть цепочка $c_{i+1}$ получается из $c_i$ заменой звена $r_{i+1}$ на подцепочку, порождающую отношение $r_{i+1}$ и содержащую отношение $r_{i+2}$. Пусть цепочка $c_{i+2}$ получается из $c_{i+1}$ заменой звена $r_{i+2}$ на подцепочку, порождающую отношение $r_{i+2}$ и содержащую отношение $r_{i+3}$. Построенная таким образом цепочка $c_{j-1}$ будет содержать отношение $r_j = r'$ и порождать отношение $r_i = r'$. Как следствие, предположение о том, что для некоторого $k$ существуют отношения, которым не присвоен уровень, не превосходящий $k - 1$, ни одному из которых нельзя присвоить уровень $k$, привело к противоречию.

Таким образом, для каждого $k$ либо все отношения в системе имеют уровень меньший, чем $k$, либо хотя бы одно отношение в системе имеет уровень $k$. Поэтому максимальный уровень отношения в системе не превосходит общего числа отношений.

Заметим, что если цепочка $c_0$ имеет уровень $m$, то существуют такие цепочки отношений $c_1 = (r_1, ..., r_p)$ и $c'_1 = (r'_1, ..., r'_q)$, что $c = (r'_1, ..., r'_{i-1}, r_1, ..., r_p, r'_{i+1}, ..., r'_q)$, где $r'_i = ChainInduces(c_1)$ и при этом уровень цепочки $c'_1$ равен 0, а уровень цепочки $c_1$ равен $m - 1$.

Пусть $c_0$ — это цепочка отношений уровня $m > \|Relations\|$. Пусть $c_1$ — это такая цепочка отношений уровня $m - 1$, что $\exists\ c'_1 \in Chains$: $c_1 = (r_1, ..., r_p)$, $c'_1 = (r'_1, ..., r'_q)$, $c = (r'_1, ..., r'_{i-1}, r_1, ..., r_p, r'_{i+1}, ..., r'_q)$, $r'_i = ChainInduces(c_1)$. Аналогично, пусть для цепочки отношений $c_k$ цепочка $c_{k+1}$ — это такая цепочка отношений уровня на 1 меньше, чем уровень цепочки $c_k$, что $\exists\ c'_{k+1} \in Chains$: $c_{k+1} = (r_1, ..., r_p)$, $c'_{k+1} = (r'_1, '..., r'_q)$, $c = (r'_1, ..., r'_{i-1}, r_1, ..., r_p, r'_{i+1}, ..., r'_q)$, $r'_i = ChainInduces(c_{k+1})$. Таким образом будут построены цепочки отношений $c_k$ для $0 \leqslant k \leqslant m - 1$, т. е. всего будет построено $m$ цепочек. Заметим, что все построенные цепочки должны порождать разные отношения. Одновременно с этим число цепочек $c_k$ больше общего числа отношений в системе, поэтому все порождаемые ими отношения не могут быть различны. Таким образом, предположение о существовании цепочки уровня большего, чем общее число отношений в системе, привело к противоречию.

Множество всех цепочек, входящих в $Chains^*$ и порождающих отношения первого уровня, совпадает с исходным множеством цепочек $Chains$ и потому является конечным. Число цепочек, порождающих отношения второго уровня, не превосходит $\Sigma_{c \in Chains} length(c) \|Chains\|$, где $length(c)$ — это длина цепочки $c$, поскольку каждая из цепочек, порождающих отношение второго уровня, получается из некоторой цепочки, порождающей отношение первого уровня, заменой одного из ее звеньев на другую цепочку, порождающую отношение первого уровня. Аналогично, $\|Chains_{k+1}\| \leqslant \Sigma_{c \in Chains_k} length(c) \|Chains_k\|$, и при этом $Chains^* = Chains_{\|Relations\|}$, что и доказывает утверждение теоремы.

## 3. Программная реализация реляционной модели ЛРД

В рамках программной реализации рассматриваемой модели предполагается, что данные целевой системы, на основе которых принимается решение о доступе, хранятся в реляционной СУБД. При этом в базе данных хранится информация обо всех парах объектов, связанных примитивными отношениями, а информация о парах объектов, связанных порожденными отношениями, должна вычисляться механизмами разграничения доступа по мере необходимости.

Реляционная модель ЛРД предназначена для использования в системах, топологические свойства социального графа которых заранее неизвестны, и поэтому программы, реализующие данную модель, должны выполнять операции обхода графа произвольного вида. Эти операции имеют достаточно высокую вычислительную сложность. Отметим, что в системе может одновременно работать большое число пользователей, каждый из которых запрашивает разрешение на доступ к разным объектам. Таким образом, механизм разграничения доступа должен обеспечивать одновременное выполнение достаточно большого числа операций (например, десятки и сотни одновременных операций в пределах короткого промежутка времени) по проверке прав доступа разных пользователей к разным объектам.

Заметим, что в числе множеств, определяемых в предлагаемой модели ЛРД, можно выделить группу "статических" множеств, которые могут быть изменены только администратором системы — множества *Relations*, *PrimitiveRelations*, *InducedRelations*, *TransitiveRelations*, *Classes*, *Types*, *Roles*, *Privileges*, *Actions*, *Chains*. Также к "статическим" могут быть отнесены отношения *UserRoles*, *RolePrivileges*, *RelationTypeAccessGranted*,

*RelationTypeAccessDenied*, *RoleTypeAccessGranted*, *RoleTypeAccessDenied* и иерархия классов.

Выделение статических элементов модели имеет важное значение при оптимизации производительности программных механизмов, реализующих реляционную модель ЛРД. Как правило, возможность доступа определенного пользователя к определенному объекту вычисляется сложной формулой логики предикатов, включающей многие из базовых множеств модели. Следует заметить, что вычисления, затрагивающие только статические элементы модели, можно проводить не каждый раз при запросе пользователя на доступ к объекту, а однократно при обновлении правил доступа. Такой подход позволяет существенно повысить производительность программных механизмов разграничения доступа.

В связи с этим обстоятельством вычисления, необходимые для определения прав доступа пользователя к объекту, целесообразно выполнять в два этапа. На первом этапе проводятся вычисления, затрагивающие только статические множества модели. Эти вычисления могут требовать значительных временных затрат и должны выполняться только при обновлении политики безопасности. На этом этапе, в частности, на основании исходного множества цепочек отношений *Chains* и функции порождения отношений *ChainInduces* рассчитываются расширенное множество цепочек *Chains** и значения функции *ChainInduces* на этом множестве. На основании доказанной теоремы о корректности для построения множества *Chains** достаточно построить множества $Chains_n$ для $n \leqslant \|Relations\|$. На втором этапе для конкретного пользователя *u* и объекта *o* при уже известном расширенном множестве цепочек проверяется, какие из них связывают пользователя *u* с объектом *o*. Этот этап выполняется каждый раз при запросе пользователя на доступ к объекту.

Заметим, что для функции проверки доступа пользователя к объекту имеет смысл возвращать результат в виде списка всех операций, которые имеет право совершать данный пользователь над данным объектом, а не значения "истина" или "ложь" для заданной конкретной операции. Возможно, например, что при загрузке веб-страницы с информацией об объекте пользователь должен видеть все действия, которые имеет право совершать с этим объектом. В этом случае целесообразно при загрузке страницы однократно проанализировать все цепочки отношений, связывающих пользователя с целевым объектом, для получения списка всех возможных действий. Отдельная обработка цепочек отношений для проверки каждой операции, допустимой для данного объекта, потребует значительно больших временных затрат.

Для реализации рассмотренного метода разработана программа-транслятор из описания используемой модели ЛРД на специальном языке в программный код функции проверки прав доступа для встраивания в веб-приложение. Эта функция принимает три аргумента — идентификатор пользователя, класс целевого объекта и идентификатор целевого объекта. Возвращаемым значением функции является список операций, которые имеет право выполнять данный пользователь с данным объектом.

Алгоритм работы программы-транслятора выглядит следующим образом. В первую очередь программа вычисляет расширенное множество цепочек отношений *Chains** и функцию порождения *ChainInduces* на этом множестве. Для этого программа инициализирует расширенное множество цепочек исходным множеством, после чего каждая из цепочек, входящих в расширенное множество, обрабатывается следующим образом. Для каждой из цепочек, входящих в расширенное множество, строятся все цепочки, которые получаются заменой каждого из входящих в нее отношений на подцепочку, также входящую в расширенное множество и содержащую это отношение. Полученные таким образом цепочки добавляются в расширенное множество цепочек отношений. После этого все цепочки, входящие в новое расширенное множество, снова обрабатываются указанным способом до тех пор, пока это приводит к добавлению в расширенное множество цепочек хотя бы одного нового элемента. Согласно теореме о корректности реляционной модели ЛРД общее число повторений данной процедуры при этом не будет больше, чем общее число отношений в системе. Выполнение каждого из шагов алгоритма при этом теоретически требует не более $N^2L$ операций, где $N$ — число цепочек в расширенном множестве на данном шаге, а $L$ — максимальная длина цепочек. Теоретически, на каждом шаге алгоритма размер расширенного множества цепочек также может возрасти не более чем в $N^2L$ раз. Однако на примерах из практики число добавляемых цепочек и трудоемкость алгоритма оказываются намного ниже. Фактическое время работы транслятора при этом не превосходит нескольких секунд.

После того как расширенное множество цепочек отношений построено, для каждого из классов объектов системы генерируется отдельный блок функции проверки доступа, который возвращает список прав доступа пользователя к объекту этого класса. При порождении этого блока кода для каждой из цепочек отношений, которая может связывать пользователя с объектом этого класса, формируется SQL-запрос к реляционной СУБД. Результат его вычисления возвращает все цепочки объектов, связывающие пользователя с целевым объектом этой последовательностью отношений. Порождаемая функция должна направить этот запрос к СУБД, после чего, если ответ на него не пуст, она добавляет к списку разрешенных виды доступа, соответствующие отношению, порождаемому этой цепочкой. Схема алгоритма выглядит следующим образом.

1. Инициализировать списки разрешенных и запрещенных операций пустыми множествами.

2. Для каждой цепочки отношений *c* из множества *Chains**, связывающей пользователя и объект доступа, выполнить запрос к базе данных, соответствующий *c*, если при этом запрос вернул непустой результат, то:

— добавить разрешенные операции, соответствующие отношению *r*, порождаемому *c*, в список разрешенных;

— добавить запрещенные операции, соответствующие отношению *r*, в список запрещенных.

3. Вернуть в качестве результата список разрешенных операций за вычетом списка запрещенных операций.

При этом конкретная последовательность запросов выбирается в зависимости от класса объекта доступа.

Заметим, что при использовании программы-транслятора дополнительные условия *ChainCondition*, накладываемые на атрибуты отношений, порождающих цепочку, хранятся в исходном файле непосредственно в виде SQL-выражений. Эти выражения присоединяются к выражению, соответствующему проверке существования пути в социальном графе, с помощью оператора "логическое И". По этой причине при использовании транслятора все используемые в модели условия *ChainCondition* должны иметь достаточно простой вид. Для указания типов объектов также используются условия, которые при необходимости присоединяются к SQL-запросу с помощью оператора "логическое И".

Для примера, следующее условие "article.date >= >= works_in.begin_date AND (works_in.end_date IS NULL OR works_in.end_date >= article.date)" означает, что дата выхода публикации должна приходиться на период работы ее автора в определенном подразделении. Таким образом, публикация считается выполненной в определенном подразделении при условии, что хотя бы один из ее авторов работает или работал в этом подразделении и при этом выполнено условие данного вида. Это условие используется при проверке доступа пользователя, являющегося ответственным за управление данными системы по подразделению, к публикациям, выполненным в этом подразделении. Например, пользователю предоставляется доступ к публикации только в том случае, если один из ее авторов работал в подконтрольном пользователю подразделении на момент выхода публикации. Запрос к реляционной СУБД, учитывающий и цепочку отношений, и дополнительное условие, может выглядеть в таком случае следующим образом: "SELECT * FROM user_manages_subdivision, works_in, is_author_of, article WHERE user_manages_subdivision.user_id = uid AND user_manages_subdivision.div_id = works_in.div_id AND works_in.author_id = is_author_of.author_id AND is_auhor_of.pub_id = pid AND article.id = pid AND article.date > = works_in.begin_date AND (works_in.end_date IS NULL OR works_in.end_date >= article.date)".

Таким образом, функция проверки доступа посылает фиксированную последовательность SQL-запросов системе управления базой данных и возвращает значение, зависящее от того, на какие из этих запросов были возвращены непустые ответы. Все вычисления с данными при этом выполняет СУБД, которая также входит в доверенную вычислительную базу. По этой причине конкретный алгоритм, используемый при проверке существования цепочек объектов, соответствующих данной цепочке отношений, неизвестен и теоретическая оценка его производительности пока не представляется возможной. Вместе с тем в силу ряда причин можно считать, что быстродействие системы окажется при этом выше, чем при выполнении сложных вычислений над данными непосредственно функцией проверки прав доступа. Одна из этих причин заключается в том, что СУБД имеет внутренние структуры данных (индексы), недоступные взаимодействующим с ней приложениям, обращение к этим структурам позволяет ускорить ряд операций с данными. Вторая причина заключается в том, что передача значительных объемов данных сценарию от СУБД также занимает время, а передача только конечного результата обработки данных при этом происходит быстрее. Кроме того, программа СУБД написана на компилируемом языке программирования. Поэтому она работает быстрее программ, выполняющих аналогичные функции, написанных на интерпретируемых языках, в том числе веб-приложений, одним из которых де-факто является ИАС "Наука-МГУ" ("ИСТИНА"). В силу перечисленных причин, при проверке доступа конкретного пользователя к конкретному объекту большая часть операций должна выполняться СУБД по неизвестному алгоритму, и теоретическая оценка производительности не представляется возможной.

Предварительная экспериментальная оценка производительности реализации модели разграничения доступа проведена на тестовой базе данных, соответствующей упрощенной модели данных в ИАС "Наука-МГУ" ("ИСТИНА") со следующими параметрами: 100 000 зарегистрированных пользователей, каждому из которых соответствует сотрудник организации; 200 000 публикаций; 1 000 000 равномерно распределенных связей автора с публикацией. Указанные размеры тестовой базы данных примерно соответствуют размерам базы данных ИАС "Наука-МГУ" ("ИСТИНА") на июль 2014 г. Для сравнения, на конец марта 2015 г. в системе зарегистрировано около 190 000 авторов и 350 000 статей.

Серия экспериментов в указанных условиях показала, что время получения списка операций, которые заданный пользователь имеет право совершать с определенным объектом, не превосходит 5 мс. При условии, что используется постоянно установленное соединение с базой данных, время получения списков операций, которые может совершать определенный пользователь с 64 случайно выбранными объектами, составляет 30...50 мс. Таким образом, эксперименты показали, что реляционная модель ЛРД может применяться в достаточно больших и высоконагруженных системах.

## Заключение

В настоящей работе представлена реляционная модель логического разграничения доступа, предназначенная для применения в многопользовательских информационных системах, включая системы управления контентом, системы управления наукометрической информацией, а также системы с элементами социальных сетей. Предложенная модель обладает рядом преимуществ по сравнению с известными моделями в применении к отдельным целевым системам. В частности, модель ориентирована на информационные системы, в которых данные хранятся преимущественно в реляционных СУБД, и позволяет формулировать широкий класс правил доступа в терминах

используемой модели данных в системе, но в более компактном представлении по сравнению с прямым использованием запросов к СУБД. Теоретически обоснованные особенности модели позволяют конструктивным образом определить возможность эффективного вычисления решения о доступе и провести надлежащую оптимизацию, что обеспечивает необходимый уровень быстродействия для применения в достаточно крупных многопользовательских веб-системах.

### Список литературы


1. **Лапонина О. Р.** Анализ возможностей языка XACML по управлению доступом // Сб. трудов V Междунар. науч.-практ. конференции "Современные информационные технологии и ИТ-образование". Москва: Изд-во МГУ, 2010 С. 473—484.

2. **Anderson A., Lockhart H.** SAML 2.0 profile of XACML v2.0, OASIS Standard, 2005. URL: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf

3. **Васенин В. А., Иткес А. А., Шапченко К. А.** О применении моделей разграничения доступа в социальных сетях к одному классу многопользовательских систем управления контентом // Программная инженерия. 2015. № 4. С. 10—19.

4. **Садовничий В. А., Афонин С. А., Бахтин А. В.** и др. Интеллектуальная система тематического исследования научно-технической информации ("ИСТИНА"). Москва: Изд-во МГУ, 2014. 262 с.

5. **Cheng Y., Park J., Sandhu R.** Relationship-Based Access Control for Online Social Networks: Beyond User-to-User Relationships // PASSAT 2012. IEEE, 2012. P. 646—655.

6. **Cheng Y., Park J., Sandhu R.** Attribute-aware Relationship-based Access Control for Online Social Networks // Data and Applications Security and Privacy XXVIII. Lecture Notes in Computer Science. 2014. Vol. 8566. P. 292—306.

7. **Hu V., Ferraiolo D., Kuhn R.** et al. Guide to Attribute-Based Access Control (ABAC) Definition and Considerations // NIST Special Publication 800-162. URL: http://dx.doi.org/10.6028/NIST.SP.800-162.



**V. A. Vasenin**[1], Professor, e-mail: vasenin@msu.ru, **A. A. Itkes**[1], Researcher, e-mail: itkes@imec.msu.ru,
**K. A. Shapchenko**[1], Senior Researcher, e-mail: shapchenko@iisi.msu.ru,
**V. Yu. Bukhonov**[2], Postgraduate Student, e-mail: bukhonovvyu@gmail.com,
[1] Scientific Research Institute of Mechanics, Lomonosov Moscow State University, Moscow
[2] Lomonosov Moscow State University, Moscow


# Relational Access Control Model Based on Chains of Relations


*A new access control model is proposed for multi-user computer systems. Traditional approaches to the implementation of access control mechanisms, namely the discretionary access control, multi-level security, and role-based models, face the challenge of dealing with dynamic properties of modern systems, including constantly changing set of users, modification of attributes of entities in the system, and rearrangement of relations between them. Under these conditions, typical for systems similar to multi-user content management systems [3] and social networks [5, 6], traditional approaches may lead to an increase in administrative actions for managing the access control mechanism and the access control rules. Declarative approaches, e.g. attribute-based access control [1, 2, 7], provide the means to reduce the verbosity of administrative actions by using more expressive logic to specify access control rules.*

*The proposed model uses so called chains of relations between entities in the target system, including its users and resources, to specify the access control rules. A chain combines a sequence of binary relations by binding entities in adjacent relations. The relation-based approach maps naturally onto the commonly used relational databases. The proposed model maintains efficient computation of access control decisions while still achieving sufficient expressiveness by supporting combination of relation chains, transitive relations, and attribute-based conditions.*

*A formal description of the model is given in the paper. A property of model leading to efficient computation of access control decisions is defined, and a constructive approach to verifying this property is provided along with the proof of the underlying theorem. Experimental evaluation of the proposed model is discussed in application to a model of a computer system "Nauka-MGU" ("ISTINA") [4] designed for user-assisted collection and management of research- and education-related data and metrics in Lomonosov Moscow State University.*



***Keywords:*** *access control, access rules, access control model, relation, RDBMS, relational access control model*


### References


1. **Laponina O. R.** Analiz vozmozhnostei yazyka XACML po upravleniyu dostupom. *Sb. tr. V Mezhdunar. nauch.-prakt. konf. "Sovremennye informatsionnye tekhnologii i IT-obrazovanie"*, Moscow, Izd-vo MGU, 2010, pp. 473—484 (in Russian).

2. **Anderson A., Lockhart H.** *SAML 2.0 profile of XACML v2.0.* OASIS Standard, 2005, available at http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf

3. **Vasenin V. A., Itkes A. A., Shapchenko K. A.** O primenenii modelei razgranicheniya dostupa v sotsial'nykh setyakh k odnomu klassu mnogopol'zovatel'skikh sistem upravleniya kontentom. *Programmnaya ingeneria,* 2015, no. 4, pp. 10—19 (in Russian).

4. **Sadovnichiy V. A., Afonin S. A., Bakhtin A.V., Bukhonov V. Yu., Vasenin V. A., Gankin G. M., Gasparyants A. E., Golomazov D. D., Itkes A. A., Kozitsyn A. S., Tumaikin I. N., Shapchenko K. A.** *Intellektual'naya sistema tematicheskogo issledovaniya nauchno-tekhnicheskoi informatsii "ISTINA"* (Intellectual system for topical analysis of scientific and technical information "ISTINA"). Moscow, Izd-vo MGU, 2014, 262 p. (in Russian).

5. **Cheng Y., Park J., Sandhu R.** Relationship-Based Access Control for Online Social Networks: Beyond User-to-User Relationships. *PASSAT 2012*. IEEE, 2012. pp. 646—655.

6. **Cheng Y., Park J., Sandhu R.** Attribute-aware Relationship-based Access Control for Online Social Networks. *Data and Applications Security and Privacy XXVIII. Lecture Notes in Computer Science*, 2014, vol. 8566, pp. 292—306.

7. **Hu V., Ferraiolo D., Kuhn R., Schnitzer A., Sandlin K., Miller R., Scarfone K.** *Guide to Attribute-Based Access Control (ABAC) Definition and Considerations.* NIST Special Publication 800-162, availavle at: http://dx.doi.org/10.6028/NIST.SP.800-162.

УДК 004.27

**В. М. Баканов**, д-р техн. наук, проф., e-mail: vbakanov@hse.ru, Национальный исследовательский университет "Высшая школа экономики" (НИУ ВШЭ), г. Москва

# Управление динамикой вычислений в процессорах потоковой архитектуры для различных типов алгоритмов

*Сформулированы задачи определения рациональных режимов работы вычислителей постфоннеймановской архитектуры с автоматическим распараллеливанием. Решение поставленных задач выполнено методами компьютерного моделирования, показана многовариантность решений и возможность выбора наиболее эффективных из них.*

***Ключевые слова:*** *распараллеливание процессов обработки данных, проблемы распараллеливания, автоматизация распараллеливания, программное и аппаратное распараллеливание, постфон-неймановские архитектуры, потоковые вычислители, интенсивность вычислений, компьютерное моделирование, стратегии управления интенсивностью вычислений*

В обозримом будущем повышение вычислительной мощности компьютеров реально лишь за счет распараллеливания процессов обработки данных. Специалисты прогнозируют значительные изменения в традиционных подходах к архитектурам параллельных систем (сейчас это в основном системы с распределенной памятью и передачей сообщений). В частности, специалист мирового уровня, разработчик вычислительных систем класса BEOWULF Томас Стерлинг считает [1], что следует обрабатывать данные сразу по мере их готовности, перемещая код к данным (в противоположность тому, как это принято сейчас в моделях передач сообщений), следует перемещать данные к тому месту, где они будут обработаны когда-то в будущем, опираться на микроархитектуру, построенную скорее на идеях обработки потока данных (DATA-FLOW), чем на процессорных ядрах, интенсивно использующих спекулятивное исполнение команд — так экономится энергия. Актуальными остаются вопросы рациональной загрузки вычислительных ядер современных многоядерных процессоров для персональных компьютеров — лишь небольшой процент полезных приложений "умеет" это делать.

Исследования в данной области представляют прямой интерес для программной инженерии, поскольку внедрение новых архитектур вычислителей не может не отразиться в будущем на базовых подходах к программированию (особенно к *параллельному программированию*) как инженерной деятельности.

В настоящее время параллельное программирование требует сложных процедур выявления скрытого параллелизма в алгоритме и распределения параллельных ветвей между независимо работающими вычислительными узлами. Хотя для многих известных алгоритмов такая задача решена, для каждого нового алгоритма ее приходится решать заново; при этом решение зачастую далеко от эффективного. В связи с этим представляют интерес архитектуры вычислителей, позволяющие выявлять параллелизм автоматически на аппаратном уровне.

Одна из таких архитектур — предложенная на рубеже 1960/1970-х гг. Джеком Б. Деннисом не фоннеймановская *потоковая* вычислительная архитектура. Эта архитектура основана на управлении ходом вычислительного процесса не программой (как в традиционном подходе CONTROL-FLOW с использованием регистра-счетчика команд, являющего серьезным препятствием к распараллеливанию), а собственно данными (конечно, корректно организованными). При этом операторы выполняются на свободных *исполнительных устройствах* (ИУ) в соответствии с принципом "*готовности к выполнению*" (ГКВ). Этот принцип, в свою очередь, является следствием "готовности" (результат присваивания значений) всех операндов данного оператора. В нашей стране исследованиями возможности применения DATA-FLOW при создании супервычислителей занимался создатель многопроцессорных вычислительных комплексов "ЭЛЬБРУС" академик В. С. Бурцев [2].

Становление идеологии DATA-FLOW прошло через несколько ступеней — от "статической" (которой присущи "детские болезни" в виде требования единократного присваивания и других подобных) до "динамической" архитектуры. Последняя позволяет выполнять любые действия, доступные для иных архитектур, и при этом не утрачивает положительных качеств автоматизма аппаратного распараллеливания. В настоящее время на пути реализации полномасштабного DATA-FLOW-вычислителя стоят чисто технические барьеры. Необходимостью является применение ассоциативной оперативной памяти (АП, привычная RAM-память в данном случае не может быть использована в силу отсутствия возможности быстрого выявления ГКВ-операторов). Такая память необходимого размера и приемлемой стоимости сейчас не производится. Известные специалистам и связанные с особенностью функционирования SMP-архитектуры проблемные вопросы являются также технологическими и в последние годы активно решаются

в связи с разработкой и производством процессоров с числом ядер в десятки и сотни. Принцип DATA-FLOW уже два десятка лет эффективно используется при работе с кэшем (технология диспетчирования и выполнения — Dispatche/Execute, DE в процессорах P6, *Pentium Pro* [3]) в процессорах общего назначения и применяется в современных специализированных процессорах.

Для классического вычислителя DATA-FLOW размер гранулы параллелизма (последовательности команд, обладающих свойством независимости по данным) равен машинной инструкции. Известны исследовательские проекты с увеличенным размером гранулы (до нескольких сотен операторов и выше), однако при этом для поиска гранул в программе требуются услуги сложного распараллеливающего компилятора.

На основе сказанного автор считает, что поисковые исследования в области функционирования вычислительных архитектур DATA-FLOW перспективны.

Работа вычислителя DATA-FLOW характеризуется дополнительными параметрами (по сравнению с параметрами функционирования вычислителя классической фон-неймановской архитектуры). В случае DATA-FLOW важно знать, каким образом будет развиваться процесс вычислений по времени (*динамика процесса*), так как от этого зависит общее время выполнения программы. Не менее важно определить методы, позволяющие управлять этим процессом. Для определения этих параметров был разработан компьютерный симулятор DATA-FLOW-вычислителя, выполняющий определенную систему команд и позволяющий учитывать длительность исполнения каждой из них [4]. Эксперименты проводились на алгоритмах, наиболее часто встречающихся в задачах моделирования процессов и явлений в механике и физике и анализе данных.

Потоковый вычислитель аппаратно реализует анализ *информационного графа алгоритма* (ИГА, граф зависимостей вида "операции ↔ операнды") в *ярусно-параллельной форме.*

Зависимость числа ГКВ-операторов от времени выполнения программы имеет ярко выраженную неравномерность. Число выполняемых в текущий момент времени операций назовем *интенсивностью вычислений* (ИВ).

Зависимость числа выполненных операций (накопленная кривая ИВ) для конкретного алгоритма от времени выполнения программы и размера обрабатываемых данных представлена на рис. 1 (см. третью сторону обложки). В общем случае зависимость *текущего* числа $N$ исполненных операций алгоритма зависит как от размера $n$ обрабатываемых данных, так и от времени $t$ выполнения программы.

Заметим, что обычно рассматривается только *функция вычислительной трудоемкости* $N(n)$, представляющая собой срез графика на рис. 1 в момент окончания вычислений $t = t_0$; таким образом, *форма кривой ИВ признается несущественной.* При описании работы потокового вычислителя естественно расширение понятия функции вычислительной трудоемкости в *части ее изменения по времени выполнения программы.*

В вычислителях DATA-FLOW-архитектуры возможно управление динамикой вычислений, полезное с точки зрения рационального использования как имеющихся исполнительных устройств (ИУ), так и нагрузки (трафика) внутрикристальных шин передачи данных. Пусть $N_{\text{ГКВ}}^{\max}$ — максимальное число готовых к выполнению команд в ходе выполнения программы; $P$ — число ИУ. Работа DATA-FLOW-вычислителя характеризуется следующими режимами (однозадачный вариант использования потокового вычислителя):

1) $P = 1$, в этом случае реализуется последовательный режим вычислений;

2) $N_{\text{ГКВ}}^{\max} > P > 1$, при этом режиме осуществляется смешанное последовательно-параллельное выполнение программы;

3) $N_{\text{ГКВ}}^{\max} \leqslant P$, этот режим характеризуется максимальным (для данной программы) распараллеливанием.

Иллюстрация этих режимов дана на рис. 2, где представлен результат моделирования управления процессом выполнения программы решения системы линейных алгебраических уравнений (СЛАУ) пятого порядка классическим методом Гаусса. Как видно, максимальное ускорение вычислений составляет чуть более 5 раз, а возможность управления динамикой вычислений (режимы интенсификации и депрессии) возможна только для второго из разобранных режимов (только в этом случае вариативность в порядке выполнения ГКВ-команд может оказывать влияние на динамику выполнения программы).

Зависимости *2* и *3* на рис. 2 соответственно иллюстрируют применение одной из стратегий управления ИВ в целях интенсификации процесса вычислений и его депрессии (по сравнению с равноприоритетной выборкой ГКВ-инструкций из буфера — кривая *1*). Интенсификация и депрессия в процессе вычислений, естественно, могут *целенаправленно* перемежаться. При значительном числе операций в алгоритме эффективность управления возрастает.

При определении приоритета выборки ГКВ-операторов из буфера команд в целях управления динамикой вычислений рационально использовать критерий "полезности" (фактически "*перспективности*") каждого оператора с точки зрения выполнения конечной цели — скорейшего завершения программы. В простейшем случае полезность оператора определяется числом иных операторов, для которых результат выполнения данного служит входным операндом (в рамках данной парадигмы возможны варианты, рис. 3). Такой подход требует просмотра ИГА минимум на одну дугу вперед.

Представляет интерес вопрос — для какого типа алгоритмов подобный метод управления является эффективным, а для какого таким не является? В целях выяснения ответа на этот вопрос была осуществлена серия вычислительных экспериментов по управлению динамикой вычислений для нескольких типов программ. Для исключения влияния различного времени выполнения отдельных арифметических инструкций это время принималось одинаковым.

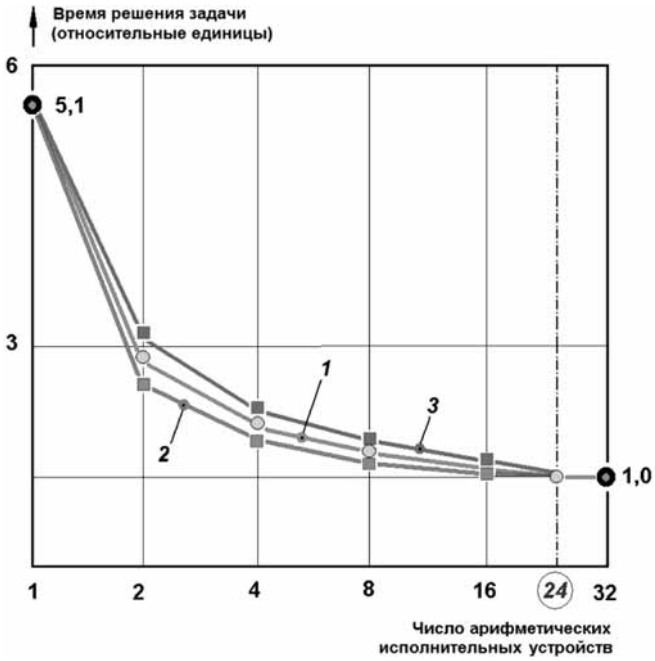


**Рис. 2. Управление интенсивностью вычислений:**

*1* — равноприоритетное выполнение ГКВ-инструкций; *2* — режим интенсификации; *3* — депрессивный режим

В табл. 1 приведены результаты моделирования решения СЛАУ десятого порядка классическим методом Гаусса (число арифметических действий 805, максимальное ускорение при распараллеливании 12,8 при $N_{\text{ГКВ}}^{\max} = 90$) для трех (из нескольких рассматривавшихся) стратегий управления динамикой вычислительного процесса.

Все три стратегии управления основаны на просмотре ИГА вперед (относительно рассматриваемой инструкции) на один шаг с дальнейшим вычислением приоритета данной инструкции при выборке из буфера ГКВ-инструкций на исполнение свободным ИУ. Например, стратегия *A* прямо использует данные

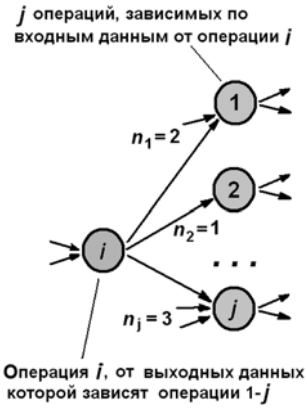


**Рис. 3. К определению "полезности" (перспективности) оператора *i*; $n_j$ — число операндов оператора *j***

о числе иных инструкций, зависящих по цепочке "результат → операнды" от выполнения данной, другие стратегии являются развитием описываемой при различных способах учета информационных зависимостей в графе. Значения таблицы показывают изменение времени выполнения программы (где "−/+" означают уменьшение/увеличение соответственно) относительно режима выборки из буфера ГКВ-инструкций в простейшем режиме (стек типа "первый вошел — первый вышел"). Прямое применение стратегии подразумевает формирование приоритета команд аналогично вышеописанному. При обратном применении приоритет вычисляется как обратная величина.

Из данных табл. 1 видно, что наиболее эффективной (из рассмотренных) является стратегия *C*, позволяющая управлять скоростью выполнения программы в пределах 4,5 % (двойной размах). Стратегии *A* и *B* показывают примерно одинаковый результат (двойной размах до 2,5 %). Заметим, что относительный "вес по времени выполнения" одной инструкции равен $1/805 \approx 0{,}12$ %. При этом стратегия *C* наиболее трудоемка и дополнительно нагружает АП при определении приоритетов.

*Таблица 1*

**Эффективность управления динамикой исполнения программы для трех из ряда рассматриваемых стратегий (решение СЛАУ десятого порядка)**

| Число ИУ | Стратегия *A* | | Стратегия *B* | | Стратегия *C* | |
|---|---|---|---|---|---|---|
| | Эффект прямого применения, % | Эффект обратного применения, % | Эффект прямого применения, % | Эффект обратного применения, % | Эффект прямого применения, % | Эффект обратного применения, % |
| 2 | −0,24 | +0,48 | −0,24 | +0,48 | −1,69 | +0,73 |
| 3 | −0,35 | +0,35 | −0,35 | +0,35 | −2,13 | +1,77 |
| 4 | −0,46 | +1,37 | −0,46 | +1,37 | −2,28 | +2,28 |
| 5 | −0,55 | +0,55 | −0,55 | +0,55 | −1,10 | +3,31 |
| 10 | −0,84 | 0 | −0,89 | 0 | −0,89 | 0 |
| 20 | −1,23 | +1,23 | −1,23 | +1,23 | −2,47 | 0 |
| 50 | 0 | +1,52 | 0 | +1,54 | 0 | +1,54 |
| 70 | 0 | 0 | 0 | 0 | 0 | +1,56 |

*Таблица 2*

**Эффективность управления динамикой исполнения программы для трех из ряда рассматриваемых стратегий (вычисление коэффициента парной корреляции по 20 точкам)**

| Число ИУ | Стратегия *A* | | Стратегия *B* | | Стратегия *C* | |
|---|---|---|---|---|---|---|
| | Эффект прямого применения, % | Эффект обратного применения, % | Эффект прямого применения, % | Эффект обратного применения, % | Эффект прямого применения, % | Эффект обратного применения, % |
| 2 | 0 | 0 | 0 | 0 | +9,41% | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | +13,5% | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | −3,79 | 0 |
| 30 | 0 | 0 | 0 | 0 | 0 | 0 |
| 40 | 0 | 0 | 0 | 0 | 0 | 0 |

Анализ процедур решения СЛАУ различного порядка показывает увеличение управляемости с ростом объема программы. Анализ процедур решения СЛАУ, требующих повышенного времени выполнения операций — операций умножения и деления, и операций сложения и вычитания, показывает приблизительно близкие данные (однако менее продуктивен вследствие априорного незнания статистики по типам операций).

Интересно, что наибольший эффект управления достигается при небольшом числе ИУ (на порядки меньшем, чем $N_{\text{ГКВ}}^{\max}$), что и будет наиболее вероятным режимом при реальной эксплуатации потоковых вычислителей.

В табл. 2 приведены данные эксперимента для программы определения коэффициента парной корреляции по 20 парам точек (число арифметических действий 168, максимальное ускорение при распараллеливании 6,72 при $N_{\text{ГКВ}}^{\max} = 62$).

По данным табл. 2 видно, что в случае вычисления коэффициента парной корреляции (похожие результаты получены для задачи приближения функции методом наименьших квадратов) эффективность управления (управляемость) динамикой выполнения программы значительно хуже, чем для решения СЛАУ (табл. 1). Для стратегий *A* и *B* она нулевая, для оценки эффективности стратегии *C* требуются дополнительные эксперименты. Очевидно, что величина управляемости определяется свойствами ИГА.

Выявление критериев управляемости — непростая задача для будущих исследований. Например, плохая управляемость динамикой вычислений в задаче определения коэффициента парной корреляции и приближения функции методом наименьших квадратов может объясняться наличием множества последовательных цепочек при вычислениях сумм типа $\Sigma x_i$, $\Sigma x_i y_i$ и подобных им при формальном "развертывании" циклов (для "разумного" компилятора естественно использовать процесс сдваивания при суммировании в целях получения алгоритма наименьшей высоты). Основой для определения управляемости (в указанном смысле) может быть уровень связности (ветвистости) ИГА данной программы.

Практический интерес представляет возможность априорного определения управляемости процесса вычислений в представленном выше понимании; это позволит более эффективно определить стратегию управления для данного алгоритма.

Логично разделить (условно) алгоритмы на "легко-" и "трудноуправляемые" (ЛУ и ТУ соответственно) в указанном смысле. К алгоритмам ЛУ-класса логично отнести алгоритмы, характеризующиеся значительным ветвлением (связностью) ИГА (число зависящих по связи "результат → операнды" от выполнения каждого оператора иных операторов велико), и к классу ТУ в противном случае. В первом случае широк выбор альтернативных путей "захвата" вершин ИГА (выполнения операций), во втором случае он более узок.

В табл. 3 дано распределение (гистограмма по числу карманов до 10) числа встречаемости связей между оператором и зависящими от него оператором по связи "результат → операнды". Например, в каждом столбце "1 → *N*" приведено общее число случаев связи по принципу "результат выполнения данной операции → число *N* зависящих от него иных операций". Программы slau_5 и slau_10 — решение СЛАУ пятого и десятого порядков соответственно, mnk_10 и mnk_20 — линейное приближение по методу наименьших квадратов для 10 и 20 точек, korr_10 и korr_20 — определение коэффициента парной корреляции для 10 и 20 точек. Параметр связности ИГА оценивался как отношение сумм числа альтернативных связей ("1 → *N*" при $N > 1$) к числу безальтернативных связей (связи "1 → 1"). Данные табл. 3 показывают, что алгоритмы ТУ-класса характеризуются параметром связности в 10—25 раз меньшим, чем алгоритмы ЛУ-класса. Разработки приемлемой во всех случаях стратегии управления динамикой вычислений продолжаются.

*Таблица 3*

**Параметр связности ИГА (гистограмма) для различных программ потокового вычислителя**

| Программа | Карманы уровней связности операций по принципу "результат данной операции → число операндов зависящей от него операции" | | | | | | | | | | Максимальное ускорение | Параметр связности |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1→1 | 1→2 | 1→3 | 1→4 | 1→5 | 1→6 | 1→7 | 1→8 | 1→9 | 1→10 | | |
| slau_5 | 89 | 5 | 7 | 9 | 4 | 0 | 0 | 0 | 0 | 0 | 4,60 | 0,281 |
| slau_10 | 698 | 6 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 9 | 12,8 | 0,152 |
| mnk_10 | 63 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4,13 | 0,032 |
| mnk_20 | 123 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4,85 | 0,016 |
| korr_10 | 85 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5,87 | 0,024 |
| korr_20 | 165 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6,72 | 0,012 |

Итак, методом имитационного моделирования показана не только собственно возможность целенаправленного управления динамикой вычислений в процессорах потоковой архитектуры путем разработки обоснованных стратегий управления приоритетом выполнения машинных инструкций, но и определена количественная оценка возможностей этого управления для априори заданных типов алгоритмов. Результаты проведенных исследований целесообразно использовать при разработке DATA-FLOW-процессоров и встроенного программного обеспечения для них.

**Список литературы**


1. **Стерлинг Т.** Многоточие Стерлинга // Суперкомпьютеры. 2010. № 3. С. 17—20.
2. **Бурцев В. С.** Вычислительные процессы с массовым параллелизмом // Электроника: Наука, Технология, Бизнес. 2002. № 2. С. 32—35.
3. **Шпаковский Г. И.** Организация параллельных ЭВМ и суперскалярных процессоров. Минск: Изд. Белорусского гос. ун-та, 1996. 296 с.
4. **Ханжин Д. А., Баканов В. М.** Симулятор DATA-FLOW-вычислителя с массовым распараллеливанием операций на уровне инструкций процессора. Свидетельство о государственной регистрации программ для ЭВМ № 2013614155. Федеральная служба РФ по интеллектуальной собственности (Роспатент), дата регистрации 24.04.2013 г.



**V. M. Bakanov**, Professor, e-mail: vbakanov@hse.ru, National Research University Higher School of Economics, Moscow


# Dynamics Control Computing in the Processor Data Flow Architecture for Different Types of Algorithms


*The paper set out the goals of definition of rational modes of calculators post Von Neumann architecture with automatic parallelization of hardware (streaming, DATA FLOW architecture) and proposes solutions to some of them. Evaluators streaming architecture have undoubted advantages in the form of a fully automatic parallelization of data processing at the hardware level and can be considered as an alternative future of modern processors classical von Neumann architecture. In contrast to the traditional architecture of calculators in this case, you can control the intensity of computation (number of simultaneous operations), defined the conditions for such management. Targeted control of the intensity calculation is useful in terms of rational use of both existing performing devices and loads (traffic) inside chip data bus and allows more efficient use of calculators streaming in single and multi-tasking. Management strategies implemented by setting priorities sample ready to run operators from the buffer memory streaming command calculator, with the possible intensification of regimes as well as the intensity of depression calculations. The concept of function computing complexity of expanding its change in time of the program. These tasks performed by computer simulation shows the multiplicity of solutions and the ability to choose the most effective one. The effectiveness of several proposed strategies for managing the intensity calculation is shown by the example of several commonly used standard algorithms.*



***Keywords:*** *parallelization of data processing problems paralleling paralleling automation, software and hardware parallelism, post the von-Neumann architecture, streaming calculators, of computing-intensive, computer simulation, strategy, controlling the intensity calculations*


**References**


1. **Sterling T.** Mnogotochie Sterlinga. *Superkompyuteryi*, 2010, no. 3, pp. 17—20 (in Russian).
2. **Burtsev V. S.** Vyichislitelnyie protsessyi s massovyim parallelizmom. *Elektronika: Nauka, Tehnologiya, Biznes*, 2002, no. 2, pp. 32—35 (in Russian).
3. **Shpakovskiy G. I.** *Organizatsiya parallelnyih EVM i superskalyarnyih protsessorov.* Minsk, Izd-vo Belorusskogo gos. un-ta, 1996. 296 p. (in Russian).
4. **Hanzhin D. A., Bakanov V. M.** Simulyator DATA-FLOW — vyichislitelya s massovyim rasparallelivaniem operatsiy na urovne instruktsiy protsessora. Svidetelstvo o gosudarstvennoy registratsii programm dlya EVM # 2013614155. Federalnaya sluzhba RF po intellektualnoy sobstvennosti (Rospatent), date of registration 24.04.2013.

УДК 004.423

**С. В. Антонов**, студент, e-mail: antonovsergey93@gmail.com,
**М. А. Кривчиков**, науч. сотр., e-mail: maxim.krivchikov@gmail.com,
Механико-математический факультет и НИИ механики МГУ имени М. В. Ломоносова, г. Москва

# Формальная модель вычислений с плавающей точкой на основе лямбда-исчисления с зависимыми типами

*Одним из важных аспектов формальной верификации программных комплексов моделирования физических процессов является доказательство корректности реализации используемых вычислительных методов. Для вычислений в высокопроизводительных средах моделирования, как правило, используют числа с плавающей точкой, соответствующие стандарту IEEE 754. В настоящей статье представлены результаты работы по построению формальной модели чисел с плавающей точкой на основе положений стандарта IEEE 754 в среде Coq и адаптированная версия этой модели для широкого класса реализаций лямбда-исчисления с зависимыми типами. Модель является исполнимой, что позволяет в том числе получать с ее помощью результаты вычислений в среде Coq.*

***Ключевые слова:*** *формальные модели, формальная верификация, лямбда-исчисление, программирование с зависимыми типами, числа с плавающей точкой, погрешность вычислений, формальная семантика*

## Введение

На настоящее время актуальны задачи, связанные с формальной верификацией унаследованных крупных программных комплексов математического моделирования физических процессов, происходящих при функционировании сложных объектов критически важных инфраструктур [1]. При реализации таких комплексов используют вычисления с плавающей точкой. В рамках процессов верификации данных комплексов возникает необходимость доказательства корректности реализации используемых численных методов. Такие доказательства осложняются особенностями выполнения вычислений над числами с плавающей точкой. В частности, в рамках доказательств сходимости и устойчивости численных методов, как правило, рассматривают погрешность, которая вносится во входные данные. Погрешность, которая вносится при замене точных арифметических операций над действительными числами на операции над числами с плавающей точкой, при этом не принимается во внимание, хотя зачастую является причиной ошибок [2].

В настоящей статье представлены результаты работы по построению формальной модели чисел с плавающей точкой стандарта IEEE 754 [3] и операций над ними в среде Coq [4]. Формальная модель была построена в целях получения метода анализа вычислительной погрешности операций над числами с плавающей точкой в рамках процессов формальной верификации программ. При этом предполагается, что формальная верификация проводится методами дедуктивной верификации, использующими формальную модель на основе лямбда-исчисления с зависимыми типами. Разработанная авторами модель является исполнимой, что позволяет получать с ее помощью результаты вычислений в среде Coq.

В целях применения модели в процессе анализа фрагментов исходного кода существующего программного комплекса для модели чисел с плавающей точкой авторами был предложен интерфейс и язык арифметических выражений. С использованием таких средств могут быть исследованы свойства кода, которые проявляются при использовании различных реализаций арифметики на приближениях действительных чисел. В дополнение к модели чисел с плавающей точкой такой интерфейс реализован авторами также для рациональных чисел с операцией приближенного вычисления арифметического квадратного корня с произвольной точностью.

Кроме того, ориентируясь на основные направления дальнейших исследований, отмеченных в заключении настоящей статьи, модель проанализирована на предмет возможности ее адаптации к другим разновидностям лямбда-исчисления с зависимыми типами, отличными от той, которая используется в среде Coq. Результатом этой работы стала адаптация модели, представленной в работе [5], к разновидности лямбда-исчисления с зависимыми типами.

## Стандарт IEEE-754

Стандарт IEEE 754—2008 [3] является основным стандартом, определяющим вычисления с плавающей точкой. В частности, этот стандарт поддерживается большим числом аппаратных реализаций вычислений над числами с плавающей точкой. Кроме того, этот стандарт (а также его предыдущие редакции и редакции стандарта ISO/IEC/IEEE 60559, идентичные по содержанию стандарту IEEE 754) используется в спецификациях распространенных языков программирования, например, таких как C, C# или Fortran.

Стандарт IEEE 754—2008 определяет следующие аспекты вычислений над числами с плавающей точкой:

— форматы чисел с плавающей точкой в двоичной и десятичной системах счисления для вычислений и обмена данными;

— операции сложения, вычитания, умножения и деления чисел с плавающей точкой, а также вычисления арифметического квадратного корня из числа с плавающей точкой;

— операции сравнения, реализующие отношение порядка на числах с плавающей точкой;

— операции прямого и обратного преобразования целого числа в число с плавающей точкой;

— операции преобразования между форматами чисел с плавающей точкой;

— операции преобразования числа с плавающей точкой из двоичной системы счисления в десятичную и обратно.

Следует отметить, что в сферу ответственности стандарта IEEE 754—2008 и настоящей модели не входят форматы представления целых чисел и способы интерпретации знака и мантиссы не-чисел (NaN).

Число с плавающей точкой в формате IEEE 754—2008 представляется как тройка (см. рисунок): *знак* (sign), принимающий значение 1, если число отрицательно, и 0 — в противном случае; *порядок* (exp) — степень основания системы счисления, используемая при вычислении значения числа, при этом порядок хранится со смещением на константу (bias), выбранную таким образом, чтобы диапазон смещенного порядка был неотрицательным; *мантисса* (significant) — дробная часть числа.

Мантисса всегда представлена в нормализованном виде — бит, стоящий до точки, всегда должен быть равен единице (т. е. число с плавающей точкой представляется в виде `1,**...*`, где знаком `*` обозначено произвольное значение бита). Поэтому такой единичный бит, стоящий до точки, не хранится в памяти компьютера. Этот бит называется неявным ведущим битом. Поскольку неявный ведущий бит не занимает места в памяти, это позволяет увеличить максимальную точность чисел.

В стандарте зарезервированы множества значений, обозначающих специальные числа с плавающей точкой — положительная и отрицательная бесконечности и не-числа (NaN). Такие значения возникают в результате операций, результат которых не определен, например при делении на ноль или извлечении квадратного корня из отрицательного числа. Бесконечность записывается в виде порядка, состоящего из единиц и нулевой мантиссы. Значение NaN записывается в виде порядка, состоящего из единиц и ненулевой мантиссы.

По заданной тройке из знака, порядка и мантиссы значение числа находится следующим образом: $(-1)^{sign} \times 2^{(exp\ -\ bias)} \times 1.significant$.

## Разработка модели чисел с плавающей точкой

В качестве типа данных для хранения одного бита был выбран тип, встроенный в систему Coq, — тип `bool`. Для хранения последовательности бит фиксированной длины используется тип вектора с элементами типа `bool`. Такой выбор обусловлен наличием стандартного модуля `Bvector` (вектор с элементами типа `bool`) в среде Coq. В этом модуле реализованы стандартные функции работы с векторами, которые используются в настоящей работе.

Введем базовые параметры модели, а именно: битовую размерность экспоненты и мантиссы (`lenExp` и `lenSign` соответственно); сдвиг и максимальное значение экспоненты (`shiftExp` и `maxExp` соответственно). С использованием этих параметров модель числа с плавающей точкой записывается в Coq так, как представлено на листинге 1.

Для модели, согласно стандарту, реализованы следующие операции:

- операции сравнения (больше, меньше, равно и пр.; основная реализация содержится в функции `compare`);
- арифметические операции (сложение, вычитание, умножение, деление, вычисление арифметиче-


**Пример числа с плавающей точкой одинарной точности (изображение распространяется под лицензией CC BY-SA 3.0 Fresheneesz at the English Wikipedia project, исходный вариант доступен по URL: http://en.wikipedia.org/wiki/File:Float_example.svg)**

```
Record fp_num : Set := mkFp
{
 sign : bool;
 exp : Bvector lenExp;
 significant : Bvector lenSign
 }.
```

**Листинг 1.** Модель числа с плавающей точкой

ского квадратного корня из числа; основная реализация содержится в функциях `sumFp`, `diffFp`, `multFp`, `divideFp`, `sqrtFp` соответственно);

- операции преобразования с потерей точности между значениями модели чисел и другими представлениями, входящими в стандартную библиотеку, а именно — целыми и рациональными числами.

В целях унификации вычислений на числах с плавающей точкой и рациональных числах был предложен интерфейс модели арифметики на приближениях действительных чисел. Интерфейс задает тип числа, арифметические операции, операции сравнения и операции конвертации над элементами типа числа. Определение интерфейса в системе Coq представлено на листинге 2.

```
Record ApproxArith := mkApproxArith {
 Num : Set;
 sum  : Num -> Num -> Num;
 diff : Num -> Num -> Num;
 mult : Num -> Num -> Num;
 div  : Num -> Num -> Num;
 sqrt : Num -> Num;
 from_rational : Q -> Num;
 to_rational : Num -> Q;
 lt : Num -> Num -> bool;
 le : Num -> Num -> bool;
 eq : Num -> Num -> bool;
 ge : Num -> Num -> bool;
 gt : Num -> Num -> bool
}.
```

**Листинг 2.** Интерфейс модели арифметики на приближениях действительных чисел

Авторами разработана реализация интерфейса для чисел с плавающей точкой и для рациональных чисел с операцией приближенного вычисления квадратного корня. Поскольку стандартные модули Coq не поддерживают вычисление арифметического квадратного корня с заданной точностью для рациональных чисел, эта арифметическая операция была реализована авторами. В качестве алгоритма был выбран метод Герона.

Для сравнения вычислительной погрешности работы программы на числах с плавающей точкой и на рациональных числах был построен простейший язык арифметических выражений. Язык поддерживает следующие операции: сложить/вычесть/умножить/разделить два числа; вычислить корень из числа; конвертировать число в/из рационального; условный оператор; получить значение переменной; записать значение в переменную. Программа на таком языке представляется как список записей, состоящих из элементов простейшего языка. Динамическая семантика языка арифметических выражений реализована в терминах интерпретатора. Интерпретатор получает на вход следующие параметры: число переменных в коде; вектор с исходными значениями переменных; код программы, представленный в виде элемента статической семантики языка арифметических выражений; реализацию интерфейса арифметики чисел. Результатом работы интерпретатора является вектор с конечными значениями переменных.

## Тестирование модели

Для тестирования модели был выбран фрагмент кода, входящий в состав программного комплекса моделирования теплогидродинамических процессов в первом и втором контурах атомных реакторов серии ВВЭР [1]. Выбранный фрагмент кода предназначен для определения наличия прямого и обратного потоков жидкости в компонентах моделируемой системы. Фрагмент кода принимает на вход следующие параметры: *p_out*, *p_in_out*, *p_in_in* — показатели давления (Па); *d1, d2* — плотность жидкости на входе и выходе компонента (кг/м$^3$); *dtt* — разрешение модели по времени (с). Выходными данными являются *part1* и *part2* — прямые и обратные потоки (кг), а также *nf* — наличие потока в компоненте.

Исследуемая функция является достаточно компактной (32 строки исходного кода на языке С), однако содержит несколько различных операций над числами с плавающей точкой, в том числе операции извлечения арифметического квадратного корня. Для тестирования исходный код функции был переписан на языке среды Coq, а также в форме статической семантики простого языка вычислений с плавающей точкой, определенного в предыдущем разделе.

Для тестирования использовалась ЭВМ с процессором Intel Core i5-2410M (2300 MHz), оперативной памятью объемом 6Gb DDR3 1333 MHz CL11. На ЭВМ установлена ОС Ubuntu версии 14.04.3 LTS (64-bit), система интерактивного доказательства теорем Coq версии 8.4pl3 и компилятор GCC версии 4.8.2.

*Таблица 1*

**Сравнение результатов вычислений модели с аппаратной реализацией чисел с плавающей точкой**

| *p_out* | *p_in_out* | *p_in_in* | *d1* | *d2* | *dtt* | *Разность part1* | *Разность part2* | *nf* |
|---|---|---|---|---|---|---|---|---|
| 100 | 150 | 200 | 961 | 1024 | 0,01 | $4{,}33 \cdot 10^{-19}$ | 0 | 1 |
| 128 | 130 | 130 | 976 | 953 | 0,08 | $2{,}16 \cdot 10^{-19}$ | 0 | 1 |
| 400 | 300 | 200 | 1056 | 968 | 0,1 | 0 | 0 | 1 |
| 350 | 349 | 350 | 953 | 976 | 0,04 | 0 | 0 | 1 |
| 375 | 375 | 400 | 1024 | 961 | 0,09 | 0 | 0 | 0 |
| 333 | 300 | 400 | 998 | 1017 | 0,06 | 0 | 0 | 0 |
| 350 | 400 | 300 | 961 | 1024 | 0,08 | 0 | $3{,}47 \cdot 10^{-18}$ | 1 |
| 325 | 350 | 310 | 1024 | 961 | 0,04 | 0 | $8{,}67 \cdot 10^{-19}$ | 1 |

*Таблица 2*

**Сравнение относительной погрешности вычислений с использованием модели чисел с плавающей точкой и модели вычислений на рациональных числах с результатами, полученными аналитически**

| Входные параметры | | | Рациональные числа | | | Числа с плавающей точкой | | |
|---|---|---|---|---|---|---|---|---|
| *p_out* | *p_in_out* | *p_in_in* | *part1* | *part2* | Время, с | *part1* | *part2* | Время, с |
| 100 | 150 | 200 | $1{,}1 \cdot 10^{-58}$ | 0 | 918 | $5 \cdot 10^{-17}$ | 0 | 1 |
| 400 | 300 | 350 | 0 | $6 \cdot 10^{-69}$ | 512 | 0 | $9 \cdot 10^{-18}$ | 1 |
| 350 | 300 | 400 | 0 | 0 | 1,5 | 0 | $8 \cdot 10^{-18}$ | 0,5 |
| 350 | 400 | 300 | $4 \cdot 10^{-58}$ | $3 \cdot 10^{-69}$ | 1442 | $5 \cdot 10^{-17}$ | $6 \cdot 10^{-17}$ | 3 |

В табл. 1 представлено сравнение результатов вычисления модели с аппаратной реализацией чисел с плавающей точкой. Входные данные были подобраны таким образом, чтобы покрывать все ветви условных операторов в исследуемой функции. С такими входными данными выполнялся исходный вариант кода на языке C, скомпилированный с ключом `mfpmath = sse`, и вариант кода на языке среды Coq. Ключ компиляции использовался для того, чтобы предотвратить использование более точных (80-битных) вычислений с плавающей точкой на устройстве FPU×87. Столбцы таблицы *Разность part1* и *Разность part2* показывают модуль разности между полученными результатами по соответствующим выходным переменным. Столбец *nf* содержит выходные данные для соответствующей переменной, которые совпали для всех наборов входных данных.

По представленным данным можно сделать вывод о том, что предлагаемая модель чисел с плавающей точкой в целом адекватно описывает аппаратную реализацию вычислений с плавающей точкой. Тем не менее причины, которыми обусловлена ненулевая погрешность, требуют дальнейшего рассмотрения.

В табл. 2 представлены результаты сравнения относительной погрешности моделей чисел с плавающей точкой и рациональных чисел (с вычислением приближенного арифметического квадратного корня) с результатом, полученным аналитически. Сравнивались результаты выполнения интерпретатора, определяющего динамическую семантику языка вычислений с плавающей точкой, с реализациями интерфейса вычислений для чисел с плавающей точкой двойной точности и для рациональных чисел на одном и том же исходном коде. Исходный код был представлен в форме статической семантики языка арифметических выражений. Точность вычисления арифметического квадратного корня для рациональных чисел была ограничена значением $10^{-50}$.

Полученные результаты позволяют утверждать, что модель может быть использована в задачах оценки погрешности, которая является следствием замены точных арифметических операций над действительными числами операциями над числами с плавающей точкой.

## Адаптация модели для других реализаций лямбда-исчисления с зависимыми типами

Основные определения в составе модели вычислений с плавающей точкой представлены как термы разновидности лямбда-исчисления с зависимыми типами, которая лежит в основе среды Coq. Далее рассмотрены основные синтаксические конструкции языка формальной спецификации и функционального программирования Gallina (является основным языком среды Coq), используемые в формальном определении модели, и соответствующие им термы лямбда-исчисления с зависимыми типами.

`Record` — синтаксическая конструкция, определяющая запись, фактически является определением типа зависимой суммы с набором дополнительных функций. В число таких функций входят следующие: конструктор, позволяющий получить элемент типа записи по значениям компонентов; функция удаления, с использованием которой покомпонентно определяются функции (зависимые произведения) из типа записи; функции доступа к компонентам. Все такие функции могут быть определены в рамках лямбда-исчисления с поддержкой зависимых сумм и терма их удаления. Следует отметить, что, согласно официальной документации, конструкция `Record` раскрывается в конструкцию определения индуктивных типов `Inductive` с одним конструктором. Несмотря на то что с позиций формального представления такое определение отличается от используемого в среде Coq определения зависимой суммы `sigT`, определяемые для записи вспомогательные функции соответствуют термам введения и удаления элемента типа зависимой суммы.

`Fixpoint` — синтаксическая конструкция, позволяющая заменить удаление терма индуктивного типа оператором сопоставления с образцом `match`. В рамках модели конструкция используется со следующими типами: тип вектора `Vector.t` (`Bvector`); тип натуральных чисел `nat`; конечные типы из двух (`bool`) и трех (`comparison`) элементов. Таким образом, для определения модели на основе некоторой разновидности лямбда-исчисления с зависимыми типами такая разновидность должна поддерживать термы удаления для перечисленных типов. При этом, как показано в работе [6], термы введения и удаления элемента типа вектора могут быть сведены к таким термам для натуральных чисел и конечных типов.

Таким образом, установлено, что разработанная авторами модель существенным образом использует следующие типы в рамках лямбда-исчисления с зависимыми типами: зависимые суммы; конечные типы из двух и трех элементов; натуральные числа. Кроме того, разновидность лямбда-исчисления, используемая для реализации, должна поддерживать термы удаления для таких типов, эквивалентные тем, которые используются в среде Coq. При этом следует отметить, что стандартные правила вывода для зависимых сумм, конечных типов и натуральных чисел в рамках теории типов (даны, например, в работах [7, 8]) удовлетворяют таким требованиям.

С использованием результатов анализа, представленных в настоящем разделе, модель вычислений с плавающей точкой была адаптирована авторами к разновидности лямбда-исчисления с зависимыми типами, предложенной в работе [5].

## Существующие результаты

Формальные модели вычислений с плавающей точкой достаточно давно являются предметом интереса исследователей. Такие модели представлены в статье [9] и в отчете [10]. При этом, в отличие от настоящей работы, исходный код таких моделей не был опубликован, а в качестве базовой формальной модели использовались модели, отличные от лямбда-исчисления с зависимыми типами. В числе современных работ следует отметить работу [11], в которой используется система Coq и модель лямбда-исчисления с зависимыми типами.

При этом в работе, результаты которой приведены в настоящей статье, в отличие от [11], операции деления и извлечения арифметического квадратного корня не требуют дополнительного указания точности вычислений. В настоящей работе используются реализации операций, соответствующие стандарту IEEE 754. В частности, для тестирования используется формат, соответствующий числам с плавающей точкой двойной точности. В числе других отличий от результатов указанной работы следует выделить компактность предлагаемой авторами модели. Код предлагаемой авторами модели в системе Coq занимает менее одной тыс. строк, тогда как модель из работы [11] вместе с реализацией операций содержит большое количество свойств и их доказательств и занимает 22 тыс. строк кода. Выделить код реализации операций в этой модели для точного сравнения не представляется возможным в связи со сложной структурой зависимостей кода.

Результаты тестирования модели, представленные в настоящей статье, включая интерфейс для модели вычислений с плавающей точкой и результаты, полученные при его применении, ближе к тем, которые опубликованы в работе [12]. Результаты этой работы показывают также, что потенциальной областью приложения модели является верифицируемая компиляция программ. Отметим, что в библиографии статьи [11] приведен ряд дополнительных ссылок на формальные модели вычислений с плавающей точкой для разных формальных систем.

## Заключение

В рамках работы, результаты которой представлены в настоящей статье, построена формальная модель чисел с плавающей точкой в соответствии с положениями стандарта IEEE 754—2008. Для модели реализованы арифметические операции сложения, вычитания, умножения, деления и извлечения арифметического квадратного корня, а также операции сравнения, которые требуют положения этого стандарта. Модель была верифицирована на примере фрагмента кода программного комплекса математического моделирования физических процессов в атомных электростанциях.

Для применения модели к задачам анализа вычислительной погрешности в рамках дедуктивной формальной верификации предложен абстрактный интерфейс вычислений над приближениями действительных чисел и язык арифметических выражений для такого интерфейса. Интерфейс реализован как для разработанной модели, так и для реализации приближенных вычислений над рациональными числами. Для языка разработана статическая и динамическая формальная семантика в форме интерпретатора. Применимость интерфейса и модели к задачам анализа вычислительной погрешности продемонстрирована на примере фрагмента кода, использовавшегося для верификации модели.

Кроме того, в целях применения модели при верификации с использованием вариантов реализаций лямбда-исчисления с зависимыми типами, отличных от принятой в среде Coq реализации, для основных компонентов модели получено представление модели для лямбда-исчисления с зависимыми типами с поддержкой зависимых сумм и натуральных чисел со стандартной схемой удаления.

Дальнейшие исследования в рассматриваемой области будут проводиться по трем перечисленным далее направлениям.

1. Реализация интерфейса вычислений над приближениями действительных чисел над моделью точных вычислений над действительными числами. Такие модели известны для реализаций лямбда-исчисления, поддерживающих коиндуктивные типы [13], в частности, для Coq [14] и Agda [15]. С использованием таких моделей может быть установлено соответствие между используемыми численными методами и их реализацией для чисел с плавающей точкой.

2. Адаптация языка арифметических выражений к общему подходу описания формальной семантики языков программирования с использованием монад и преобразователей монад. Решение этой задачи позволит использовать модель в качестве отдельного модуля при описании формальной семантики языков программирования.

3. Реализация процедур частичного разрешения и DSL на основе модели в рамках реализации методов дедуктивной верификации для языково-ориентированного программирования [5, 16].

Исходный код модели для среды Coq доступен по следующему URL: https://github.com/antonovsergey93/floating-point-model

### Список литературы


1. **Васенин В. А., Астапов И. С., Роганов В. А.** и др. Распараллеливание теплогидравлического расчетного кода CMS в составе полномасштабной суперкомпьютерной модели "Виртуальная АЭС" // Ломоносовские чтения. Тезисы научной конференции. Секция механики. Москва, 15—23 апреля 2013 г. М.: Изд-во МГУ, 2013. С. 39—39.

2. **Goldberg D.** What every computer scientist should know about floating point arithmetic // ACM Computing Surveys. 1991. Vol. 23, N 1. P. 5—48.

3. **IEEE** std. 754-2008 (revision of IEEE std. 754-1985), IEEE standard for floating-point arithmetic. 2008.

4. **Coq** Development Team. The Coq Proof Assistant Reference Manual. INRIA-Rocquencourt, 2012. URL: https://coq.inria.fr/distrib/current/refman/

5. **Васенин В. А., Кривчиков М. А.** Предметно-ориентированные языки с заданной формальной семантикой на основе лямбда-исчисления с зависимыми типами // Международная конференция "Мальцевские чтения". Тезисы докладов. Новосибирск, 2014. С. 25—25. URL: http://www.math.nsc.ru/conference/malmeet/14/Malmeet2014.pdf

6. **Chlipala A.** Certified programming with dependent types: A pragmatic introduction to the Coq proof assistant. The MIT Press, 2013. 440 p.

7. **Thompson S.** Type theory and functional programming. Redwood City, CA: Addison Wesley Longman Publishing Co., Inc, 1991, 338 p.

8. **The Univalent** Foundations Program. Homotopy type theory: Univalent foundations of mathematics. Princeton, NJ: Institute for Advanced Study, 2013, 609 p.

9. **Barrett G.** Formal methods applied to a floating-point number system // IEEE Transactions on Software Engineering. 1989. Vol. 15, N 5. P. 611—621.

10. **Carreno V. A.** Interpretation of IEEE-854 floating-point standard and definition in the HOL system. NASA Langley Technical Report Server, 1995.

11. **Boldo S., Melquiond G.** Flocq: A unified library for proving floating-point algorithms in Coq. IEEE, 2011. P. 243—252.

12. **Boldo S., Jourdan J., Leroy X., Melquiond G.** Verified compilation of floating-point computations // Journal of Automated Reasoning. 2015. Vol. 54, N 2. P. 135—163.

13. **Berger U., Hou T.** Coinduction for exact real number computation // Theory of Computing Systems. 2007. Vol. 43, N 3—4. P. 394—409.

14. **Krebbers R., Spitters B.** Type classes for efficient exact real arithmetic in Coq // Logical Methods in Computer Science / Ed. by M. Escardó. 2013. Vol. 9, N 1. doi: 10.2168/LMCS-9(1:1)2013.

15. **Chuang C. M.** Extraction of programs for exact real number computation using Agda: PhD thesis. Swansea: Dept. of Computer Science, Swansea University, 2011.

16. **Васенин В. А., Кривчиков М. А.** Языково-ориентированное программирование для формальной верификации программного обеспечения // Четвертая научно-практическая конференция "Актуальные проблемы системной и программной инженерии". Тезисы докладов. МИЭМ НИУ ВШЭ, М.: Изд-во НИУ ВШЭ, 2015. С. 30—31.

**S. V. Antonov**, Student, e-mail: antonovsergey93@gmail.com,
**M. A. Krivchikov**, Researcher, e-mail: maxim.krivchikov@gmail.com, Faculty of Mechanics and Mathematics and Institute of Mechanics, Lomonosov Moscow State University, Moscow, Russia

# A Formal Model for the Floating Point Computations for the Dependently-Typed Lambda Calculus

*The proof of correctness for the computational methods which are used in the complex physical simulation software is an important part of the formal verification process for such a software. The high-performance physical simulation algorithms are usually implemented by the means of floating point computing model based on IEEE 754 Standard.*

*In this paper a new formal model for IEEE 754 floating point numbers and arithmetic operations is presented. The model is implemented in Coq proof assistant. The arithmetic operations are executable so that it is possible to obtain the results of computation in Coq. The model is abstract enough to be expressed in any dependently-typed lambda calculi with support for the dependent sums, finite types with two and three elements and the natural numbers. The model is tested by comparison of the computation results for model and for hardware floating point implementations using the fragment of code from production physical simulation package.*

*The floating-point computations model includes the abstract interface for the arithmetic operations on some approximation of real numbers. Based on such interface, the authors specify a simple language of arithmetic expressions with replaceable implementations of arithmetic. By means of the language, the error which is introduced in the mentioned code fragment by replacing exact real number operations with inexact floating point equivalents is justified.*

**Keywords:** *formal models, formal verification, lambda-calculus, programming with dependent types, floating point numbers, floating point inaccuracy, formal semantics*

**References**

1. **Vasenin V. A., Astapov I. S., Roganov V. A., Maydanik V. N., Krivchikov M. A.** Rasparallelivanie teplogidravlicheskogo raschetnogo koda CMS v sostave polnomasshtabnoi superkomp'yuternoi modeli "Virtual'naya AES". *Lomonosovskie chteniya. Tezisy nauchnoi konferentsii. Sektsiya mekhaniki.* Moscow, 15—23 April 2013, Moscow, Izd-vo MGU, 2013, pp. 39—39 (in Russian).

2. **Goldberg D.** What every computer scientist should know about floating point arithmetic, *ACM Computing Surveys,* 1991, vol. 23, no. 1. pp. 5—48.

3. **IEEE** std. 754-2008 (revision of IEEE std. 754-1985), IEEE standard for floating-point arithmetic. 2008.

4. **Coq** Development Team. The Coq Proof Assistant Reference Manual. INRIA-Rocquencourt, available at: https://coq.inria.fr/distrib/current/refman/

5. **Vasenin V. A., Krivchikov M. A.** Predmetno-orientirovannye yazyki s zadannoi formal'noi semantikoi na osnove lyambda-ischisleniya s zavisimymi tipami. *Mezhdunarodnaya konferentsiya "Mal'tsevskie chteniya". Tezisy dokladov.* Novosibirsk, 2014, pp. 25—25, available at: http://www.math.nsc.ru/conference/malmeet/14/Malmeet2014.pdf (in Russian).

6. **Chlipala A.** *Certified programming with dependent types: A pragmatic introduction to the Coq proof assistant.* The MIT Press, 2013, 440 p.

7. **Thompson S.** *Type theory and functional programmin*g. Redwood City, CA: Addison Wesley Longman Publishing Co., Inc, 1991, 338 p.

8. **The Univalent** Foundations Program. Homotopy type theory: Univalent foundations of mathematics. Princeton, NJ: Institute for Advanced Study, 2013, 609 p.

9. **Barrett G.** Formal methods applied to a floating-point number system, *IEEE Transactions on Software Engineering,* 1989, vol. 15, no. 5. pp. 611—621.

10. **Carreno V. A.** Interpretation of IEEE-854 floating-point standard and definition in the HOL system. NASA Langley Technical Report Server, available at: http://shemesh.larc.nasa.gov/fm/papers/NASA-95-tm110189.pdf

11. **Boldo S., Melquiond G.** Flocq: A unified library for proving floating-point algorithms in Coq, *20th IEEE Symposium on Computer Arithmetic (ARITH)*, 2011, pp. 243—252.

12. **Boldo S., Jourdan J., Leroy X., Melquiond G.** Verified compilation of floating-point computations, *Journal of Automated Reasoning*, 2015, vol. 54, no. 2, pp. 135—163.

13. **Berger U., Hou T.** Coinduction for exact real number computation. *Theory of Computing Systems*, 2007, vol. 43, no. 3—4, pp. 394—409.

14. **Krebbers R., Spitters B.** Type classes for efficient exact real arithmetic in Coq. *Logical Methods in Computer Science* / Ed. by M. Escardo. 2013, vol. 9, no. 1. doi: 10.2168/LMCS-9(1:1)2013.

15. **Chuang C. M.** Extraction of programs for exact real number computation using Agda: PhD thesis. Swansea: Dept. of Computer Science, Swansea University, 2011.

16. **Vasenin V. A., Krivchikov M. A.** Yazykovo-orientirovannoe programmirovanie dlya formal'noi verifikatsii programmnogo obespecheniya. *Chetvertaya nauchno-prakticheskaya konferentsiya "Aktual'nye problemy sistemnoi i programmnoi inzhenerii". Tezisy dokladov.* Moscow, Izd-vo NIU VShE, 2015, pp. 30—31 (in Russian).

УДК 004.9

**А. А. Бездушный**, аспирант, e-mail: andrey.bezdushny@gmail.com,
Московский физико-технический институт (Государственный университет), г. Долгопрудный

# Система MemoPIM: управление личной информацией и знаниями с использованием семантических технологий

*В ходе повседневной деятельности человек сталкивается со все большими объемами информации, значительная часть которой хранится в цифровом формате. В настоящей работе на основе библиографических исследований рассмотрены требования к системе MemoPIM, совмещающей функциональные возможности управления личной информацией и знаниями. Рассмотрены вопросы ведения контекста работы, автоматизированной категоризации данных и выявления связей между ресурсами. Знания предложено представлять в RDF-формате, используя OWL-онтологии для описания их структуры.*

***Ключевые слова:*** *управление личной информацией, управление личными знаниями, семантический рабочий стол, семантическая сеть, связанные открытые данные, распространение активации, рекомендации тегов, меры схожести*

## Введение

В ходе своей разноплановой деятельности человек регулярно сталкивается с необходимостью изучения новой информации, значительная часть которой хранится в цифровом формате и распределена по различным источникам. В процессе систематизации и изучения информации вырабатывается собственное понимание сведений, формируются связи между новой и изученной ранее информацией. В ходе этого процесса формируются личные знания, которые человек в дальнейшем использует при решении различных задач профессиональной или повседневной деятельности. Со временем часть знаний неизбежно забывается, что порождает необходимость переноса знаний в цифровой формат и организации эффективной работы с ними.

Вопросы организации личных знаний и ведения работы с ними рассматриваются в рамках задач управления личной информацией (*Personal Information Management*) и управления личными знаниями (*Personal Knowledge Management*). *Управление личной информацией* включает вопросы работы с информационными ресурсами, такими как электронные и бумажные документы, веб-страницы, почтовые сообщения. В задаче *управления личными знаниями* рассматривают вопросы формирования знаний, сопровождения и оперативной работы с ними. Знания накапливаются человеком в ходе его повседневной деятельности. Они могут быть связаны с информационными ресурсами или произвольными понятиями, фактами или идеями, важными для человека. На настоящее время большинство существующих систем подобного назначения ориентированы либо только на управление информацией, либо только на управление знаниями, хотя, как правило, между знаниями и информацией существует тесная взаимосвязь. В настоящей работе рассмотрена система MemoPIM, объединяющая функции управления информацией и знаниями. Система MemoPIM расширяет прототип системы, представленной в работе [1].

## 1. Системы управления личной информацией и знаниями

В данном разделе рассмотрим используемые в настоящее время подходы к управлению личной информацией и управлению личными знаниями.

Популярным направлением исследований в области *управления личной информацией* является Semantic Desktop [2] — подход к организации данных на персональном компьютере. В соответствии с этим подходом все виды информации, хранимые в цифровом формате, — файл, email-сообщение или событие календаря, рассматривают как RDF-ресурсы (RDF — Resource Description Framework) с уникальным идентификатором. Совокупность этих ресурсов образует *информационное пространство* пользователя. Системы управления личной информацией осуществляют сбор сведений из различных источников, объединяют их и предоставляют интерфейсы работы с ним. Данный подход применяется в работах [2—7]. Далее, в самом общем объеме перечислены основные требования, предъявляемые в этих работах к системам управления личной информацией, а именно наличие следующих механизмов:

— предоставления OWL-онтологии, определяющей структуру ресурсов информационного про-

странства, а также возможности ее расширения пользователем;

— загрузки в систему и представления в RDF-формате информации, сформированной сторонними приложениями (почтовые сообщения, календари, контакты);

— проставления связей между только загруженными и существовавшими ранее ресурсами (например, между человеком из списка контактов и отправителем электронного письма);

— предоставления интерфейсов, навигации и поиска по информационному пространству;

— поддержки совместной работы с ресурсами (комментирования, обсуждений).

*Системы управления личными знаниями* организуют работу со знаниями, которые человек накапливает в ходе своей повседневной деятельности. Для автоматизации работы с этими знаниями они сначала должны быть переведены в цифровой формат. В результате этого процесса формируются так называемые *компьютеризированные знания* [8], которые являются лишь "тенью" настоящих знаний, хранящихся в голове человека. Полноценный перенос всех знаний в цифровой формат является трудоемким процессом и в большинстве случаев не оправдывающим ресурсозатраты на его реализацию. По этой причине в работе [9] предложено переносить в цифровой формат только ту часть знаний, которая необходима человеку для извлечения нужной информации из памяти в определенный момент времени. Компьютеризированные таким образом знания — это набор взаимосвязанных ключей (подсказок), увидев которые человек вспомнит, с какой целью они создавались и какие реальные знания с ними ассоциированы. В качестве таких ключей могут выступать электронные документы, заметки, картинки и другие виды информации. В связи с этим важной функцией системы управления знаниями является помощь пользователю в запоминании ключей и последующем их извлечении.

Можно выделить два подхода к представлению знаний и работе с ними — *графический* и *гипертекстовый*. В графических способах записи знания обычно представляются в виде графа, вершинами которого являются ресурсы базы знаний, а ребрами — связи между ними. В гипертекстовом подходе каждый ресурс базы знаний имеет собственное детальное представление, из которого выводятся все его атрибуты и связи с другими ресурсами. Переход между ресурсами осуществляется с помощью ссылок.

Среди наиболее распространенных способов графической записи знаний можно выделить карты памяти (*Mind Map* [10]) и карты понятий (*Concept Map* [11]). В работе [12] в качестве интерфейса для работы со знаниями предложено использовать тематические карты (*Topic Maps* — ISO-стандарт графического представления знаний). В работе [13] рассмотрена система iMapping, предоставляющая интерфейс для графического описания знаний. Основной особенностью iMapping является использование масштабируемых пользовательских интерфейсов для представления знаний. Все ресурсы базы знаний располагаются на неограниченной плоскости, а их содержимое становится доступно по мере увеличения масштаба интерфейса. Система Docear [14] позволяет использовать карты памяти при ведении и анализе библиографической информации.

В работе [9] использован гипертекстовый подход к работе со знаниями. Авторы описывают модель *Conceptual data structures* (CDS), в основе которой лежит представление знаний в виде троек "объект—предикат—субъект". В этой работе также описана система "*Hypertext Knowledge Workbench*", предоставляющая гипертекстовый интерфейс для создания и управления знаниями. В работах [15, 16] рассмотрены вопросы проектирования модели организации знаний, согласующейся с устройством памяти человека, а также вопросы применения механизма распространения активации для ведения контекста работы пользователя. Другим способом гипертекстового представления знаний является подход семантических Wiki [17, 18]. Семантические Wiki расширяют стандартные Wiki, позволяя задавать семантику связей между страницами.

Большинство рассмотренных систем поддерживают либо только управление знаниями, либо только управление информацией, хотя личные знания человека часто связаны с информационными ресурсами, хранимыми на его компьютере или в сети. Также мало внимания уделяется вопросам анализа информации и знаний, автоматизации действий, выполняемых пользователем. На основании проведенного анализа были определены ключевые функции системы MemoPIM, объединяющей управление личной информацией и знаниями.

## 2. Архитектура системы MemoPIM

Предлагаемое автором решение развивает идеи *Semantic Desktop*, дополняя функции системы управления информацией механизмами, обеспечивающими работу со знаниями. Помимо информации, изначально хранящейся в цифровом формате (электронные документы, email-сообщения), система поддерживает работу с личными знаниями человека. Пополняя сведения, выгруженные из внешних источников дополнительными метаданными, и проставляя связи между ресурсами, пользователь формирует цифровое представление личных знаний. На рис. 1 приведена архитектура системы.

Рассмотрим основные модули, представленные на схеме. В основе системы лежит база знаний пользователя. Структура базы знаний задается системной онтологией и ее пользовательскими расширениями. Более подробно системная онтология будет рассмотрена в следующих разделах. База знаний пополняется либо вручную пользователем, с помощью интерфейсов системы, либо автоматизировано, из внешних источников.

Внешними источниками для пополнения базы знаний пользователя являются файловая система

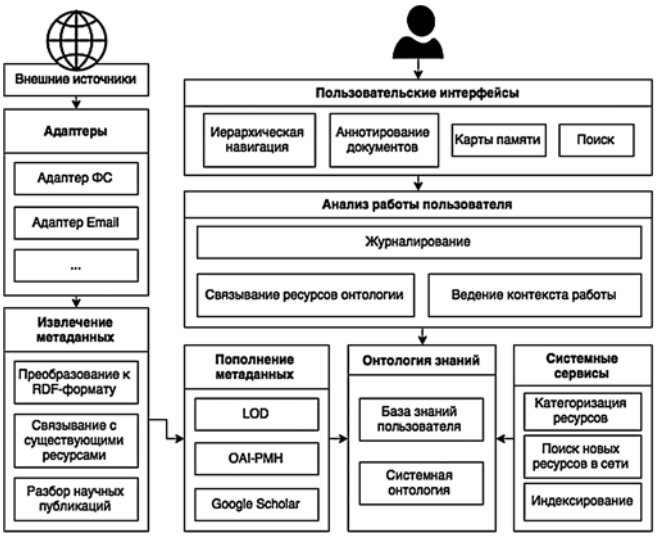


**Рис. 1. Архитектура системы MemoPIM**

(ФС), почтовый сервер и др. В фоновом режиме адаптеры осуществляют синхронизацию ресурсов базы знаний с внешними источниками.

После выгрузки информации из внешних источников она приводится к RDF-формату в соответствии с системной онтологией. Далее загруженные данные связываются с существующими ресурсами, например, отправитель письма ассоциируется с ресурсом, представляющим соответствующую персону в базе знаний. После этого для некоторых типов ресурсов из их содержимого выделяются дополнительные метаданные. Например, из текста научной публикации выделяются название, авторы, список литературы. Более подробно процесс выделения указанных метаданных из текстов публикаций рассмотрен в работе [19].

Следующим этапом является пополнение созданного ресурса дополнительными метаданными из внешних источников. Например, для научных публикаций на основании названия публикации осуществляется поиск полной библиографической информации. В случае успеха найденная информация переводится в RDF-формат и сохраняется в базе знаний пользователя. Более детально модуль, реализующий такое пополнение, рассмотрен в работе [19].

Для работы с базой знаний системой предоставляются иерархические и графические пользовательские интерфейсы. В режиме иерархического представления пользователю выводится дерево ресурсов, соединенных иерархическими связями. Просмотр полной информации о ресурсе доступен с помощью его детального представления. С помощью графического интерфейса работы пользователь может создавать карты памяти, узлами которых являются ресурсы базы знаний. Интерфейс аннотирования позволяет размечать и комментировать текстовое содержимое ресурсов базы знаний. Интерфейс поиска позволяет осуществлять полнотекстовый поиск по базе знаний, а также поиск по ресурсам с учетом их структуры.

По мере работы пользователя системой ведется журналирование выполняемых им действий. Записи из журналов в дальнейшем используются для ведения контекста пользователя и поиска новых связей между ресурсами. Более подробно эти модули рассмотрены в следующих разделах.

Системные сервисы проводят фоновый анализ базы знаний пользователя. Сервис автоматизированной категоризации отвечает за предоставление пользователю рекомендации по присвоению ресурсам тегов. Сервис поиска новых ресурсов в сети отвечает за предоставление пользователю потенциально интересных ему ресурсов на основании анализа тех, которые хранятся в базе знаний. Например, на основании имеющихся в базе знаний публикаций в соответствии с рекомендациями Google Scholar система предлагает другие публикации, потенциально интересные пользователю. Сервис индексирования отвечает за ведение индексов, необходимых для эффективного поиска информации в базе знаний.

## 3. Системная онтология

Системная онтология описывает основные понятия и классы, используемые различными модулями системы. Онтология состоит из десяти частей, разделенных по предметным областям. Используются классы из следующих онтологий: *Simple Knowledge Organization System* (SKOS)[1], *Dublin Core Terms* (DCTERMS)[2], *Friend of a Friend* (FOAF)[3], *NEPOMUK Information Element Ontology* (NIE)[4], *NEPOMUK File Ontology* (NFO)[5], *NEPOMUK Contact Ontology* (NCO)[6], *NEPOMUK Message Ontology* (NMO)[7], *NEPOMUK Calendar Ontology* (NCAL)[8], *Personal Information Model Ontology* (PIMO)[9], *Task Model Ontology* (TMO)[10], *FRBR-aligned Bibliographic Ontology* (FaBiO)[11], *Citation Typing Ontology* (CiTO)[12], *User Interaction Context Ontology* (UICO) [20]. На рис. 2 (см. третью сторону обложки) представлены ключевые классы системной онтологии и связи между ними. Штриховыми линиями обозначено наследование, сплошными — связи между классами. В начале названий всех классов стоит пространство имен, соответствующее одной из перечисленных выше онтологий. Классы, созданные в рамках данной работы, отмечены пространством имен "mpim".

В основе системных онтологий лежит *онтология знаний*, в которой определяется класс Resource, являющийся родительским для всех остальных классов. Для структурирования знаний вводятся ассоциатив-

[1] http://www.w3.org/TR/skos-reference/
[2] http://dublincore.org/documents/dcmi-terms/
[3] http://xmlns.com/foaf/spec/
[4] http://www.semanticdesktop.org/ontologies/2007/01/19/nie
[5] http://www.semanticdesktop.org/ontologies/2007/03/22/nfo
[6] http://www.semanticdesktop.org/ontologies/2007/03/22/nco
[7] http://www.semanticdesktop.org/ontologies/2007/03/22/nmo
[8] http://www.semanticdesktop.org/ontologies/2007/04/02/ncal
[9] http://www.semanticdesktop.org/ontologies/2007/11/01/pimo/
[10] http://www.semanticdesktop.org/ontologies/2008/05/20/tmo/
[11] http://purl.org/spar/fabio
[12] http://purl.org/spar/cito

ные/иерархические связи между ресурсами и теги. Выделение иерархических связей необходимо для поддержания иерархического режима представления знаний.

## 4. Карты памяти

Карты памяти (*Mind Maps* [10]) — это диаграммы, обычно используемые для представления идей, мыслей, задач или других видов знаний. В центре карты памяти располагается узел, представляющий основную идею. По радиусу от него распространяются ветви, образуя древовидную структуру, в вершинах которой находятся различные идеи, слова или понятия. Карты памяти часто используют для представления знаний при обучении, для классификации идей или при проведении "мозгового штурма".

Можно выделить несколько направлений исследований, связанных с картами памяти. В работе [14] на основании содержимого карт памяти пользователю предоставляются рекомендации по потенциально интересным ему научным публикациям. В работе [21] рассмотрены вопросы использования онтологий при работе с картами памяти.

В рамках системы MemoPIM карты памяти применяют для изучения, анализа информации, регистрации новых знаний. Карты памяти, с одной стороны, являются визуально удобной для восприятия человека формой записи знаний, а с другой стороны, легко могут быть преобразованы в машинно-читаемую форму — RDF-данные. В работе [21] к недостаткам карт памяти относят отсутствие формальной семантики, что приводит к двусмысленной интерпретации знаний, усложняет их совместное использование. В разрабатываемой системе узлами карты памяти являются ресурсы базы знаний, а ребрами — связи между ними. Таким образом, знания, формализованные в карте памяти, описываются системной онтологией и ее пользовательскими расширениями.

## 5. Ведение контекста работы пользователя

Люди, как правило, испытывают трудности при одновременной работе в разных сферах предметной деятельности, что связано с необходимостью регулярных переключений между текущими задачами. Чтобы возобновить работу в другой сфере, человеку необходимо вернуться в состояние, в котором он был на момент переключения, вспомнить, о чем он думал и в каком направлении планировал развивать мысли. Поэтому важной задачей системы управления личной информацией является ведение контекста работы пользователя. По мере выполнения действий система должна отслеживать ресурсы, с которыми работает пользователь, запоминать их и упрощать дальнейший доступ к ним.

Для задачи ведения контекста можно руководствоваться принципами работы с*емантической (долгосрочной) памяти человека*. Семантическую память можно рассматривать в виде графа, узлами которого


**Рис. 3. Граф понятий семантической памяти**

являются понятия, а ребрами — связи между ними (рис. 3). В процессе размышления о некотором понятии человек вспоминает также и о других понятиях, связанных с ним в этом графе.

Общепринятым принципом извлечения информации из долгосрочной памяти считается механизм *распространения активации* (*Spreading Activation* [22]). В соответствии с ним получение информации из памяти начинается с внешнего стимула, провоцирующего активацию одного из понятий, находящихся в долгосрочной памяти. Например, если человек видит пожарную машину, то происходит активация соответствующего понятия в памяти. Далее активация распространяется по ребрам графа понятий, при этом большее значение активации получают элементы, связанные ребрами с наибольшим весом, т.е. элементы, более близкие по смыслу. Например, при активации термина "пожарная машина" для дальнейшей работы становятся доступны такие понятия, как "красный", "огонь", "дом".

В рамках проектируемой системы механизм распространения активации может использоваться для ведения текущего контекста работы пользователя. В таком случае внешними стимулами являются действия пользователя (например, просмотр, редактирование или поиск ресурсов), а в роли графа понятий выступают ресурсы базы знаний и имеющиеся между ними связи. По мере работы с системой формируется группа ресурсов, обладающих наибольшим значением активации. Такие ресурсы представляют собой *текущий контекст работы пользователя* — наиболее актуальные для него сведения на конкретный момент времени.

### 5.1. Алгоритм распространения активации

Стандартный алгоритм распространения активации описан в работе [23]. В данном разделе рассмотрим основные факторы, оказывающие влияние на результаты работа алгоритма.

При применении алгоритма распространения активации важно, чтобы между ресурсами, которые часто используются совместно, существовала связь в графе. Частично данный вопрос решается использованием тегов, однако этого может быть недостаточно.

За формирование дополнительных связей отвечает отдельный модуль системы, который будет рассмотрен в разд. 7.

Другим дополнением, которое может улучшить расчет активации, является ведение нескольких уровней активации одновременно. Данный подход применен в работе [16]. В ней авторы предлагают различать три вида активации — *краткосрочную* (STA), *среднесрочную* (MTA) и *долгосрочную* (LTA). При превышении STA некоторого порогового значения увеличивается значение MTA. Аналогично увеличение MTA ведет к росту LTA. Каждому виду активации соответствует свое значение коэффициента затухания $d_s$. Значения среднесрочной и долгосрочной активации учитываются при расчете краткосрочной — ресурсы, обладающие среднесрочной или долгосрочной активацией, оказывают большее влияние на распространение краткосрочной активации.

### 5.2. Онтология контекста работы и онтология действий пользователя

Онтология контекста работы описывает основные классы, необходимые для применения механизма распространения активации. Класс Context представляет контекст работы пользователя, контекстом может быть проект (Project) или задача (Task). В рамках некоторого контекста каждому ресурсу может быть проставлено значение *активации* (Activation). Активация делится на краткосрочную (ShortTermActivation), среднесрочную (MiddleTermActivation) и долгосрочную (LongTermActivation). Класс RelationWeight позволяет задать вес связи, используемый при расчете активации.

Онтология действий пользователя описывает различные типы пользовательских действий, которые инициируют процесс распространения активации. В ней используются классы онтологии *User Interaction Context Ontology* (UICO), описанной в работе [20].

## 6. Автоматическая категоризация ресурсов

Традиционно выделяют два основных подхода к категоризации ресурсов — с помощью иерархических структур и с помощью тегов. Разрабатываемая система объединяет оба вида представлений, позволяя создавать иерархические связи между тегами.

По мере наполнения базы знаний пользователя ресурсами и соответствующими им тегами появляется возможность проставлять теги к новым ресурсам автоматически. Эту задачу можно рассматривать как задачу рекомендации возможных тегов для ресурсов. На настоящее время такие рекомендации являются весьма популярным направлением исследований. Обычно рекомендации тегов используются в системах, предоставляющих возможность ведения фолксономий — пользовательских таксономий, например *CiteULike*[13], *Bibsonomy*[14], *Delicious*[15].

[13] http://www.citeulike.org/
[14] http://www.bibsonomy.org/
[15] https://delicious.com/

### 6.1. Обзор существующих решений

Можно выделить три основных подхода к формированию рекомендаций для документа: на основании статистики использования тегов с этим документом [24—26], на основании статистики использования тегов и содержания документа [27—34], только на основании содержания документа [35, 36]. Первый подход применяется в основном для больших фолксономий, в которых ко многим документам теги назначаются различными пользователями. Второй и третий подходы применимы для менее обширных фолксономий.

В работе [24] предложен алгоритм *FolkRank*, позволяющий отсортировать узлы в трехдольном графе пользователи — документы — теги. В основе *FolkRank* лежат такие же соображения, что и в алгоритме *PageRank*, который применяется для ранжирования веб-страниц поисковыми системами. Согласно этому алгоритму ресурс, который отмечен важным тегом, сам считается важным. Другой подход к анализу статистики использования тегов применен в работах [26, 27]. В них исследована совместная встречаемость тегов, т. е. новые теги к ресурсу предлагаются на основании других проставленных к нему тегов. Основным недостатком подходов, в которых рекомендации строятся только на основании статистики применения тегов, является требование хорошей связанности фолксономии, что не позволяет решить проблему "холодного старта".

Для решения проблемы "холодного старта" часто предлагается при построении рекомендаций использовать содержимое документов. В ряде работ ([27—31]) рассматривали схожесть документов по текстовому содержимому целиком. В других работах ([29—34]) исследовали зависимости между отдельными словами и тегами, которые присвоены документам.

В первой группе работ, перечисленной выше, предложены различные метрики для расчета схожести документов, чаще всего основанные на векторном представлении текстов и различных мерах важности слов в тексте, таких как TF-IDF и Okapi BM25. Основные различия заключаются в методе выбора подходящих тегов. В работах [28—30] при рекомендации тегов для нового документа предложено отбирать на основании меры схожести набор документов, схожих с ним. Далее эти теги ранжируются и предлагаются пользователю как подходящие. В работах [31, 32] для подбора подходящих тегов предложено делать анализ графа пользователи — документы — теги, но в отличие от работы [24] дополнительные ребра предложено проводить между самими документами. Вес этим ребрам ставится пропорционально схожести документов.

В работах второго направления ([29—34]) исследована связь отдельных слов с тегами. В начале из текстового содержимого отбирают ключевые слова, которые в дальнейшем используются при анализе. В работе [29] в рамках метода *ContentCoOcc* введена функция, определяющая степень связанности слов

и тегов. В работе [32] предложен гибридный подход к построению рекомендаций, сочетающий анализ взаимосвязей между словами и тегами, статистику совместного использования тегов и предпочтения пользователя. В работе [33] описан алгоритм для нахождения ассоциативных связей между словами и тегам. Результатом алгоритма являются правила вида A, B → C (из слов A и B следует тег C). В работах [30, 35] исследован граф слово — тег — пользователь, в работе [29] применена модификация алгоритма *FolkRank* [24], а в работе [34] для расчета схожести между тегами и словами построена нейронная сеть.

Последним направлением исследований является автоматическое выделение тегов непосредственно из текстового содержимого документа. В работе [35] рассмотрена задача автоматического выделения ключевых слов из текстов научных публикаций. Для каждого слова определяется набор характеристик (например, вхождение слова в название статьи или аннотацию, положение в документе), которые в сумме определяют, насколько слово характеризует данную статью. Вес для каждой характеристики определяют с помощью алгоритмов машинного обучения на тестовой выборке. В работе [36] предложены два подхода. Первый, требующий наличия обучающей выборки, основан на машинном обучении с учителем и во многом аналогичен методу, представленному в работе [35]. Второй метод основан на графовом представлении текста. Согласно ему каждому слову текста ставится в соответствие вершина в графе, ребро между вершинами проставляется в том случае, когда эти слова встречаются в одном предложении. Далее проводится ранжирование графа с помощью алгоритма, основанного на алгоритме *PageRank*.

В работе [29] проведено сравнение эффективности пяти алгоритмов рекомендации тегов. Протестированы следующие алгоритмы: 1) графовый, без учета содержимого; 2) графовый с вершинами "документ", "пользователь", "тег"; 3) графовый с вершинами "слово", "пользователь", "тег"; 4) неграфовый, основанный на схожести документов; 5) неграфовый, основанный на зависимостях между словами и тегами. По результатам тестирования на различных наборах данных авторы пришли к выводу, что графовые методы не дают существенного прироста в качестве рекомендаций, обладая при этом существенно большей вычислительной сложностью. Перечисленные выше методы 4 и 5 выдают схожие результаты.

### 6.2. Реализация

В системе MemoPIM используется неграфовый метод, учитывающий содержимое документов. Основные идеи метода взяты из работ [29, 32]. Поскольку в большинстве случаев теги и ресурсы в базах знаний различных пользователей отличаются, в качестве источника для рекомендаций автором было решено использовать только теги и ресурсы из базы знаний самого пользователя.

Перед формированием рекомендаций из каждого слова выделяется его основа (проводится стемминг слов), а также отбрасываются стоп-слова. Далее для всех оставшихся слов высчитывается связь между словом и тегом. Значение этой функции определяет, насколько слово характеризует ресурс.

$$WT(w,t) = \sum_{r \in Res(w,t)} \left(\delta_t(w,r) + \delta_c(w,r)\right) Tfidf(w,r).$$

Здесь $r$ — ресурс, Res($w$, $t$) — множество ресурсов, отмеченных тегом $t$ и содержащих слово $w$;

$$\delta_t(w,r) = \begin{cases} 1, w \in T(r) \\ 0, \quad \text{иначе} \end{cases}, \quad \delta_c(w,r) = \begin{cases} 1, w \in C(r) \\ 0, \quad \text{иначе} \end{cases};$$

$T(r)$, $C(r)$ — множества слов, входящих в название и содержание ресурса соответственно;

*Tfidf* — мера информативности слова, которая вычисляется по следующей формуле:

$$Tfidf(w,r) = \frac{n_w}{N} \log \frac{|R|}{|R_w|},$$

где $n_w$ — число вхождений слова $w$ в текст; $N$ — общее число слов в тексте; $|R|$ — общее число ресурсов; $|R_w|$ — число ресурсов, содержащих слово $w$.

При подборе тегов для рекомендации рейтинг тега рассчитывается по следующей формуле:

$$RT(r,t) = \sum_{w \in r} Tfidf(w,r) WT(w,t).$$

Таким образом, влияние слова на рейтинг тега складывается из двух факторов:

- насколько это слово характеризует ресурс;
- насколько часто слово встречается в ресурсах, помеченных тегом, а также насколько оно характеризует эти ресурсы.

## 7. Связывание ресурсов базы знаний

В рамках данного модуля проводится анализ базы знаний в целях построения дополнительных связей между ресурсами. Для поиска связанных между собой ресурсов обычно применяют различные меры схожести (*similarity measures*). Рассмотрим меры, применяемые для расчета схожести между RDF-ресурсами. В работе [37] схожесть ресурсов рассчитывали на основании общих ключевых слов, которые их характеризуют. При расчете используется мера схожести ресурсов *Semantic Connectivity Score* (*SCS*):

$$SCS(W_1,W_2) = \left( \sum_{\substack{e_1 \in E_1 \\ e_2 \in E_2 \\ e_1 \neq e_2}} SCS_e(e_1,e_2) + \frac{|E_1 \cap E_2|}{2} \right) \frac{1}{|E_1||E_2|}$$

$$SCS_e(e_1,e_2) = \sum_{l=1}^{n} \beta^l \left| path_{(a,b)}^{<l>} \right|,$$

где $W_1$, $W_2$ — сравниваемые ресурсы; $e_1$, $e_2$ — ключевые слова; $E_1$, $E_2$ — наборы ключевых слов, ассоциированных с ресурсами $W_1$ и $W_2$; $SCS_e(e_1, e_2)$ — мера схожести ключевых слов; β — коэффициент затухания; $n$ — число возможных путей между вершинами $a$ и $b$; $|\,\text{path}^{<l>}_{(a,b)}\,||$ — длина пути между ключевыми словами. К недостаткам данного подхода можно отнести тот факт, что мера рассчитана только на наличие одного вида связей между ресурсами, а также отсутствие нормирования меры частотой использования ключевых слов. Нормирование позволило бы уменьшить влияние наиболее распространенных ключевых слов на близость ресурсов.

Другой подход для расчета схожести описан в работе [38]. Авторами предложена мера семантической близости между вершинами RDF-графа

$$\text{Sim}(a,b) = \max_{i=\overline{1,k}} \prod_{j=1}^{n_i} pv_{i,j},$$

где $k$ — число возможных путей между вершинами $a$ и $b$; $n_1$ — число возможных семантических отношений между вершинами $a$ и $b$ на пути $i$; $pv_{i,j}$ — значение веса ребра на основе $j$-го предиката на пути $i$. На основании семантической близости между вершинами графа авторы предлагают рассчитывать близость между триплетами и наборами триплетов. К недостаткам данного подхода можно отнести необходимость ручного задания весов.

Еще один подход, использующий пути в RDF-графе для расчета схожести, описан в работе [39]. При этом расчете предложено использовать меру информативности пути между ресурсами. Для такого расчета используется мера *Predicate Frequency — Inverse Document Frequency* (PFITF). Функции $PFITF_i^u(p)$ и $PFITF_o^u(p)$ определяют меру информативности входящих (incoming) и исходящих (outgoing) связей ресурса $u$ посредством предиката $p$:

$$PFITF_i^u(p) = \frac{\left|T_i^u(p)\right|}{\left|T_i^u\right|} \log \frac{|T_i|}{|T_i(p)|}.$$

Здесь $T_i^u(p)$ — множество (?, $p$, $u$); $T_i^u$ — множество (?, ?, $u$); $T_i$ — общее множество троек в репозитории; $T_i(p)$ — множество (?, $p$, ?). Аналогичным образом вводится мера $PFITF_o^u(p)$, где вместо троек (?, $p$, $u$) используются тройки ($u$, $p$, ?). На основании посчитанных мер вычисляется мера информативности пути в графе:

$$I\left(w\left(u_s,u_f\right)\right) = \sum_{(u_i,u_{i+1})\in w} I(u_i,u_{i+1},p_i),$$

$$I(s,f,p) = \frac{PFITF_o^s(p) + PFITF_i^f(p)}{2}.$$

Здесь $I(s, f, p)$ — информативность тройки ($s$, $p$, $f$), $s$, $f$ — вершины графа, $p$ — соединяющее их ребро; $I(w(u_s, u_f)$ — мера информативности пути из ресурса $u_s$ в $u_f$. К недостаткам данного подхода можно отнести то обстоятельство, что нормирование выполняется по частоте использования предикатов, но не выполняется по частоте использования конкретных ресурсов. Наличие последнего уменьшит влияние наиболее распространенных ресурсов на информативность пути. Кроме того, нормирование по частоте использования предикатов может быть малоэффективно в случае, когда в большинстве троек используется один и тот же предикат. В качестве такого примера можно привести задачу, рассмотренную в работе [37], где единственным видом метаданных были ключевые слова.

### 7.1. Реализация

В системе MemoPIM для расчета схожести ресурсов базы знаний проводится анализ не только самого RDF-графа, но и того, как пользователь строит свою работу с ним. Длина ребра ($a$, $r$, $b$) рассчитывается по формуле

$$d(a,b) = 1 - C(a,b)\alpha - RFITF(r,a,b)\beta,$$

где $C(a, b)$ — частота совместного использования ресурсов $a$ и $b$ — чем чаще в журналах работы пользователя два ресурса используются в близкий промежуток времени, тем меньше расстояние между этими ресурсам в RDF-графе. Второй величиной, определяющей близость ресурсов, является функция *Relation Frequency — Inverse Triple Frequency* (*RFITF*), схожая с функцией, предлагаемой в работе [39] (коэффициенты α, β определяют влияние описанных выше слагаемых на итоговое значение меры схожести):

$$RFITF(r,a,b) = \frac{\left|T(r,a,b)\right|}{\left|T(r,a)\right|} \log\left(\frac{\left|T(r)\right|}{\left|T(r,b)\right|}\right) \log\left(\frac{|T|}{\left|T(r)\right|}\right),$$

где $|T(r, a, b)|$ — мощность связи ресурса $a$ с ресурсом $b$ посредством предиката $r$. Мощность может задаваться для разных предикатов по-разному. Например, для предиката, связывающего ключевое слово с текстовым документом, мощность — это количество вхождений ключевого слова в документ;

$|T(r, a)|$ — число троек ($a$, $r$, ?) и (?, $r$, $a$);

$|T(r)|$ — число троек (?, $r$, ?);

$|T|$ — общее число троек в репозитории пользователя.

Мера схожести между ресурсами определяется по формуле

$$S(A,B) = \sum_{p\in P} \gamma^{l(p)} \frac{1}{d(p)},$$

$$d(p) = \sum_{a,b\in p} d(a,b),$$

где γ — коэффициент затухания; $P$ — множество существующих путей в графе между вершинами $A$ и $B$; $l(p)$ — число ребер в пути; $d(p)$ — длина пути $p$.

## 8. Интерфейсы управления знаниями

Система поддерживает два вида представления знаний — *иерархическое* (рис. 4) *и графическое* (рис. 5). Информация, которая выгружается из внешних источников (файловой системы, почтового сервера), обычно сгруппирована иерархически. При импорте создаются соответствующие RDF-ресурсы, которые соединяются иерархическими связями. В режиме иерархического представления знаний рабочая область делится на три части. Слева выводится дерево ресурсов, соединенных иерархически-

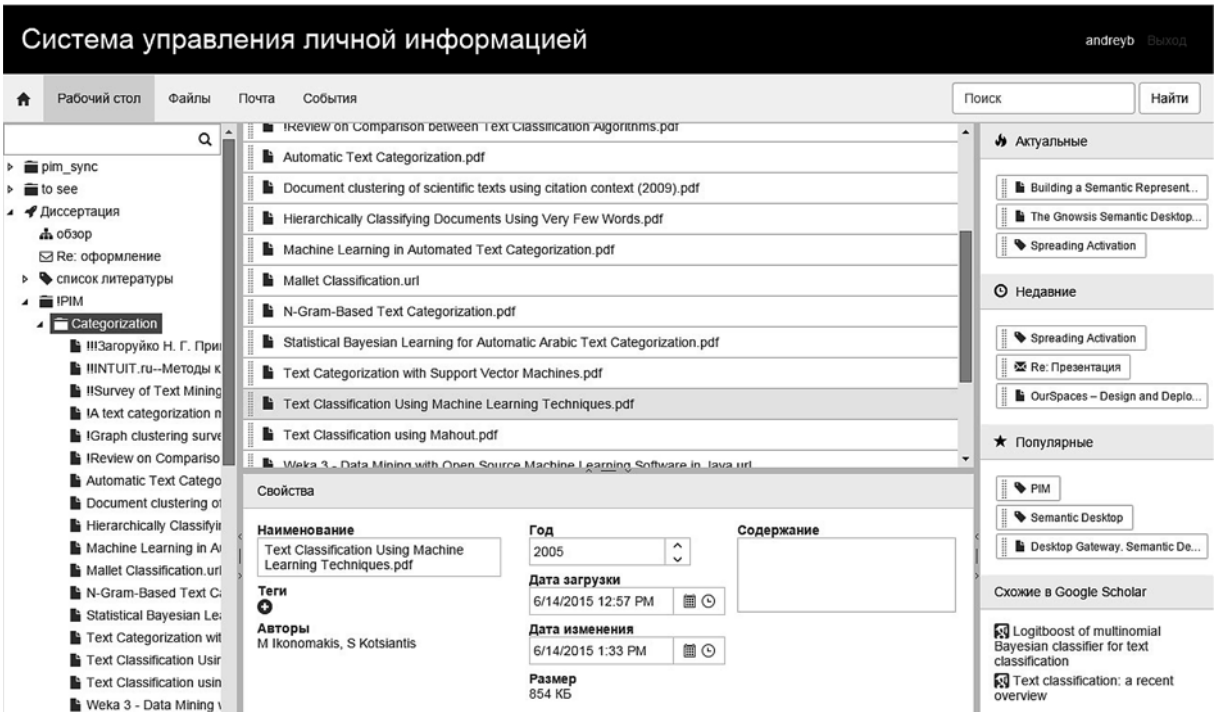


**Рис. 4. Иерархический интерфейс работы со знаниями**


**Рис. 5. Представление знаний в виде карт памяти**

ми связями. При выделении ресурса в центральной части рабочей области отображается его детальное представление, содержащее все атрибуты данного ресурса и ссылки на другие ресурсы, связанные с ним. В правой части выводятся ресурсы, представляющие текущий контекст работы пользователя.

Другим режимом работы со знаниями, который предоставляется системой, являются карты памяти. Пользователь может создать диаграмму, узлами которой являются ресурсы базы знаний, а ребрами — связи между ними.

Помимо описанных интерфейсов создания и редактирования знаний, в дальнейшем планируется реализовать интерфейсы аннотирования текстовых документов, просмотра журналов работы с ресурсами.

## Заключение

На основе библиографического исследования представлены основные требования, предъявляемые к системам управления личной информацией и знаниями. На основании их анализа спроектирована система MemoPIM, расширяющая прототип системы, описанный в работе [1]. Знания в системе хранятся в RDF-формате. Для формализации структуры RDF-данных спроектирована OWL-онтология, описывающая основные понятия и классы, используемые различными модулями системы. Для работы со знаниями предоставляются графический и иерархический интерфейсы. К функциям системы также относятся ведение контекста работы, автоматизированная категоризация данных, связывание ресурсов. По каждой функции проведен обзор существующих решений и способов реализации.

В рамках продолжения работы планируется развитие и апробация прототипа системы MemoPIM. К основным направлениям этой деятельности можно отнести предоставление пользователю возможностей расширения системной онтологии, поддержку интерфейсов создания новых классов и связей между ними. Кроме того, в перспективе пользователю может предоставляться возможность настройки интерфейсов представления ресурсов.

### Список литературы


1. **Бездушный А. А.** Концептуальные положения и архитектура системы семантического управления личной информацией // Программная инженерия. 2014. № 9. С. 30—37.
2. **Sauermann L. Grimnes G.A., Kiesel M.** et al. Semantic desktop 2.0: The gnowsis experience // The Semantic Web — ISWC 2006. Berlin: Heidelberg Springer, 2006. P. 887—900.
3. **Dong X. L., Halevy A.** A platform for personal information management and integration // In Proceedings of CIDR 2005. January 4—7, 2005. Asilomar, CA, USA. ACM SIGMOD, 2005. P. 119—130.
4. **Karger D. R., Bakshi K., Huynh D.** et al. Haystack: A General-Purpose Information Management Tool for End Users Based on Semistructured Data // In Proceedings of CIDR 2005. January 4—7, 2005. Asilomar, CA, USA. ACM SIGMOD 2005. P. 13—26.
5. **Gemmell J., Bell G., Lueder R.** et al. MyLifeBits: fulfilling the Memex vision // Proceedings of the tenth ACM international conference on Multimedia. December 1—6, 2002. France, Juan les Pins. ACM, 2002. P. 235—238.
6. **Chirita P. A., Costache S., Nejdl W.** et al. Beagle + +: Semantically enhanced searching and ranking on the desktop // The Semantic Web: Research and Applications. Berlin Heidelberg: Springer, 2006. P. 348—362.
7. **Chen J., Guo H., Wu W., Wang W.** iMecho: an associative memory based desktop search system // CIKM '09: Proceedings of the 18th ACM Conference on Information and Knowledge Management. November 02—06, 2009. Hong Kong, China. ACM, 2009. P. 731—740.
8. **Maurer H.** The heart of the problem: Knowledge management and knowledge transfer // Proceedings of ENABLE'99. June 2—5, 1999. Espoo, Finland. EVITech Digital Press, 1999. Vol. 99. P. 8—11.
9. **Völkel M., Haller H.** Conceptual data structures for personal knowledge management // Online Information Review. 2009. Vol. 33, N 2. P. 298—315.
10. **Buzan T.** Use both sides of your brain: new mind-mapping techniques. New York: Plume Books, 1991.
11. **Novak J. D., Cañas A. J.** The theory underlying concept maps and how to construct them. URL: http://cmap.ihmc.us/docs/theory-of-concept-maps (дата обращения 22.06.2015).
12. **Richter J., Völkel M., Haller H.** DeepaMehta — A Semantic Desktop // In Proceedings of the Semantic Desktop Workshop at ISWC. Galway, Ireland. November 6, 2005. URL: http://www.deepamehta.de/sites/default/files/deepamehta-paper-iswc2005.pdf (дата обращения 22.06.2015).
13. **Haller H., Abecker A.** iMapping — A Zooming User Interface Approach for Personal and Semantic Knowledge Management // Proceedings of the 21st ACM conference on Hypertext and hypermedia. June 13—16, 2010. Toronto, ON, Canada, ACM, 2010. P. 119—128.
14. **Beel J., Langer S., Genzmehr M., Gipp B.** Utilizing Mind-Maps for Information Retrieval and User Modelling // User Modeling, Adaptation, and Personalization. Lecture Notes in Computer Science. 2014. Vol. 8538. P. 301—313.
15. **Trullemans S., Signer B.** Towards a Conceptual Framework and Metamodel for Context-aware Personal Cross-Media Information Management Systems // Conceptual Modeling. Lecture Notes in Computer Science. 2014. Vol. 8824. P. 313—320.
16. **Katifori A., Vassilakis C., Dix A.** Ontologies and the Brain: Using Spreading Activation through Ontologies to Support Personal Interaction // Cognitive Systems Research. 2010. Vol. 11, N 1. P. 25—41.
17. **Oren E., Völkel M., Breslin J. G., Decker S.** Semantic wikis for personal knowledge management // Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA 2006), Krakow, Poland. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2006. Vol. 4080. P. 509—518.
18. **Schaffert S.** IkeWiki. A Semantic Wiki for Collaborative Knowledge Management // Proceedings of the Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. UK, Manchester: IEEE, 2006. P. 388—396.
19. **Бездушный А. А.** Организация и управление личными каталогами научных публикаций с использованием технологий Semantic Web // XV Российская конференция с участием иностранных ученых "Распределенные информационные и вычислительные ресурсы" — DICR-2014. 02.11—05.11.2014 Новосибирск. Материалы конференции. Новосибирск: ИВТ СО РАН, 2014. URL: http://konf.ict.nsc.ru/files/conferences/dicr2014/fulltext/248753/248755/Bezdushny_DICR2014.pdf (дата обращения 22.06.2015).
20. **Devaurs D., Rath A. S., Lindstaedt S. N.** Exploiting the user interaction context for automatic task detection // Applied Artificial Intelligence. 2012. Vol. 26, N 1. P. 58—80.
21. **Křemen P., Mička P., Blaško M.** Ontology-driven mindmapping // Proceedings of the 8th International Conference on Semantic Systems. September 05—07 2012, Austria, Graz. ACM, 2012. P. 125—132.
22. **Collins A. M., Loftus E. F.** A spreading-activation theory of semantic processing // Psychological review. 1975. Vol. 82, N 6. P. 407.
23. **Crestani F.** Application of spreading activation techniques in information retrieval // Artificial Intelligence Review. 1997. Vol. 11, N 6. P. 453—482.
24. **Jäschke R., Marinho L., Hotho A.** et al. Tag recommendations in social bookmarking systems // Ai Communications. 2008. Vol. 21, N 4. P. 231—247.
25. **Gemmell J., Schimoler T., Ramezani M., Mobasher B.** Adapting K-Nearest Neighbor for Tag Recommendation in

Folksonomies // Intelligent Techniques for Web Personalization (ITWP). 2009. Vol. 528. P. 51—62.
26. **Menezes G. V., Almeida J. M., Belém F.** et al. Demand-driven tag recommendation // Machine Learning and Knowledge Discovery in Databases. Berlin Heidelberg: Springer, 2010. P. 402—417.
27. **Lu Y. T., Yu S. I., Chang T. C., Hsu J. Y. J.** A Content-Based Method to Enhance Tag Recommendation // International Joint Conference on Artificial Intelligence (IJCAI). AAAI Press, 2009. Vol. 9. P. 2064—2069.
28. **Sood S. C., Owsley S. H., Hammond K. J., Birnbaum L.** TagAssist: Automatic Tag Suggestion for Blog Posts // Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007). USA, Boulder, Colorado, 2007. P. 26—28.
29. **Landia N.** Content-awareness and graph-based ranking for tag recommendation in folksonomies: дис. ... University of Warwick. 2013.
30. **Landia N.** Utilising document content for tag recommendation in folksonomies // Proceedings of the sixth ACM conference on Recommender systems. September 09—13, 2012. Dublin, Ireland. ACM, 2012. P. 325—328.
31. **Guan Z., Bu J., Mei Q.** et al. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects // Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. July 19—23, 2009. Boston, MA, USA, ACM, 2009. P. 540—547.
32. **Lipczak M., Hu Y., Kollet Y., Milios E.** Tag sources for recommendation in collaborative tagging systems // Proceedings of the ECML-PKDD discovery challenge workshop. Bled, Slovenia: Springer, 2009. P. 157—172.
33. **Wang J., Hong L., Davison B. D.** Tag recommendation using keywords and association rules // Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining. Bled, Slovenia: Springer, 2009. P. 261—274.
34. **Lee S. O. K., Chun A. H. W.** Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ANN semantic structures // Proceedings of the 6th Conference on WSEAS International Conference on Applied Computer Science (ACOS'07). Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society, 2007. Vol. 7. P. 88—93.
35. **Romary P. L. L.** Automatic Key Term Extraction from Scientific Articles in GROBID. // SemEval 2010 Workshop. Uppsala, Sweden, Jul 2010. P. 4.
36. **Litvak M., Last M.** Graph-based keyword extraction for single-document summarization // Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization. Manchester, UK: Association for Computational Linguistics, 2008. P. 17—24.
37. **Nunes B. P., Fetahu B., Casanova M. A.** Cite4Me: Semantic Retrieval and Analysis of Scientific Publications // In Proceedings of the LAK Data Challenge, International Learning Analytics & Knowledge Conference. April 08—12, 2013. Leuven, Belgium. ACM, 2013. P. 10—15.
38. **Хоай Л., Тузовский А. Ф.** Разработка семантических электронных библиотек на основе онтологических моделей // Сборник докладов XV Всероссийской научной конференции RCDL. Ярославль, 2013. С. 291—299.
39. **Pirro G.** Reword: Semantic relatedness in the web of data // Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence. Toronto, Ontario, Canada: AAAI Press, 2012. P. 129—135.

**A. A. Bezdushny**, Postgraduate Student, e-mail: andrey.bezdushny@gmail.com,
Moscow Institute of Physics and Technology

# MemoPIM: Personal Information and Knowledge Management with Semantic Technologies

*In the course of daily activities, one is confronted with increasingly volume of information, a large part of which is stored in digital format. In this paper the MemoPIM system is presented. MemoPIM supports management of personal information and personal knowledge and develops ideas of the Semantic Desktop — an approach to the personal information space organization in accordance with principles of Semantic Web and Linked Open Data.*

*Knowledge and information is stored in RDF repository, based on OWL ontology. Ontology describes classes from different domains, such as file system ontology, publication ontology, calendar ontology etc. Knowledge can be imported from external sources or created manually by user. System provides adapters for following sources: file system, email, calendar, browser bookmarks. Imported information is transformed to RDF in accordance with system ontology. In some cases additional metadata is created, for example author, annotation and title is extracted from scientific articles.*

*During user work, actions performed by him are logged to knowledge base. These logs are later used for several tasks such as context inference, knowledge base analysis and tag recommendation. Context inference is based on spreading activation algorithm over RDF graph. Activation values for RDF nodes are recalculated after each user action. To create additional relations in RDF data, knowledge base analysis is performed. Another service provided by system is automatic categorization of resources. Categorization is based on tag recommendation algorithm.*

*System provides two types of user interfaces for knowledge management — based on hierarchical presentation of data and based on mind maps.*

***Keywords:*** *Personal Information Management, Personal Knowledge Management, Semantic Desktop, Semantic Web, RDF, Linked Open Data, Spreading activation, Tag recommendation, RDF similarity measures*

### References

1. **Bezdushny A. A.** Kontseptual'nye polozheniya i arkhitektura sistemy semanticheskogo upravleniya lichnoi informatsiei (Conceptual provisions and architecture of semantic personal information management system). *Programmnaya ingeneria*, 2014, vol. 9, pp. 30—37 (in Russian).
2. **Sauermann L. Grimnes G. A., Kiesel M., Fluit C., Maus H., Heim D., Nadeem D., Horak B., Dengel A.** Semantic desktop 2.0: The gnowsis experience. *The Semantic Web — ISWC 2006*. Berlin Heidelber, Springer, 2006, pp. 887—900.
3. **Dong X. L., Halevy A.** A platform for personal information management and integration. *In Proceedings of CIDR 2005*. January 4—7, 2005. Asilomar, CA, USA. ACM SIGMOD, 2005, pp. 119—130.

4. **Karger D. R., Bakshi K., Huynh D., Quan D., Sinha V.** Haystack: A General-Purpose Information Management Tool for End Users Based on Semistructured Data. *In Proceedings of CIDR 2005*. January 4—7, 2005. Asilomar, CA, USA. ACM SIGMOD, 2005, pp. 13—26.

5. **Gemmell J., Bell G., Lueder R., Drucker S., Wong C.** MyLifeBits: fulfilling the Memex vision. *Proceedings of the tenth ACM international conference on Multimedia*. December 1—6, 2002. France, Juan les Pins. ACM, 2002, pp. 235—238.

6. **Chirita P. A., Costache S., Nejdl W., Paiu R.** Beagle++: Semantically enhanced searching and ranking on the desktop. *The Semantic Web: Research and Applications*. Berlin Heidelberg, Springer, 2006, pp. 348—362.

7. **Chen J., Guo H., Wu W., Wang W.** iMecho: an associative memory based desktop search system. *CIKM '09: Proceedings of the 18th ACM Conference on Information and Knowledge Management*. November 02—06, 2009. China, Hong Kong. ACM, 2009, pp. 731—740.

8. **Maurer H.** The heart of the problem: Knowledge management and knowledge transfer. *Proceedings of ENABLE'99*. June 2—5, 1999. Espoo, Finland. EVITech Digital Press, 1999, vol. 99, pp. 8—11.

9. **Volkel M., Haller H.** Conceptual data structures for personal knowledge management. *Online Information Review*, 2009, vol. 33, no. 2, pp. 298—315.

10. **Buzan T.** *Use both sides of your brain: new mind-mapping techniques*. New York: Plume Books, 1991.

11. **Novak J. D., Canas A. J.** *The theory underlying concept maps and how to construct them*, available at: http://cmap.ihmc.us/docs/theory-of-concept-maps (date of access 22.06.2015).

12. **Richter J., Völkel M., Haller H.** DeepaMehta — A Semantic Desktop. In *Proceedings of the Semantic Desktop Workshop at ISWC*. Galway, Ireland, November 6, 2005, available at: http://www.deepamehta.de/sites/default/files/deepamehta-paper-iswc2005.pdf (date of access 22.06.2015).

13. **Haller H., Abecker A.** iMapping — A Zooming User Interface Approach for Personal and Semantic Knowledge Management. *Proceedings of the 21st ACM conference on Hypertext and hypermedia*. June 13—16, 2010, Toronto, ON, Canada. ACM, 2010, pp. 119—128.

14. **Beel J., Langer S., Genzmehr M., Gipp B.** Utilizing Mind-Maps for Information Retrieval and User Modelling. *User Modeling, Adaptation, and Personaliz*ation*, Lecture Notes in Computer Science,* 2014, vol. 8538, pp. 301—313.

15. **Trullemans S., Signer B.** Towards a Conceptual Framework and Metamodel for Context-aware Personal Cross-Media Information Management Systems. *Conceptual Modeling, Lecture Notes in Computer Science*, 2014, vol. 8824, pp. 313—320.

16. **Katifori A., Vassilakis C., Dix A.** Ontologies and the Brain: Using Spreading Activation through Ontologies to Support Personal Interaction. *Cognitive Systems Research*, 2010, vol. 11, no. 1, pp. 25—41.

17. **Oren E., Vokel M., Breslin J. G., Decker S.** Semantic wikis for personal knowledge management. *Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA 2006),* Krakow, Poland. Lecture Notes in Computer Science. Berlin Heidelberg, Springer, 2006, vol. 4080, pp. 509—518.

18. **Schaffert S.** IkeWiki. A Semantic Wiki for Collaborative Knowledge Management. *Proceedings of the Enabling Technologies: Infrastructure for Collaborative Enterprises*. Manchester, UK, IEEE, 2006, pp. 388—396.

19. **Bezdushny A. A.** Organizaciya i upravlenie lichnymi katalogami nauchnyh publikacij s ispol'zovaniem tekhnologij Semantic Web (Organization and management of personal publication directories with use of Semantic Web Technologies). *XIV Russian Conference with international participation "Distributed information and computing resources" (DICR-2014)*. Novosibirsk, 2014, available at: http://konf.ict.nsc.ru/files/conferences/dicr2014/fulltext/248753/248755/Bezdushny_DICR2014.pdf. (in Russian).

20. **Devaurs D., Rath A. S., Lindstaedt S. N.** Exploiting the user interaction context for automatic task detection. *Applied Artificial Intelligence*, 2012, vol. 26, no. 1, pp. 58—80.

21. **Kremen P., Micka P., Blasko M.** Ontology-driven mindmapping. *Proceedings of the 8th International Conference on Semantic Systems*. September 05—07, 2012, Graz, Austria, ACM, 2012, pp. 125—132.

22. **Collins A. M., Loftus E. F.** A spreading-activation theory of semantic processing. *Psychological review,* 1975, vol. 82, no. 6, pp. 407.

23. **Crestani F.** Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 1997, vol. 11, no. 6, pp. 453—482.

24. **Jaschke R., Marinho, L., Hotho A., Schmidt-Thieme L., Stumme G.** Tag recommendations in social bookmarking systems. *Ai Communications*, 2008. vol. 21, no. 4, pp. 231—247.

25. **Gemmell J., Schimoler T., Ramezani M., Mobasher B.** Adapting K-Nearest Neighbor for Tag Recommendation in Folksonomies. *Intelligent Techniques for Web Personalization — ITWP*, 2009. vol. 528, pp. 51—62.

26. **Menezes G. V., Almeida J. M., Belem F., Goncalves M. A., Lacerda A., De Moura E. S., Pappa G., Veloso A., Ziviani N.** Demand-driven tag recommendation. *Machine Learning and Knowledge Discovery in Databases*, Berlin Heidelberg: Springer, 2010, pp. 402—417.

27. **Lu Y. T., Yu S. I., Chang T. C., Hsu J. Y. J.** A Content-Based Method to Enhance Tag Recommendation. *International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2009, vol. 9, pp. 2064—2069.

28. **Sood S., Owsley S., Hammond K. J., Birnbaum L.** TagAssist: Automatic Tag Suggestion for Blog Posts. *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*. Colorado, USA, Boulder, 2007, pp. 26—28.

29. **Landia N.** Content-awareness and graph-based ranking for tag recommendation in folksonomies. Diss. ...University of Warwick, 2013.

30. **Landia N.** Utilising document content for tag recommendation in folksonomies. *Proceedings of the sixth ACM conference on Recommender systems*. September 09—13, 2012. Dublin, Ireland. ACM, 2012, pp. 325—328.

31. **Guan Z., Bu J., Mei Q., Chen C., Wang C.** Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. July 19—23, 2009. Boston, MA, USA. ACM, 2009, pp. 540—547.

32. **Lipczak M., Hu Y., Kollet Y., Milios E.** Tag sources for recommendation in collaborative tagging systems. *Proceedings of the ECML-PKDD discovery challenge workshop*. Bled, Slovenia: Springer, 2009, pp. 157—172.

33. **Wang J., Hong L., Davison B. D.** Tag recommendation using keywords and association rules. *Proceedings of ECML PKDD (The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases) Discovery Challenge*. Bled, Slovenia: Springer, 2009, pp. 261—274.

34. **Lee S. O. K., Chun A. H. W**. Automatic tag recommendation for the web 2.0 blogosphere using collaborative tagging and hybrid ANN semantic structures. *Proceedings of the 6th Conference on WSEAS International Conference on Applied Computer Science (ACOS'07)*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society, 2007, vol. 7, pp. 88—93.

35. **Romary P. L. L.** Automatic Key Term Extraction from Scientific Articles in GROBID. *SemEval 2010 Workshop*. Uppsala, Sweden, Jul 2010, pp. 4.

36. **Litvak M., Last M.** Graph-based keyword extraction for single-document summarization. *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*. Manchester, UK, Association for Computational Linguistics, 2008, pp. 17—24.

37. **Nunes B. P., Fetahu B., Casanova M. A**. Cite4Me: Semantic Retrieval and Analysis of Scientific Publications. *Proceedings of the LAK Data Challenge, International Learning Analytics & Knowledge Conference*. April 08—12, 2013. Leuven, Belgium. ACM, 2013, pp. 10—15.

38. **Hoai L., Tuzovsky A. F.** Razrabotka semanticheskih ehlektronnyh bibliotek na osnove ontologicheskih modelej (Development of Semantic Digital Libraries on the Basis of the Ontological Models). *Proceedings of the RCDL 2013. Digital Libraries: Advanced Methods and Technologies*. Jaroslavl', 2013, pp. 291—299 (In Russian).

39. **Pirro G.** Reword: Semantic relatedness in the web of data. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. Toronto, Ontario, Canada, AAAI Press, 2012, pp. 129—135.

УДК 004.89

**В. П. Тельнов**, канд. техн. наук, доц., e-mail: telnov@bk.ru,
**А. В. Мышев**, канд. физ.-мат. наук, доц., e-mail: mishev@iate.obninsk.ru,
Обнинский институт атомной энергетики НИЯУ "МИФИ"

# Семантическая паутина и поисковые агенты для высшей школы

*Статья посвящена вопросам внедрения технологий семантического веба в образовательные порталы высшей школы. Рассмотрены поисковые агенты, работающие с международными базами знаний, а также адаптивная технология контекстного поиска во всемирной паутине, когда первичное извлечение контента выполняется известными поисковыми машинами. Представлены архитектурные, технологические и проектные решения в нотации диаграмм компонентов и диаграмм последовательности.*

***Ключевые слова:*** *информационные технологии, семантическая паутина, база знаний, поисковый агент, высшее образование*

## Введение

В Российской Федерации на семантический веб как на информационно-образовательный ресурс впервые обратили серьезное внимание в 2001—2002 гг. [1, 2]. В те годы начали предлагать и рассматривать футуристические концепции семантических образовательных порталов, которые по замыслу их авторов были призваны обеспечить глубокую персонализацию образовательных услуг.

В упомянутых концепциях речь шла о том, что образовательные услуги должны максимально полно и точно соответствовать потребностям, уровню подготовки и когнитивным особенностям каждого конкретного учащегося. Было справедливо подмечено, что традиционные веб-технологии (в настоящее время их иногда обозначают как WEB 1.0 и WEB 2.0) не обеспечивают поиск и навигацию в среде распределенных знаний на семантическом уровне. Предполагалось, что задача может быть решена на базе технологий семантического веба путем применения так называемых интеллектуальных агентов, которые для извлечения знаний будут использовать общепризнанные онтологии и правила вывода [2]. Под агентами понимались программные компоненты, которые могут работать без непосредственного управления со стороны человека для достижения поставленных перед ними целей. Типичной является ситуация, когда агенты собирают, фильтруют и обрабатывают информацию, найденную в глобальной сети и в специальных базах данных, иногда путем взаимодействия с другими агентами.

Предлагаемый подход к персонализированному обучению основывался на выделении двух типов сущностей: концептов (фрагментов знаний) и собственно учебных объектов (произвольных веб-ресурсов). Процесс обслуживания образовательного запроса в общем случае должен был, по замыслу авторов, включать два этапа.

На первом этапе предполагалось определить образовательные потребности клиента. Результатом этапа должна являться некая индивидуальная программа обучения, построенная из концептов (классов) онтологии некоторой конкретной предметной области, которая на текущий момент времени принята большинством участников соответствующего профессионального сообщества.

На втором этапе должно выполняться "покрытие" программы обучения, составленной из концептов, доступными в образовательном пространстве учебными объектами. Поскольку в открытой образовательной среде доступно большое число учебных объектов, для каждого концепта найдется множество вариантов "покрытия". При этом все используемые образовательные ресурсы (учебные объекты) должны быть аннотированы с использованием стандартизованных словарей метаданных [3]. Предполагалось, что в построении индивидуальной программы обучения будут задействованы интеллектуальные агенты нескольких типов, например, такая комбинация: агент учащегося, агент — построитель программ обучения и поисковый агент.

Что касается вопросов практического программирования, интеллектуальные агенты предполагалось создавать в основном на технологической платформе JAVA Agent DEvelopment Framework (JADE) [4]. Выбор инструментария JAVA был обусловлен широким распространением данного языка программирования как универсального средства создания машинно-независимых сетевых приложений. Это было время относительной молодости семантического веба, и появление продвинутых интеллектуальных образовательных порталов казалось делом недалекого будущего.

## Современная российская практика

За почти полтора десятка лет, прошедших с момента первой публикации на тему семантического веба, разработан ряд стандартов, рекомендаций и реализовано множество проектов [5—7]. Однако, несмотря на очевидные успехи, до сих пор (и это признает Т. Бернерс-Ли [8]) нельзя сказать, что основные идеи семантического веба уже в достаточной мере реализованы на практике. Данное соображение представляется особенно верным в отношении российских образовательных порталов.

Посетив несколько десятков официальных сайтов ведущих российских вузов, можно убедиться в том, что доминирующей платформой для их образователь-

ных порталов на сегодняшний день являются специализированные системы управления курсами (CMS), также известные как системы управления обучением (LMS) или виртуальные обучающие среды (VLE). Система CMS Moodle [9], как свободно распространяемый программный продукт с открытым исходным кодом, пользуется наибольшей популярностью. Очевидно, что подобные программные решения слабо связаны с идеями семантических образовательных порталов.

В 2011 г. Минобрнауки России определило победителя конкурса по теме "Создание публичного ресурса открытых данных в области науки и техники, интегрированного в единое международное пространство знаний Linked Open Data" [10, 11]. Попытки обнаружить в рунете следы долгожданного публичного ресурса пока остаются безуспешными. Приходится согласиться с тем, что сейчас в России производится крайне мало взаимосвязанных открытых данных. Основными источниками данных для российских пользователей семантического веба по-прежнему остаются международные базы знаний, включающие русскоязычный контент, прежде всего "DBpedia" [5], "Wikidata" [6] и "Freebase" [7]. Это главным образом справочные и библиографические данные, информация из области мультимедиа и прочие сведения энциклопедического характера.

Необходимо также упомянуть работы преподавателей российских университетов, где в той или иной мере задействованы онтологии и другие технологии семантического веба, например работы [12, 13]. Вместе с тем общедоступная рабочая версия семантического образовательного веб-портала пока наблюдается лишь в одной из стран ближнего зарубежья (см. работы [14, 15]).

## Портал "Кафедра онлайн"

Предметом рассмотрения настоящей статьи является пилотный проект, связанный с применением технологий семантического веба и поисковых агентов в вузах РФ. Проект реализуется как часть более общего образовательного портала "Кафедра онлайн" [16, 17]. Условное наименование компонента — "Семантическая паутина" (рис. 1). Доступ к компоненту "Семантическая паутина" возможен через главную страницу образовательного портала "Кафедра онлайн" по сетевому адресу http://ksst.obninsk.ru или по прямой ссылке http://ksst.obninsk.ru/semantic.

## Поисковый агент "DBpedia"

Широко известный в мире семантического веба проект "DBpedia" [5, 18] представляет собой результат совокупных усилий профессионального сообщества по извлечению частично структурированных данных, представленных в Википедии, в их связи с другими открытыми базами знаний и последующей публикации извлеченной информации в формате RDF (**Resource Description Framework**) во всемирной паутине для ее свободного использования. Система, созданная в рамках проекта "DBpedia", позволяет принимать и обслуживать сложные запросы на языке SPARQL [19] к имеющейся базе знаний через публичные точки доступа.

На настоящее время проект "DBpedia" представляет собой одну из крупнейших в мире баз знаний, которая обеспечивает свободный доступ к более чем 280 млн элементов данных и 2,6 млн концепций. Проект охватывает широкий спектр областей знаний и имеет высокую степень концептуального перекрытия с иными


**Рис. 1. Образовательный портал "Кафедра онлайн"**

открытыми наборами данных. Развивающаяся онтология "DBpedia" выступает в качестве своего рода структурного каркаса для организации этих данных.

Одно из преимуществ "DBpedia" заключается в том, что данный проект обеспечивает семантические отношения, соединяющие различные домены специализированных предметно-ориентированных наборов данных. Система "DBpedia" имеет около 5 млн исходящих ссылок RDF, что делает ее хорошей отправной точкой для семантических поисковых роботов. В свою очередь, несколько десятков крупных внешних источников связанных данных устанавливают RDF-ссылки на "DBpedia".

Чтобы воспользоваться потенциалом мировых источников связанных данных, отечественным университетам необходимы общедоступные технологичные способы организации запросов к этим данным. Публичные точки входа SPARQL призваны удовлетворить потребность учащихся в связанных данных, но количество и доступность таких точек в настоящее время оставляют желать лучшего.

Поисковый агент "DBpedia" интегрирован в компонент "Семантическая паутина" образовательного портала "Кафедра онлайн" [17]. Данный поисковый агент адаптирован для использования в учебной деятельности теми студентами, которые специализируются в информатике и компьютерных дисциплинах. Для этой категории учащихся создание запросов на языке SPARQL [19] не должно представлять принципиальных трудностей, поскольку в программе их обучения присутствуют дисциплины, связанные с изучением баз данных и языка SQL.

Всемирная паутина наполнена связанными данными, однако открытыми остаются вопросы создания удобных технологичных способов их извлечения. В поисковом агенте "DBpedia" для доступа к связанным данным применяется технология Linked Data Fragments (LDF) [20]. Такой подход позволяет выполнить "расщепление" сложного SPARQL-запроса на RDF-триплеты на промежуточном LDF-сервере и тем самым снизить нагрузку на серверы целевых баз знаний [21].

Укрупненная диаграмма компонентов для поискового агента "DBpedia" показана на рис. 2 (см. четвертую сторону обложки). Данная UML-диаграмма и все последующие диаграммы выполнены в соответствии с международным стандартом ISO/IEC 19505-2:2012 Information technology (UML) [22] с помощью программного обе-

спечения Visual Paradigm for UML [23]. Диаграмма компонентов (*Component Diagram*) на рис. 2 — это диаграмма физического уровня, которая служит для представления программных компонентов и зависимостей между ними. Диаграмма компонентов обеспечивает визуализацию общей структуры программного обеспечения и спецификацию исполняемого кода (единиц развертывания).

## Поисковый агент "Wikidata"

Проект "Wikidata" [6], запущенный в 2012 г. фондом Wikimedia Foundation, предлагает базу знаний, построенную на основе Википедии, для гибридного использования компьютерами и людьми. При этом вносить знания в нее могут и пользователи, и машины. Одно из главных отличий проекта "Wikidata" от более раннего проекта "DBpedia" состоит в том, что в последнем клиенты-машины не могли вносить правки в семантическую базу данных. Если в проекте "DBpedia" основой структурирования связанных данных являются RDF-триплеты (субъект, предикат, объект), то в проекте "Wikidata" этим единицам примерно соответствует следующая тройка: элемент (*item*), свойство (*property*) и значение (*value*).

По состоянию на 2014 г. централизованный репозиторий проекта "Wikidata" имел следующие метрики: около 15 млн статей (концепций) и более 30 млн утверждений, в том числе 20 млн ссылок, 8 млн строковых данных, более 1 млн хронологических дат и 1 млн географических координат объектов. В 2015 г. после миграции коллаборативной базы знаний Freebase на платформу "Wikidata" (анонсировано компанией Google) можно ожидать значительного увеличения данных показателей.

Необходимо отметить, что работа с базой знаний "Wikidata" требует от пользователей определенного понимания принципов организации Википедии, а также некоторых технических навыков. Это связано в первую очередь с тем, что программное обеспечение данного проекта ("Wikibase") по существу представляет собой расширение основного движка Википедии ("MediaWiki").

Поисковый агент "Wikidata" для извлечения связанных данных использует готовый инструмент AutoList из проекта "Wikidata", который запускается в дочернем окне и затем асинхронно взаимодействует с выделенным сервером для отработки поисковых запросов к базе знаний. Для подготовки собственно поисковых запросов предлагается визуальный редактор "Wikidata Query Editor", который спроектирован и реализован как независимый компонент программного обеспечения.

В рамках обсуждаемого пилотного проекта оба поисковых агента — "DBpedia" и "Wikidata" — имеют схожий пользовательский интерфейс, выполненный в единой стилистике (см. http://ksst.obninsk.ru/semantic). Однако внутренняя архитектура этих поисковых агентов и логика их работы различаются в значительной степени. На рис. 3 (см. четвертую сторону обложки) показана укрупненная диаграмма компонентов, соответствующая поисковому агенту "Wikidata".

Два взаимодополняющих поисковых агента — "DBpedia" и "Wikidata" — включены в состав компонента "Семантическая паутина" в целях их апробации и сопоставления возможностей в ходе бета-тестирования, которое выполняется силами студентов и преподавателей. Представляется, что достигнутый к настоящему времени уровень развития технологий семантического веба пока не позволяет отдать предпочтение ни одному из программных продуктов или баз знаний.

## Поисковый агент "AjaxSearch"

Современные реалии высшей школы в России таковы, что подавляющее число студентов и преподавателей не подозревают о существовании семантического веба и вообще связанных открытых данных. Они продолжают практиковать поиск информации во всемирной паутине по ключевым словам, используя для этого публичные поисковые машины. Немалую роль здесь играют традиции, а также простота и высокая скорость формирования поискового запроса в сравнении с поисковыми запросами к семантической паутине. В интересах этой обширной категории потребителей сетевой информации разработан поисковый агент "AjaxSearch", пример пользовательского интерфейса которого представлен по адресу http://ksst.obninsk.ru/semantic.

В связи с изложенным выше уместен вопрос: "Зачем нужен еще один поисковый агент, если для поиска по ключевым словам имеются общедоступные Google и Яндекс?" Здесь целесообразно отметить несколько представленных далее характерных особенностей публичных поисковых машин, о которых Google с Яндексом обычно умалчивают, но которые хорошо известны их пользователям.

- Чрезмерное число результатов поиска, не соответствующих запросу. Пользователь тратит много времени и сил, просматривая первые страницы результатов, и ему не всегда хватает терпения перейти к оставшимся ссылкам.
- Найденные документы ранжируются поисковой машиной в соответствии с ее внутренним алгоритмом, который не всегда отвечает интересам конкретного пользователя.
- Ссылки на "раскрученные" коммерческие сайты имеют больший рейтинг по сравнению с прочими результатами поиска.
- Пользователям не всегда удобно управлять контекстом поискового запроса, уточнять и направлять поиск.

При работе поискового агента "AjaxSearch" (рис. 4, см. четвертую сторону обложки) глобальный поиск документов, так же как и поиск по конкретным специализированным ресурсам, изначально выполняется штатными поисковыми машинами Google Ajax Search, Yahoo и Яндекс, взаимодействие с которыми по сети происходит асинхронно. Найденный контент является своего рода "сырьем" для дальнейшей обработки.

Иначе реализован поиск документов, с которыми имеет дело компонент "Облачный кабинет" портала "Кафедра онлайн". Число документов в "Облачном кабинете" может исчисляться многими тысячами единиц. Эти документы расположены в удаленных хранилищах данных где угодно в глобальной сети, поэтому не всегда индексируются публичными поисковыми машинами. Доступ ко многим документам изначально ограничен (например, только для студентов, для преподавателей, для проектных групп). Навигация по таким документам возможна через деревья "Облачного кабинета" или через поисковый агент "AjaxSearch". Фактическая работа при этом выполняется с использованием поисковой машины Dojo Object Storage [24].

В агенте "AjaxSearch" результаты работы упомянутых выше поисковых машин всегда проходят дополнительную обработку, селекцию и сортировку, прежде чем они появляются на мониторе. Укрупненная диаграмма компонентов для поискового агента "AjaxSearch" показана на рис. 4.

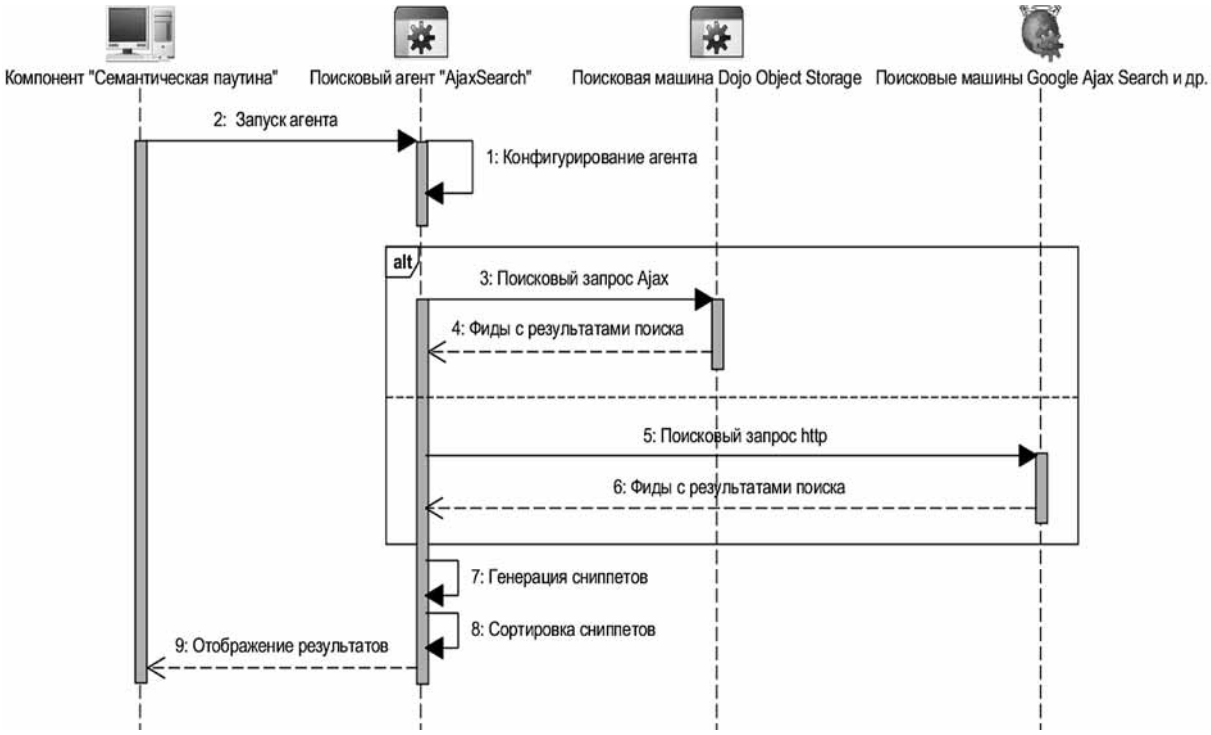


**Рис. 5. Укрупненная диаграмма последовательности для агента "AjaxSearch"**

Под релевантностью документа понимается мера его соответствия поисковому запросу. Наиболее релевантные документы (точнее, представляющие их образы — сниппеты поисковой системы [25]) поднимаются наверх списка результатов поиска. Также наверх списка результатов поиска "всплывают" те сниппеты, которые в наибольшей мере соответствуют контексту поискового запроса, т. е. обладают наибольшей пертинентностью.

В поисковом агенте "AjaxSearch" контекст поискового запроса изначально весьма широк и охватывает все дисциплины, которые преподаются на данной кафедре. Номенклатура дисциплин кафедры с соответствующими таксономиями хранится в деревьях "Облачного кабинета" портала "Кафедра онлайн". Контекст поискового запроса можно сузить, ограничив его конкретными группами преподаваемых дисциплин или направлениями подготовки студентов. Наконец, контекст возможно задать строго индивидуально, в форме отдельного текстового документа или специализированного глоссария. Для оценки меры соответствия сгенерированных сниппетов контексту поискового запроса (т. е. их пертинентности) поисковый агент "AjaxSearch" вычисляет метрики Левенштейна [26]. Возможна тонкая настройка так называемых цен отдельных операций в метриках Левенштейна (вставка, удаление, замена символов).

Укрупненная диаграмма последовательности для поискового агента "AjaxSearch" показана на рис. 5. Диаграмма последовательности (*Sequence Diagram*) служит для представления взаимодействия между компонентами программного обеспечения (объектами, другими сущностями) в форме последовательности сообщений и соответствующих им событий на линиях жизни сущностей. В известном смысле диаграмма последовательности отражает логику взаимодействия компонентов. Представленный на рис. 5 комбинированный фрагмент "Альтернатива" (*alt*) демонстрирует выбор необходимой поисковой машины на линии жизни компонента "AjaxSearch".

## Заключение

Статья не претендует на всестороннее освещение заявленной темы, но представляет один из вариантов практического решения задачи. Осознанно выбран облегченный стиль изложения материала, свободный от громоздких выкладок и формальных построений, которые могли бы отпугнуть малоискушенных неофитов семантического веба из среды учащихся. Вместе с тем программный продукт "Семантическая паутина" призван посеять в головах студентов и других продвинутых пользователей сети Интернет зерна сомнения относительно того, правильно ли они ищут и получают информацию в глобальной сети.

Правомерен критический взгляд на перспективы использования семантического веба в высшей школе. Общеизвестен пример одного из лидеров открытого интернет-образования — проект MIT OpenCourseWare [27] Массачусетского технологического института по публикации в свободном доступе материалов всех курсов института. Инициатива MIT создала прецедент, которому последовали многие другие крупные университеты. В случае с проектом MIT OpenCourseWare и аналогичных ему апологеты семантических технологий будут вынуждены констатировать, что их идеи пока еще далеки от полномасштабного внедрения в реальную образовательную деятельность ведущих мировых университетов.

Что касается перспектив конкретного пилотного проекта "Семантическая паутина", цель которого — подвигнуть учащихся к знакомству с миром семантического веба через использование поисковых агентов, то здесь здоровый скептицизм также представляется оправданным. Господ-

ствующая в настоящее время в России тенденция к упрощению высшего образования оставляет семантическим интернет-технологиям мало надежд. Исключением из данной ситуации могли бы стать студенты (магистранты) вузов, которые специализируются в информатике и компьютерных дисциплинах.

### Список литературы


1. **Пантелеев М. Г., Пузанков Д. В., Татаринов Ю. С.** Перспективы использования технологий семантического Web в образовательных порталах. URL: http://window.edu.ru/resource/126/49126/files/portal3-18.pdf (дата обращения 17.05.2015).
2. **Татаринов Ю. С., Пантелеев М. Г., Сазыкин П. В.** Мультиагентный сервис построения персонализированных программ обучения для семантических образовательных порталов. URL: http://old.lvk.cs.msu.su/files/mco2005/tatarinov.pdf (дата обращения 17.05.2015).
3. **IEEE** 1484.12.1—2002. Learning Object Metadata standard. URL: http://en.wikipedia.org/wiki/Learning_object_metadata (дата обращения 17.05.2015).
4. **JAVA** Agent DEvelopment Framework. URL: http://jade.tilab.com/ (дата обращения 17.05.2015).
5. **DBpedia**. URL: https://ru.wikipedia.org/wiki/DBpedia (дата обращения 17.05.2015).
6. **Wikidata**. URL: http://www.wikidata.org (дата обращения 17.05.2015).
7. **Freebase**. URL: https://ru.wikipedia.org/wiki/Freebase (дата обращения 17.05.2015).
8. **Berners-Lee T.** Linked Data. URL: http://www.w3.org/DesignIssues/LinkedData.html (дата обращения 17.05.2015).
9. **Moodle**. URL: https://moodle.org/ (дата обращения 17.05.2015).
10. **О проекте** "Создание публичного ресурса открытых данных в области науки и техники, интегрированного в единое международное пространство знаний Linked Open Data" // Известия. 2011. 12 сентября. URL: http://izvestia.ru/news/500270 (дата обращения 17.05.2015).
11. **Определен** победитель конкурса по теме "Создание публичного ресурса открытых данных в области науки и техники, интегрированного в единое международное пространство знаний Linked Open Data" // Минобрнауки России. URL: http://old.mon.gov.ru/work/zakup/lot/8521/ (дата обращения 17.05.2015).
12. **Аникин А. В., Жукова И. Г.** Онтологическая модель представления знаний для поиска и интеграции образовательных ресурсов в открытых образовательных сетях // Известия Волгоградского государственного технического университета. 2014. Том 21, № 12. URL: http://cyberleninka.ru/article/n/ontologicheskaya-model-predstavleniya-znaniy-dlya-poiska-i-integratsii-obrazovatelnyh-resursov-v-otkrytyh-obrazovatelnyh-setyah (дата обращения 17.05.2015).
13. **Филозова И. А., Добрынин В. Н.** Структурно-функциональное обеспечение электронного семантического сборника образовательных ресурсов // Материалы Девятой Всероссийской научной конференции "Электронные библиотеки: перспективные методы и технологии, электронные коллекции" RCDL—2007. Переславль-Залесский, 15—18 октября 2007. URL: http://www.rcdl2007.pereslavl.ru/doc/2012/paper39.pdf (дата обращения 17.05.2015).
14. **Титенко С. В.** Структурные основы онтологически-ориентированной системы управления информационно-учебным Web-контентом // Управляющие системы и машины: информационные технологии. 2012. № 2. С. 35-42. URL: http://www.setlab.net/?view = Titenko-usim-new (дата обращения 17.05.2015).
15. **Портал** знаний. URL: http://www.znannya.org/ (дата обращения 17.05.2015).
16. **Тельнов В. П., Мышев А. В.** "Кафедра онлайн": облачные технологии в высшем образовании // Программные продукты и системы. 2014. № 4. С. 91—99. URL: http://www.swsys.ru/index.php?page = article&id = 3903 (дата обращения 17.05.2015).
17. **Образовательный** портал "Кафедра онлайн". URL: http://ksst.obninsk.ru (дата обращения 17.05.2015).
18. **Kobilarov G., Bizer C., Auer S., Lehmann J.** DBpedia — A Linked Data Hub and Data Source for Web and Enterprise Applications // Proceedings of Developers Track of 18th International World Wide Web Conference (WWW 2009), April 20th—24th, Madrid, Spain 2009. URL: http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/DBpedia-WWW2009-DevTrack-Abstract.pdf (дата обращения 17.05.2015).
19. **SPARQL**. URL: https://ru.wikipedia.org/wiki/SPARQL (дата обращения 17.05.2015).
20. **Linked** Data Fragments. Concept. URL: http://linkeddatafragments.org/concept/ (дата обращения 17.05.2015).
21. **Verborgh R., Vander Sande M.**, **Colpaert P. et** al. Web-Scale Querying through Linked Data Fragments // Proceedings of the 7th Workshop on Linked Data on the Web, April, 2014, URL: http://events.linkeddata.org/ldow2014/papers/ldow2014_paper_04.pdf (дата обращения 17.05.2015).
22. **Международный** стандарт ISO/IEC 19505—2:2012 Information technology — Object Management Group Unified Modeling Language (OMG UML) — Part 2: Superstructure. URL: http://webstore.iec.ch/preview/info_isoiec19505-2{ed1.0}en.pdf (дата обращения: 17.05.2015).
23. **Программное** обеспечение Visual Paradigm for UML. URL: http://www.visual-paradigm.com/features/uml-and-sysml-modeling/ (дата обращения 17.05.2015).
24. **Dojo** Toolkit. URL: http://dojotoolkit.org (дата обращения 17.05.2015).
25. **Сниппеты в** поисковых системах.URL: https://ru.wikipedia.org/wiki/%D0 %A1 %D0 %BD%D0 %B8 %D0 %BF %D0 %BF%D0 %B5 %D1 %82 (дата обращения 17.05.2015).
26. **Левенштейн В. И.** Персональная страница. URL: http://www.keldysh.ru/departments/dpt_10/lev.html (дата обращения 17.05.2015).
27. **MIT** OpenCourseWare. URL: http://ocw.mit.edu (дата обращения 17.05.2015).



**V. P. Telnov**, Associate Professor, e-mail: telnov@bk.ru, **A. V. Mishev**, Associate Professor, e-mail: mishev@iate.obninsk.ru, Obninsk Institute for nuclear power engineering of national research nuclear University "MEPhI"


# The Semantic Web and Search Agents for High School


*In the prologue article outlines the ideas and motivations underlying the concept of semantic educational portals and intelligent agents. Question of availability the Linked Open Data in the Russian segment of the World Wide Web. The presentation focused on the problems of the practical implementation of the ideas of the Semantic Web for educational portals of Russian universities.*

*In the main part of the article presents and discusses three search agents, mutually complementary. Of these, the first two agents — agents "DBpedia" and "Wikidata" — dealing with similar international knowledge bases. The third search agent "AjaxSearch" offers adaptive technology of the contextual information search in the world wide web, when the initial retrieval of content from the global network performed well-known search engines (Google, Yahoo, Yandex), who have the appropriate software interface. The key architectural, technological and design decisions in UML notation diagrams of components, sequence diagrams and deployment diagrams are given. Showing examples of the user interface.*

*The conclusion presents the arguments for and against the introduction of semantic web technologies in educational practice. The article focused on a wide range of professionals interested in the application of semantic technologies in educational activities of higher educational institutions, as well as students and professors specializing in the field of informatics and computer science.*

***Keywords:*** *information technologies, semantic web, knowledge base, search agent, higher education*

## References

1. **Panteleev M. G., Puzankov D. V., Tatarinov Yu. S.** *Perspektivy ispol'zovaniya tekhnologij semanticheskogo Web v obrazovatel'nykh portalakh*, available at: http://window.edu.ru/resource/126/49126/files/portal3-18.pdf (date of access 17.05.2015, in Russian).

2. **Tatarinov Yu. S., Panteleev M. G., Sazykin P. V.** *Mul'tiagentnyj servis postroeniya personalizirovannykh programm obucheniya dlya semanticheskikh obrazovatel'nykh portalov*, available at: http://old.lvk.cs.msu.su/files/mco2005/tatarinov.pdf (date of access 17.05.2015, in Russian).

3. **IEEE** 1484.12.1—2002. Learning Object Metadata standard, available at: http://en.wikipedia.org/wiki/Learning_object_metadata (date of access 17.05.2015).

4. **JAVA** Agent DEvelopment Framework, available at: http://jade.tilab.com/ (date of access 17.05.2015).

5. **DBpedia**, available at: https://ru.wikipedia.org/wiki/DBpedia (date of access 17.05.2015).

6. **Wikidata,** available at: http://www.wikidata.org (date of access 17.05.2015).

7. **Freebase,** available at: https://ru.wikipedia.org/wiki/Freebase (date of access 17.05.2015).

8. **Berners-Lee T.** Linked Data, available at: http://www.w3.org/DesignIssues/LinkedData.html (date of access 17.05.2015).

9. **Moodle**, available at: https://moodle.org/ (date of access 17.05.2015).

10. **O proekte** "Sozdanie publichnogo resursa otkrytykh dannykh v oblasti nauki i tekhniki, integrirovannogo v edinoe mezhdunarodnoe pro-stranstvo znanij Linked Open Data. *Izvestiya*, 2011, 12 September, available at: http://izvestia.ru/news/500270 (date of access17.05.2015, in Russian).

11. **Opredelyon** pobeditel' konkursa po teme "Sozdanie publichnogo resursa otkrytykh dannykh v oblasti nauki i tekhniki, integrirovanno-go v edinoe mezhdunarodnoe prostranstvo znanij Linked Open Data". Minobrnauki Rossii, available at: http://old.mon.gov.ru/work/zakup/lot/8521/ (date of access 17.05.2015, in Russian).

12. **Anikin A. V., Zhukova I. G.** Ontologicheskaya model' predstavleniya znanij dlya poiska i integratsii obrazovatel'nykh resursov v otkrytykh obrazovatel'nykh setyakh. *Izvestija Volgogradskogo gosudarstvennogo tehnicheskogo universiteta*, 2014, vol. 21, no 12, available at: http://cyberleninka.ru/article/n/ontologicheskaya-model-predstavleniya-znaniy-dlya-poiska-i-integratsii-obrazovatelnyh-resursov-v-otkrytyh-obrazovatelnyh-setyah (date of access 17.05.2015, in Russian).

13. **Filozova I. A., Dobrynin V. N.** Strukturno-funktsional'noe obespechenie ehlektronnogo semanticheskogo sbornika obrazovatel'nykh resursov. *Materialy Devjatoj Vserossijskoj nauchnoj konferencii "Jelektronnye biblioteki: perspektivnye metody i tehnologii, jelektronnye kollekcii" RCDL—2007.* Pereslavl'-Zalesskij, 15—18 October 2007, available at: http://www.rcdl2007.pereslavl.ru/doc/2012/paper39.pdf (date of access 17.05.2015, in Russian).

14. **Titenko S. V.** Strukturnye osnovy ontologicheski-orientirovannoj sistemy upravleniya informatsionno-uchebnym Web-kontentom. *Upravljajushhie sistemy i mashiny: informacionnye tehnologii*, 2012, no 2, pp. 35—42, available at:http://www.setlab.net/?view = Titenko-usim-new (date of access 17.05.2015, in Russian).

15. **Portal znanij**, available at: http://www.znannya.org/ (data of access 17.05.2015, in Russian).

16. **Tel'nov V. P., Myshev A. V.** "Kafedra onlajn": oblachnye tekhnologii v vysshem obrazovanii. *Programmnye produkty i sistemy*, 2014, no 4, pp. 91—99, available at: http://www.swsys.ru/index.php?page = article&id = 3903 (date of access 17.05.2015, in Russian).

17. **Obrazovatel'nyj** portal "Kafedra onlajn", available at:http://ksst.obninsk.ru (date of access 17.05.2015, in Russian).

18. **Kobilarov G., Bizer C., Auer S., Lehmann J.** DBpedia — A Linked Data Hub and Data Source for Web and Enterprise Applications. *Proceedings of Developers Track of 18th International World Wide Web Conference (WWW 2009)*, April 20th—24th, Madrid, Spain, 2009, available at: http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/DBpedia-WWW2009-DevTrack-Abstract.pdf (date of access 17.05.2015).

19. **SPARQL**, available at: https://ru.wikipedia.org/wiki/SPARQL (date of access 17.05.2015).

20. **Linked** Data Fragments. Concept, available at:http://linked-datafragments.org/concept/ (date of access 17.05.2015).

21. **Verborgh R., Vander Sande M., Colpaert P. Coppens S., Mannens E., Van de Walle R.** Web-Scale Querying through Linked Data Fragments. *Proceedings of the 7th Workshop on Linked Data on the Web,* April, 2014, available at: http://events.linkeddata.org/ldow2014/papers/ldow2014_paper_04.pdf (date of access 17.05.2015).

22. **ISO/IEC** 19505—2:2012 Information technology — Object Management Group Unified Modeling Language (OMG UML). — Part 2: Superstructure, available at: http://webstore.iec.ch/preview/info_isoiec19505-2{ed1.0}en.pdf (date of access 17.05.2015).

23. **Design** software with Unified Modeling Language (UML), available at: http://www.visual-paradigm.com/features/uml-and-sysml-modeling/ (date of access 17.05.2015).

24. **Dojo** Toolkit, available at: http://dojotoolkit.org (date of access 17.05.2015).

25. **Snippety** v poiskovykh sistemakh, available at:https://ru.wikipedia.org/wiki/%D0 %A1 %D0 %BD%D0 %B8 %D0 %BF %D0 %BF%D0 %B5 %D1 %82 (date of access 17.05.2015, in Russian).

26. **Levenshtejn V. I.**, personal page, available at: http://www.keldysh.ru/departments/dpt_10/lev.html (date of access 17.05.2015, in Russian).

27. **MIT** OpenCourseWare, available at: http://ocw.mit.edu (date of access 17.05.2015).

**ООО "Издательство "Новые технологии".** 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева*. Корректор *З. В. Наумова*

Сдано в набор 06.07.2015 г.Подписано в печать 18.08.2015 г.Формат 60×88 1/8.Заказ PI915
Цена свободная.

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