УДК 681.3

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА АЛГЕБРЫ АЛГОРИТМИКИ НА ПЛАТФОРМЕ WEB 2.0.

В.А. Иовчев, А.С. Мохница

Институт программных систем НАН Украины,
03187, Киев, проспект Академика Глушкова, 40,
e-mail: iovchev.v@gmail.com, mohnitsa@isofts.kiev.ua

Рассматривается разрабатываемый на платформе Web 2.0 инструментарий проектирования, трансформации и синтеза параллельных алгоритмов и программ, основанный на алгеброалгоритмических спецификациях. Функциональность инструментария демонстрируется на иллюстративном примере алгоритма броуновского движения частиц в кристалле.

Web 2.0-based means for design, transformation and synthesis of parallel algorithms and programs using algebraic algorithmic specifications are considered. The functionality of tools is demonstrated on the illustrative example of the algorithm of Brownian motion of particles in the crystal.

Введение

Широкое распространение многоядерного и многопоточного параллелизма в современных вычислительных системах привело к потребности создания специального инструментария для разработки и реинженерии параллельного программного обеспечения, который охватывает все этапы жизненного цикла программы. В Институте программных систем НАН Украины на протяжении длительного периода развивается теория, методология и инструментарий для автоматизированного проектирования параллельных программ, основанные на средствах высокоуровневой алгеброалгоритмической формализации и автоматизации преобразований программ [1–5]. Разработана экспериментальная инструментальная система ИПС для конструирования и оптимизации параллельных программ [6–8]. Система символьных вычислений TermWare создана на основе парадигмы переписывающих правил, дополняет возможности системы ИПС и используется для автоматизации инженерии программного кода параллельных программ [9–10].

Данная работа посвящена инструментарию, использующему онлайновую реализацию метода диалогового конструирования синтаксически правильных программ (ОДСП), и продолжающему линию исследований систем МУЛЬТИПРОЦЕССИСТ и ИПС. Система ОДСП предназначена для проектирования параллельных программ для современных сред выполнения, в которых алгеброалгоритмические спецификации и преобразования алгоритмов объединены с мощными средствами автоматизации и синтеза программ на основе Web 2.0. Отметим, что под современными средами выполнения понимаются мультиядерные, мультипотоковые, кластерные, Грид, облачные системы и т. д. [11]. Функционирование системы ОДСП иллюстрируется на примере преобразования последовательной схемы алгоритма «броуновского движения» частиц в кристалле в параллельную схему.

Изложение материала работы подчиненно следующей структуре: раздел 1 посвящен алгебро-алгоритмическим основам инструментария ОДСП; в разделах 2 и 3 приведена архитектура системы ОДСП, а также характеризуется ее функциональность; в разделе 4 рассмотрены инструментарий трансформации схем алгоритмов и программ, и подключаемый модуль трансформации для системы ОДСП; в разделе 5 на иллюстративном примере демонстрируется процесс трансформационной сводимости схем алгоритмов с помощью системы ОДСП и модуля трансформации.

1. Алгеброалгоритмические основы средств проектирования и синтеза алгоритмов и программ

Синтезатор МУЛЬТИПРОЦЕССИСТ является ярким примером открытых систем проектирования схем алгоритмов и программ для некоторых классов задач [12]. Данная система, используя алгоритмические проекты, оформленные на языке САА-схем, генерирует тексты программ на целевых языках программирования (Ассемблер, Си, Паскаль и др.). САА-схемы представляют собой естественно-лингвистические проекты алгоритмов, в основе которых лежит аппарат систем алгоритмических алгебр (САА) Глушкова. В Украине первая алгебра программ была построена В.М. Глушковым в рамках работ по системам алгоритмических алгебр [1] в середине 60-х годов, за несколько лет до того, как задача построения подобной алгебры была предложена Э. Дейкстрой. Более того, еще в 1959 году [13] Л.А. Калужнин предложил алгебру граф-схем как инструмент для математизации программирования, что послужило отправной точкой для создания САА. В 1982 году была создана система синтеза программ МУЛЬТИПРОЦЕССИСТ [12], в качестве входного языка была использована параллельная модификация САА– САА-М [14].

Рассматриваемый инструментарий (в отличие от системы МУЛЬТИПРОЦЕССИСТ) состоит в интеграции всех трех представлений алгоритма [2]: аналитического (формула в избранной алгебре), естественно-лингвистического (САА-схемы) и графового (граф-схемы Калужнина) при его конструировании.

Аналитическое представление базируется на алгебрах и является компактной записью алгоритма, направленной на его дальнейшее преобразование (минимизация, оптимизация по разным критериям) на базе аппарата соотношений и тождеств, развитых в алгебрах.

Естественно-лингвистическое представление (в виде текста), также базируется на аппарате алгебр. В процессе модификации с помощью метаправил декомпозируется на инвариантную часть (неинтерпретированную схему), которая представляет собою верхний уровень проекта и собственно интерпретации (нижний уровень), зависящие от избранной предметной области. После свертки инвариантная часть может быть снова проинтерпретирована, но уже с помощью других средств нижнего уровня.

Графовое представление, главное преимущество которого – наглядность, также опирается на аппарат алгебр и ориентировано на визуализацию конструируемого алгоритма. Основой для построения этого представления (для аналитического также) может быть выбрана инвариантная часть САА-схемы с дальнейшей интерпретацией в соответствии с избранным нижним уровнем.

Пример. Рассмотрим последовательную схему алгоритма, моделирующего броуновское движение n частиц (примесей) в одномерном кристалле (массиве) длины $l > 0$ в течение m итераций [15]. В начале все частицы находятся в первой (слева) ячейке кристалла. Каждая итерация заключается в переходе частицы в соседнюю (левую или правую) ячейку кристалла в зависимости от сгенерированного случайного числа (если оно больше или равно предварительно заданному пределу p, то происходит перемещение частицы на клетку вправо, если нет – то влево).

На массиве-кристалле введена следующая разметка. Начало и конец массива помечены маркерами Н и К, соответственно. Перед началом исполнения в первой клетке находится число n (отвечающее общему количеству частиц), остальные клетки содержат нули. Кроме того, первая клетка содержит указатели $У_1$, …, $У_n$, моделирующие перемещение частиц, по одному на каждую частицу. При перемещении указателя (частицы) из клетки в соседнюю, число, находящегося в данной клетке, уменьшается на единицу, число, находящееся в результирующей клетке, соответственно увеличивается. По окончании каждой итерации происходит вывод состояния массива.

Формула последовательного алгоритма броуновского движения примесей (частиц) в кристалле имеет следующий вид:

SeqBrownPart :: == СТАРТ * ИНИЦ * {[LastIteration] ПРИНТ * {[LastParticle] ГЕН(Х) * *([(Х>=Limit) И (НЕ(d($У_i$, К)))] СДВИГ_П($У_i$), Е) * ([(Х<Limit) И (НЕ (d($У_i$, Н)))] СДВИГ_Л($У_i$), Е)}}* *ПРИНТ * ФИН;

где

СТАРТ – получение начальных параметров алгоритма (количество частиц, размер кристалла, величина вероятностного барьера и т.д.);

ИНИЦ – инициализация переменных, в т.ч. массива-кристалла;

ПРИНТ – вывод текущего состояния кристалла;

ГЕН(Х) – генерация случайного числа Х в интервале (0, …, 1);

СДВИГ_П – переход частицы на одну ячейку кристалла вправо;

СДВИГ_Л – переход частицы на одну ячейку кристалла влево;

ФИН – действия, предпринимаемые по окончании алгоритма;

Е – тождественный оператор;

Н, К – маркеры начала и конца массива-кристалла; Уi – указатель на текущую частицу;

d($У_i$, Н) (d($У_i$, К)) – предикат, истинный при достижении текущей частицей начала (конца) массива-кристалла;

LastIteration, LastParticle – предикаты, означающие соответственно достижение последней итерации и последней частицы;

Limit – величина вероятностного барьера, (Х>=Limit) ((Х<Limit)) – предикат, истинный, если случайное число Х больше или равно (соответственно меньше) вероятностного барьера.

В последовательной версии алгоритма во внешнем цикле осуществляется вывод текущего состояния кристалла и «отработка» перехода для всех частиц, а во вложенном цикле – переход для каждой частицы.

Естественно-лингвистическое представление имеет вид:

СХЕМА ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ БРОУНОВСКОГО ДВИЖЕНИЯ ПРИМЕСЕЙ (ЧАСТИЦ) В КРИСТАЛЛЕ ====

"Последовательный алгоритм броуновского движения примесей (частиц) в кристалле"

КОНЕЦ КОММЕНТАРИЯ

"SeqBrownPart"==

==== "СТАРТ"

ЗАТЕМ

"ИНИЦ"

ЗАТЕМ

ПОКА НЕ 'LastIteration'

```
ЦИКЛ
  "ПРИНТ"
  ЗАТЕМ
  ПОКА НЕ 'LastParticle'
  ЦИКЛ
    "ГЕН(X)"
    ЗАТЕМ
    ЕСЛИ 'X>=Limit'
         И
        НЕ('Указатель У(i) в конце (М)')
    ТО "СДВИГ_П(У(i))"
    ИНАЧЕ "Пустой оператор"
    КОНЕЦ ЕСЛИ
    ЗАТЕМ
    ЕСЛИ 'X<Limit'
         И
        НЕ('Указатель У(i) в начале (М)')
    ТО "СДВИГ_Л(У(i))"
    ИНАЧЕ "Пустой оператор"
    КОНЕЦ ЕСЛИ
  КОНЕЦ ЦИКЛА
КОНЕЦ ЦИКЛА
ЗАТЕМ
"ПРИНТ"
ЗАТЕМ
"ФИН"
```

КОНЕЦ СХЕМЫ ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ БРОУНОВСКОГО ДВИЖЕНИЯ ПРИМЕСЕЙ (ЧАСТИЦ) В КРИСТАЛЛЕ

Граф-схема описанного алгоритма показана на рис. 1.

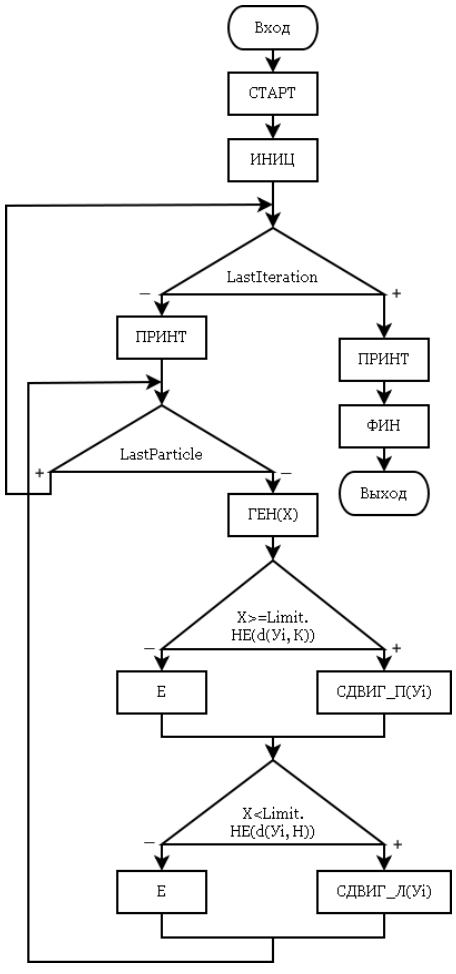

Рис. 1. Граф-схема последовательного алгоритма броуновского движения

2. Архитектура инструментальной системы ОДСП

Предлагаемый инструментарий состоит из компонентов, показанных на рис. 2 [16]. Как упоминалось ранее, с помощью инструментария осуществляется диалоговое проектирование и синтез объектно-ориентированных параллельных алгоритмов и программ с использованием элементов базы знаний. А так же их последующая отладка, эксплуатация, реинженерия и т.д. Отметим, что компоненты инструментария автономны [17], гибко связанны и имеют согласованный протокол обмена данными, т.е. данная модель, по сути, является сервисно-ориентированной [11].

Клиент представляет собой интерфейс для диалогового взаимодействия пользователя с инструментарием и его ресурсами. А именно возможность проектирования последовательного или параллельного алгоритма в вышеупомянутых формах и синтеза исходного кода в целевом языке программирования, а так же его дальнейшей эксплуатации (запуск в доступных современных вычислительных средах, распараллеливание и другое).

Диспетчер – ядро инструментария, которое организовывает связь между базой данных, генератором, расширениями, с помощью шлюза, с современными средами выполнения и пользователями (посредством клиентов). Организовывает проектирование, синтез и выполнение (отладка, реинженерия) параллельных программ в современных средах выполнения.

База данных состоит из базы алгоритмических знаний инструментария и технической базы данных. База алгоритмических знаний инструментария вмещает следующие разделы:

•схемы алгоритмов (разработанные алгоритмы из разных предметных областей);

•стратегии обработки (схемы, которые описывают классы алгоритмов и подлежат дальнейшей детализации);

•метаправила свертки, развертки и трансформации (обеспечивают абстрагирование, детализацию и переинтерпретацию схем, а также содержат тождества и соотношения для преобразования схем);

•базисные понятия и их программные реализации (ориентированные на проектирование алгоритмов и синтез программ в данной предметной области на избранном целевом языке);

•графические элементы, используемые для представления граф-схем алгоритмов (различные виды стрелок, блоков и др.).

Техническая база данных содержит настройки и данные пользователей инструментария, зарегистрированных сред выполнения, расширений, а также настройки и стратегии работы инструментария в целом.

Генератор – это сервис, осуществляющий синтез программ.

Интерфейс расширений – это сервис, ориентированный на расширение возможностей инструментария путем включения дополнительных сервисов для работы с алгоритмами и программами. Например, оценка сложности алгоритма [15].

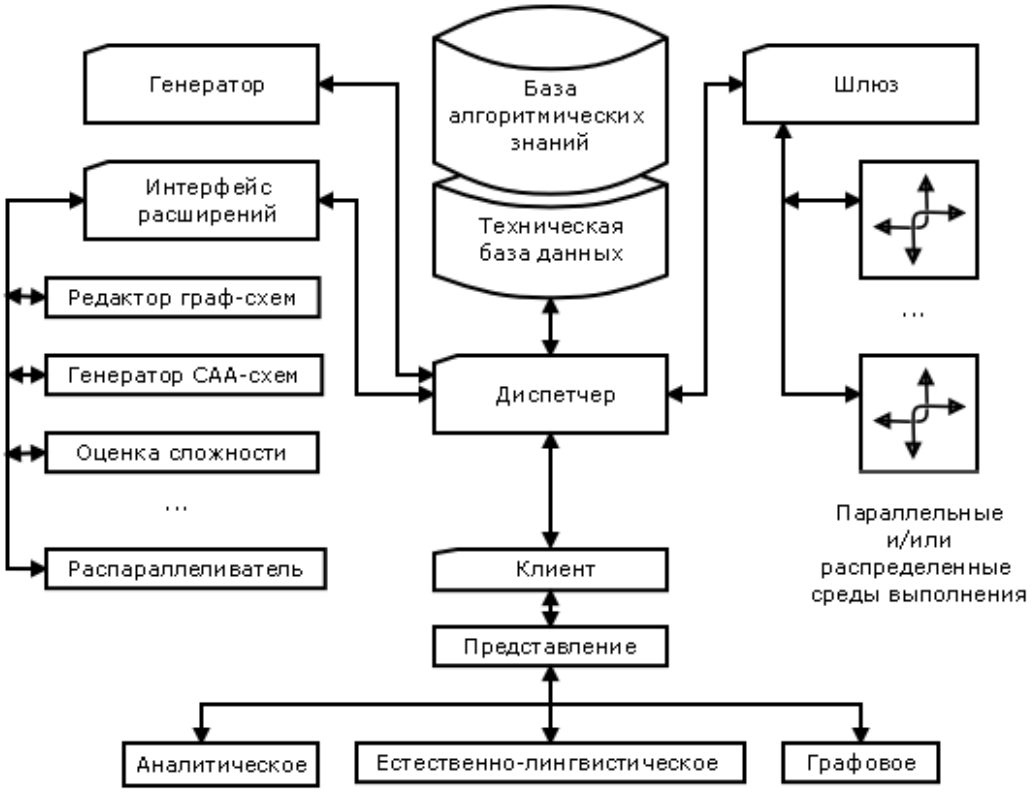

Рис. 2. Архитектура ОДСП

Редактор граф-схем ориентирован на визуальное представление алгоритмов. При этом изменения, внесенные во время редактирования, соответствующим образом отобразятся на другие представления алгоритма.

Параметрически управляемый генератор САА-схем [18] ориентирован на синтез схем алгоритмов и программ. Посредством спецификаций более высокого уровня, называемых регулярными гиперсхемами (РГС) [12, 19]. РГС применяются, в частности, для представления алгоритмов управления выводом в грамматиках структурного проектирования (ГСП) [16, 20, 21]. Проектирование гиперсхем, как и САА-схем, выполняется в диалоговом режиме.

Шлюз – это сервис, обеспечивающий запуск, анализ и получение результатов выполнения программ в современных средах выполнения.

3. Средства проектирования и синтеза параллельных объектно-ориентированных программ

Вышеупомянутая система ОДСП продолжает линию исследований МУЛЬТИПРОЦЕССИСТ [7, 8, 12] и ИПС [16]. Главным отличием от предшественников в первую очередь является ориентация на многопользовательское использование через Интернет и распределенную архитектуру, как самой системы, так и проектируемых с ее помощью приложений.

Основным компонентом рассматриваемого инструментария является онлайновый ДСП-конструктор, ориентированный на автоматизированное проектирование и синтез синтаксически правильных алгоритмов и программ. В основу его функционирования положен диалоговый режим с использованием меню подстановок, FIFO (память типа очередь) и дерева конструирования алгоритма. Меню состоит из операторных и логических конструкций, суперпозиция которых позволяет создавать алгоритмы в упомянутых ранее формах (см. разд. 1). Данные конструкции входят в сигнатуру операций модифицированных САА (САА-М) [2, 14, 22], ориентированных на формализацию последовательных и параллельных вычислений. Выбранные пользователем конструкции, а также операторные и логические переменные, входящие в них, отображаются в дереве с дальнейшей детализацией переменных. В зависимости от типа выбранной переменной система предлагает соответствующий компонент меню или открывает для пользователя архив базисных понятий из базы знаний. Отметим «поуровневый» стиль конструирования алгоритма, а также возможность переходов на различные уровни (узлы дерева) с продолжением процесса диалогового конструирования, причем подобный переход сопровождается соответствующим изменением состояния FIFO [22].

С помощью ОДСП-конструктора было выполнено диалоговое «поуровневое» проектирование алгоритма броуновского движения приведенного ранее в качестве примера (разд. 1). Базовая рабочая область ОДСП-конструктора (см. рис. 3) состоит из меню пользователя и трех автономных компонентов (типа виджет), двое из которых отображают соответствующие представления алгоритма в процессе его конструирования (САА-схема, формула, дерево алгоритма). Верхний левый компонент предназначен для отображения и выбора элементов подстановки из базы знаний. Слева внизу расположен компонент расширение для получения САА-схемы. Дерево алгоритма для каждого составного оператора схемы отображается на отдельной вкладке компонента справа. На рис. 3. изображены дерево конструирования и САА-схема основного составного оператора.

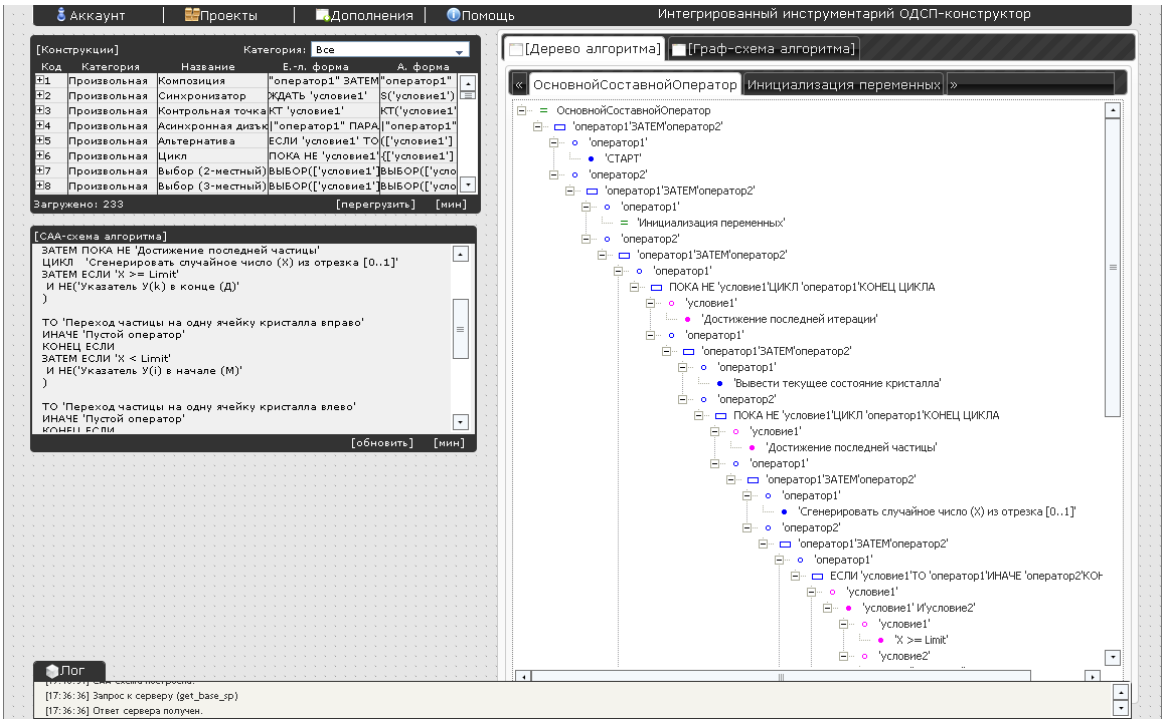

Рис. 3. Проект алгоритма броуновского движения частиц в ОДСП-конструкторе

Помимо языковых конструкций и элементарных операторов и предикатов база знаний содержит также стратегии обработки — схемы алгоритмов с переменными. ОДСП-конструктор можно использовать как при проектировании новых схем, так и в процессе детализации уже имеющихся стратегий. При этом проверка синтаксических ошибок и построение дерева осуществимы с помощью синтаксического анализатора, базирующегося на соответствующих средствах алгоритмики [2, 22].

По дереву алгоритма, реализациям элементарных операторов и условий, а также другим фрагментам программ на целевом объектно-ориентированном языке программирования (Java, C++ и др.), ОДСП-конструктор выполняет синтез программы. В процессе синтеза управляющие конструкции схемы отображаются в соответствующие операторы языка программирования, а вместо базисных элементов подставляются их реализации на этом же языке. На вход синтезатора поступает также файл, содержащий каркасное описание основного класса приложения (без реализаций методов), в который выполняется подстановка синтезированного кода. Реализации базисных элементов написаны с использованием программных компонентов для решения задач символьной обработки [22, 23], которые содержат описание данных (массива, указателей, маркеров и др.) и методов доступа к ним. Описать структуру упомянутых классов и выполнить генерацию каркасного программного кода можно с помощью системы Rational Rose [22]. Синтез кода для асинхронных алгоритмов связан с использованием потоков [22–28].

4. Инструментарий трансформации схем и подключаемый модуль Трансформатор

В рамках работы над системой ИПС с помощью аппарата алгебры алгоритмики был спроектирован и реализован на языке программирования Delphi инструментарий трансформации схем алгоритмов и программ (Трансформатор) [29, 30]. Внешний вид Трансформатора представлен на рис. 4. Интерфейс приложения состоит из следующих элементов:

- табличного блока выбора правил трансформации, хранящихся в базе данных Трансформатора;
- поля редактирования, содержащего трансформируемое аналитическое выражение на данном этапе преобразования, а также на все предыдущих этапах;
- поля редактирования, отображающего информацию о правилах (соотношениях), примененных к трансформируемому выражению для каждого этапа процесса преобразования.

Кроме того, программа предлагает отдельное окно для ввода новых или редактирования существующих тождеств с сохранением в базе тождеств и соотношений.

Первоначально Трансформатор базировался на алгоритме последовательной статической декомпозиции ДЕК/С [2], осуществляющем посимвольный анализ входной цепочки и накладываемой на нее формы с последующей переинтерпретацией выходной цепочки с помощью получивших значение переменных формы. В настоящее время Трансформатор использует более эффективный алгоритм последовательной рекурсивной декомпозиции на основе таблиц иерархии, предложенный авторами в [30].

Для реализации возможности эквивалентных преобразований для системы ОДСП авторами был спроектирован и создан на языке JavaScript [26, 27] подключаемый модуль Трансформатор (см. рис. 5), аналогичный по внешнему виду и функциям своему предшественнику.

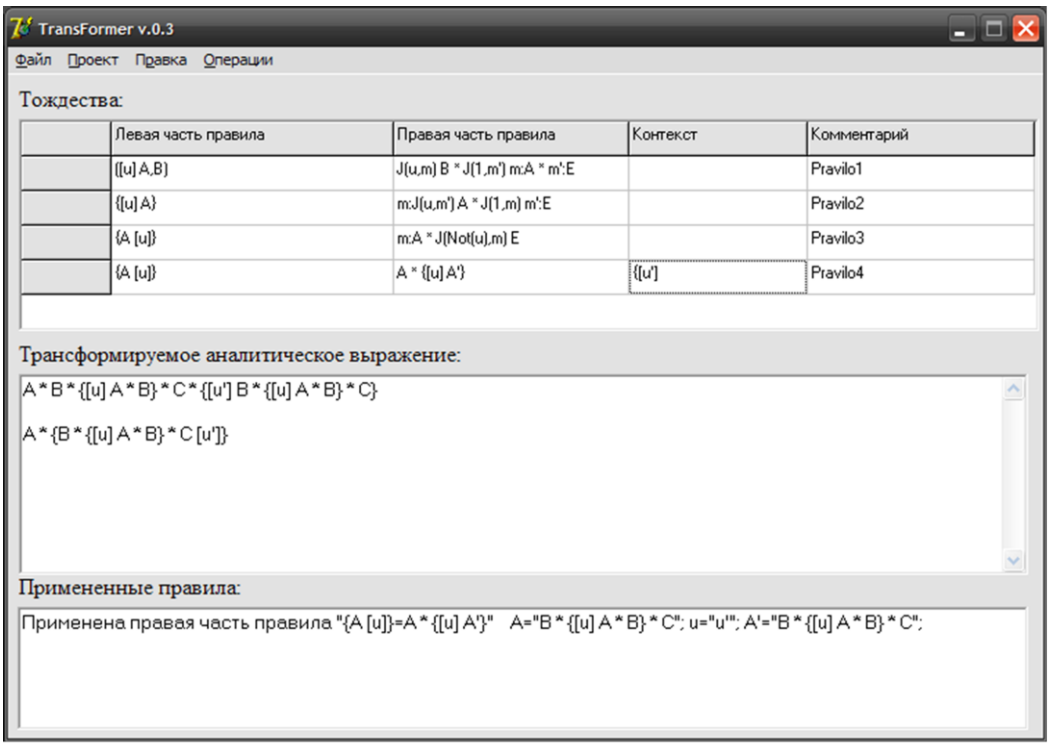

Рис. 4. Инструментарий трансформации схем алгоритмов

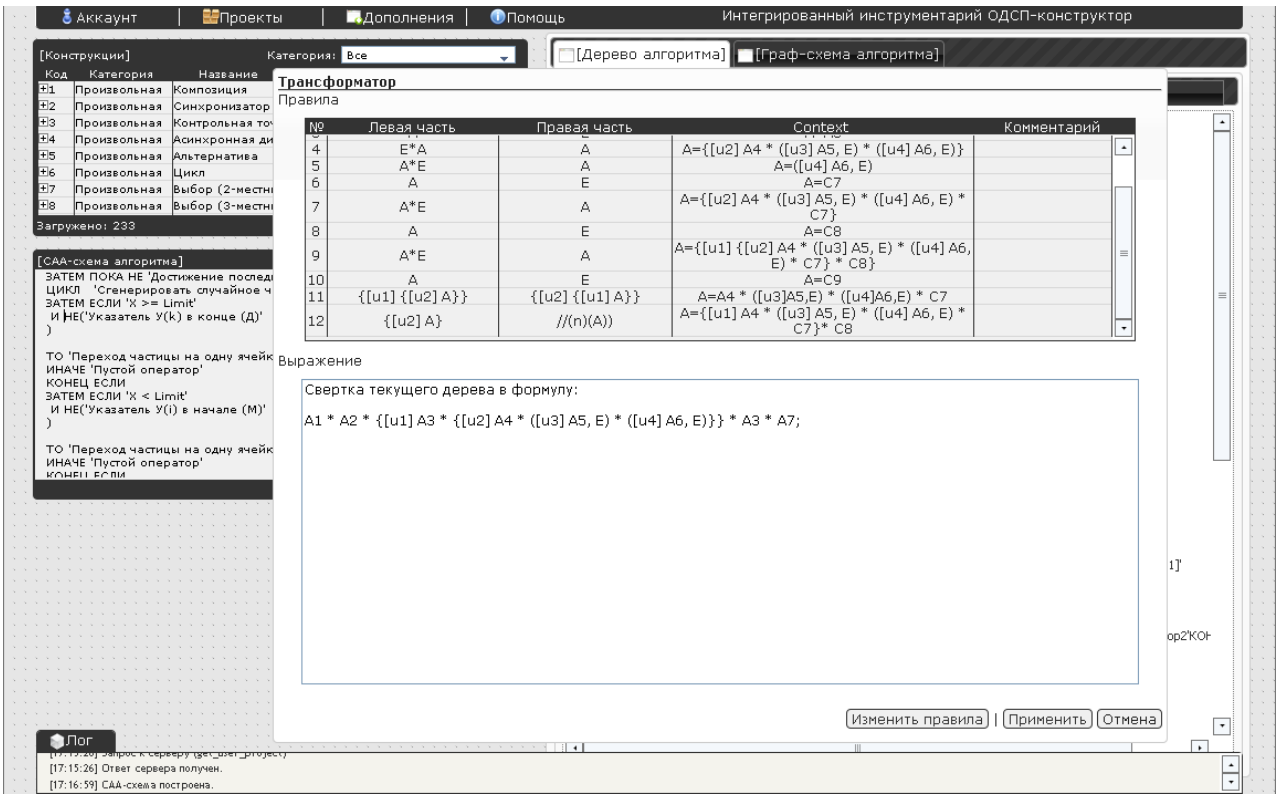

Рис. 5. Неинтерпретированная формула аналитического представления последовательного алгоритма

5. Трансформационная сводимость в системе ОДСП

Продемонстрируем с помощью системы ОДСП и модуля Трансформатор преобразование последовательного алгоритма броуновского движения (см. раздел 1) к параллельному.

Аналитическое представление (формула) параллельного алгоритма броуновского движения примесей (частиц) в кристалле имеет следующий вид:

ParBrownPart :: == СТАРТ * ИНИЦ * ПРИНТ * $\dot{\vee}(n)$({[LastIteration] ГЕН(X) *([(X>=Limit) И (НЕ(d(Уi, К)))] ПСДВИГ_П($У_i$), Е) * ([(X<Limit) И (НЕ(d($У_i$, Н)))] ПСДВИГ_Л($У_i$), Е)* ППРИНТ} * T(u)) * S(AllProcsFinished) * ПРИНТ * ФИН,

где

$\dot{\vee}(n)$ – оператор асинхронной дизъюнкции (n процессов-ветвей);

AllProcsFinished – предикат, истинный, если все ветви асинхронной дизъюнкции закончили обработку;

T(u) – контрольная точка, введенная для синхронизации ветвей, u – условие, истинное с момента достижения контрольной точки в процессе вычислений в ветви (при ошибке – не определено);

S(AllProcsFinished) – синхронизатор – оператор, реализующий ожидание пока не выполнено условие AllProcsFinished;

$У_i$ – указатель на текущую ветвь;

ПСДВИГ_П($У_i$), ПСДВИГ_Л($У_i$), ППРИНТ – версии операторов СДВИГ_П($У_i$), СДВИГ_Л($У_i$), ПРИНТ, ориентированные на функционирование в ветвях асинхронной дизъюнкции.

Особенностью параллельной версии алгоритма по сравнению с последовательной, является замена внешнего цикла асинхронной дизъюнкцией, каждая ветвь которой обрабатывает отдельную частичку (указатель). Для каждой частицы на протяжении всех итераций вложенный в тело дизъюнкции цикл осуществляет переход (сдвиг) и вывод текущего состояния кристалла для данной частицы.

Покажем трансформационную сводимость аналитических представлений упомянутых алгоритмов.

1 шаг – свертка SeqBrownPart, в результате получим следующую формулу (см. рис. 5):

SBP:: == A1 * A2 * {[u1] A3 * {[u2] A4 * ([u3] A5, E) * ([u4] A6, E)}} * A3 * A7;

далее свернем ParBrownPart, в результате чего получим следующую формулу:

PBP:: == B1 * B2 * B3 * $\dot{\vee}$ ({[v1] B4 * ([v2] B5, E) * ([v3] B6, E) * B7} * B8) * B9 * B3* B10;

2 шаг – приведение SBP к промежуточной формуле (см. рис. 6, шаг 10), по структуре аналогичной PBP. Осуществляется домножением на тождественный оператор с последующей переинтерпретацией, либо же заменой «лишних» операторов тождественным с последующим сокращением (правила A*E=A; E *A = A).

SBP2:: == A1 * A2 * C3 * {[u1] {[u2] A4 * ([u3] A5, E) * ([u4] A6, E) * C7} * C8} * C9* A3 * A7;

3 шаг – вынос «вложенной» итерационной скобки перед «внешней» (см. Рис. 6, шаг 11), в результате чего условием внешнего цикла становится u2, а условием вложенного – u1 (правило {[u1] {[u2] A}}= {[u2] {[u1] A}} (в данном случае оператор A не оказывает влияние ни на u1, ни на u2)).

SBP3:: == A1 * A2 * C3 * {[u2] {[u1] A4 * ([u3] A5, E) * ([u4] A6, E) * C7}* C8} * C9* A3 * A7;

4 шаг – замена внешнего цикла на асинхронную дизъюнкцию (см. рис. 6, шаг 12). Преобразование возможно благодаря известному количеству и взаимной независимости итераций внешнего цикла (правило {[u2] A} = $\dot{\vee}$ (*n*)(A)):

SBP4:: == A1 * A2 * C3 * $\dot{\vee}$ (*n*)({[u1] A4 * ([u3] A5, E) * ([u4] A6, E) * C7}* C8) * C9* A3 * A7;

PBP:: == B1 * B2 * B3 * $\dot{\vee}$ (*n*)({[v1] B4 * ([v2] B5, E) * ([v3] B6, E) * B7} * B8) * B9 * B3* B10;

5 шаг – переинтерпретация формулы SBP4, в результате которой получим формулу ParBrownPart.

В дальнейшем, данная формула может с помощью ОДСП быть собрана и передана на исполнение на параллельную распределенную систему по выбору пользователя.

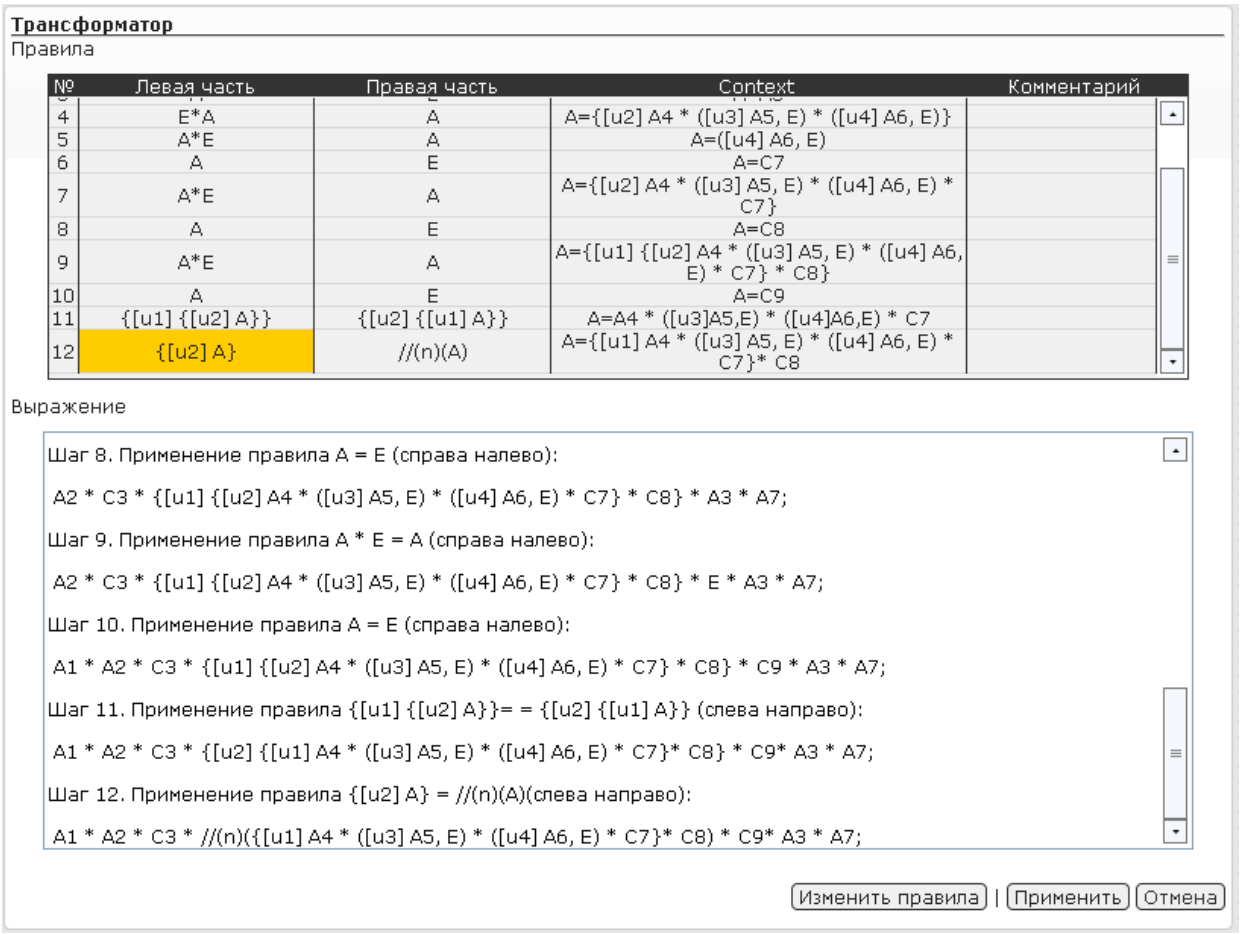

Рис. 6. Этапы преобразования

Заключение

Рассмотрена инструментальная система ОДСП, продолжающая линию исследований МУЛЬТИПРОЦЕССИСТ и ИПС, предназначенная для проектирования программного обеспечения для параллельных и распределенных сред с помощью алгебро-алгоритмических спецификаций на основе средств Web 2.0. Для ОДСП в качестве подключаемого модуля реализовано на языке JavaScript инструментальное средство трансформации аналитических спецификаций алгоритмов и программ (Трансформатор). Возможности системы ОДСП по конструированию, трансформации и сборке демонстрируются на примере преобразования последовательной схемы алгоритма броуновского движения частиц в кристалле в параллельную.

В перспективе планируется дальнейшее развитие и обогащение системы новыми подключаемыми модулями (в т.ч. и для системы TermWare), а также звуковое сопровождение интерфейса для пользователей с проблемами зрения.

1. *Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л.* Алгебра. Языки. Программирование. – Киев: Наук. думка, 1-е изд., 1974. – 327 с.; 2-е изд., перераб., 1978. – 318 с.; 3-е изд., перераб. и доп, 1989. – 376 с.
2. *Цейтлин Г.Е.* Введение в алгоритмику. – Киев: Сфера, 1998. – 310 с.
3. *Doroshenko A., Tseitlin G.* Models and Parallel Programming Abstractions to Enhance Concurrency of Parallel Programs, Fundamenta Informaticae. – 2004. – Vol. 60, № 1–4. –P. 99–111.
4. *Андон Ф.И., Дорошенко А.Е., Цейтлин Г.Е., Яценко Е.А.* Алгеброалгоритмические модели и методы параллельного программирования. – Киев: Академпериодика, 2007. – 634 с.

5. *Дорошенко А.Е.* Математические модели и методы организации высокопроизводительных параллельных вычислений. Алгебродинамический подход. – Киев: Наук. думка, 2000. – 177 с.
6. *Цейтлин Г.Е., Яценко Е.А.* Элементы алгебраической алгоритмики и объектно-ориентированный синтез параллельных программ // Математические машины и системы. – 2003. – № 2. – С. 64–76.
7. *Яценко Е.А.* Алгебры гиперсхем и интегрированный инструментарий синтеза программ в современных объектно-ориентированных средах // Кибернетика и системный анализ. – 2004. – № 1. – С. 47 – 52.
8. *Яценко О.А.* Середовище конструювання алгоритмічних знань та інструментарій синтезу програм // Проблеми програмування. — 2006. – № 2–3. – С. 349–359.
9. *Doroshenko, A.E., Shevchenko R. A.* Rewriting Framework for Rule-Based Programming Dynamic Applications // Fundamenta Informaticae. – 2006. – 72. – P. 95–108.
10. *TermWare* – http://www.gradsoft.com.ua/products/termware_rus.html
11. *Дорошенко А.Е., Алистратов О.В., Тырчак Ю.М., Розенблат А.П.* Системы GRID-вычислений — перспектива для научных исследований // Проблеми програмування. – 2005. – № 1. –Р. 14–38.
12. *Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К.* Многоуровневое структурное проектирование программ: Теоретические основы, инструментарий. – М.: Финансы и статистика, 1989. – 208 с.
13. *Калужнин Л.А.* Об алгоритмизации математических задач // Проблемы кибернетики. – 1959. – Вып. 2. – С. 51–69.
14. *Ющенко Е.Л., Цейтлин Г.Е., Галушка А.В.* Алгебро-грамматические спецификации и синтез структурированных схем программ // Кибернетика. – 1989. – № 6. – С. 5–16.
15. *Дорошенко А.Е., Жереб К.А., Яценко Е.А.* Об оценке сложности и координации вычислений в многопоточных программах // Проблеми програмування. – 2007. – № 2. – С. 41–55.
16. *Дорошенко А.Е., Цейтлин Г.Е., Иовчев В.А*. Высокоуровневые средства автоматизации проектирования параллельных алгоритмов // Проблеми програмування. - 2009. – № 3. – 19–29 с.
17. *IBM* Autonomic Computing. – http://www.research.ibm.com/autonomic/
18. *Яценко Е.А.* Алгебры гиперсхем и интегрированный инструментарий синтеза программ в современных объектно-ориентированных средах. // Кибернетика и системный анализ. – 2004. – № 1. – С. 47–52.
19. *Цейтлин Г.Е.* Алгебры Глушкова и теория клонов // Кибернетика и системный анализ. – 2003. – № 4. – С. 48–58.
20. *Цейтлин Г.Е., Суржко С.В., Ющенко К.Л., Шевченко А.И.* Алгоритмические алгебры. – Киев, 1997. – 342 с.
21. *Цейтлин Г.Е., Иовчев В.А., Мусихин А.А.* Ментальные аспекты методов символьной мультиобработки // Проблеми програмування. – 2008. – № 1. – С. 60–67.
22. *Яценко Е.А., Мохница А.С.* Инструментальные средства конструирования синтаксически правильных параллельных алгоритмов и программ // Проблеми програмування. – 2004. – № 2–3. – С. 444 – 450.
23. *Цейтлин Г.Е., Яценко Е.А.* Элементы алгебраической алгоритмики и объектно-ориентированный синтез параллельных программ // Математические машины и системы. – 2003. – № 2. – С. 64 – 76.
24. *Бишоп Д.* Эффективная работа: Java 2. – СПб.: Питер, 2002. – 592 с.
25. *Timothy R.* Fisher, Java Phrasebook//Sams Publishing,USA – 2007. – 224 p.
26. *Dave Crane*. Bear Bibeault with Tom Locke, AJAX. Prototype & Scriptaculous in action.//Manning. – 2007. – 544 p.
27. John Resing, Pro JavaScript Techniques//Apress,USA - 2006, 359 p.
28. *Глушаков С.В., Лукошкина С.Н.* Технология JAVA2//Харьков: Фолиою – 2006. – 606 с.
29. *Мохница А.С.* Алгебра алгоритмики и трансформационная сводимость схем алгоритмов и программ // Проблеми програмування. Матер. 6-й Междунар. научно-практической конф. по программированию УкрПРОГ'2008. – 2008. – № 2/3. – С. 341–347.
30. *Мохница А.С., Иовчев В.А., Андрющенко Е.А.* Особенности реализации средств трансформационного синтеза параллельных алгоритмов. // Проблеми програмування. – 2009. – № 4.