Вопросы автоматно-сетевого моделирования вычислительных систем с управлением потоком данных

С.М. Салибекян, П.Б. Панфилов

Аннотация. В статье представлены основы разрабатываемого формального аппарата (ОА-сеть) для описания и анализа вычислительного процесса в системах и приложениях, работающих в парадигме dataflow (управление вычислениями с помощью потока данных). Основой аппарата стали объектно-атрибутный подход к организации вычислительного процесса, теория конечных автоматов и теория процессных сетей Кана. Разработанный аппарат позволяет описывать все нюансы параллельного вычислительного процесса в распределенной вычислительной системе и найдет применение в имитационном моделировании, для оценки параметров вычислительной системы и поиска тупиковых ситуаций.

Ключевые слова: объектно-атрибутная архитектура, параллельные вычисления, моделирование, вычислительная система с управлением потоком данных, dataflow, процессная сеть Кана, теория конечных автоматов.

1. Актуальность

Парадигма управления вычислительным процессом с помощью потока данных (dataflow) считается весьма перспективной для создания высокопараллельных вычислительных систем [1]. Однако несмотря на свою 40-летнюю историю и свои теоретические преимущества парадигма пока не смогла занять достойного места среди параллельных вычислительных систем и приложений. И причиной тому не только сложность в аппаратной и программной реализации данного класса вычислительных систем (ВС), но и недостаточная формальная проработка dataflow-модели вычислений, т.к. почти все существующие математические модели неудобны для описания dataflow-вычислений. Так, по мнению профессора Массачусетского технологического института, США, Карла Хьюита (англ. Carl Hewitt), сетям Петри применительно к моделированию dataflow-систем присущи следующие критические недостатки [2]:

- сети Петри моделируют управление потоком, но не сам поток данных;
- сложность описания одновременных действий, выполняемых во время вычислительного процесса;
- физическая интерпретация перехода в сетях Петри весьма сомнительна.

В [3] авторами была предложена доработка аппарата сетей Петри, с помощью которой он был приспособлен к моделированию dataflow-систем. Однако данная модель, к сожалению, не обеспечивает работу со сложными типами данных и не позволяет описывать процесс обмена данными между вычислительными узлами через оперативную память, что существенно ограничивает ее применение.

Аппарат системы взаимодействующих автоматов [4], хоть и приспособлен для описания параллельных вычислительных процессов, однако не подходит для моделирования dataflow-систем, т.к. с его помощью описывается последовательность действий, а не поток данных.

Теория процессных сетей Кана [5] была разработана специально для моделирования dataflow-систем. Однако она не способна описать обработку сложно структурированных

данных и взаимодействие вычислительных узлов через общую или распределенную оперативную память.

Поэтому перед авторами встала задача разработки формальной модели, которая бы позволяла: описывать работу параллельной распределенной ВС; обмен данными между вычислительными узлами как напрямую, так и через общую или распределенную оперативную память; обработку сложно структурированных данных. Основой модели послужили объектно-атрибутная (ОА) архитектура ВС, разработанная в Московском институте электроники и математики Национального исследовательского университета «Высшая школа экономики» [3, 6], теория конечных автоматов и процессная сеть Кана. Комбинация этих подходов и позволяет обеспечить адекватную формализацию dataflow-вычислительного процесса.

2. ОА-алгебра

Предлагаемую формальную модель dataflow-вычислительного процесса можно разделить на две части: пассивную (ОА-алгебра) и активную (объектно-атрибутная автоматно-сетевая модель). Пассивная часть представляет собой описание модели сложно структурированных данных и операций сравнения и преобразования данных, а активная часть – описание сети ОА-автоматов, которые, собственно, и реализуют обработку данных. ОА-автомат представляет собой совокупность элементов любой природы (счетные и несчетные множества, списки, вектора, функции, вектора функций и т.д.), сгруппированных в множества, вместе с функцией изменения этих элементов. Согласно ОА-подхода, как это представлено, например, в работах [3,6], в прикладной (программной или аппаратной) ОА-системе ОА-автомат является моделью функционального устройства (ФУ), которое осуществляет обработку информации.

Теперь более подробно остановимся на пассивной части модели и ее описанию предпошлем ряд базовых определений и утверждений.

Определение 1. Все данные (хранимые и участвующие в обмене информацией между ОА-автоматми) в формальной ОА-модели представляются в виде информационных пар (ИП). ИП – это двойка $c=\{a, l\}$, включающая атрибут $a \in A$ (A – множество индексов атрибутов) и нагрузку $l \in L = \{\Re \cup nil \cup S \cup \Omega\}$, где L – множество данных в нагрузке, $\Re$ – множество рациональных чисел, nil – обозначение пустой нагрузки, Ω – множество адресов (номеров) глобальной памяти; $S \in \Sigma^*$ – множество цепочек символов, принадлежащих алфавиту Σ. Все ИП можно разделить на два класса, а именно: информационные (служат для описания каких-либо объектов) и милликоманды (применяются для обмена информаций между ОА-автоматами и управления ОА-автоматами).

Дадим определение **цепочки ИП** в нашей модели (в ОА-ВС цепочка ИП называется **информационной капсулой** [3, 6]).

Определение 2. η является **цепочкой ИП**, если выполняются следующие условия:

1. Если ω - пустая цепочка ИП (не содержит ни одной ИП), то ω - цепочка ИП;
2. Если γ - цепочка ИП и c – ИП, то γc (или $\gamma \cup c$) – цепочка ИП (где $\cup$ - знак объединения цепочек ИП);
3. η - цепочка ИП тогда и только тогда, когда она является таковой в силу (1) и (2).

Примечание: в цепочку ИП могут входить одинаковые ИП.

Индексом ИП в цепочке является порядковый номер ИП в цепочке, например, η_i, где $i = 1,2,\ldots |\eta|$, а $|\eta|$ – длина (количество ИП) цепочки η.

Для реализации информационного поиска и обработки информационных конструкций введем следующее понятие эквивалентности цепочек.

Утверждение 1. **Цепочки ИП** $\eta 1$ и $\eta 2$ **эквивалентны** (т.е. $\eta 1=\eta 2$) в том случае, если выполняются следующие условия:

1. совпадают длины цепочек ИП $\eta 1$ и $\eta 2$, т.е. $|\eta 1| = |\eta 2|$
2. $\eta 1=\eta 2=\omega$ или $\eta 1_i=\eta 2_i$, $i=1,2,\ldots,|\eta 1|$.

Определим **операции над цепочками ИП**, чтобы иметь возможность их обработки.

Определение 3. Операция **конкатенации (объединения)**: если α и β - цепочки ИП, то цепочка ИП $\alpha\beta$ ($\alpha \cup \beta$) называется конкатенацией (сцеплением, объединением) цепочек ИП α и β. $\alpha\omega=\omega\alpha=\alpha$, где ω - пустая цепочка ИП.

Определение 4. Операция **пересечения цепочек ИП**: если α и β - цепочки ИП, то цепочка ИП γ является пересечением цепочек ИП α и β (α ∩ β) в том случае, если она содержит в себе ИП, удовлетворяющие следующему условию:

$\gamma = \{a_i \in \alpha, \exists\ b_j \in \beta: a_i=b_j\}$, где i=1,2,…, |α|, j=1,2,…, |β|.

Определение 5. **Связка цепочек ИП (W)** – это взаимосвязанная структура данных, состоящая из цепочек ИП. В ячейках связки W могут содержаться цепочки ИП $W=\{w_1,w_2,\ldots w_{|W|}\}$, где |W| – длина связки цепочек символов, $w_1,\ldots,w_{|W|}$. Для индексации конкретной ИП в цепочке цепочек ИП можно пользоваться двойным индексом: W_{ij} – ИП, которая содержится в j-ой ИП i-ой цепочке ИП из W.

Связка цепочек ИП W будет использоваться в нашей модели в качестве общей памяти, в которой располагаются данные, к которым имеют доступ все вычислительные устройства.

Определение 6. Объектно-атрибутный граф или **ОА-граф** представляет собой связку цепочек ИП (капсул), объединенных ссылками, расположенными в нагрузках ИП (ссылка принадлежит множеству адресов глобальной памяти Ω) (Рис. 1). ОА-граф является связным графом. ОА-граф можно использовать для описания сложно структурированных данных, программ и объектов.

Для отображения элементов множества Ω на ИП, расположенные в общей памяти W, вводится функция **FromMem**, а для преобразования индексов ИП в W на множество Ω - функция **ToMem**.

Для удобства анализа ОА-графа составим связку цепочек ИП V, в которую будут входить все цепочки ИП, входящие в ОА-граф, и отныне будем считать, что ОА-граф – это связка таких цепочек ИП, где ссылки, расположенные в нагрузках ИП этих цепочек, указывают только на ИП, входящие в состав ОА-графа, т.е.

$\forall\ c=\{a,l\} \in V, l \in \Omega : \mathrm{FromMem}(l) \in V$, где V – множество всех ИП в ОА-графе.

Благодаря этому введению появляется возможность сравнения ОА-графов. Дадим еще одно определение, необходимое нам для осуществления сравнения ОА-графов.

Определение 7. **Переиндексацией ОА-графа** будем называть изменение последовательности (индексов) цепочек ИП (или индексов ИП) в V.

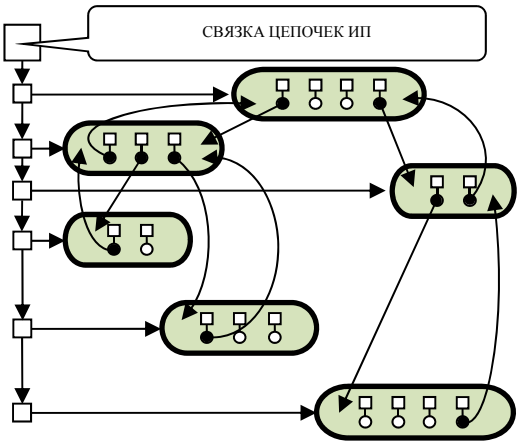

Рис. 1. Объектно-атрибутный граф

Также введем функцию индексации памяти **IdMem**, которая задает однозначное отображение элементов множества адресов глобальной памяти $\omega_i \in \Omega$, на множество индексов I ОА-графа (отображение является инъекцией) ($I=\overline{1-|V|}$). Функция IdMem понадобится для формирования ОА-графа из цепочек ИП, расположенных в глобальной памяти W.

Утверждение 2. **ОА-графы G1 и G2 эквивалентны**, если выполняются следующие условия:

1. |G1|=|G2|;
2. $\forall\ i \in 1,2,\ldots|G1|,\ G1_i=G2_i$.

Утверждение 3. **АО-графы G1 и G2 изоморфны (G1 ≈ G2)**, если выполняются следующие условия:

1. |G1|=|G2|;
2. Можно сделать такую переиндексацию ОА-графа и подобрать такие функции индексации памяти IdMem1 для G1 и IdMem2 для G2, что $\forall\ i \in 1,2,\ldots|G1|,\ G1_i=G2_i$.

Если G1 и G2 изоморфны, то это, например, значит, что оба графа описывают один и тот же объект.

Утверждение 4. **ОА-граф G1 частично изоморфен ОА-графу G2 (G1 >≈ G2)**, если можно подобрать такую переиндексацию G1 и такие функции индексации памяти для графов G1и G2, что $G1_i \cap G2_i = G1_i$, где i=1,…|G1|.

Частичная изоморфность означает, что ОА-граф G1 является подграфом G2; или, напри-

мер, обозначает, что объект G1 входит в состав объекта G2.

3. Автоматно-сетевая модель dataflow-вычислительной системы

Теперь представим обобщенное описание работы объектно-атрибутной автоматной сети (ОА-сети) как модели dataflow-вычислительного процесса.

Определение 8. **Объектно-атрибутный автомат (ОА-автомат)** представляет собой обработчик информации, представленной в виде ИП, цепочек ИП и связок цепочек ИП.

Как отмечалось выше, ОА-автомат – это модель функционального устройства в объектно-атрибутной архитектуре dataflow-ВС [3, 6]. В отличие от классического автомата, его состояние задается не дискретным конечным множеством состояний, а контекстом.

Определение 9. **Контекст ОА-автомата *B*** – это набор абстрактных сущностей любой природы (скалярные или векторные величины, счетные или непрерывные множества и т.д.).

Определение 10. Если представить каждую сущность, входящую в контекст ОА-автомата, как измерение в многомерном пространстве, то **состояние ОА-автомата *K*** будет представлять собой точку в этом пространстве (пространство состояний или фазовое пространство).

Определение 11. Изменение состояния ОА-автомата задается **функцией изменения состояния автомата *F***:

$$K'=F(K),$$

где F – функция перехода ОА-автомата из текущего состояния K в новое состояние K'.

Функцию F, например, можно задать с помощью вектора предикатов переходов $\overline{P}$ и вектора функций перехода ($\overline{F}$). При этом размерности векторов должны совпадать, т.е. $|\overline{P}| = |\overline{F}|$. Аргументами предиката P_i и функции перехода в новое состояние F_i, где i=1,2,…, $|\overline{P}|$, являются все параметры, входящие в контекст K. F_i представляет собой вектор функций, где каждая функция возвращает значение одного из параметров контекста $\overline{F}_i=[F_{i1},F_{i2},\ldots,F_{i|K|}]$, $K_j'=F_j(K)$, где j=1...$|K|$.

Переход ОА-автомата из состояния K в новое состояние K' осуществляется следующим образом: происходит перебор предикатов из вектора предикатов и, если $\exists i$: $P_i(K)$=true, то $K'=F_i(K)$ (т.е. $K_j=F_{ij}(K)$, где i=1,2,..,$|\overline{F}|$, j=1,2,…, $|K|$) и перебор заканчивается. В том случае, когда ни один из предикатов P_i не принимает истинного значения, контекст ОА-автомата остается неизменным. Функция F входит в контекст ОА-автомата и может быть изменена после перехода ОА-автомата в новое состояние: таким образом, алгоритм работы (поведение) автомата может изменяться непосредственно во время вычислительного процесса.

ОА-автоматы делятся на несколько типов, каждый из которых реализует определенный функционал и имеет свой индивидуальный контекст. Однако контекст ОА-автомата любого типа обязательно включает в себя входную и выходную очереди милликоманд (аналогично процессной сети Кана, где у каждого исполнительного устройства имеется очередь входных сообщений). Так, функция изменения контекста F может быть активизирована только в том случае, когда во входной очереди милликоманд присутствует хотя бы одна милликоманда (после активации функции F милликоманда из головы очереди милликоманд удаляется). Во время изменения своего контекста ОА-автомат может помещать информацию в конец своей выходной очереди милликоманд и тем самым передавать данные другим ОА-автоматам, для которых эта очередь является входной. При этом одна очередь милликоманд может входить в контекст одного ОА-автомата в качестве выходной и в контекст другого ОА-автомата в качестве входной очереди. В контекст любого типа ОА-автомата также в обязательном порядке входит общая оперативная память W. ОА-автоматы, таким образом, получают возможность работы с данными, расположенными в ней. Передача милликоманд между ОА-автоматами может происходить через автомат-коммутатор типа «Шина» (Bus) или напрямую.

Определение 12. Совокупность взаимодействующих ОА-автоматов будем называть **ОА-сетью**.

Структура ОА-сети, описывающей параллельный dataflow-вычислительный процесс, приведена на Рис. 2.

Определение 13. **Выполнением *S* ОА-сети** будем называть последовательность активаций функций изменения контекстов ОА-автоматов: $S=F_{i1},F_{i2}\ldots$.

ОА-сеть, в том числе, производит обработку ОА-графов, расположенных в глобальной памяти W.

Примечание: в один момент может быть активирована только одна функция изменения контекста ОА-автомата (аналогично сети Петри, где в один момент может быть активирован только один переход).

Определение 14. **Начальной конфигурацией ОА-сети** будем называть конфигурацию перед началом процесса моделирования.

Функция перехода F_i, где i есть номер ОА-автомата, считается разрешенной к активации в том случае, если во входной очереди ОА-автомата присутствует хотя бы одна милликоманда.

Определение 15. Выполнение ОА-сети будем считать завершенным в том случае, когда входные очереди всех ОА-автоматов пусты или выполняется другое условие окончания моделирования. Такая конфигурация называется **конечной конфигурацией ОА-сети**.

4. Пример ОА-сети

Применение предложенного объектно-атрибутного автоматно-сетевого формализма описания dataflow-вычислений можно продемонстрировать на примере параллельного приложения семантического разбора естественного языка (англ. Natural Language Processing, NLP).

В последнее время данное научное направление привлекает большое внимание научной общественности. Во-первых, семантический анализ востребован во многих областях: web-поиск, машинный перевод, филология и т.д. Во-вторых, в данной области остается много нерешенных проблем, включая полноценный семантический анализ естественного языка, для реализации которого необходимо разработать робастую методику семантического разбора языка. Существующие подходы [7], например, на основе теории конечных автоматов, теории

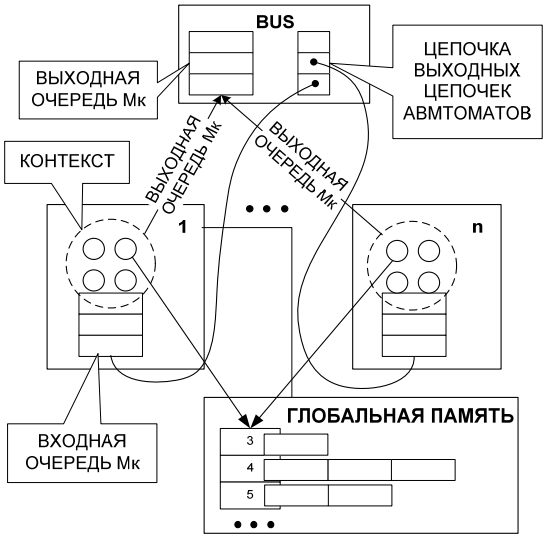

Рис. 2. Сеть ОА-автоматов

фреймов или концепции «смысл-текст» [8], не могут похвастаться надежностью, универсальностью и гибкостью. ОА-сетевой подход лучше всего подходит для решения такой задачи и имеет ряд преимуществ, включая возможность реализации объектного принципа, эффективное распараллеливание вычислений, реализацию на распределенных ВС и др.

В состав ОА-сети и для реализации вычислительного процесса семантического анализа языка должны входить ОА-автоматы (функциональные устройства) следующих типов:

- **Lexica** осуществляет обработку входной последовательности символов, выделение из нее лексем (лексический анализ), формирование ИП с описанием лексем для дальнейшей обработки.
- **IcManager** отвечает за формирование и модификацию новых ИП, цепочек ИП и ОА-графов.
- **List** – это автомат для сравнения капсулы с капсулами, находящимися в списке капсул, и выполнения последовательности действий в зависимости от результата сравнения.
- **Find** – это автомат для сравнения капсулы (цепочки ИП) с эталонной капсулой и выполнения последовательности действий в зависимости от результата сравнения.

Опишем более подробно автомат Find, в контекст которого входят ссылки на три цепоч-

ки ИП, а именно: эталонную цепочку ИП, последовательность милликоманд для выполнения в случае совпадения сравниваемой цепочки с эталонной и последовательность милликоманд для реализации при несовпадении цепочки с эталоном.

Контекст автомата Find будет представлять собой следующий набор элементов (так называемых внутренних регистров):

$$AF=\{A, L, MkIn, MkOut, W, F, T, \eta s, \eta f\},$$

где A и L – это множество атрибутов милликоманд и множество данных в нагрузке ИП, W – общая память (связка цепочек ИП);

MkIn и MkOut – очереди входных и выходных милликоманд, соответственно;

F – функция изменения контекста;

T – эталонная капсула;

ηs и ηf – капсулы, передаваемые в выходную очередь милликоманд в случаях успешного (success) и неуспешного (failure) поиска, соответственно (аналогично блокам программ then и else при условном ветвлении в классическом программировании).

Для управления автоматом Find используются следующие **атрибуты милликоманд:**

Set – установить эталонную цепочку ИП (т.е. записать в регистр T адрес эталонной цепочки в глобальной памяти);

SuccessProgSet и FailProgSet – установить цепочку ИП с программой, передаваемой в выходную очередь милликоманд в случае, если в результате сравнения с эталонной цепочкой ИП получилась непустая (success) и пустая цепочка ИП ω (failure), соответственно (т.е. запись адреса в поля ηs и ηf контекста);

FindOr и FindAnd – сравнение по правилу ИЛИ (достаточно хотя бы одного совпадения в эталонной цепочке ИП) и по правилу И (все ИП должны совпадать), соответственно, эталонной цепочки ИП с цепочкой ИП из нагрузки милликоманды.

Правила изменения контекста будут выглядеть следующим образом (справа от знака «→» находится предикат перехода, слева – функция изменения контекста ОА-автомата; знак «∧» обозначает логическую операцию «И»):

F:

F_1: MkIn=Set → T=Load(MkIn)

F_2: MkIn= FailProgSet → ηf =Load(MkIn)

F_3: MkIn= SuccessProgSet → ηs =Load(MkIn)

F_4: MkIn= FindOr ∧ Load(MkIn) ∩ T ≠ ω → MkOut= MkOut ∪ ηf

F_5: MkIn= FindOr ∧ Load(MkIn) ∩ T = ω → MkOut= MkOut ∪ ηs

F_6: MkIn= T ∩ FindAnd ∧ Load(MkIn) = T → MkOut= MkOut ∪ ηf

F_7: MkIn= T ∩ FindAnd ∧ Load(MkIn) ≠ T → MkOut= MkOut ∪ ηs

В результате выполнения правил F_4 и F_6 в выходную очередь милликоманд добавляется последовательность милликоманд, запускаемая при успешном поиске, F_5 и F_7 – при неуспехе поиска.

Семантический анализ языка в ОА-сети происходит следующим образом. Распознаваемый текст поступает на ОА-автомат Lexica, который выделяет из исходного текста лексемы и оформляет их в виде ИП. Далее лексемы обрабатываются автоматами Find и List, а автомат IcManager формирует ОА-графы, исходя из смысла, заложенного в тексте. Затем на основе запроса пользователя на анализируемом языке формируется ОА-граф запроса и производится определение частичной изоморфности ОА-графа запроса ОА-графу, описывающему смысл текста [9].

Заключение

Предлагаемый формальный аппарат описания dataflow-вычислительного процесса позволяет проводить математическое и имитационное моделирование распределенных вычислительных систем и приложений, реализуемых в dataflow-парадигме. Модель выглядит более универсальной, чем такие часто применяемые модели параллельных ВС, как автоматные, на основе сетей Петри или процессных сетей Кана, т.к. позволяет моделировать параллельный вычислительный процесс, dataflow и объектные системы.

Следует отметить, что предложенная модель уже реализована на практике в среде ОА-программирования и моделирования, с помощью которой можно как описывать алгоритм решения прикладной задачи по принципам ОА-архитектуры, так и проводить имитационное моделирование вычислительного процесса в распределенной ВС. Данная модель уже ис-

пользовалась для моделирования параллельных ВС, а также систем семантического (смыслового) распознания языка [9].

Литература

1. Бурцев В.С. Выбор новой системы организации выполнения высокопараллельных вычислительных процессов, примеры возможных архитектурных решений построения суперЭВМ. // Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ. М, 1997. URL: http://www.computer-museum.ru/books/Burcev_parallelizm.pdf
2. Carl Hewitt. A historical perspective on developing foundations for iInfo™ information systems: iConsult™ and iEntertain™ apps using iInfo™ information integration for iOrgs™ information systems. URL: http://arxiv.org/ftp/arxiv/papers/0901/0901.4934.pdf
3. Салибекян С.М., Панфилов П.Б. Моделирование суперкомпьютерной вычислительной системы объектно-атрибутной архитектуры с управлением потоком данных // Информационные технологии и вычислительные системы.—2013.—No. 1.—С.3-10.
4. Автоматное управление асинхронными процессами в ЭВМ и дискретных системах. / Под ред. В.И. Варшавского. – М.: Наука, 1986.
5. Ключев А.О., Кустарев П.В., Ковязина Д.Р., Петров Е.В. Программное обеспечение встроенных вычислительные систем. – СПб.: СПбГУ ИТМО, 2009. – 212 с. http://window.edu.ru/resource/411/63411/files/itmo368.pdf
6. Salibekyan, S.M. and Panfilov, P.B. Object-Attribute Architecture for Design and Modeling of Distributed Automation Systems. Automation and Remote Control, Vol.73, No.3/March 2012, pp.587-595. DOI:10.1134/S0005117912030174.
7. Нгуен Ба Нгок, Тузовский А.Ф. Обзор подходов семантического поиска // Доклады томского государственного университета систем управления и радиоэлектроники. Томский государственный университет систем управления и радиоэлектроники. Томск. 2010. URL: http://www.tusur.ru/filearchive/reports-magazine/2010-2-2/34.pdf
8. Мельчук И.А. Опыт теории лингвистических моделей «СМЫСЛ <–>ТЕКСТ» - М.: Школа «Языки русской литературы», 1999.
9. Салибекян С.М., Халькина С.Б., Тиновицкий К.Д. Объектно-атрибутный подход для семантического анализа естественного языка. // Объектные системы - 2014: материал VIII Международной научно-практической конференции (Ростов-на-Дону, 10-12 мая 2014 г.) / Под общ. ред. П.П. Олейника. - Ростов-на-Дону: ШИ ЮРГТУ (НПИ), 2014. - С. 80-86 URL: http://objectsystems.ru/files/2012/Object_Systems_2014_Proceedings.pdf

Салибекян Сергей Михайлович. Доцент Московского института электроники и математики НИУ «Высшая школа экономики». Окончил МИЭМ в 2000 году. Кандидат технических наук. Автор более 30 печатных работ. Область научных интересов: распределенная и параллельная обработка данных, вычислительные системы и системы имитационного моделирования, компьютерная лингвистика, системы искусственного интеллекта. E-mail**:** salibek@yandex.ru

Панфилов Петр Борисович. Профессор Школы бизнес-информатики НИУ «Высшая школа экономики». Окончил МИЭМ в 1982 году. Кандидат технических наук, доцент. Автор более 70 печатных работ и 1 монографии. Область научных интересов: распределенная и параллельная обработка данных, вычислительные системы и системы имитационного моделирования реального времени, визуальный компьютинг, интерактивные системы визуализации и виртуальные среды. E-mail: panfilov@miem.edu.ru