РАЗВИТИЕ АЛГОРИТМА МИВАРНОЙ МАШИНЫ ЛОГИЧЕСКОГО ВЫВОДА

Елисеев Дмитрий Владимирович,
г. Москва, Россия, d.eliseev@mivar.ru

Сергушин Георгий Сергеевич,
г. Москва, Россия, g.sergushin@mivar.ru

Хадиев Андрей Маратович,
г. Москва, Россия, a.khadiev@mivar.ru

Чувиков Дмитрий Алексеевич,
г. Москва, Россия, d.chuvikov@mivar.ru

Аннотация. Детально рассмотрен алгоритм поиска миварного логического вывода. Описаны используемые структуры, которые использовались для построения логического вывода. Рассмотрен алгоритм без прочистки от лишних правил, а так же способ решения данной проблемы. Рассмотрены случаи ветвления логического вывода, а так же способы организации подобного вывода в строгую последовательность правил. Приведены блок-схемы указанных алгоритмов.

Ключевые слова: мивар; логический вывод; алгоритм; миварная машина логического вывода; интеллектуальные системы

Сведения об авторах: Елисеев Д.В., к.т.н., ведущий программист, старший научный сотрудник МГТУ им. Н.Э,Баумана, НИИ «Мивар»;
Сергушин Г.С., Магист-инженер, младший научный сотрудник МАДГТУ (МАДИ), НИИ «Мивар»;
Хадиев А.М., Магист-инженер, младший научный сотрудник МАДГТУ (МАДИ), НИИ «Мивар»;
Чувиков Д.А., аспирант МАДГТУ (МАДИ), инженер-программист НИИ «МИВАР».

Миварная технология обладает большим потенциалом при должном развитии. Миварный движок, позволяющий решать задачи используя описанные в миварной нотации — первый шаг в создании интеллектуальных систем основанных на миварном подходе. В процессе создания данного движка решалось большое количество вопросов, начиная с выбора языка написания заканчивая тонкостями реализации алгоритма поиска решения. В данной статье описан алгоритм, с помощью которого происходит поиск миварного логического вывода.

Ниже представлена упрощенная блок-схема алгоритма по поиску миварного логического вывода.

В блоках 1 и 2 происходит ввод известных параметров и передаётся список искомых параметров. В блоке 3 из известных параметров формируется очередь или стек. Алгоритм позволяет использовать очереди разных типов (LIFO, FIFO…), в зависимости от выбранного вида будет меняться лишь блок 13.

В блоке 4 происходит проверка на то, известны ли уже все параметры. Она не только позволяет завершить поиск, когда все искомые параметры станут известными, но так же позволяет избежать запуск поиска алгоритма в том случае, если искомые переменные были так же переданы в списке известных параметров.

Если результатом блока 4 является ИСТИНА, то происходит вывод найденных значений и, если производился поиск, алгоритма в блоке 5. Если же результатом блока 4 является ЛОЖЬ, то запускается проверка в блоке 6.

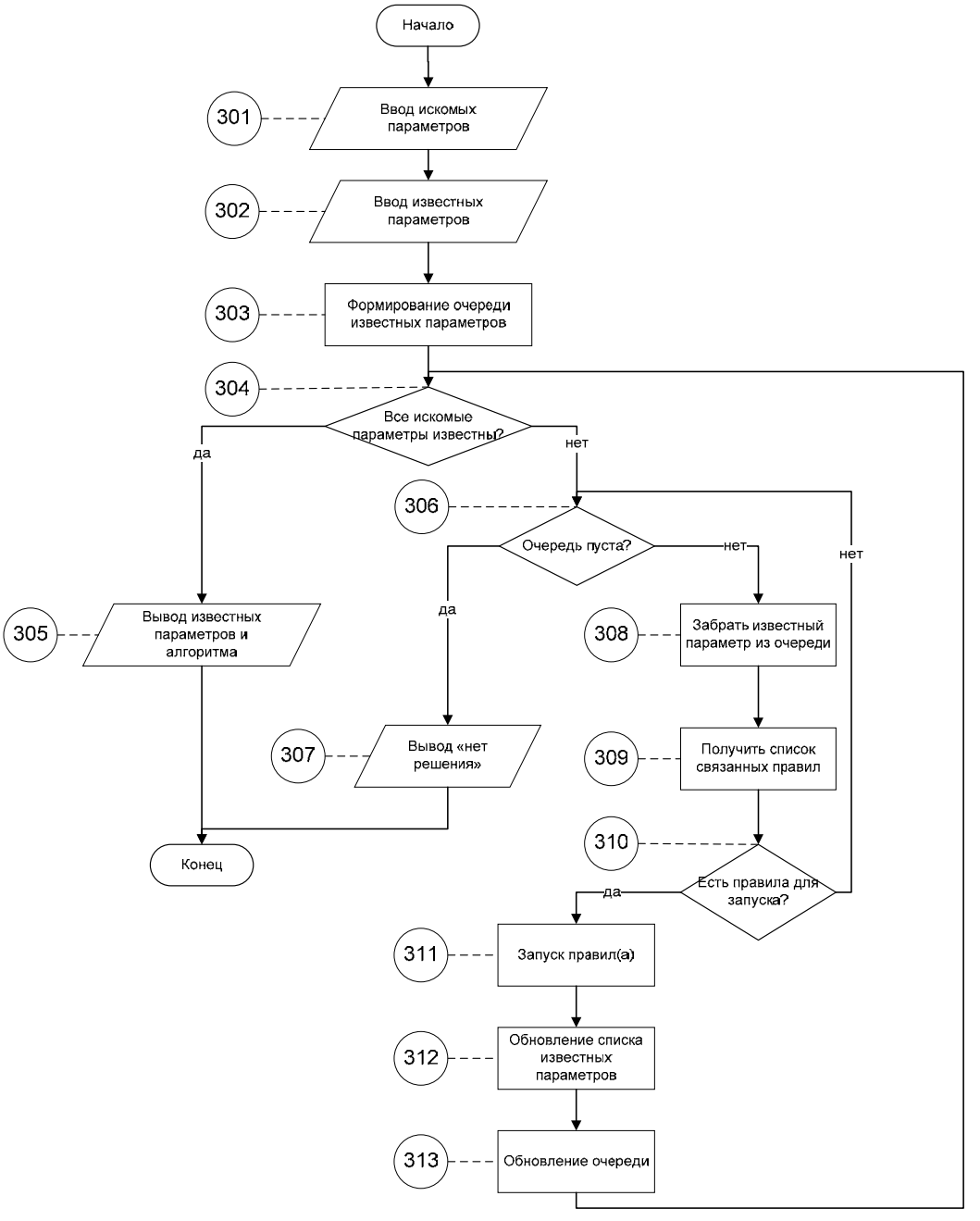

Рис. 1. Общий вид блок-схемы алгоритма работы ММЛВ

Блок 6 проверяет остались ли ещё не обработанные параметры в очереди известных. Если результатом блока 6 является ИСТИНА, то вызывается блок 7. В зависимости от конкретной реализации и цели в блоке 7 может выводится как простое «нет решения», так и вывод списка тех искомых параметров, которые удалось найти, вместе с их алгоритмом и списка не найденных параметров.

Если результатом блока 6 является ЛОЖЬ, то в блоке 8 забирается один параметр из очереди. В блоке 9 получают связанные с ним правила. В блоке 10 проверяют, можно ли запустить какие-либо правила из списка. Запустить можно лишь те правила, в которых известны все входные параметры и которые не были запущены раньше. Если нет таких правил, то возвращаются к блоку 6 и повторяют блоки 8,9,10.

Если результатом блока 10 является ИСТИНА, то в блоке 11 происходит запуск правил. По результатам блока 11 в блоке 12 обновляется список известных параметров и их значений, а в блоке 13 найденные в блоке 11 параметры заносятся в очередь известных параметров. Затем происходит возврат к блоку 4 и выполняется его проверка.

Данный метод поиска позволяет уйти от полного перебора всех возможных правил на каждом шаге и, тем самым, уйти от NP-полной задачи в сторону линейного уровня сложности.

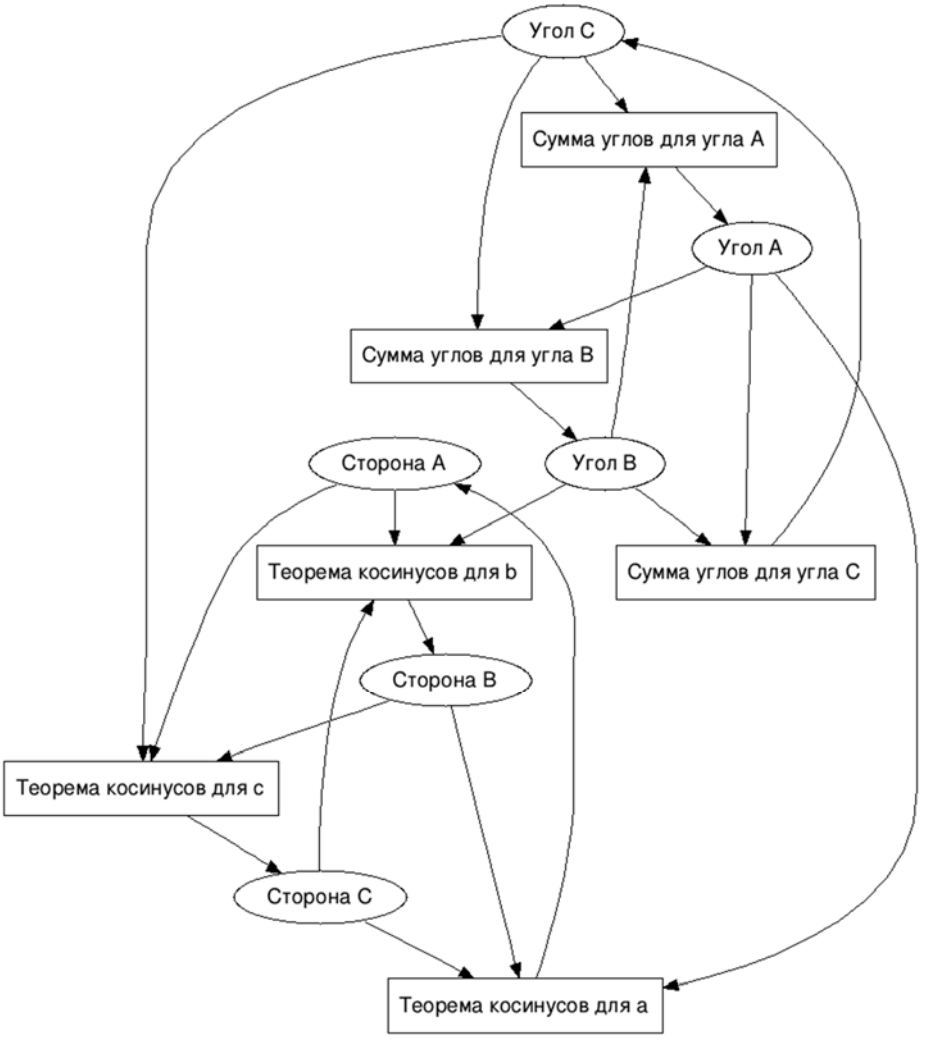

Рис. 2. Пример треугольника и графа по нему

Как говорилось выше, МС может быть изображена в виде двудольного ориентированного графа. На рис. 2 представлена схема треугольника и возможный граф по нему. В эллипсах указаны параметры, а в прямоугольниках правила. На этом графе наглядно показано, что у объектов хранится информация о том, с какими правилами они связаны (на графе это представлено в виде стрелок) и их роль в этих правилах (направление стрелок), а в правилах хранится информация о том, с какими объектами они связаны (на графе эти связи отображены в виде стрелок). В конкретных реализациях подобные связи могут быть реализованы за счёт различного рода списков. Например, внутри описания параметра «Сторона A» может лежать:

– Список правил, в которых «Сторона A» участвует в качестве входа («теорема косинусов для b», «теорема косинусов для c»);

– Список правил, в которых «Сторона A» является выходом («теорема косинусов для a»);

– Объединение предыдущих списков.

Аналогичная информация содержится в правилах в списках входных и выходных параметров. Такая архитектура позволяет уменьшить количество переборов правил на каждой итерации логического поиска по миварной матрице (блоки 9 и 10, рисунок 1), т.к. перебираются не все правила, а правила связанные лишь с выбранным параметром.

Однако, могут возникнуть ситуации, когда в ходе логического вывода в алгоритм были добавлены лишние шаги. На рисунке 3а показано изначальное состояние системы. Известен параметр P1, P4 нужно найти. На первом шаге можно запустить правила R1 и R2 и узнать P2 и P3 соответственно, что и происходит на рисунке 3б. После этого остаётся только запустить правило R3 и узнать P4.

Правило\Параметр	P1	P2	P3	P4	N+1
R1	X	Y			
R2	X		Y		
R3		X		Y	
M+1	Z			W	

а

Правило\Параметр	P1	P2	P3	P4	N+1
R1	X	Y			2
R2	X		Y		2
R3		X		Y	
M+1	Z	Z	Z	W	

б

Правило\Параметр	P1	P2	P3	P4	N+1
R1	X	Y			2
R2	X		Y		2
R3		X		Y	2
M+1	Z	Z	Z	Z	

Рис. 3. Пример лишних шагов в алгоритме
(а – изначальное состояние, б – после первого прохода, в – результат)

В итоге мы запустили правила R1, R2 и R3. Но, если присмотреться к нашей сети, станет очевидно, что правило R2 нам было не нужно для вычисления P4.

В большинстве случаев такие лишние шаги не критичны для времени расчёта. Но в целях дополнительного ускорения вычисления и создания более точных прецедентов указанный выше подход может быть расширен блоком удаления всех необязательных шагов вычисления. Для этого при стандартном поиске алгоритма в каждый найденный параметр заносится, какое правило было для этого запущено, а в каждое правило добавляется информация, из какого параметра в него пришли. Дальнейший алгоритм может быть представлен в виде блок-схемы, изображенной на рисунке 4.

На первых шагах мы получаем списки правил и параметров с информацией о том, кто кого вызвал и к кому привел. На следующем шаге формируется список из искомых параметров. В конкретных реализациях допускается использование связанных списков, так как скорость вставки в них выше, чем скорость вставки в другие структуры хранения данных. Затем очищается список найденных параметров. Во время стандартного прохода в него могли попасть параметры, чей расчёт необязателен для итогового алгоритма (см. рисунок 3).

В блоке 4 осуществляем проверку, пустой ли связанный список искомых параметров. Если результатом блока 4 является ИСТИНА, то происходит завершение алгоритма. Если результатом блока 4 была ЛОЖЬ, то вызывается блок 5.

В блоке 5 из списка забирается параметр и правило, по которому высчитали взятый параметр. В блоке 6 проверяют, добавлено ли оно в общий алгоритм.

Если результатом блока 6 является ЛОЖЬ, то происходит переход к блоку 7, где формируется локальный алгоритм вычисления. Локальный алгоритм – это список переходов от искомого параметра к известным. Формат записи локального алгоритма может отличаться в зависимости от реализации.

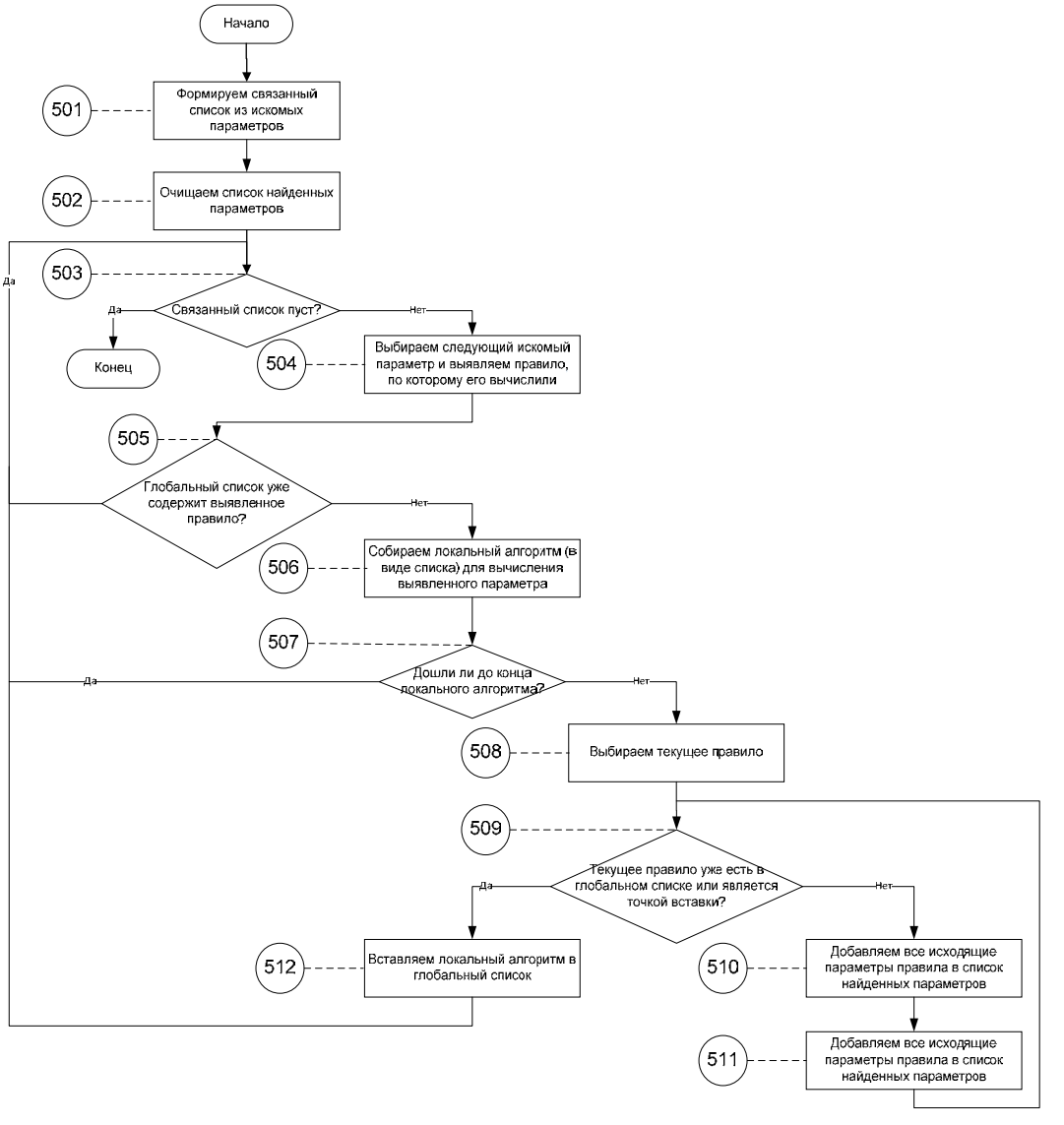

Рис. 4. Построение алгоритма логического вывода без лишних шагов

Например, в случае с P5 из рисунка 5а, локальный алгоритм может хранить в себе следующее: P5<-R3, R3<-P3, P3<-R1, R1<-P1, где «<-» обозначает направление перехода. Как уже говорилось неоднократно выше, вершины графа (параметры) знают о своих исходящих и входящих связях (правилах), а правила, в свою очередь, знают свои входные и выходные параметры, поэтому допускается более сокращённая запись алгоритма в виде последовательности правил, например R3,R1. Как видно из приведённой записи локального алгоритма и рисунка 5б, существуют ситуации, когда в правиле более одного входа и, следовательно, есть параметры, информация о которых не хранится в списке переходов. Такие параметры заносятся в начало списка искомых параметров, чтобы в дальнейшем построить их локальный алгоритм.

Так как алгоритм – последовательность шагов, то, чтобы не нарушить общий алгоритм вычисления, при добавлении параметра в список искомых параметров, в системе так же сохраняется место вставки его локального алгоритма. В общем случае, место вставки – указание, до какого правила должен быть вставлен локальный алгоритм в общий. Во время построения алгоритма могут возникнуть два класса ситуаций, они изображены на рисунке 5. Первый мы называем «веткой» («ветка вниз» - 5а, «ветка вверх» - 5б). Как видно из рисунка 5а, производя поиск P7 и P5, мы получаем две разные ветки алгоритма. В данном случае, R5 не запустится и не попадёт в алгоритм, если не будет известен P6.

Дойдя до R5, P4 будет добавлен в список искомых с указанием места вставки «до R5». Здесь стоит отметить, что в локальном алгоритме расчёта P6 встречается P4 и он будет удалён из списка искомых.

В ситуации «ветка вверх», чтобы посчитать P7, нужно сначала посчитать от P1 до P4, от P4 до P6 и от P3 до P5. Здесь место вставки R5, алгоритмы расчёта P5, P6 и P4 нужно вставить выше него.

На рисунке 5в представлена «цикличная» ситуация. Наш алгоритм на время разделяется и затем сходится опять, так как, чтобы найти P7 нам нужны и P6, и P5. Точка вставки опять же R5.

Пошаговый пример применения места вставки будет разобран после словесного описания всего алгоритма, изображенного на рисунке 4.

В блоке 8 проверяем, дошли ли мы до конца локального алгоритма. Если результатом блока 8 является ИСТИНА, то происходит переход к блоку 4. Если результатом блока 8 является ЛОЖЬ, то происходит переход к блоку 9.

В блоке 9 забирается первое правило из списка. Затем в блоке 10 проверяют, есть ли это правило в общем алгоритме, как в случае с R1 из примера, приведённого раньше или является оно местом вставки, определённом в пункте 5. Если результатом блока 10 является ИСТИНА, то происходит переход в блок 4. Если результатом блока 10 является ЛОЖЬ, то происходит переход в блок 11.

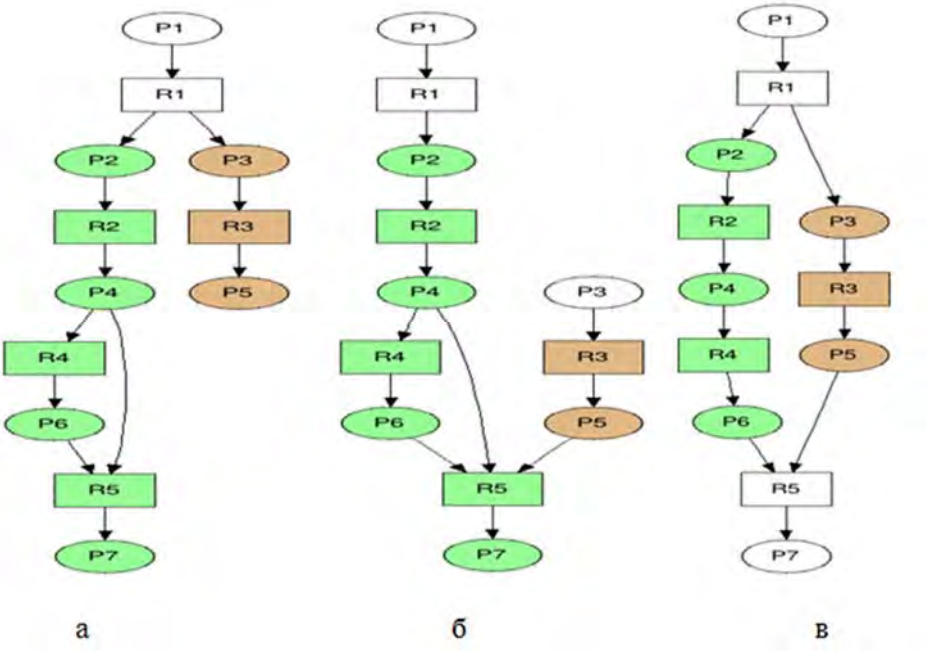

Рис. 5. Пример ветвления (а – тип «ветка вниз», б – тип «ветка вверх», в –тип «цикличное»)

В блоке 11 все выходные параметры правила добавляются в список «найденных», при этом, конечно же, они проверяются на дублирование. Заодно проверяется список искомых на наличие этих параметров. При совпадении они удаляются из списка искомых. Схожий пример мы видели при рассмотрении рис. 5а. В локальном алгоритме P6 встречалось P4, которое было удалено из списка искомых.

В блоке 12 текущее правило добавляется в общий алгоритм относительно его места вставки и происходит переход к блоку 8.

Описанный выше алгоритм позволяет очистить общий алгоритм вычисления задачи от лишних шагов и осуществить запуск только необходимых для решения задачи правил, что позволяет ускорить работу всего предлагаемого логического вывода на больших задачах.

Давайте рассмотрим ситуацию 5б более внимательно. Изначально, в списке искомых параметров у нас значится только P7. В переменной хранится информация, что её рассчитали в правиле R5. В R5 известно, что у него три входа P4, P5 и P6. Причём допустим, что “родителем”(инициатором, параметром, при обработке которого было добавлено это правило) R5 в алгоритме является P4. Поэтому P6 и P5 добавляются в начало списка известных параметров. Местом вставки их локальных алгоритмов указывается R5. Дальнейший локальный алгоритм содержит P4<-R2, R2<-P2, P2<-R1, R1<-P1. Полный спи-

сок локальных переходов для P7 выглядит следующим образом: P7<-R5, R5<-P4, P4<-R2, R2<-P2, P2<-R1, R1<-P1. В общий алгоритм мы заносим только последовательность правил и получаем: R5<-R2<-R1.

Теперь давайте построим локальный алгоритм P6. В общем виде его можно записать следующим образом: P6<-R4, R4<-P4, P4<-R2, R2<-P2, P2<-R1, R1<-P1. Берём правило R4. У него только один вход, поэтому мы не добавляем ничего в список искомых параметров. Для P6 местом вставки является R5, поэтому R4 ставим над ним. Общий алгоритм на этом шаге выглядит следующим образом: R5<-R4<-R2<-R1.Теперь берём правило R2. Оно уже есть в общем алгоритме, значит вся оставшаяся часть локального алгоритма уже содержится в общем и нет смысла тратить время на лишние проверки.

Мы извлекаем P5 из списка искомых. Его алгоритм довольно короток: P5<-R3, R3<-P3. Берём R3. Его нет в общем алгоритме, место вставки выше R5. Получаем: R5<-R3<-R4<-R2<-R1.

Теперь рассмотрим сложность переборов. Все известные элементы добавляются в очередь, далее проходя по каждому элементу из очереди, в конец очереди заносятся все смежные объекты (с указанием из какого объекта в них пришли), но объекты заносятся в очередь только в том случае, если они еще не были посещены. Благодаря этому миварный проход не может зациклиться или пройти один и тот же элемент более одного раза. Это обусловлено тем, что в процессе обхода миварной матрицы, в каждый элемент была записана информация, о том, из какого элемента произошел переход в текущий элемент миварной сети. Эта информация позволяет построить цепочки от искомых элементов к известным. Из блок-схем видно, что параметры на обработку вставляются при необходимости в список, поэтому при реализации была выбрана структура «связанный список», вставка элементов в структуры такого типа занимает O(1). В алгоритме происходит частая проверка на то, был ли добавлен элемент в логический вывод, для этого использовалась такая структура данных, как множество, для определения содержания элемента в множестве в среднем требуется время O(1). Отсюда следует, что сложность алгоритма зависит от количества объектов в модели и в данном случае стремится к O(n) (где n – количество элементов миварной матрицы, как правил, так и параметров). А при решении реальных локальных задач сложность, зачастую, гораздо меньше и равна O(m), где m – количество задействованных во время просчёта элементов и m<=n.

Что же касается когнитивных карт и онтологий, то в них используются более медленные алгоритмы поиска, зачастую, поиск в глубину. Например, когнитивные карты представляют собой взвешенный граф взаимовлияний объектов (факторов) друг на друга и имеют как положительный, так и отрицательные веса. При поиске решения осуществляют поиск пути по графу с минимальным или максимальным весом. Вследствие чего происходит перебор всех вершин, что порождает большую вычислительную сложность, в худшем случае стремящуюся к полному перебору.

Литература

1. Максимов В.И., Корноушенко Е.К., Качаев С.В. Когнитивные технологии для поддержки принятия управленческих решений.// Информационное общество, 1999, вып. 2, С. 50–54.

2. Чувиков Д.А. Разработка электронного образовательного ресурса (ЭОР) «МИВАР». «МИВАР» - логический искусственный интеллект. Саарбрюкен, Германия: LAP LAMBERT Academic Publishing Gmbh & Co. KG, 2015. 65 с. ISBN: 978-3-659-33033-9.

3. Варламов О.О. Эволюционные базы данных и знаний для адаптивного синтеза интеллектуальных систем. Миварное информационное пространство. М.: Радио и связь, 2002. 288 С.

4. Варламов О.О. Системный анализ и синтез моделей данных и методы обработки информации в самоорганизующихся комплексах оперативной диагностики : диссертация на соискание ученой степени доктора технических наук. М.: МАРТИТ, 2003. 307 с.

5. Varlamov O.O., Adamova L.E.E., Eliseev D.V., Mayboroda Yu.I., Antonov P.D., Sergushin G.S., Chibirova M.O. Mivar Thechnologies in Mathematical Modeling of Natural Language, Images and Human Speech Understanding // International Journal of Advanced Studies. 2013. Т. 3. № 3. С. 17–23.

6. Варламов О.О. Практическая реализация линейной вычислительной сложности логического вывода на правилах «если-то» в миварных сетях и обработка более трех миллионов правил // Автоматизация и управление в технических системах. 2013. № 1. С. 60–97.

7. Варламов О.О. Создание интеллектуальных систем на основе взаимодействия миварного информационного пространства и сервисно-ориентированной архитектуры // Искусственный интеллект. 2005. № 3. С.

8. Варламов О.О. Обзор 25 лет развития миварного подхода к разработке интеллектуальных систем и создания искусственного интеллекта // Труды НИИР. 2011. № 1. С. 34–44.

9. Chuvikov D.A., Kazakova N.A., Varlamov O.O., Goloviznin A.V. 3D modeling and 3D objects creation technology analysis for various intelligent systems // International Journal of Advanced Studies. 2014. Т. 4. № 4. С. 16–22. DOI: 10.12731/2227-930X-2014-4-3.

10. Варламов О.О. Разработка адаптивного механизма логического вывода на эволюционной интерактивной сети гиперправил с мультиактивизаторами, управляемой потоком данных // Искусственный интеллект. 2002. № 3. С. 363–370.

11. Варламов О.О. Основы многомерного информационного развивающегося (миварного) пространства представления данных и правил // Информационные технологии, 2003. № 5. С. 42–47.

12. Елисеев Д.В. Методика обработки темпоральной реляционной базы данных в миварном пространстве: 05.13.17: автореф. … дис. ктн; МГТУ им. Н.Э. Баумана. М., 2011. 16 с.

13. Алгебра многомерных матриц для обработки адаптируемой модели данных [Электронный ресурс] / Елисеев Д.В., Балдин А.В. Электрон. журн. М.: «Наука и образование: электронное научно-техническое издание», 2011 – Режим доступа: http://technomag.edu.ru/doc/199561.html, свободный, (Дата обращения: 17.03.2015)

14. Варламов О.О. Системы обработки информации и взаимодействие групп мобильных роботов на основе миварного информационного пространства // Искусственный интеллект. 2004. № 4. С. 695–700.

15. Варламов О.О. Анализ взаимосвязей GRID и САС ИВК, SOA и миварного подхода // Искусственный интеллект. 2005. № 4. С. 4–11.

16. Варламов О.О. Логический искусственный интеллект создан на основе миварного похода! МИВАР: активные БД с линейным логическим выводом > 3млн правил => понимание смысла+ сингулярность в виртуальной реальности. Саарбрюкен, Германия: LAP LAMBERT Academic Publishing Gmbh & Co. KG, 2012. 700 с. ISBN: 978-3-8473-1953-5.

ALGORITHM DEVELOPMENT OF MIVAR INFERENCE MACHINE

Dmitry Vladimirovich Eliseev,
Moscow, Russia, d.eliseev@mivar.ru

Georgij Sergeevich Sergushin,
Moscow, Russia, g.sergushin@mivar.ru

Andrei Maratovich Khadiev,
Moscow, Russia, a.khadiev@mivar.ru

Dmitry Alekseevich Chuvikov,
Moscow, d.chuvikov@mivar.ru

Abstract. The search algorithm of mivar inference is considered in detail. We had decribed the structure used for building inference. We consider algorithm with unnecessary rules and solution of this problem. We consider branching case of inference and ways of organizing a similar conclusion in a strict sequence of rules. Shows a block diagram of those algorithms.

Keywords: mivar; inference; algorithm; mivar inference machine; intelligent system

References

1. Maksimov V.I., Kornoushenko E.K., Kachaev S.V. (1999), “Cognitive technologies to support management decisions”, Informacionnoe obshhestvo, vol. 2, pp. 50-54.

2. Chuvikov D.A. (2015), Razrabotka elektronnogo obrazovatelnogo resursa (EOR) “MIVAR”. “MIVAR” - logicheskij iskusstvennyj intellekt [Development of electronic educational resources (EER) “MIVAR”. “MIVAR” - logical AI], Saarbrucken, Germany: LAP LAMBERT Academic Publishing Gmbh & Co. KG – 65 p. ISBN: 978-3-659-33033-9.
3. Varlamov O.O. (2002), Evolyucionnye bazy dannyx i znanij dlya adaptivnogo sinteza intellektualnyx sistem. Mivarnoe informacionnoe prostranstvo [Evolutionary knowledge and data base for adaptive synthesis of intelligent systems. Mivar information space], Moscow, Russia: Radio and communication, – 288 p. ISBN 5-256-01650-4.
4. Varlamov O.O. (2003), “System analysis and synthesis of data models and methods of information processing in the self-organizing complexes of online diagnostics”, IT, no 3, pp. 299.
5. Varlamov O.O., Adamova L.E.E., Eliseev D.V., Mayboroda Yu.I., Antonov P.D., Sergushin G.S., Chibirova M.O. (2013), “Mivar Thechnologies in Mathematical Modeling of Natural Language, Images and Human Speech Understanding”, International Journal of Advanced Studies. vol. 3, no 3, pp. 17-23.
6. Varlamov O.O. (2013), “Practical realization of linear computational complexity of logical reasoning based on “IF-THEN” rules in mivar networks and handling more than three million production rules”, Automation and control in technical systems, no 1, pp. 60-97.
7. Varlamov O.O. (2005), “Creation of intellectual systems on the basis of interaction of mivar information space and service-oriented architecture”, AI, no 3, pp.13.
8. Varlamov O.O. (2011), “Review of 25 years of development mivar approach to the development of intelligent systems and the creation of artificial intelligence”, Trudy NIIR, no 1, pp. 34-44.
9. Chuvikov D.A., Kazakova N.A., Varlamov O.O., Goloviznin A.V. (2014), “3D modeling and 3D objects creation technology analysis for various intelligent systems”, International Journal of Advanced Studies, vol. 4, no 4, pp. 16-22, DOI: 10.12731/2227-930X-2014-4-3.
10. Varlamov O.O. (2002), “Development of adaptive inference engine on the evolutionary interactive network with hyper multiactivators rules, controls the flow of data”, AI, no 3, pp. 363-370.
11. Varlamov O.O. (2003), Osnovy mnogomernogo informatsionnogo razvivayushchegosya (mivarnogo) prostranstva predstavleniya dannykh i pravil [Fundamentals of developing a multi-dimensional information (mivar) space of data and rules representation], Informatsionnyie tehnologii, , no. 5, pp. 42-47.
12. Eliseev D.V. (2011), Metodika obrabotki temporal'noi relyatsionnoi bazy dannykh v mivarnom prostranstve [Technique of processing temporal relational database in the mivar space. PhD in Technique. sci. diss.], Moscow, Bauman MSTU Publ., 149 p.
13. Eliseev D.V., Baldin A.V. (2011), Algebra mnogomernykh matrits dlya obrabotki adaptiruemoi modeli dannykh [Multidimensional matrix algebra to processing adaptive data model], Moscow, Nauka I obrazovanie: nauchnoe isdanie VGTU im. N. E. Baumana, no. 7, p. 4. Available at: http://technomag.edu.ru/doc/199561.html. (accessed 17.03.2015)
14. Varlamov O.O. (2004), Sistemy obrabotki informacii i vzaimodejstvie grupp mobil'nyh robotov na osnove mivarnogo informacionnogo prostranstva [Information processing systems and the interaction of groups of mobile robots based on the mivar information space], AI, no. 4, pp. 695-700.
15. Varlamov O.O. (2005), Analiz vzaimosvjazej GRID i SAS IVK, SOA i mivarnogo podhoda [Analysis of the relationship GRID and SAS CPI, SOA, and mivar approach], AI, no 4, pp. 4-11.
16. Varlamov O.O. (2012), Logicheskij iskusstvennyj intellekt sozdan na osnove mivarnogo poxoda! MIVAR: aktivnye bazy dannyx s linejnym logicheskim vyvodom > 3mln pravil => ponimanie smysla+ singulyarnost v virtualnoj realnosti [Logical AI was created based on mivar technologies. MIVAR: active databases with linear logical reasoning > 3 million rules > meaning understanding + singularity of virtual reality], Saarbrucken, Germany: LAP LAMBERT Academic Publishing Gmbh & Co. KG – 700 p. ISBN: 978-3-8473-1953-5.

Information about authors:

Eliseev D.V., PhD, Software Engineer of LTD. «MIVAR».
Khadiev A.M., doctoral student, Software Engineer of LTD. «MIVAR».
Sergushin G.S., doctoral student, Software Engineer of LTD. «MIVAR».
Chuvikov D.A., doctoral student, Software Engineer of LTD. «MIVAR».