ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ ПРИНЦИПА РЕЗОЛЮЦИИ С ИСПОЛЬЗОВАНИЕМ УПРАВЛЯЮЩИХ ЭВРИСТИК

Д.С. Зарецкий

1. Введение

Несмотря на свою более чем полувековую историю, автоматический логический вывод с использованием принципа резолюции до сих пор играет важнейшую роль в современной жизни. Текущие тенденции развития интеллектуальных систем, диктующие требования надежности, корректности и универсальности систем логического вывода полностью удовлетворяются алгоритмом, предложенным Дж. А. Робинсоном в 1965 году[1]. Значение же, казалось бы, неразрешимой проблемы экспоненциального роста поискового пространства приглушается за счет наращивания вычислительных мощностей, причем, не только качественного, в виде увеличения частоты процессоров, но и количественного, за счет использования параллельных реализаций алгоритмов резолюции.

Однако, как бы то ни было, попытки, если не контроля, то хотя бы снижения роста пространства поиска за счет ввода эвристических функций и предположений на данный момент являются приоритетными для разработчиков алгоритмов резолюции. К тому же, немаловажную роль начинает играть организация параллельности вычислений: стремление наиболее эффективно использовать имеющиеся, пусть и заметно увеличивающиеся год от года, вычислительные мощности наряду с оптимизацией самих вычислений также является немаловажным фактором и даже критерием оценки разрабатываемых алгоритмов.

Настоящая статья направлена на сравнение двух основных подходов к распараллеливанию алгоритма резолюции, а также исследованию двух потенциально интересных эвристик, направленных на оптимизацию и ускорение самого алгоритма.

Рассмотрим, что же из себя представляет принцип резолюции.

2. Постановка и анализ задачи.

Пусть задано некоторое множество дизъюнктов S. Основной идеей принципа резолюции является проверка: невыполнимо (противоречиво) ли множество S. Это достигается за счет попарного резольвирования всех дизъюнктов, содержащихся в S.

В силу полноты принципа резолюции для достижения требуемого результата в общем случае необходимо и достаточно обеспечить резольвирование всех дизъюнктов из S, а также полученных ранее резольвент (результатов резольвирования), если в ходе этого процесса был получен пустой дизъюнкт, следовательно, множество S невыполнимо, в обратном случае – выполнимо[2].

Таким образом, в общем случае классический алгоритм метода резолюции выглядит следующим образом:

Для каждого дизъюнкта $D_a \in S$ и для каждого дизъюнкта $D_p \in S$ произвести резольвирование. Если результат – пустой дизъюнкт, тогда завершить, если нет – добавить результат в S.

Очевидным является то, что структура алгоритма состоит из двух циклов: выбор дизъюнктов D_a (будем называть их активными дизъюнктами) и дизъюнктов D_p (пассивные). Причем, сами операции резольвирования при этом друг с другом не связаны, а значит, могут выполняться параллельно.

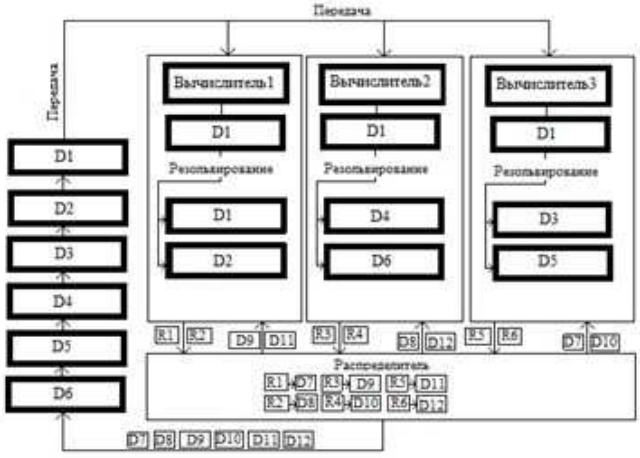


Рис. 1а. Распределение пассивных дизъюнктов

Таким образом, достаточно логичным являются два подхода к параллельной реализации метода резолюции:

- распределение по узлам-вычислителям пассивных ( ) дизъюнктов, которые в свою очередь будут резольвироваться с текущим активным дизъюнктом («пассивный алгоритм»),
- распределение по узлам-вычислителям активных ( ) дизъюнктов с задачей резольвирования оных со всеми дизъюнктами множества S (активный алгоритм).

Графически это будет выглядеть следующим образом (рис 1):

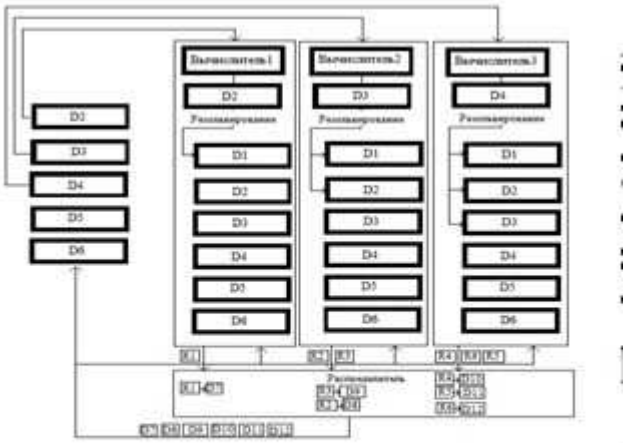


Рис. 1б. Распределение активных дизъюнктов

Несмотря на общую природу, эти два алгоритма имеют, однако, ряд отличий в значениях критериев оценки эффективности параллельной работы (Таблица 1):

Характеристика	«Активный» алгоритм	«Пассивный» алгоритм
Корректность алгоритма (требование полноты)	Есть	Есть
Максимальное количество используемой памяти	Высокое (требуется хранить копию базы дизъюнктов на каждом вычислителе). Max_mem ~ (k+1)*Dis_mem, где Mem_i = Dis_mem Dis_mem – память, требуемая для хранения всех возможных дизъюнктов. Mem_i – память, занимаемая дизъюнктами на i-ом узле	Малое (база дизъюнктов распределяется по вычислителям). Max_mem < 2*Dis_mem Mem_i – настраивается.
Требование к синхронности	Абсолютная асинхронность. Поддержание синхронности не требуется.	Требуется синхронизация.
Отношение работа/передача сообщений между узлами	Абсолютно неравномерное. Неконтролируемое.	Неравномерное. Возможна регуляция.
Максимальный объем передаваемых сообщений.	Высокий Max_mes=(k+2)*Dis_mem	Высокий Max_mes=(k+2)*Dis_mem
Централизация	средняя	высокая
Отказоустойчивость	Высокая	Высокая

Таблица 1: априорное сравнение параллельных реализаций резолюции.

Остановимся подробнее на некоторых из пунктов:

1. Корректность обоих алгоритмов очевидна: само двуцикличное построение последовательного алгоритма, а как следствие, и эквивалентных ему параллельных, характерно для задач полного перебора и гарантирует, что каждая пара дизъюнктов будет прорезольвирована, что в свою очередь в силу полноты принципа резолюции гарантирует, что пустой дизъюнкт будет выведен всегда, когда это возможно.
2. Количество памяти характеризуется тем, что у «активного» алгоритма необходимо хранить полную базу дизъюнктов на каждом узле-вычислителе, а также на узле-распределителе. В свою очередь «пассивному алгоритму» требуется хранить всю базу только на распределителе, вторая копия базы полностью распределяется между узлами-вычислителями. Кроме того, часть дизъюнктов динамически может удаляться из базы, что будет показано ниже. Помимо этого, объем необходимой памяти на каждом конкретном узле может регулироваться на фазе распределения дизъюнктов.
3. Асинхронность «активного» алгоритма очевидна. Количество резольвируемых пар на каждом узле-вычислителе абсолютно различно для каждого конкретной сессии (резольвирование «активного» дизъюнкта с «пассивной» базой). В свою очередь, не только количество, но и качество резольвируемых пар на каждом из вычислителей настраивается за счет фазы распределения. Возможно применение эвристик, сводящих сессии резольвирований к как можно более равному промежутку времени.
4. Объем передачи сообщений. Казалось бы, «пассивный» алгоритм должен по этому показателю быть более выигрышным, однако тот выигрыш, получаемый за счет отсутствия требования поддержания полной базы дизъюнктов на каждом из узлов, теряется на фазе распределения заданий.
5. Централизация «активного» алгоритма заключается только лишь в распределении «активных» дизъюнктов. Роль же «центра» в «пассивном» алгоритме значительно более высокая: только на центральном узле хранится полная база дизъюнктов, а также информация о распределении дизъюнктов по узлам-вычислителям, а это значит, что операции, требуемые наличия полной информации о системе возможны только на центральном узле.
6. Отказоустойчивость обоих алгоритмов, как любых алгоритмов, имеющих централизованную структуру достаточно высока. При отказе любого узла, кроме центрального, по имеющейся информации его структура, может быть восстановлена, а, так как после выполнения каждой сессии, идет синхронизация результатов с центральным узлом, то потеря информации будет ограничиваться рамками одной сессии.

Очевидно, что оба алгоритма имеют право на жизнь. Они оба обладают такими необходимыми свойствами как корректность и отказоустойчивость. Однако, несмотря на это, «пассивный» алгоритм менее требователен к объемам памяти, предоставляемым узлами (что, в силу объемности задачи, немаловажно) и более гибок в настройках, что позволяет не только минимизировать избыточность вычислений, но и применять различные эвристики (вроде описанного ниже метода сортировки и границы) с целью направить вычисления по более эффективному пути. Единственным недостатком алгоритма является требование к синхронизации работы узлов-вычислителей, однако, и этот недостаток может быть решен за счет применение управляющих эвристик, равномерно распределяющих нагрузки на узлы-вычислители.

Рассмотрим некоторые эвристики, позволяющие повысить эффективность алгоритма принципа резолюции.

3. Управляющие эвристики принципа резолюции.

Рассмотрим действие принципа резолюции на несложном примере:

Задача:

1. Если на кубике x лежит кубик y, то кубик x лежит под кубиком y
2. Если на кубике z лежит кубик y, а кубик x лежит под кубиком z, то x лежит под кубиком y.
3. На кубике A лежит кубик B
4. На кубике B лежит кубик C
5. Требуется доказать, что кубик A лежит под кубиком C.

Переведем эти утверждения во множество дизъюнктов, используя следующие предикаты:

- on(x,y): на кубике x лежит кубик y,
- above(x,y): кубик x лежит под кубиком y.

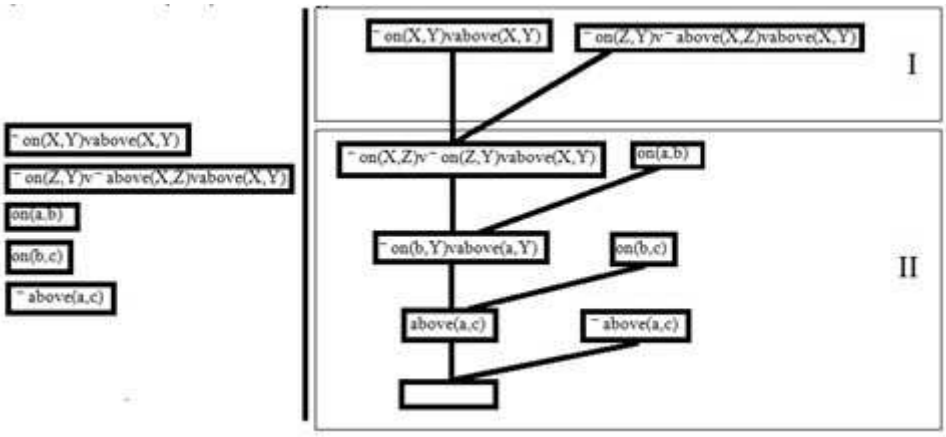
Получим следующий результат:

1) $\neg on(X,Y) \vee above(X,Y)$
2) $\neg on(Z,Y) \vee \neg above(X,Z) \vee above(X,Y)$
3) $on(a,b)$
4) $on(b,c)$
5) $\neg above(a,c)$

Как мы видим, процесс вывода пустого дизъюнкта можно разделить на два этапа:

На первом этапе (I), назовем его конструктивным, из дизъюнктов, чья длина – две и три литеры соответственно был получен дизъюнкт длины три.

На втором же этапе (II), будем называть его деструктивным, за счет присутствия в паре дизъюнкта единичной длины длина результирующего дизъюнкта поступательно уменьшалась, пока не был получен пустой дизъюнкт.



На самом деле этот момент весьма показателен и прослеживается во всех случаях применения принципа резолюции. Действительно, любой вывод, каким бы сложным и неочевидным бы он ни был, всегда заканчивается серией применения дизъюнктов единичной длины, причем, чем более «длинные» дизъюнкты применялись на ранних шагах вывода, тем большее количество более коротких дизъюнктов требуется применить, чтобы «разрушить» результат их применения. Таким образом, можно с высоким уровнем уверенности утверждать обратную зависимость между длинами дизъюнктов и частотой их применения, из чего в свою очередь следует вывод: имеет смысл применять более короткие дизъюнкты на ранних стадиях, чтобы сокращать длину имеющихся в базе дизъюнктов.

Кроме того, классический принцип резолюции имеет одну весьма неприятную особенность: дизъюнкт, получаемый в ходе вывода, добавляется в конец. Это означает, что прорезольвировать его можно будет только тогда, когда будут прорезольвированы все дизъюнкты, находящиеся выше него. Таким образом, даже имея в множестве дизъюнктов пару дизъюнктов, дающих пустой дизъюнкт, результат будет получен только тогда, когда до этой пары дойдет очередь, что приводит к серьезным временным потерям.

Обе описанные проблемы может помочь решить эвристика, основанная на динамическом упорядочивании базы дизъюнктов S в ходе работы алгоритма. Ее суть заключается в следующем: «активная» часть дизъюнктов (то есть та часть, из которой еще не выбирались активные дизъюнкты) всегда должна быть упорядоченной, что позволяет в качестве активных дизъюнктов выбирать дизъюнкты наименьшей длины. Конечно же, упорядочивать множество S каждый раз после добавления новой резольвенты достаточно накладно с точки зрения расхода вычислительных мощностей, однако, эта проблема решается с помощью приема, аналогичного приему, используемому в алгоритме сортировки включением. А именно: пусть исходная база дизъюнктов упорядочена, а в качестве последнего дизъюнкта был выбран дизъюнкт под номером i. Тогда, чтобы сохранить остаток множества дизъюнктов упорядоченным, нужно включить новый дизъюнкт D перед первым дизъюнктом, имеющим такую же длину. С точки зрения вычислительной сложности данная операция несравнимо проще сортировки, и позволяет осуществить добавление нового дизъюнкта без существенной затраты времени.

Использование подобного подхода позволяет решить такие проблемы как:

- наращивание ненужных «мусорных» дизъюнктов в ходе резольвирования дизъюнктов большой длины – в первую очередь будут резольвироваться дизъюнкты, направленные на уменьшение длины результирующего дизъюнкта,
- избыточные вычисления перед получением пустого дизъюнкта – если дизъюнкт, требуемый для вывода пустого дизъюнкта, был получен, то, в силу его обязательной однолитеральности, он уже на следующем шаге будет выбран в качестве активного.

Кроме того, за счет упорядочивания системы дизъюнктов достигается равномерность заполнения узлов-вычислителей, что немаловажно с точки зрения параллельной реализации алгоритма.

Также, очевидно, что подобное переупорядочивание дизъюнктов никак не влияет на полноту алгоритма, а значит, обеспечивает его корректность.

Однако, несмотря на всю свою привлекательность, упорядочивание дизъюнктов не является панацеей. Рассмотрим следующий пример, являющийся частью вывода в задаче Steamroller[3].

I. A(W)
II. A(F)
III. ¯E(F,W)
IV. ¯A(x)v ¯A(z)v E(x,z)

В силу специфики задачи, результативное (приводящие в конечном итоге к пустому дизъюнкту) применение дизъюнкта I к дизъюнкту IV может быть осуществлено только после того, как к нему будет применен дизъюнкт II или дизъюнкт III. А это, в свою очередь, представляется возможным тогда и только тогда, когда в качестве активного дизъюнкта будет выбран результат резольвирования дизъюнктов II и IV или III и IV, то есть не однолитеральный дизъюнкт. Понятно, что в случае, если в задаче порождается огромное количество дизъюнктов длины большей единицы (как раз как в случае с задачей Steamroller), то необходимый дизъюнкт может быть погребен под этой лавиной, что вызывает желание по возможности сократить пространство поиска. Для этого может быть использована эвристика, основанная на методе границы.

Основной идеей метода границы является ввод динамически изменяемого ограничения на некоторый параметр, используемый при выводе. Пусть таким параметром будет длина одного из дизъюнктов. Введем порог, чье значение изначально равно единице. Далее, будем резольвировать только ту пару дизъюнктов, в которой есть хотя бы один дизъюнкт, чья длина не превышает заданного порога. В случае, если результат (пустой дизъюнкт) не был получен, то увеличиваем порог на единицу и продолжаем работу. Когда величина порога станет равна длине максимального дизъюнкта, то эвристика сама собой станет обычной резолюцией. Если и при этом не будет получен результат, то это будет означать в силу полноты резолюции, что множество дизъюнктов выполнимо, при этом значение порога станет больше длины максимального дизъюнкта, что и будет сигнализацией к завершению работы.

Как и первый метод динамической сортировки, очевидно, что метод границы сохраняет полноту задачи, более того, оба эти метода имеют перекрывающиеся цели – резольвирование более коротких дизъюнктов. Это позволяет использовать их по отдельности. Однако, в случае их общего использования, эвристика порога вступает в работу только тогда, когда заканчиваются все активные однолитеральные дизъюнкты, после чего позволяет более эффективно использовать дизъюнкты большей длины, «возвращая» управление методу сортировки, как только появляется первый дизъюнкт, чья длина не превышает значение порога. Вкупе с низкой вычислительной стоимостью метода границ это делает совместное использование метода границ и динамической сортировки крайне предпочтительным.

4. Сравнение методов в реальных условиях.

Перед началом работы была поставлена задача: сравнить два подхода к параллельной реализации принципа резолюции, а также влияния управляющих эвристик на процесс ускорения вычислений. В качестве прототипа «активного» алгоритма была выбрана реализация алгоритма Anode[4]. «Пассивный» же алгоритм представлен модулем Passive, реализованным в рамках данной работы, оснащенным также описанными выше эвристиками.

Для тестирования алгоритмов были выбраны классические задачи логического вывода[5]:

- ханойские башни с 5 кольцами
- ханойские башни с 6 кольцами
- steamroller

В качестве критерия эффективности распараллеливания было выбрано параллельное ускорение, в качестве критерия эффективности применения эвристик – чистое время выполнения. Также для «пассивного» алгоритма представлены диаграммы загруженности процессоров, показывающие возможности саморегулирования системы за счет применения эвристики упорядочивания

Результаты (Таблицы 2-4) были получены с использованием двухядерной системы Intel(R) Core(TM)2 Duo CPU P8700 @ 2.53GHz, объем ОЗУ: 3 Гб, Операционная система Windows 7.

Ханойские башни(5 колец):

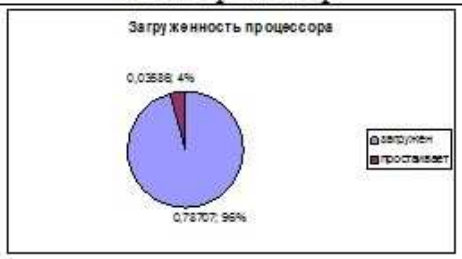
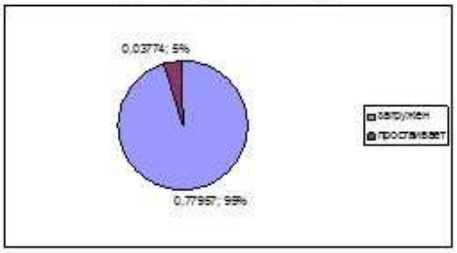
1-ый процессор	2-ой процессор
	

Таблица 2а. Загруженности процессоров.

категория	Anode	Различие	Passive
Время выполнения на 1 узле. (сек)	65,115	39,367	1,65407
Время выполнение на 2 узлах (сек)	39,343	48,091	0,8181
Ускорение	1,655	22,18%	2,022

Таблица 2б. Сравнительные характеристики алгоритмов Anode и Passive.

Ханойские башни(6 колец):

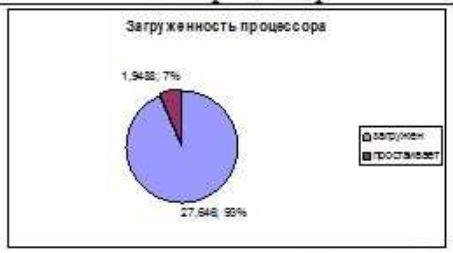
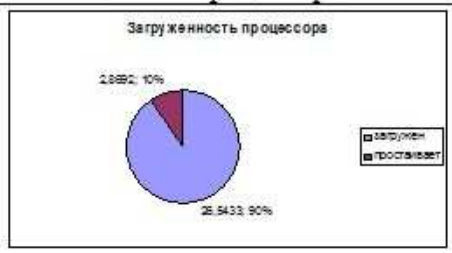
1-ый процессор	2-ой процессор
	

Таблица 3а. Загруженности процессоров.

категория	Anode	Различие	Passive
Время выполнения на 1 узле. (сек)	7500,1344	132,988	56,3972
Время выполнение на 2 узлах (сек)	6151,1532	209,113	29,4154
Ускорение	1,219	56,030%	1,902

Таблица 3б. Сравнительные характеристики алгоритмов Anode и Passive.

Steamroller:

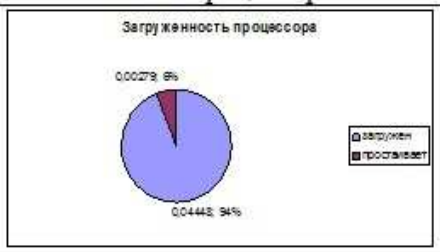
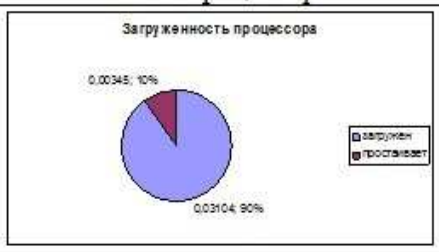
1-ый процессор	2-ой процессор
	

Таблица 4а. Загруженности процессоров.

категория	Anode	Различие	Passive
Время выполнения на 1 узле. (сек)	3,2526	41,615	0,07816
Время выполнение на 2 узлах (сек)	2,7191	58,100	0,0468
Ускорение	1,196	39,632%	1,670

Таблица 4б. Сравнительные характеристики алгоритмов Anode и Passive.

4. Заключение.

В данной статье были описаны и исследованы два направления параллельной реализации принципа резолюции для систем дедуктивного вывода, а также предложены и исследованы два эвристических метода, позволяющие ускорить процесс получения результата. Сравнительное исследование показало значительный прирост эффективности по сравнению с обычным алгоритмом.

Конечно же, на основании полученных результатов сложно утверждать, что данный прирост эффективности обусловлен не спецификой задачи и такие же результаты следует ожидать в общем случае, однако, это является достаточным поводом для дальнейшего исследования.

Кроме того, хотелось бы заметить, что кроме эвристики упорядочивания алгоритм Passive не использует больше никаких механизмов управления процессом загруженности вычислительных узлов, а значит, также является предметом для дальнейшего исследования.

Тем не менее, первичное экспериментальное тестирование выявило, что описанный в статье алгоритм и предложенные эвристики представляют собой достаточный интерес и обладают немалым потенциалом, а очерченные направления усовершенствования позволяют говорить о перспективности дальнейших исследований.

ЛИТЕРАТУРА:

1. Robinson J.A. «A Machine Oriented Logic Based on the Resolution Principle» Journal of the ACM, vol. 12, 1965.
2. Ч.Чень, Р.Ли «Математическая логика и автоматическое доказательство теорем», М.: Наука,1983 – 358с.
3. Stickel M.E. «Schubert's Steamroller Problem: Formulations and Solutions» Journal of Automated Reasoning, vol. 2, 1986.
4. В.П. Кутепов, К.Ю. Хотимчук «Параллельный логический вывод на многоядерном Компьютере», 2009.
5. В.Н. Вагин, Е.Ю. Головина, А.А. Загорянская, М.В. Фомина. Достоверный и правдоподобный вывод в интеллектуальных системах. Под ред. В.Н.Вагина и Д.А.Поспелова. М.: Изд. 2-ое исправл. и дополн. Физматлит, 2008, 714с.