УДК 004.89

И. А. Бессмертный

ПРИМЕНЕНИЕ РЕЛЯЦИОННЫХ ОПЕРАЦИЙ ДЛЯ ЛОГИЧЕСКОГО ВЫВОДА В ПРОДУКЦИОННЫХ СИСТЕМАХ

Рассматривается задача логического вывода в системах искусственного интеллекта, использующих большое число фактов и правил. Основной проблемой таких систем является быстрый рост сложности логического вывода с увеличением числа фактов и правил. Предлагается устранить проблему комбинаторной сложности задачи логического вывода путем использования операций реляционной алгебры над множествами кортежей переменных. Приводятся результаты измерений скорости логического вывода в среде СУБД MS ACCESS и в среде Prolog.

Ключевые слова: *искусственный интеллект*, *реляционная алгебра*, *логический вывод*.

Введение. Исследования в области практической реализации искусственного интеллекта (ИИ), моделирующего мыслительные процессы, показывают необходимость создания обширной базы знаний, состоящей из миллионов фактов [1]. В отличие от экспертных систем, ориентированных на решение конкретных задач, системы ИИ, позволяющие порождать новые знания, должны использовать неизмеримо большее число правил. Комбинаторная сложность задачи поиска решений делает невозможным создание программ для извлечения новых знаний классическими методами логического вывода [2]. Существующие методы ускорения логического вывода, в частности алгоритм *Rete* [3], предполагают предварительную подстановку фактов в условия правил, т.е. по сути полное решение задачи. В результате пользователь сразу получает список решений для каждого выбранного правила. „Узким“ местом алгоритма *Rete* является необходимость его повторного запуска после каждого изменения базы фактов, чем и обусловлено ограниченное применение этого алгоритма.

В настоящей работе рассматривается возможность применения теоретико-множественных операций к задаче логического вывода в продукционных системах аппарата, в частности, аппарата реляционных СУБД.

Модель базы знаний. Основной единицей знаний (атомом) в семантической сети является факт вида „субъект—отношение—объект“ или „субъект—свойство—значение“. Множество фактов базы знаний $F = \{f\}$ образуют триплеты $f = (s,p,o)$, или $f = (s,p,v)$, где s — субъект, p — предикат, o — объект, v — значение. Мощность множества фактов обозначим n. Множества субъектов и объектов связываются друг с другом предикатами, образуя сетевую структуру. Данная модель позволяет воспроизвести как реляционный граф, в котором субъект и объект — сущности, так и граф *Растье* [4], где субъект — действие или процесс.

Продукционная модель представления знаний помимо фактов основывается на правилах вида ЕСЛИ—ТО. Наиболее известным языком представления правил в семантической сети является SWRL, имеющий синтаксис расширения языка XML (<http://www.w3.org/Submission/SWRL/>). Применение правила заключается в интерпретации кортежа $<R, EC, ER, L, S, LV>$, где R — множество ресурсов, LV — множество литералов, $LV \subseteq R$, EC — отображение классов и типов данных на подмножества R и LV, ER — отображение свойств на бинарные отношения в R, L — отображение литералов, используемых в правиле, на элементы LV, S — отображение индивидуальных имен, заданных в правиле, на элементы EC.

В связи с нечитаемостью текстов SWRL в синтаксисе XML для редактирования используется другое представление, более удобное для пользователя и близкое к синтаксису языка Prolog:

$$\text{hasParent}(?\text{x1},?\text{x2}) \wedge \text{hasBrother}(?\text{x2},?\text{x3}) \Rightarrow \text{hasUncle}(?\text{x1},?\text{x3}).$$

Здесь с вопросительного знака начинаются переменные, а предикат находится не в центре триплета, а вынесен за скобки. Таким образом, в большинстве правил условия совпадают с фактами базы знаний, а переменные получают значения в ходе развертывания правил.

Пусть правило имеет k условий $c(s_1,p_1,o_1)$, $c(s_2,p_2,o_2)$,..., $c(s_k,p_k,o_k)$, где p_i — предикат, s_i, o_i — либо константа, либо переменная. Правила такого вида имеют ограниченную логику, поскольку позволяют устанавливать только совпадение с фактами. Стандарт SWRL предусматривает возможность наряду с предикатами из фактов использовать в правилах функции над переменными. Для целей настоящего исследования ограничимся предикатами неравенства $c(s_i,differentFrom,o_i)$. Заметим, что в стандарте SWRL отсутствуют отрицания. На первый взгляд, это существенно ограничивает возможности правил, однако вполне соответствует концепции открытого мира (*Open World Assumption*), в которой отсутствие информации о факте не означает автоматически отсутствия факта. Предикаты также не могут быть переменными, что соответствует логике первого порядка. Следует отметить, что написание очень сложных правил требует высокой квалификации, в то время как любые сложные правила могут быть подвергнуты декомпозиции. Кроме того, логический вывод из цепочек правил может быть более быстрым за счет анализа и отсечений промежуточных результатов, заведомо нерелевантных запросу.

Наивный логический вывод. Логический вывод из правила, состоящего из условий $c(?x_i,p_i,?y_i)$, представляет собой решение системы уравнений с количеством неизвестных, равным числу переменных в правиле. Одну часть уравнений образуют пары $?x_i = ?y_j$, если переменная используется более чем в одном условии правила, вторую — уравнения $?x_i$=*значение*, $?y_i$=*значение* или неравенства $?x_i \neq$*значение*, $?y_i \neq$*значение*, получаемые подстановкой в условия правила подходящих фактов $f=(s,p,o)$. Наивный вывод состоит из следующих шагов.

1. С условием правила $c(?x_1,p_1,?y_1)$ сопоставляется первый подходящий факт $f_1 = (s_1,p_1,o_1)$, и если переменные свободны, то они получают значения $?x_1 = s_1$, $?y_1 = o_1$, а если какая-либо переменная уже связана, то выполняется ее сравнение с константой из факта.

2. Если сопоставление условия правила с фактом успешно, то выполняется переход к следующему условию правила, иначе машина вывода пытается сопоставить условие правила со следующим фактом, т.е. выполняется откат после неудачи.

3. Если все факты исчерпаны, а условие правила не выполнено, происходит откат к предыдущему условию правила и в него подставляется очередной факт.

4. После успешного выполнения всех условий значения переменных подставляются в результирующую часть правил. Чтобы получить все возможные решения, выполняется откат, как если бы правило не было решено успешно.

Таким образом, наивный вывод является чрезвычайно простым, но требует очень много времени. Общее число попыток унификации k условий с n фактами может достигать n^k.

Обработка правил в среде СУБД MS ACCESS. В работе [5] представлен метод индексации и предварительного отбора фактов для условий правила. Использование предварительного отбора фактов позволяет сократить размерность задачи поиска решений обратно пропорционально числу фактов, задействованных для каждого правила. Если обработке подвергается существенная часть всей базы знаний, эффект индексации нивелируется. В качестве побочного эффекта индексации фактов обнаружилось, что для простых правил, включающих в себя единственную переменную, предварительный отбор фактов сразу дает искомый результат без подстановки в правило. Рассмотрим, можно ли обойтись без подстановки переменных в правила в более сложных случаях — когда число переменных в правиле больше единицы.

Пусть в результате отбора фактов для условий правила $c(s_1,p_1,o_1)$, $c(s_2,p_2,o_2)$,..., $c(s_k,p_k,o_k)$, где s_i, o_i — либо константа, либо переменная, получены множества кортежей $T = \{\{t_i\}\}$, $t_i=(x_{i1},x_{i2})$, если в условии правила две переменные, $t=(x_{i1})$ — если одна переменная. Таким образом, получаем k таблиц приблизительно следующего вида.

x_{11}	x_{12}

x_{21}	x_{22}

... | x_{i1} | ... | x_{k1} | x_{k2} |

Каждая таблица должна иметь по меньшей мере одну общую переменную хотя бы еще с одной таблицей. В противном случае результат будет состоять в декартовом произведении с данной таблицей, что обычно лишено смысла. Таблицы могут иметь связи следующих типов.

1. Соединение двух таблиц по совпадению значений одной или более переменных у пары таблиц. В этом случае две таблицы объединяются реляционным оператором INNER JOIN.

2. Фильтрация таблицы по условию сравнения значения переменных между собой или переменной и значения. Данная функция выполняется с помощью условия WHERE и операторов сравнения.

Рассмотрим на конкретных примерах, как условия правил могут транслироваться в операторы SQL. Для упрощения импорта базы знаний в СУБД создадим для фактов единственную таблицу F, состоящую из столбцов S, P, O, для хранения субъекта, предиката и объекта соответственно. Запрос на языке SQL для вывода фактов о вышеупомянутом отношении „дядя—племянник(ца)“ будет выглядеть следующим образом:

```
SELECT F_1.Object AS Uncle, F.Object AS Nephew
FROM F INNER JOIN F AS F_1 ON F.Subject=F_1.Subject
WHERE (((F.Predicate)="is_parent") AND ((F_1.Predicate)="has_brother")).
```

Более сложное правило для отношения „падчерица“ на языке SWRL

$$\text{hasChild}(?x1,?x2) \wedge \text{hasSpouse}(?x1,?x3) \wedge \text{hasSex}(?x2,\text{female}) \wedge \text{hasChild}(?x4,?x2) \wedge \text{differentFrom}(?x1,?x4) \wedge \text{differentFrom}(?x3,?x4) \Rightarrow \text{hasStepDaughter}(?x3,?x2)$$

будет иметь следующий эквивалент на языке SQL:

```
SELECT DISTINCT F_1.Object AS Step_parent, F_2.Object AS Step_daughter
FROM ((F INNER JOIN F AS F_1 ON F.Subject=F_1.Subject)
INNER JOIN F AS F_2 ON F.Object=F_2.Object)
```

```
INNER JOIN F AS F_3 ON F_3.Subject=F_2.Object
WHERE (F.Predicate="is_parent" And F_1.Predicate="is_spouse" And
F_2.Predicate="is_parent" And F.Subject<>F_2.Subject And F_1.Object<>F_2.Subject
And F_3.Predicate="has_sex" And F_3.Object="female").
```

Таким образом, поскольку мощность языка SQL существенно выше, чем SWRL, любая конструкция SWRL легко транслируется в SQL.

Время обработки правил SWRL как запросов SQL. Среднее время обработки одного правила T с использованием СУБД равно

$$T = \frac{t_{\text{load}}}{m} + \left(t_{\text{sel}} + t_{\text{store}}\right),$$

где t_{load} — время загрузки фактов в базу данных, t_{sel} — время выполнения запроса, t_{store} — время добавления результатов запроса в новые факты базы знаний, m — количество запросов к СУБД в ходе поиска решения.

Для определения времени поиска с помощью генератора случайных чисел была создана база фактов, отражающих отношения „родителя“ и „супруга“. Длительность логического вывода вычислялась для SQL-запросов, выполняемых в среде СУБД MS ACCESS, в сравнении с наивным выводом, реализованным на языке Visual Prolog 7.2 и представленным в работе [6]. На языке Prolog помимо наивного вывода были также реализованы правила, выполняющие операцию INNER JOIN, в сочетании с индексацией фактов.

На рис. 1 приведены графики зависимости времени обработки правила (N — количество фактов) для отношения „отчим/мачеха—пасынок/падчерица“, которые показывают, что обработка запросов в среде СУБД выполняется более чем на три порядка быстрее, нежели наивный логический вывод (кривая *1*). Реализация логического вывода методами реляционной алгебры в среде Prolog (*3*) показывает скорость, вполне сопоставимую с SQL-запросом в СУБД ACCESS (*2*). Тем не менее достигнутое ускорение по меньшей мере не хуже публикуемых данных для данного алгоритма по сравнению с *Rete* [3].

На рис. 2 приведены данные о длительности загрузки данных и индексации фактов в СУБД ACCESS (кривая *1*) и в Prolog-программу (*2*). СУБД здесь демонстрирует приблизительно в пять раз более высокую скорость. Если на одной базе знаний обрабатывается множество правил, то среднее время обработки одного факта в СУБД и в Prolog-программе нивелируется, как показывает формула.

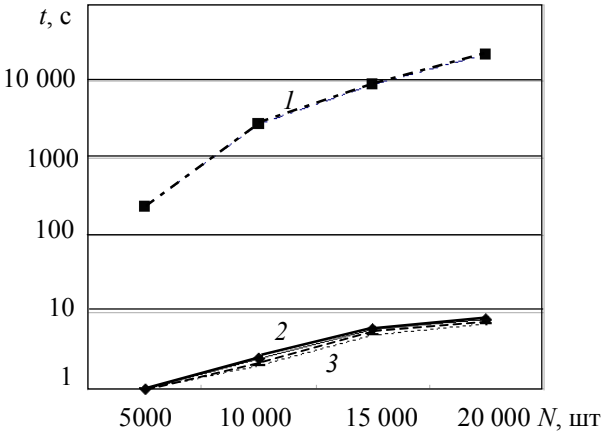

Рис. 1

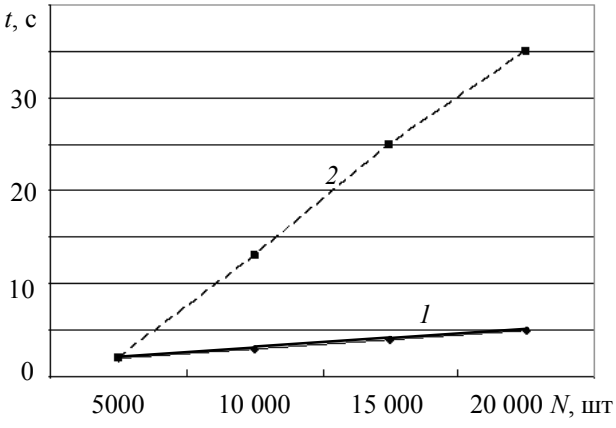

Рис. 2

Заключение. Приведенные выше результаты исследования демонстрируют возможность реализации быстрого логического вывода в продукционных моделях знаний с использованием методов реляционного исчисления. Ускорение по сравнению с наивным логическим выводом превышает три порядка, что вполне сопоставимо с последними версиями алгоритма *Rete* при отсутствии недостатков, присущих ему. Недостатками СУБД применительно

к данной задаче являются относительно большое время загрузки данных и необходимость разработки транслятора с языков представления правил, например, SWRL, в SQL-запросы.

СПИСОК ЛИТЕРАТУРЫ

1. 2001's Computer as Dream and Reality. The Discover Interview: Marvin Minsky [Electronic resource]: <http://discovermagazine.com/2007/jan/interview-minsky/>.
2. *Doorenbos R. B.* Production Matching for Large Learning Systems. PhD Theses. University of South California, 1995. 208 p.
3. *Forgy C. L.* RETE: A fast algorithm for the many pattern / many object pattern match problem // Artificial Intelligence. 1982. Vol. 19. P. 17—37.
4. *Héber L.* Tools for Text and Image Analysis: An Introduction to Applied Semiotics. Paris: ADAGP; Montréal: SODRAC, 2006.
5. *Бессмертный И. А.* Теоретико-множественный подход к логическому выводу в базах знаний // Науч.-технич. вестн. СПбГУ ИТМО. 2010. Вып. 66. С. 43—48.
6. *Bessmertny I.* Visual Prolog and Semantic Networks at Knowledge Visualization // Proc. of Visual Prolog Application & Language Conf.: VIP-ALC ’08. St. Petersburg, Russia, 2008. P. 107—111.

Сведения об авторе

Игорь Александрович Бессмертный — канд. техн. наук, доцент; Санкт-Петербургский государственный университет информационных технологий, механики и оптики, кафедра вычислительной техники;
E-mail: igor_bessmertny@hotmail.com

Рекомендована кафедрой вычислительной техники

Поступила в редакцию 16.04.10 г.