УДК 004.89

И. А. Бессмертный

МЕТОДЫ ПОИСКА ИНФОРМАЦИИ В ПРОДУКЦИОННЫХ СИСТЕМАХ

Представлен обзор основных направлений исследований и достигнутых результатов в области построения машин логического вывода в продукционных системах: в частности, индексации и предварительного отбора фактов, применения методов реляционной алгебры для логического вывода, в том числе систем управления базами данных и операций над множествами в среде Prolog.

Ключевые слова: *продукционные системы, реляционная алгебра, язык Prolog.*

Введение. Представление знаний в виде фактов и правил совместно с их извлечением посредством машины логического вывода образуют продукционную систему. Идея продукционной модели данных получила материальное воплощение в экспертных системах, которые активно разрабатывались в 80-е гг. XX века [1]. Увеличение объемов баз знаний сделало актуальной проблему комбинаторной сложности задачи логического вывода. Одним из методов, успешно ускоряющим логический вывод, стал алгоритм RETE [2], используемый, в частности, в популярной оболочке экспертных систем JESS (www.jessrules.com).

Объявленная W3C консорциумом (World Wide Web Consortium) концепция глобальной семантической сети (Semantic Web) [3] также предполагает использование продукционной модели для представления знаний. Поскольку объемы баз знаний, образуемых ресурсами сети Интернет, могут значительно превышать масштабы экспертных систем, исследование методов логического вывода является целесообразным и актуальным. В настоящей статье приводится краткий обзор исследований в данной области, проводимых на кафедре вычислительной техники Санкт-Петербургского государственного университета информационных технологий, механики и оптики.

Индексация и предварительный отбор фактов. Пусть множество фактов базы знаний $F = \{f\}$ образуют атомы (триплеты) $f = (s, p, o)$, где s — субъект, p — предикат, o — объект. Проиндексируем факты следующим образом. Присвоим каждому факту в базе знаний порядковый номер i, тогда нумерованный факт может быть описан следующим образом:

$$f(i) = (i, s, p, o).$$

Для множества термов $T = \{t\}$, содержащихся в фактах, сформируем индекс в виде

$$X = \{x\} = \{(t, w, \{i_{tw}\})\},$$

где w — место данного терма в атоме (в качестве субъекта, предиката или объекта), $\{i_{tw}\}$ — множество номеров фактов, имеющих терм t в качестве w=(s; p; o).

Факты используются правилами, тело каждого из которых состоит из множества условий $C = \{c_1, c_2, \ldots, c_k\}$, где $c_j = (s_j, p_j, o_j)$, s_j — субъект, p_j — предикат, o_j — объект, при этом s; $o = (t; v)$, v — переменная, $p_j = t$. Для каждого из условий c_j правила извлечение релевантных фактов для перечисленных сочетаний термов заключается в нахождении пересечений множеств индексов:

$$I_j = \{i_{ts}\} \cap \{i_{tp}\} \cap \{i_{to}\}, s_j = \text{const}, p_j = \text{const}, o_j = \text{const};$$

$$I_j = \{i_{tp}\} \cap \{i_{to}\}, p_j = \text{const}, o_j = \text{const};$$

$$I_j = \{i_{ts}\} \cap \{i_{tp}\}, s_j = \text{const}, p_j = \text{const};$$

$$I_j = \{i_{tp}\}, p_j = \text{const}.$$

Каждой переменной v, используемой в j-м условии, из списков I_j можно поставить в соответствие множество кортежей $\{i, u_i\}$, где $i \in I_j$, u_i — значение переменной v, извлекаемое из i-го факта. Если переменная v используется более чем в одном условии правила, пересечение

$$U_v = \{u_v\} = \bigcap_{j \in C} \{u\}_j$$

множеств значений переменной позволит сократить число фактов, требуемых для унификации этих условий. Для получения списка фактов I_{vj}, содержащих переменную v для j-го условия правила, где эта переменная встречается, достаточно выполнить операцию реляционного деления

$$I_{vj} = \{i, ui\} \div \{uv\}.$$

Наконец, если в условии c_j правила содержится более одной переменной, то пересечение списков

$$I_j = \bigcap_{v \in c_j} I_{vj}$$

для каждой из двух переменных даст окончательный список фактов, которые отвечают j-му условию правила.

Индексация и предварительный отбор позволяют подставлять в каждое правило только факты, которые обеспечивают успешное выполнение его условий. Экономия времени пропорциональна доле таких фактов. При использовании в правиле 100 % фактов экономия, естественно, нулевая.

Применение реляционных операций для логического вывода. В работе [4] показаны результаты исследования метода индексации и предварительного отбора фактов, где демонстрируется, что если в заголовке правила присутствует только одна переменная, то обработка индекса сразу дает множество решений, т.е. обработка правила становится излишней. В этой связи возникает вопрос, нельзя ли обойтись без подстановки фактов в условия правила в более сложных случаях. Пусть в результате отбора фактов для условий $c(s_1, p_1, o_1)$, $c(s_2, p_2, o_2)$, ..., ..., $c(s_k, p_k, o_k)$, где s_i, o_i — либо константа, либо переменная, получены множества кортежей $T = \{\{t_i\}\}$, где $t_i=(x_{i1}, x_{i2})$, если в условии правила содержатся две переменные, и $t_i=(x_{i1})$, если присутствует 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 представлять собой декартово произведение с данной таблицей, что обычно лишено смысла. Таблицы могут иметь связи следующих типов:

— соединение двух таблиц по совпадению значений их одной или более переменных; в этом случае две таблицы объединяются реляционным оператором INNER JOIN;

— фильтрация таблицы по условию сравнения значений переменных между собой или сравнения переменной и константы; данная функция выполняется с помощью условия WHERE и операторов сравнения.

Использование более сложных конструкций требуется в случае присутствия отрицания и кванторов в условиях правил. Однако в базах знаний, построенных на допущении открытого мира (Open World Assumption), отрицания не допускаются, а кванторы устраняются с помощью алгоритмов *сколемизации* [5]. Таким образом, условия правил могут

быть преобразованы в запросы на языках СУБД. Ниже приведен пример правила для отношения дядя—племянник(ца):

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

и соответствующего ему запроса SQL:

```
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)="hasParent") AND ((F_1.Predicate)="hasBrother"));
```

В настоящее время проводятся исследования по разработке средств автоматической загрузки баз знаний в СУБД и трансляции правил на языки запросов.

Быстрый логический вывод в среде Prolog. Язык программирования Prolog имеет встроенный механизм обратного логического вывода, соответствующий методу поиска „сначала вглубь“. Известно, что этот метод является самым экономичным в части использования памяти, но требует значительных временных затрат и, кроме того, не гарантирует достижения цели ввиду опасности бесконечного углубления в рекурсии [5]. В этой связи заслуживает внимания опыт применения для логического вывода реляционных операций, описанных выше.

В составе библиотек Visual Prolog имеются предикаты, реализующие операции над множествами. Однако скорость выполнения этих операций слишком велика, поэтому использование таких предикатов, в частности *intersection, difference, join, union*, вместо „наивного“ логического вывода лишено смысла.

Основная причина медленного выполнения таких предикатов состоит в том, что при обработке двух списков их элементы сопоставляются каждый с каждым. Реализация операций пересечения, разности и объединения двух списков $X=\{x\}$ и $Y=\{y\}$ предполагает в среднем развертывание $n_x n_y/2$ вершин дерева поиска (список Y сканируется до появления искомого значения), где n_x, n_y — количество фактов в множествах X и Y соответственно. Для операции соединения развертывается $n_x n_y$ вершин, поскольку соединение представляет собой декартово произведение с фильтрацией.

Отсортируем списки $X=\{x\}$ и $Y=\{y\}$ по возрастанию значений. В этом случае оба списка можно обрабатывать совместно, и дерево поиска будет состоять из n_x+n_y вершин. Таким образом, ожидаемое ускорение обусловлено переходом от квадратичной к линейной зависимости сложности поиска от числа фактов. Ниже приведен пример предиката пересечения отсортированных списков, разработанных автором для среды программирования Visual Prolog 7.2:

```
    predicates
    intersectSorted : (Elem* ListX, Elem* ListY) -> Elem* IntersectionXY.
clauses
    intersectSorted([],_) = [] :-!.
    intersectSorted(_,[]) = [] :-!.
        intersect-
Sorted([Y|Xs],[Y|Ys]) = [Y|intersectSorted(Xs,[Y|Ys])] :-!.
    intersectSorted([X|Xs],[Y|Ys]) = intersectSorted([X|Xs],Ys) :- X>Y,!.
    intersectSorted([_|Xs],[Y|Ys]) = intersectSorted(Xs,[Y|Ys]).
```

Выигрыш во времени при использовании логического вывода по сравнению с „наивным“ выводом в среде Prolog составляет приблизительно три порядка.

Заключение. Ускорение логического вывода в продукционных системах может быть достигнуто различными способами, в том числе посредством применения апробированных средств СУБД. Среда программирования Prolog также является привлекательной, поскольку позволяет в компактной форме реализовать многие алгоритмы искусственного интеллекта. На кафедре вычислительной техники СПбГУ ИТМО проводятся исследования и в других направлениях решения данной задачи, в частности по применению в задаче поиска методов случайного блуждания.

СПИСОК ЛИТЕРАТУРЫ

1. *Уотермен Д.* Руководство по экспертным системам / Пер. с англ.; Под ред. *В. Л. Стефанюка.* М.: Мир, 1989. 388 с.

2. *Forgy C. L.* RETE: A fast algorithm for the many pattern / many object pattern match problem // Artificial Intelligence. 1982. Vol. 19. P. 17—37.

3. *Berners-Lee T., Hendler J., Lassila Ora.* The semantic web // Sci. Amer. Magazine. 2001. May. P. 29—37.

4. *Бессмертный И. А.* Теоретико-множественный подход к логическому выводу в базах знаний // Науч.-техн. вестн. СПбГУ ИТМО. 2010. Вып. 02 (66). С. 43—48.

5. *Рассел С., Норвиг П.* Искусственный интеллект: Современный подход: Пер. с англ. М.: Изд. дом „Вильямс“, 2006. 1408 с.

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 18.01.11 г.