4 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.89

ТЕОРЕТИКО-МНОЖЕСТВЕННЫЙ ПОДХОД К ЛОГИЧЕСКОМУ ВЫВОДУ В БАЗАХ ЗНАНИЙ

И.А. Бессмертный

Рассматривается проблема комбинаторной сложности задачи поиска решений при резолюции правил в системах искусственного интеллекта. Предлагается подход к ускорению извлечения знаний путем индексации фактов и сокращения числа используемых фактов с помощью операций над множествами индексов. В отличие от известных алгоритмов, предполагающих предварительный отбор фактов для каждого правила, индексы абстрагируются от правил, что позволяет логически и физически разделить базы фактов и базы правил, а также упростить модификацию базы знаний. Демонстрируется возможность замены логического вывода реляционными операциями над кортежами переменных.

Ключевые слова: искусственный интеллект, индексация фактов, логический вывод.

Состояние проблемы и существующие решения

В большинстве систем искусственного интеллекта (ИИ) имеется база знаний, хранящая факты и правила для предметной области. Применение правил к известным фактам позволяет порождать новые знания. Для извлечения знаний используется интеллектуальный агент (машина вывода), который либо находит существующие факты, либо порождает новые факты путем логического вывода из правил. Вывод из правил представляет собой классическую задачу неинформированного поиска [1] и в случае небольших баз знаний может решаться путем последовательного перебора всех фактов для каждого из условий правила.

Пусть множество фактов базы знаний $F = \{f\}$ образует атомы (триплеты) $f = (s, p, o)$, где s – субъект, p – предикат, o – объект. Мощность множества фактов обозначим n. База знаний содержит также множество правил $R = \{r\}$. Каждое правило r состоит из результирующей части и тела (условия правила). Для простоты будем считать, что результирующая часть содержит один атом, а тело – множество атомов $C = \{c\}$. Пусть среднее число условий в одном правиле равно k, а мощность множества правил равна m. Запуск процедуры применения всех правил к множеству фактов вызывает последовательный перебор всех фактов для каждого из условий правил. Общее число A попыток унификации условий с фактами будет равно

$$A = mn^k.$$

Данная формула показывает линейный рост сложности поиска при добавлении новых правил, степенную зависимость от числа фактов и показательную – от среднего числа условий в правиле. Даже трехзначные значения числа фактов в базе знаний и однозначные числа условий в правилах порождают миллионы вершин дерева поиска. Таким образом, комбинаторная сложность задачи даже для небольших баз знаний не позволяет решать ее путем «буквального» или «наивного» логического вывода.

Наиболее известным методом ускорения резолюции правил является алгоритм *Rete* [2], используемый в экспертных системах *CLIPS, Jess, Soar* и др. В основу алгоритма положено префиксное дерево, узлами которого являются условия правил. В каждом узле префиксного дерева создается список фактов из базы знаний, которые удовлетворяют условиям правила. Фактически это означает многократное дублирование фактов (рабочей памяти) в области правил (операционной памяти). При перемещении от корня префиксного дерева до листа списки фактов подвергаются операции пересечения,

и в листе находятся только факты, необходимые и достаточные для резолюции правила. Перемещение от корня до листа как раз и означает резолюцию правила.

Узким местом алгоритма *Rete* является изменение префиксного дерева при изменении фактов базы знаний. При добавлении, изменении или удалении фактов префиксное дерево должно строиться заново либо модифицироваться. Большинство модификаций алгоритма *Rete*, например, *Modify-in-place, Scaffolding, Decision Tree* [3], нацелены именно на изменение префиксного дерева. Между тем, модификация множества фактов происходит в системах ИИ постоянно, поскольку каждое правило дает в качестве результата новые факты, которые должны сразу же использоваться в ходе резолюции.

Индексация и предварительный отбор фактов

Проиндексируем факты следующим образом. Присвоим каждому факту в базе знаний порядковый номер i, тогда нумерованный факт будет выглядеть следующим образом:

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

Для множества термов $T = \{t\}$, встречающегося в фактах, построим индекс в виде

$$X = \{x\} = \{(t, w, \{i_{tw}\})\}, \quad (1)$$

где w – место данного терма в атоме (в качестве субъекта, предиката или объекта), $\{i_{tw}\}$ – множество номеров фактов, имеющих терм t в роли w, w=(*'s';'p';'o'*). Резолюция правила заключается в установлении истинности условий и присвоении значений переменным. Теперь при обращении к правилу, тело которого состоит из множества условий $\{c_1, c_2, \ldots, c_k\}$, где $c_j = (s_j, p_j, o_j)$, s_j – субъект, p_j – предикат, o_j – объект, $(s; o) = (t; v)$, v – переменная, $p_j = t$. Поскольку в логике первого порядка не допускается использование переменных в качестве предиката, каждое из условий c_j может иметь одно из четырех сочетаний термов и переменных: (t, t, t); (t, t, v); (v, t, t); (v, t, v).

Для каждого из c_j условий правила извлечение релевантных фактов для перечисленных сочетаний термов заключается в нахождении пересечений множеств индексов:

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

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

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

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

Каждой переменной v, используемой в j-м условии, из списков I_j можно поставить в соответствие множество кортежей $\{i, u_i\}$, где i – номер факта, $i \in I_j$, 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 во всех условиях C правила, в которых эта переменная используется, 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 правила c_j участвуют 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.

Рассмотрим данный алгоритм на простом примере. Пусть имеется база знаний, состоящая из фактов, имеющих сквозную нумерацию (здесь и далее будем придерживаться синтаксиса языка Prolog):

fn(1, *sergey, has_sex, male*).
fn(2, *nikita, has_sex, male*).
fn(3, *sergey, is_a, person*).
fn(4, *natalia, is_a, person*).
fn(5, *nikita, is_a, person*).
fn(6, *nikita, has_sex, male*).
fn(7, *natalia, has_sex, female*).
fn(8, *sergey, parent, nikita*).
fn(9, *sergey, parent, andrey*).
fn(10, *natalia, parent, nikita*).
fn(11, *natalia, parent, andrey*).
fn(12, *nikita, parent, stepan*).
fn(13, *andrey, parent, egor*).

Построим для этих фактов индекс, как показано в формуле (1):

x(*sergey, s,* [1,3,8,9]).
x(*nikita, s,* [2,5,6,12]).
x(*natalia, s,* [4,7,10,11]).
x(*andrey, s,* [13]).
x(*has_sex, p,* [1,2,6,7]).
x(*is_a, p,* [3,4,5]).
x(*parent, p,* [8,9,10,11,12,13]).
x(*male, o,* [1,2,6]).
x(*person, o,* [3,4,5]).
x(*female, o,* [7]).
x(*nikita, o,* [8,10]).
x(*andrey, o,* [9,11]).
x(*stepan, o,* [12]).
x(*egor, o,* [13]).

Создадим правила в виде *r*(*conditionList, resultingList*) с использованием переменных, начинающихся с вопросительного знака, где *conditionList* – список условий, *resultingList* – список триплетов результата. Первое правило устанавливает, что субъект является мужчиной, если он является человеком и имеет мужской пол.

r([*c*(*“?x”, is_a, person*), *c*(*“?x”, has_sex, male*)], [*f*(*“?x”, is_a, man*)].

	1-е условие правила	2-е условие правила
Исходное условие	*?x, is_a, person*	*?x, has_sex, male*
Индекс для используемых термов	*x*(*is_a, p,* [3,4,5]). *x*(*person, o,* [3,4,5]).	*x(has_sex, p,* [1,2,6,7]). *x(male, o,* [1,2,6]).
Пересечение по номерам	*?x, s,* [3,4,5]	*?x, s,* [1,2,6]
Значения переменной	*?x* = [*sergey, natalia, nikita*]	*?x* = [*sergey, nikita, nikita*]
Пересечение по значениям	[*sergey, nikita*]	
Отфильтрованный индекс	*?x, s,* [3, 5]	*?x, s,* [1,2,6]

Таблица 1. Предварительный отбор фактов для правила *?x is_a man*

Используем индекс для отбора фактов и находим пересечение значений переменной *?x*, используемых во всех условиях.

[sergey, natalia, nikita] ∩ [sergey, nikita, nikita] = [sergey, nikita]

Фильтруем списки фактов, требуемых для резолюции каждого условия правила, включая в них только значения термов, попавшие в пересечение. Для первого условия это факты [3,5], а для второго – [1,2,6]. Таким образом, для резолюции правила требуется унифицировать условия правила с пятью фактами. Если не использовать индекс, то каждое условие правила нужно сопоставлять со всеми 13-ю фактами, всего 26 фактов.

Создадим более сложное правило, определяющее отношение «прародитель» (grandparent):

r([*c*(*“?x”, parent, “?y”*), *c*(*“?y”, parent, “?z”*)], [*f*(*“?x”, grandparent, “?z”*)].

	1-е условие правила	2-е условие правила
Исходное условие	*“?x”, parent, “?y”*	*“?y”, parent, “?z”*
Индекс для используемых термов	*x*(*parent, p,* [8,9,10,11,12,13]).	*x*(*parent, p,* [8,9,10,11,12,13]).
Пересечение по номерам	*?x, s,* [8,9,10,11,12,13]). *?y, o,* [8,9,10,11,12,13]).	*?y, s,* [8,9,10,11,12,13]). *?z, o,* [8,9,10,11,12,13]).
Значения переменной	[*nikita, andrey, nikita, andrey, stepan, egor*]	[*sergey, sergey, natalia, natalia, nikita, andrey*]
Пересечение по значениям	[*nikita, andrey*].	
Отфильтрованный индекс	[8,9,10,11]	[12,13]

Таблица 2. Предварительный отбор фактов для правила *?x grandparent ?z*

Более одного раза в правиле встречается только переменная *?y*; пересечение ее допустимых значений в первом и втором условиях равно

[nikita, andrey, nikita, andrey, stepan, egor] ∩ [sergey, sergey, natalia, natalia, nikita, andrey] = [nikita, andrey].

В качестве допустимых значений переменной *?y* мы получили список объектов, которые являются одновременно родителями и детьми. Прореживаем список фактов для первого условия [8,9,10,11,12,13], оставляя в нем только факты, имеющие термы [*nikita, andrey*] в качества объекта. Получаем список [8,9,10,11]. Аналогично получаем список фактов [12,13] для второго условия. Находим пересечение номеров фактов, соответствующих каждой из переменных, в каждом из условий. Для первого условия [8,9,10,11,12,13] ∩ [8,9,10,11] = [8,9,10,11], для второго – [12,13] ∩ [8,9,10,11,12,13] = [12,13]. Таким образом, мы добиваемся того, что каждому из условий правила в процессе унификации будут предъявляться только факты, гарантированно порождающие результаты.

Добавление нового факта в базу знаний вручную либо как результата резолюции правила не требует сложных вычислений. Для этого нужно присвоить факту номер и модифицировать три индекса, соответствующие субъекту, предикату и объекту, либо создать новые индексы, если данные сущности встречаются в первый раз.

Оценка быстродействия алгоритма

Предлагаемый здесь метод реализован в программе Semantic, разработанной автором специально для изучения принципов построения систем искусственного интеллекта, визуализации знаний, а также для исследований способов построения интеллектуальных агентов. Поскольку прямой логический вывод предполагает независимую обработку каждого правила, длительность вывода имеет линейную зависимость от числа правил. Следовательно, оценку быстродействия можно оценивать на одном правиле. Тестирование данного метода проводилось на базе знаний, описывающей родственные отношения. Факты вида субъект–is_parent–объект создавались с помощью генератора случайных чисел. Правило, которое использовалось для измерения времени вывода, описывает отношение типа «прапрародитель» и имеет три условия:

r([*c*(*"?x1","is_parent","?x2"*), *c*(*"?x2","is_parent","?x3"*), *c*(*"?x3","is_parent","?x4"*)], [*f*(*"?x1","is_grandgrandparent","?x4"*)]).

Приведенный ниже график (см. рис.) показывает зависимость времени работы программы логического вывода от числа фактов в базе знаний.

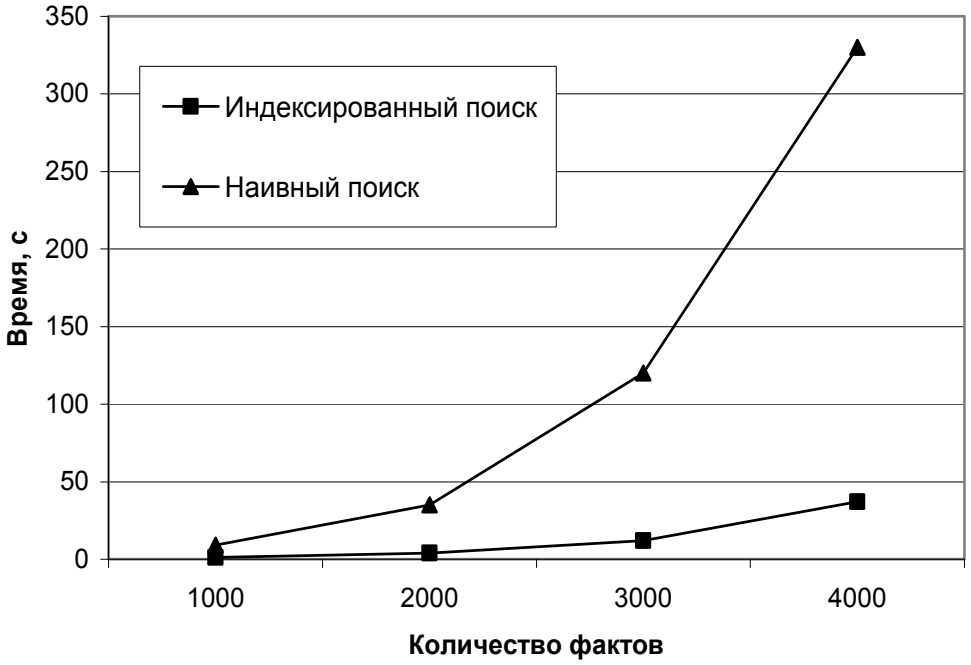

Рис. Зависимость времени логического вывода от числа фактов в базе знаний

Эксперименты на тестовом наборе фактов показали, что эффективность индексации фактов зависит также от результативности правил. Если количество успешных применений правил невелико, то время вывода с использованием индексов стремится к нулю. Если каждая комбинация исходных фактов приводит к успешной резолюции (ситуация, на практике, маловероятная), то использование индексов только увеличивает время за счет издержек на индексацию. В использованном примере успешными были приблизительно 15% всех применений правил.

Замена унификации правил операциями реляционной алгебры

В приведенном примере резолюции правила «субъект является мужчиной» был получен список значений переменной *?x:* [*sergey, nikita*]. Но поскольку эта переменная в правиле единственная, то задача практически уже решена, и новые факты могут быть получены подстановкой значений переменной в заголовок правила: *f*(*sergey, is_a, man*)*; f*(*nikita, is_a, man*).

Рассмотрим, можно ли обойтись без интерпретации правил во втором примере. Извлечем из базы знаний факты [8,9,10,11], а также [12,13] и получим кортежи переменных *?x* и *?y* для первого условия и *?y* и *?z* для второго условия

?x	*?y*		*?y*	*?z*
sergey	*nikita*		*nikita*	*stepan*
sergey	*andrey*		*andrey*	*egor*
natalia	*nikita*			
natalia	*andrey*			

Выполнив для этих таблиц операцию соединения по совпадающим значениям *?y*, получим кортеж

?x	*?y*	*?z*
sergey	*nikita*	*Stepan*
sergey	*andrey*	*Egor*
natalia	*nikita*	*Stepan*
natalia	*andrey*	*Egor*

Результат операции соединения дает допустимые сочетания значений переменных *?x* и *?z*, т.е. искомое решение:

f(*sergey, grandparent, stepan*).
f(*sergey, grandparent, egor*).
f(*natalia, grandparent, stepan*).
f(*natalia, grandparent, egor*).

В рамках данной работы не рассматриваются отношения отрицания в правилах, а также отношения сравнения, отличные от равенства, хотя интерпретация правил, содержащих отношения «больше», «меньше» и др., приводит всего лишь к модификации операций соединения. Очевидным недостатком является возможное чрезмерное разрастание индексов для отношений объект–свойство–значение в связи с тем, что количество возможных значений свойств обычно существенно больше количества сущностей. Здесь целесообразно использовать хеширование значений свойств, как это делается в некоторых модификациях алгоритма *Rete*.

Заключение

Предлагаемый метод существенно отличается от алгоритма *Rete*. Главное отличие заключается в том, что только индексация фактов выполняется заблаговременно, а операция предварительного отбора фактов выполняется непосредственно перед резолюцией. Это означает, что факты и правила могут существовать отдельно, в том числе на разных сетевых ресурсах, что соответствует концепции Глобальной семантической сети [4]. В ходе предварительного отбора фактов для каждого условия правила создаются множества кортежей значений переменных, используя которые, можно в отдельных случаях отказаться от резолюции правил, а, используя операции реляционной алгебры над кортежами, сразу получить множество решений правила. Тестирование метода индексации фактов показало его работоспособность и ускорение приблизительно на порядок по сравнению с «наивным» логическим выводом. Дальнейшее ускорение возможно путем полноценной реализации замены унификации правил операциями реляционной алгебры.

Литература

1. Рассел С., Норвиг П. Искусственный интеллект: Современный подход. 2-е изд. / пер. с англ. – М.: Изд. дом «Вильямс», 2006.
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. Doorenbos R. B.. Production Matching for Large Learning Systems // PhD Theses. University of South California, 1995. – 208 pp.
4. Berners-Lee T., Hendler J., Lassila O. The Semantic Web // Scientific American Magazine. – May, 2001.

Бессмертный Игорь Алексаендрович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, кандидат технических наук, доцент, igor_bessmertny@hotmail.com