СОВМЕЩЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ МОДЕЛЕЙ ЯЗЫКА ДЕЙТАЛОГ

Жирный Д.Г., Дацун Н.Н.
Кафедра ПМИ, ДонГТУ
datsun@pmi.donetsk.ua

Abstract

Zhirney D.G, Datsun N . N . Combining calculational models of Datalog language. *The combined model of calculation of the Datalog programs is considered, the effective algorithm of calculation of results is developed, the completeness, correctness and finitness of calculations are analysed. The applicability of existing methods of optimization to the combined model is considered, the alternative method is offered.*

1.Введение

Дейталог - это язык запросов к базам данных, основанный на парадигме логического программирования. Дейталог появился как альтернатива языку Пролог в контексте дедуктивных баз данных. Он является непроцедурным, множественным, нечувствительным к порядку следования предикатов в правилах и не имеющим специальных предикатов языком. Дейталог был специально разработан для приложений, использующих большое количество данных, хранящихся в реляционной БД.

2.Отрицание в Дейталоге

Для использования Дейталога в качестве языка запросов к базам данных, ему не хватает многих свойств, которыми обладают многие коммерческие языки (в частности, SQL [2]). Мы рассмотрим отрицание как наиболее нужное расширение классического Дейталога.

Дейталог~ - язык, синтаксис которого полностью совпадает с синтаксисом Дейталога, кроме того, что в теле правил разрешены литералы с отрицанием. Каждая переменная, встречающаяся в негативном литерале, должна также встречаться в позитивном литерале в том же теле Отрицание в "чистом" Дейталоге отсутствует, но, применяя аксиому о закрытости мира (АЗМ), можно вывести отрицательные факты из множества “чистых” Дейталог-высказываний. В контексте Дейталога АЗМ можно сформулировать следующим образом если факт логически не следует из множества высказываний Дейталог-программы, тогда отрицание этого факта есть истина АЗМ - не универсальное правило, а только принцип, который может подходить или не подходить к семантике проблемы. Формально АЗМ звучит так: “Литерал L является ложным в данной программе, если он не принадлежит единственной минимальной модели Эрбрана [I] данной программы”.

Отрицательные факты Дейталога - это позитивные основные литералы, перед которыми стоит знак отрицания. В реальной жизни часто необходимо описывать правила, содержащие отрицательную информацию В “чистом” Дейталоге нет способа выразить такие правила.

Существуют следующие проблемы вычисления программ на Дейталоге с отрицанием:

- немонотонность оператора наименьшей неподвижной точки [3]. Для решения этой проблемы предлагаются различные методы задания строгого порядка вычисления правил;
- бесконечность (например, *a(x):-~b(x).* а - бесконечно). Введём новое правило: каждая переменная, входящая в предикат с отрицанием, должна входить также в обычный литерал;
- противоречивость (например, *a(x):-~a(x), b(x).*). Есть несколько походов к решению этой проблемы (обобщенная АЗМ, слабообощенная АЗМ, стабильное отрицание и др. [4, 5, 6]);
- проблема выбора минимальной модели Эрбрана.

Существуют две основные модели вычисления Дейталог-программ с отрицанием: стратификационный и инфляционный Дейталог. Кроме того, авторами предлагается совмещенная модель вычисления Дейталог-программ.

2.1. Стратификационная модель

Стратификационная модель вычисления [1] позволяет выбрать наиболее "естественную" Эрбрановскую модель для заданной программы. Она основана на следующем интуитивно понятном правиле: "При вычислении правила с несколькими негативными литералами в его теле, сначала вычисляем предикаты, относящиеся к этим негативным литералам". Затем АЗМ "локально" применяется к этим предикатам.

Достоинства стратификационной модели:

- запрос естественней и лёгок в понимании (похожим образом можно было бы сформулировать запрос, например, на Прологе [2]);
- необходим дополнительный анализ программы перед вычислением для построения стратификации;
- вычисление по стратификационной модели всегда производит минимальную Эрбрановскую модель.

Недостаток - только стратифицируемая программа может быть вычислена.

2.2 *Инфляционная модель*

Инфляционная модель вычисления [1, 10, 11] применима для всех Дейталог-программ. Вычисление производится последовательно таким образом, что все правила программы обрабатываются параллельно на каждом шаге. Новые факты добавляются к существующим в конце каждого шага. На каждом последующем шаге берутся факты из ЭБД и факты, выведенные на предыдущем шаге, при этом АЗМ применяется "временно " во время вычисления в телах правил: считается, что отрицание фактов, которое ещё не выведены, есть истина. Алгоритм останавливается, когда ни один дополнительный факт не может выведен.

Достоинства инфляционной модели:

- любая программа может быть вычислена;

- метод достаточно эффективен и легко может быть распараллелен.

Недостатки:

- запросы, которые можно выразить с помощью инфляционной модели, но нельзя выразить с помощью стратификационной, не очень существенны даже для перспективных приложении баз данных [10, 11];
- вычисление по инфляционной модели в общем случае не производит минимальную Эрбрановскую модель;
- "неестественное" вычисление запросов - результаты запросов, вычисленные с помощью инфляционной семантики, трудно интерпретировать.

3. Совмещенная модель вычисления дейталог-программ

В [1] приведено доказательство того, что инфляционная модель более выразительна, чем стратификационная. Таким образом, инфляционная модель кажется более предпочтительной по сравнению с стратификационной. Однако результаты вычисления по инфляционной модели довольно трудно интерпретировать как для нестратифицируемых, так и для стратифицируемых программ. Учитывая достоинства и недостатки обеих моделей вычисления программ на Дейталоге~, можно предложить объединить их. Сначала следует проанализировать расширенный граф зависимостей РГЗ(Р). Если этот граф содержит циклы, в которых хоть одна дуга помечена "~", то вычислять программу с помощью инфляционной модели, иначе - с помощью стратификационной. Такая модель сможет вычислить любой запрос, при этом "обычные" программы (имеющие стратификацию) будут вычисляться "естественным" образом (подобно Прологу [2]), а программы, имеющие рекурсию через отрицание - с помощью инфляционной модели. При анализе графа РГЗ(Р) можно выяснить некоторые дополнительные свойства программы, в частности её линейность и разбить на страты [10, 11].

Пусть HB - базис Эрбрана, EHB - его экстенсиональная часть, IHB - его интенсиональная часть [1, 10, 11]. Тогда программа Р на Дейталоге~ может быть описана отображением μ_p из EHB в IHB, где для любых возможных экстенсиональных БД $E \subseteq EHB$, μ_p может быть определено следующим образом: $\mu_p(E)=cons(P \cup E) \cap IHB$.

Предлагается следующий алгоритм вычисления μ_p (E):

Алгоритм вычисления по совмещенной модели:

Аргументы: программа на Дейталоге~ Р, экстенсиональная БД Е

Результаты: множество кортежей $F= \mu_p (E)$

Метод:

1. Выполняем стратификацию для программы Р согласно алгоритму в [1].
2. Если алгоритм стратификации завершился с успехом (стратификация Р выполнена, получено k страт: $(P_1,...P_k\})$, то

2.1. i:=l; F:-{}

2.2 F:=F∪вычисленные кортежи по алгоритму INFERI~ [1] из предикатов, определенных в страте P_i;

2.3. i:=i+1;

2.4 если i=k тогда конец алгоритма, иначе переходим к 2.2

3. Иначе вычисляем программу P по инфляционному алгоритму [1].

КОНЕЦ

Достоинства совмещенной модели:

- любая программа может быть вычислена;
- формулировка запроса достаточно естественна;
- метод может быть эффективно реализован

Недостаток - достаточно сложная семантика

Вообще, метол вычисления должен быть:

- корректным (в результат не должны попадать кортежи, которые не принадлежат результату),
- полным (метод должен производить все кортежи результата);
- конечным (вычисление должно происходить за конечное время при конечности исходных данных)

Ниже мы рассмотрим каждый из этих критериев применительно к вычислению Дейталог-программ с помощью совмещенной модели.

3.1 Выразительность и полнота

Язык запросов должен обладать хорошими абстрактными свойствами, придающими ему гибкость в условиях непрерывно меняющейся среды. Обычно в этом смысле говорят, что язык должен быть полным. Однако не ясно, что в точности следует понимать под полнотой языка запросов. В настоящий момент предложено несколько критериев полноты [7]: критерий Кодда, критерий Банцелона, критерий Чандра-Харела, критерий Ливчака.

Рассмотрим полноту языка по критерию Кодда.

Критерий Кодда Язык L полон, если он не слабее логики первого порядка, т.е. EP(L) ϵ EP (логика первого порядка), где EP(L) - выразительная сила языка L, EP(L) - множество всех запросов, выразимых в языке запросов L.

Рассмотрим выразительную силу языка Дейталог~. Согласно совмещенной модели вычислений, представленной ранее, можно вычислить *любую* программу.

Предлагается следующее утверждение:

Утверждение: Вычислительная мощность совмещенной модели такая же, как и у инфляционного Дейталога~.

3.2 Корректность вычислений

Результаты вычисления программы по инфляционной модели довольно трудно интерпретировать из-за того, что инфляционное вычисление производит в общем случае не минимальную Эрбрановскую модель и при попытке сопоставления

результирующих кортежей с программой возникают вопросы об адекватной интерпретации результата.

Учитывая то, что инфляционный алгоритм включен как составная часть в совмещенный алгоритм, этот вопрос актуален Рассмотрим пример:

p(x):-a(x).~p(x).

При инфляционном вычислении данной программы с ЭБД=(a('a')) алгоритм остановится с результатом {p('a')}, однако при подстановке данного факта в правило, из которого и получен данный результат, мы получим противоречие:

a(‘a’)& ~p(‘a’) → p(‘a’)

Совмещенная модель тоже имеет этот недостаток. Один из механизмов, предложенных для предотвращения подобных результатов логического вывода - это введение трехзначной логики: {Истина, Ложь, Неопределено) [2].

3.3 Конечность вычислений

Запрос к БД конечен, если результат состоит из конечного числа кортежей. Для запросов, сформулированных в виде множества хорновских правил, проблема определения конечности, вообще говоря, нерешаема [8]. Сформулируем, и докажем для совмещенной модели теорему о конечности вычислений:

Теорема: Для конечного множества дизъюнктов Деталога~ S алгоритм вычисления результата по совмещенной модели (описанный выше) конечен.

Доказательство: В [1] показано, что алгоритмы стратификации, вычисления по стратификационной модели и по инфляционной модели завершаются при условии конечности 5. Очевидно, что совмещенный алгоритм вычисления также завершается при конечности 5, так как он является комбинацией указанных алгоритмов через оператор "если-то-иначе" [9]

3.4 Эффективная реализация алгоритма

Алгоритм вычисления по совмещенной модели может быть очень эффективно реализован. Действительно, на шаге 1 производится анализ РГЗ программы и строится ее стратификация (если таковая существует). На этом этапе также возможно выяснить свойства программы, в частности, ее линейность и отсутствие рекурсивных правил (анализ структуры программы), релевантность предикатов к запросу (участвует ли данный предикат в вычислении целевого дизъюнкта - анализ цели) и многие другие полезные свойства, необходимые для применения оптимизационных алгоритмов и альтернативных методов вычисления.

Можно показать, что практически все методы оптимизации (для стратификационной модели вычисления и классического Дейталога) можно применять к совмещенному методу Кроме того, авторами предлагается оценивать релевантность предикатов в правилах в алгоритме вычисления по стратификационной модели на этапе анализа РГЗ(Р) для дополнительной оптимизации. Кратко опишем метод оптимизации на основе оценки релевантности предикату цели:

Предикат P_i релевантен к предикату P_k в программе Р, если в РГЗ(Р) есть путь из P_i в P_k. Очевидно, что мы можем переписать программу Р таким образом, чтобы заменить цель в виде: *?-*$p_i(x_i,...x_n)$, ... $p_k(x_1,...x_i)$ на дополнительный предикат ИБД в

виде: *result({X}):- $p_1(x_1,...x_n)$, ... $p_k(x_1....x_1)$.*, где {X} - множество переменных у всех предикатов в запросе.

Заметим, что мы обобщаем вид запроса (который обычно представляется в виде ?-$p_k(x_1,...x_n)$.). Теперь мы можем вычислить множество всех предикатов, релевантных предикату result, и вычислять в методе Гаусса-Зейделя кортежи, относящиеся только к релевантным фактам. Такую оптимизацию в неявном виде выполняет также метод запрос/подзапрос [1,2]. Формальное описание метода выходит за рамки данной статьи.

Заключение

В данной работе рассмотрена совмещённая модель вычисления Дейталог-программ, разработан эффективный алгоритм вычисления результатов, проанализирована полнота, корректность и конечность вычислений. Рассмотрены методы оптимизации алгоритма, предложен альтернативный метод.

Результаты работы предполагается реализовать в экспериментальном проекте Дейталог для Windows [12], применяемого в настоящее время для обучения студентов специальности ПО ВТ и АС в курсе "Функциональное и логическое программирование".

Литература

1. Чери С, Готлоб Г., Танка Л. Логическое программирование и базы данных. -М.: Мир, 1992 г.-352 с.
2. Логическое программирование; Пер. с англ. и фр. - М.: Мир, 1988. - 369 с.
3. Задорожный В.И. Расширение дедуктивных языков запросов конструкциями с ограниченным квантором всеобщности. -Программирование, 2, 1995, с. 9-20
4. http://www-ssdi.di.fct.unl.pt/-lmp/publications/Biblio.ljtml - Bibliography of Luis Moniz Pereira
5. http://www.isu.ru/-slava/my_list.htm - Logic Programming Research Related Pages
6. http://karna.cs.umd.edu:3264/papers/ - University of Maryland: Seminar Docs Home Page
7. Ливчак А.Б. Полнота языков запросов. -Программирование, 2,1994, с. 31-42
8. http://www.cs.umd.edu/-kifer - Michael Kifer Home Page
9. Линтер Р. и др. Теория и практика структурного программирования. М.: Мир, 1982 г.
10. Ceri S., Gottlob G., Tanca L. Datalog: a self-contained tutorial. -Программирование, 4, 1991, с. 20-40
11. Ceri S., Gottlob G., Tanca L. Datalog: a self-contained tutorial. - Программирование, 5, 1991, с. 9-31
12. Жирний Д.Г., Дацун Н.Н. Реалізація інтегрованого середовища ДЕЙТАЛОГу для навчання логічному програмуванню та його оцінка / Всеукраїнська конференція молодих науковців “Інформаційні технології в науці та освіті”. 15-18 квітня 1997. Черкаси. - Черкаси :Черкаський державний університет, 1997. - с. 27.