Санкт-Петербургский государственный университет

Кафедра системного программирования

Разработка транслятора подмножества языка Datalog в операции над многомерными матрицами

Александра Юрьевна Лановая, группа 22.Б15-мм

Научный руководитель: С. В. Григорьев , доцент кафедры системного программирования, к. ф.-м. н.

Санкт-Петербург
2024

Datalog

- Является языком запросов, основанным на реляционной алгебре и логическом программировании
- Запросы строятся с помощью правил, выражающих логические связи между отношениями в базе данных
- Позволяет работать с производными (выводимыми) отношениями, которые формируются на базе существующих отношений при выполнении запросов
- Применение в различных областях
  - Системы управления знаниями
  - Анализ программного кода
  - Кибербезопасность

Синтаксис Datalog-a

- Pr — множество предикатов, $arity : Pr \to \mathbb{N}$ — функция арности, Cs — множество констант, V — множество переменных.
- **Term**:
  $t ::= x \mid c$, где $x \in V,\ c \in Cs$
- $p \in Pr$ — предикат и $\overline{t}$ — последовательность термов, $|\overline{t}| = arity(p)$. **Atom** A:
  $A ::= p(\overline{t})$
- **Clause** C:
  $C ::= H \leftarrow B_1, \cdots, B_m$, где H — голова, $B_1, \cdots B_m$ — тело.
- Когда $m = 0$, то есть $C \equiv H$ — **Fact**. В случае, когда $m \geq 1$ — **Rule**.
- **Program** P:
  $P ::= C_0, \cdots, C_k$.

Рассматриваемое подмножество Datalog

- $DatalogProgram = \{P \mid P = (Clauses, Chain_rule_extension)\}$ — множество всех Datalog-программ
- $Clauses$ — список всех фактов и правил в программе P
- $Chain_rule_extension$ — предикат, определяющий, что каждое правило в $Clauses$ удовлетворяет следущим условиям
  $\forall c \in Clauses$
  - Если c является фактом ($ClauseFact$), то $Chain_rule_extension(c) = True$
  - Если c является правилом ($ClauseRule\ r$), то пусть
    - $head_pred = r.(head).(predicate)$ — предикат в головной части правила
    - $head_terms = r.(head).(terms)$ — термы в головной части правила
    - $body_atoms = r.(body)$ — атомы в теле правила

    Тогда $Chain_rule_extension(c) = \exists t \in head_terms : (\forall b \in body_atoms : \exists p,\ ts : b.(predicate) = p \wedge b.(terms) = ts \wedge t \in ts)$
- $Datalog_subset = \{P \in DatalogProgram \mid P.(Chain_rule_extension) = True\}$ — рассматриваемое подмножество Datalog-программ

Пример

```
family(ann, bill, claire).
family(bill, donna, eric).
ancestor(X, Y, Z) :- family(X, Y, Z).
ancestor(X, Y, Z) :- family(X, W, Z), family(W, Y, Z).
```

Современные трансляторы и многомерные матрицыы

- Современные трансляторы
  - Soufflé
  - LogicBlox
    - Мемоизация промежуточных результатов, использование индексов для ускорения доступа к данным, параллелизм вычислений
  - D4
  - Datafun
    - Использование алгебраической переформулировки для оптимизации, применение статистических моделей
- Многомерные матрицы
  - Применение в логическом программировании не исследовано должным образом
  - Интеграция линейной алгебры в логические системы — перспективное направление
  - Играют ключевую роль в оптимизации запросов

Постановка задачи

Целью работы является реализация транслятора Datalog в многомерные матрицы
Задачи:

- Написать библиотеку для операций над многомерными матрицами
- Разработать и реализовать транслятор подмножества языка Datalog в операции над многомерными матрицами
- Провести тестирование разработанных компонент
- Автоматизировать доказательство корректности транслятора

Библиотека операций над многомерными матрицами

- Создание многомерных матриц
- Упрощение матриц
  - Исключение двух наименьших последовательных измерений, содержащих по одному элементу
  - Исключение наибольшего измерения многомерной матрицы, содержащей один элемент
- Сложение матриц
- Изменение структуры многомерной матрицы
  - Операция среза для многомерных матриц
  - Операция flatten
  - Операция reshape
- Операции set и get
- Умножение матриц

Процесс трансляции

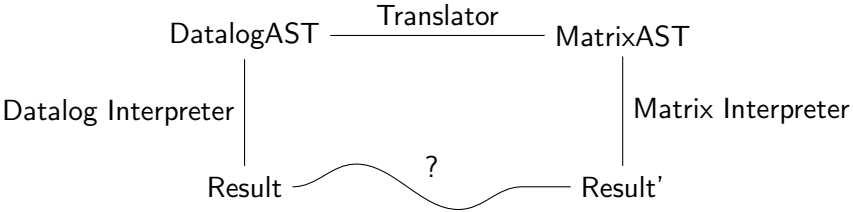
1. Определение доменов для предикатов
2. Трансляция списка атомов в список матриц
3. Трансляция фактов и правил
4. Корректировка размерностей матриц
5. Сложение матриц, являющихся результатом трансляции одного головного правила атома

Пример

```
family(ann, bill, claire).
family(bill, donna, eric).
ancestor(X, Y, Z) :- family(X, Y, Z).
ancestor(X, Y, Z) :- family(X, W, Z), family(W, Y, Z).
```

$$\text{family_matrix} = \begin{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix}, \text{ancestor_matrix} = \begin{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \end{bmatrix}$$

Тестирование транслятора



Доказательство корректности транслятора

- Определение синтаксиса и семантики транслируемого подмножества Datalog
- Определение синтаксиса и семантики операций над многомерными матрицами
- Трансляция атомов в матрицы
- Трансляция правил в операции над многомерными матрицами
- Трансляция всей программы Datalog в последовательность операций над многомерными матрицами

Theorem translateDatalogProgram_correct :

$$forall\ p\ q,\ eval_datalog\ p\ q\ =\ eval_matrix\ (translateDatalogProgram\ p)\ q.$$

Результаты

Были получены следующие результаты:

- Была разработана библиотека операций над многомерными матрицами[1]
- Реализованный транслятор успешно выполняет преобразование подмножества Datalog-программ в операции над многомерными матрицами
- Проведено тестирование всех компонент
- Автоматизировано доказательство корректности транслятора

[1] Ссылка на GitHub