Анализ времени связывания для реляционных программ

Ирина Артемьева, Екатерина Вербицкая

JetBrains Research, Университет ИТМО

16.05.2020

Реляционное программирование

- Программы реляционного программирования = математические отношения
- Можно исполнять в различных направлениях: зафиксировав часть аргументов, находить остальные
- $append^o \subseteq [a] \times [a] \times [a]$ — отношение, связывающее три списка; третий аргумент — конкатенация первых двух ($?$ — выходной):
  - $append^o \; x \; y \; ?$ — конкатенация x и y
  - $append^o \; ? \; y \; z$ — разность z и y
  - $append^o \; ? \; ? \; z$ — все пары списков, конкатенация которых равна z
  - $append^o \; ? \; ? \; ?$ — все списки отношения

Преимущество реляционного программирования

Выбор направления: написав одну программу, получаем несколько целевых функций

- по интерпретатору языка и набору тестов синтезируем программу
- по предикату ”последовательность вершин в графе формирует желаемый путь” синтезируем генератор таких путей

Язык miniKanren

- Основной представитель парадигмы реляционного программирования
- Встраивается в языки общего назначения
- Все языковые конструкции обратимы
  - в отличие от Prolog, который использует необратимую операцию cut, что делает невозможным выполнение по направлениям

Абстрактный синтаксис miniKanren

Конструкции на примере отношения $append^o$:

```
:: append^o x y z =
  (x === [] /\ z === y) \/
  ([h t r:
    (x === h % t /\
     z === h % r /\
     {append^o t y r}])
```

Проблема скорости вычисления

- При написании программы подразумевается конкретное направление, называемое прямым; все другие — обратные
- Выполнение в обратном направлении обычно крайне неэффективно

Подходы к ускорению выполнения

- Специализация
  - Конъюнктивная частичная дедукция (CPD)
    - из отношения удаляются избыточные вычисления, зависимые от известного входа
  - Суперкомпиляция
    - отношение исполняется с сохранением истории вычислений — на её основе происходит оптимизация
- ***Трансляция***
  - По отношению с фиксированным направлением генерируется функция на функциональном языке программирования

Трансляция

Отношение $+$ Направление $\Rrightarrow$ Функция

- Несколько выходных переменных объединятся в кортеж
- Вариативность результа выражается списком
- ***Неопределённость порядка вычислений***

Неопределённость порядка вычислений

- Рассмотрим порядок вычислений 2 дизъюнкта $append^o$ в прямом и обратном направлении

```
:: appendᵒ x y z =
  (x === [] /\ z === y) \/
  ([h t r:
    (x === h % t /\
     z === h % r /\
     {appendᵒ t y r}])
```

- $(h \,\%\, t) \;:=\; x$
- $r \;:=\; append^o\; t\; y$
- $z \;:=\; (h \,\%\, r)$
- $return\; z$

- $(h \,\%\, r) \;:=\; z$
- $(t,\; y) \;:=\; append^o\; r$
- $x \;:=\; h \,\%\, t$
- $return\; (x,\; y)$

- Решение — ***анализ времени связывания***

Анализ времени связывания

- Вход: программа на miniKanren и имена входных переменных
- Каждой переменной сопоставляется время её связывания
- Ранее не существовало для miniKanren

Цель и задачи

- Цель: указать порядок, в котором имена переменных связываются со значениями
- Задача: разработать алгоритм анализа времени связывания для miniKanren

Аналоги анализа времени связывания

- Для offline-специализации
  - какие данные известны статически и могут быть учтены при специализации
- Mercury — для эффективной компиляции
  - два типа аннотаций — недостаточно
  - использует граф типов — в miniKanren нет типов
- ***Лямбда-исчисление с функциями высшего порядка***
  - определяется порядок связывания переменных
  - типы аннотаций — $\{0, 1, \ldots, N\}$

Аннотирование

- Время связывания переменной — число или $Undef$ (время связывания неизвестно)
- Процесс подбора чисел назовем *аннотированием*
- Изначально входные переменные аннотируются 0, остальные — $Undef$
- Аннотация никогда не заменяется на меньшую

Пример аннотирования $append^o$

- Число над переменной — её аннотация
- Рассмотрим прямое и обратное направление

```
:: append^o x^0 y^0 z^1 =
   (x^0 === [] /\
    y^0 === z^1) \/
   ([h, t, r:
       x^0 === h^1 % t^1 /\
       z^3 === h^1 % r^2 /\
       {append^o t^1 y^0 r^2}])
```

```
:: append^o x^1 y^1 z^0 =
   (x^1 === [] /\
    y^1 === z^0) \/
   ([h, t, r:
       x^3 === h^1 % t^2 /\
       z^0 === h^1 % r^1 /\
       {append^o t^2 y^2 r^1}])
```

- $(h \,\%\, t) \;:=\; x$
- $r \;:=\; append^o\; t\; y$
- $z \;:=\; (h \,\%\, r)$
- $return\; z$

- $(h \,\%\, r) \;:=\; z$
- $(t,\; y) \;:=\; append^o\; r$
- $x \;:=\; h \,\%\, t$
- $return\; (x,\; y)$

Алгоритм аннотирования

- Аннотировать отношение = аннотировать все его дизъюнкты
- Аннотировать дизъюнкт = аннотировать все его конъюнкты
  - Переменная в конъюнктах одного дизъюнкта имеет одну аннотацию — согласованность аннотаций
- Аннотировать конъюнкт = аннотировать унификацию или вызов отношения

Аннотирование унификации

- $x^{Undef} \equiv [y^m\ z^n] \rightarrow$ аннотация $x = max(m, n) + 1$
- $x^n \equiv [y^m\ z^{Undef}\ w^k] \rightarrow$ аннотация $z = n + 1$
- $C\ Name\ [t_0^{i_0}, \ldots, t_k^{i_k}] \equiv C\ Name\ [s_0^{j_0}, \ldots, s_k^{j_k}] \rightarrow$ аннотируем аргументы конструкторов попарно
- Остальные случаи симметричны

Аннотирование вызова

- Аннотирование самого вызова: $Undef$-аннотации аргументов равны $n + 1$, где n — максимальная аннотация аргументов
  - Аннотирование невозможно, если все аргументы $Undef$ — недостаточно информации
- Аннотирование тела вызываемого отношения
  - Во избежание повторного аннотирования отношения по тому же направлению, сохраним название и направление в “стеке вызовов”

Несколько вызовов в дизъюнкте

- Последовательность нескольких вызовов влияет на направления их трансляции
- Пусть z — входная переменная

$$
\begin{array}{l}
::\ rel^o\ \ \mathtt{x}\ \ \mathtt{y}\ \ z^0\ \ = \\
\quad f^o\ \ \mathtt{x}\ \ \mathtt{y}\ \ \wedge \\
\quad h^o\ \ z^0\ \ \mathtt{y}\ \ \wedge \\
\quad g^o\ \ \mathtt{x}\ \ z^0
\end{array}
$$

не достаточно информации

$$
\begin{array}{l}
::\ rel^o\ \ \mathtt{x}\ \ \mathtt{y}\ \ z^0\ \ = \\
\quad h^o\ \ z^0\ \ y^1\ \ \wedge \\
\quad f^o\ \ x^2\ \ y^1\ \ \wedge \\
\quad g^o\ \ x^1\ \ z^0
\end{array}
$$

$x \ := \ f\ y$

$$
\begin{array}{l}
::\ rel^o\ \ \mathtt{x}\ \ \mathtt{y}\ \ z^0\ \ = \\
\quad h^o\ \ z^0\ \ y^1\ \ \wedge \\
\quad g^o\ \ x^1\ \ z^0\ \ \wedge \\
\quad f^o\ \ x^1\ \ y^1
\end{array}
$$

предикат $f\ x\ y$

- Решение: в случае неудавшегося аннотирования запустим алгоритм еще раз, изменив порядок вызовов

Терминируемость алгоритма аннотирования

- Повторное аннотирование отношений не производится $\rightarrow$ каждому отношению сопоставляется конечное количество уникальных аннотаций
- В случае нескольких вызовов в дизъюнкте количество перестановок вызовов конечно $\rightarrow$ конечно количество запусков алгоритма

Результаты

- Разработан алгоритм анализа времени связывания для miniKanren
  - он определяет порядок, в котором связываются переменные данного отношения с учётом направления его вычисления
- Успешно интегрирован в транслятор