Применение метода суперкомпиляции для специализации реляционных программ

Мария Куклина, M4236

Университет ИТМО
Научный консультант: Вербицкая Екатерина Андреевна

Реляционное программирование

Определение

Вид декларативного программирования, в котором программы представляются как набор отношений между аргументами.

Пример

Пример *запросов* для отношения умножения $\text{mul}^o \subseteq \text{Int} \times \text{Int} \times \text{Int}$:

- `mul`^{o}`(2, 2, 4)` — проверка корректности отношения.
- `mul`^{o}`(2, 2, C)` — поиск всех `C`, таких что `2 * 2 = C`.
- `mul`^{o}`(A, 1, 4)` — поиск всех `A`, таких что `A * 1 = 1`.
- `mul`^{o}`(A, B, C)` — поиск всех троек `A, B, C`, таких что `A * B = C`.

miniKanren

Встраиваемый предметно-ориентированный язык реляционного программирования[1].

Применение

- Замена тяжеловесной подсистемы Prolog для ряда задач.
- Генерация программ по спецификации входов и выходов на основе *реляционного интерпретатора*.
- Порождение решения задач поиска по решению задачи распознавания[2].
- Поиск лечения редких генетических заболеваний в точной медицине[3].

---

[1] *"Relational Programming in miniKanren: Techniques, Applications, and Implementations"*, Byrd, 2009

[2] *"Relational Interpreters for Search Problems"*, Lozov, Verbitskaia и Boulytchev, 2019

[3] *"The Algorithm for Precision Medicine (Invited Talk)"*, Might, 2019

Постановка проблемы

- Сложные алгоритмы в miniKanren работают медленно.
- Многие отношения на деле являются функциональными, из-за чего запуск в “обратном” направлении очень неэффективен.
  Примеры
  - Программы, порождённые инструментом по трансляции из функционального языка в miniKanren[4].
  - Порождение решения задач поиска.

[4] *“Typed Relational Conversion”*, Lozov P., 2018

Специализация

Определение

Автоматизированная техника оптимизации программ, при которой из программы удаляются избыточные вычисления, зависимые от частично известного входа[5].

- Частичная дедукция — класс методов специализации для логический языков, в частности, для Prolog.[6]
- Специализация miniKanren на основе *конъюнктивной частичной дедукции (CPD)*[7].
  - Сложна в поддержке, даёт нестабильные результаты.
  - Однако предоставляет библиотеку для построения специализаторов.

[5] *Partial evaluation and automatic program generation*, Jones, Gomard и Sestoft, 1993

[6] *Advanced Techniques for Logic Program Specialisation*, Leuschel, 1997

[7] *"Relational Interpreters for Search Problems"*, Lozov, Verbitskaia и Boulytchev, 2019

Суперкомпиляция

Определение

Техника автоматической трансформации и анализа программ, при которой программа символьно исполняется с сохранением истории вычислений, на основе которой принимаются решения о трансформации и оптимизации.

- Суперкомпиляторы применяются во основном для функциональных языков.
- Суперкомпиляция показывает хорошие результаты при специализации.
- Полуавтоматическая позитивная суперкомпиляция для Prolog[8].

[8] *A Prolog Positive Supercompiler*, Diehl, 1997

Цели и задачи

Цель

Улучшение результатов специализации реляционных программ путём применения метода суперкомпиляции.

Задачи

- Реализовать базовый суперкомпилятор для miniKanren.
- Рассмотреть возможные методы улучшения получившегося суперкомпилятора.
- Протестировать результаты и сравнить их с результатами CPD и с оригинальными программами.

μKanren

Определение

Минимальное подмножество miniKanren, содержащее в себе только основные операции языка: конъюнкция, дизъюнкция, унификация, введение свежей переменной и вызов реляционного отношения.

- Программа на μKanren представляет собой логическую формулу, атомы которой — это либо унификация двух термов, либо вызов отношения.
- Не содержит операторов miniKanren с эффектами.
- Библиотека для специализации работает с μKanren[9].

[9] https://github.com/kajigor/uKanren_transformations

Суперкомпиляция для μKanren

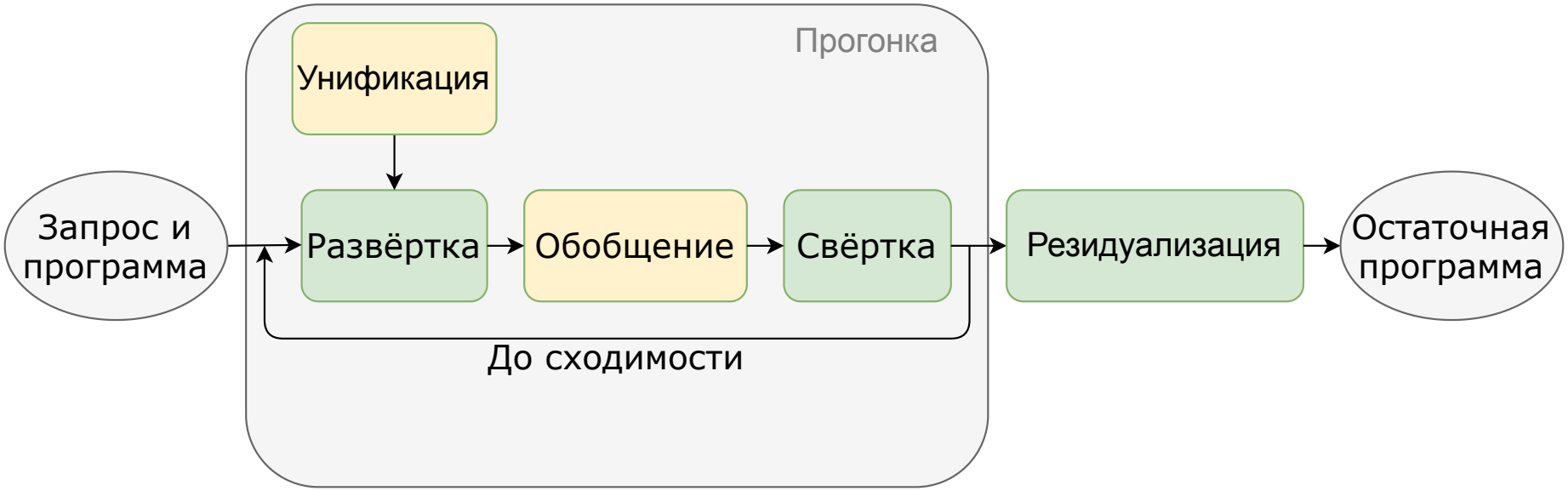

Рис.: Схема алгоритма суперкомпиляции

— библиотека по специализации miniKanren с дополнениями

— собственная разработка

Результаты задачи

- Реализован базовый алгоритм суперкомпиляции.
  - Используемый алгоритм обобщения основан на алгоритме для конъюнктивной частичной дедукции, для которого доказана терминируемость.
- Придуман и реализован алгоритм построения оптимизированной программы по графу суперкомпиляции.

Улучшение суперкомпиляции для μKanren

Проблемы

- Повторение символьных вычислений.
- Обобщение только на родителей: слишком много вычислений перед свёрткой.
- Обобщение вверх: в некоторых случаях нет эффекта от того, что часть входных данных известна.
- Тривиальный шаг прогонки порождает слишком много ветвей исполнения.
- Нет способа эффективно сообщить μKanren, что можно прервать вычисление, из-за чего суперкомпилятору приходится вычислять заведомо избыточные ветви.

Улучшение суперкомпиляции для μKanren

Стратегии вычисления выражения на шаге прогонки

Разные стратегии вычисления выявляют разные возможности для оптимизации.

- Всегда вычислять первый вызов за шаг.
- Вычислять все вызовы за шаг.
- Последовательно вычислять вызовы.
- В первую очередь вычислять не рекурсивные вызовы.
- В первую очередь вычислять рекурсивные вызовы.
- В первую очередь вычислять выражения с минимальным числом ветвлений.
- В первую очередь вычислять выражения с максимальным числом ветвлений.

Результаты задачи

- Применены подходы по улучшению алгоритма суперкомпиляции.
  - Добавлено кэширование.
  - Добавлено обобщение на все вычисленные вершины.
  - Добавлена возможность частично или полностью исключить обобщение вверх.
  - Проанализированы стратегии развёртывания.
- Расширение библиотеки для специализации неравенствами.

Тестирование

Сценарий тестирования:

1. Специализация реляционной программы.
2. Трансляция остаточной программы в DSL языка реализации miniKanren.
3. Запуск программы.
4. Сравнение времени исполнения с CPD и оригинальной программой.

Реализация miniKanren: проект OCanren[10]

Реализация CPD для miniKanren: проект uKanren_transformations[11]

Реализация CPD для Prolog: проект ECCE[12]

Платформа: Intel Core i5-6200U CPU, 2.30GHz, DDR4, 12GiB.

[10] https://github.com/JetBrains-Research/OCanren

[11] https://github.com/kajigor/uKanren_transformations/

[12] https://github.com/leuschel/ecce

Тестирование

Программы для тестирования

sort Алгоритм реляционной сортировки.
Запрос : сортировка случайного списка длины 50

isPath Проверка принадлежности пути графу.
Запрос: поиск произвольного пути длины 10, принадлежащих графу с 21 вершиной и 50 рёбрами.

logint Реляционный интерпретатор формул логики высказываний.
Запрос: поиск 1000 истинных формул в данной подстановке.

lam Реляционный интерпретатор лямбда-выражений.
Запрос: поиск n термов в указаной форме.

Результаты тестирования

Базовый суперкомпилятор

Параметр	*Оригинал*	*Ecce*	*Cpd*	*CK*
sort	случайный список длины 50			
	8.42	12.28	13.2	0.26
isPath	произвольный путь длины 10			
граф 1	> 300	9	10	0.25
граф 2		2.7	2.8	0.1
logint	размер подстановки			
0	> 300	0.17	2.7	0.09
1		0.09	1.7	0.07
lam	редуцируются			
10 термов к себе	0.17	0.001	0.008	0.002
50 термов к себе	> 300	2.98	4.32	1.79
1000 термов к const	1.01	0.126	0.263	0.274

Результаты тестирования

Улучшения

Результаты работы

- Реализован и протестирован суперкомпилятор для задачи специализации.
- Применены подходы по улучшению качества суперкомпиляции для задачи специализации.
- Добавлены неравенства в библиотеку по специализации.
- Реализованы реляционные интерпретаторы для тестирования.
- Проведено тестирование и анализ результатов.
- Исправление багов библиотеки для специализации.

Спасибо за внимание!

- Работа будет представлена во второй половине мая на воркшопе по трендам логического программирования TEASE-LP.
- Ссылка на репозиторий:
  `https://github.com/RehMaar/uKanren-spec`