Применение метода суперкомпиляции для специализации реляционных программ

Мария Куклина, M42363

Университет ИТМО
Научный руководитель: Близнец Иван Александрович
Научный консультант: Вербицкая Екатерина Андреевна
2020

Реляционное программирование

Определение

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

Пример

Пример *запросов* для отношения "меньше или равно" $\text{leq}^o \subseteq \text{Int} \times \text{Int}$:

- $\text{leq}^o(1, 2)$ — проверка корректности отношения.
- $\text{leq}^o(X, Y)$ — поиск значений X и Y, при которых отношение выполняется.

miniKanren

Встраиваемый предметно-ориентированный язык реляционного программирования, представленный как набор операторов[1].

Применение

- Легковесная логическая подсистема проекта.
- Поиск лечения редких генетических заболеваний в точной медицине[2].
- Порождение решения задач поиска по решению задачи распознавания[3].

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

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

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

Постановка проблемы

- Сложность реализации эффективных программ.
- Поиск входов по выходам отношения работает значительно медленнее, чем “прямой” запуск.

Специализация

Определение

Автоматизированная техника оптимизации программ, при которой из программы удаляются избыточные вычисления, зависимые от частично известного входа[4].

- Частичная дедукция — класс методов специализации для логический языков, в частности, для Prolog.[5]
- Специализация miniKanren на основе *конъюнктивной частичной дедукции (CPD)*[6].
  - Сложна в поддержке, даёт нестабильные результаты.
  - Однако предоставляет библиотеку для построения специализаторов.

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

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

[6] *"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 об оптимизации.

- Суперкомпиляторы применяются во основном для функциональных языков[7].
- Полуавтоматическая суперкомпиляция для Prolog[8].
- Теоретические доводы для автоматической суперкомпиляции для Prolog[9].

[7] *"Introduction to Supercompilation"*, Sørensen и Glück, 1998

[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.

Суперкомпиляция для miniKanren

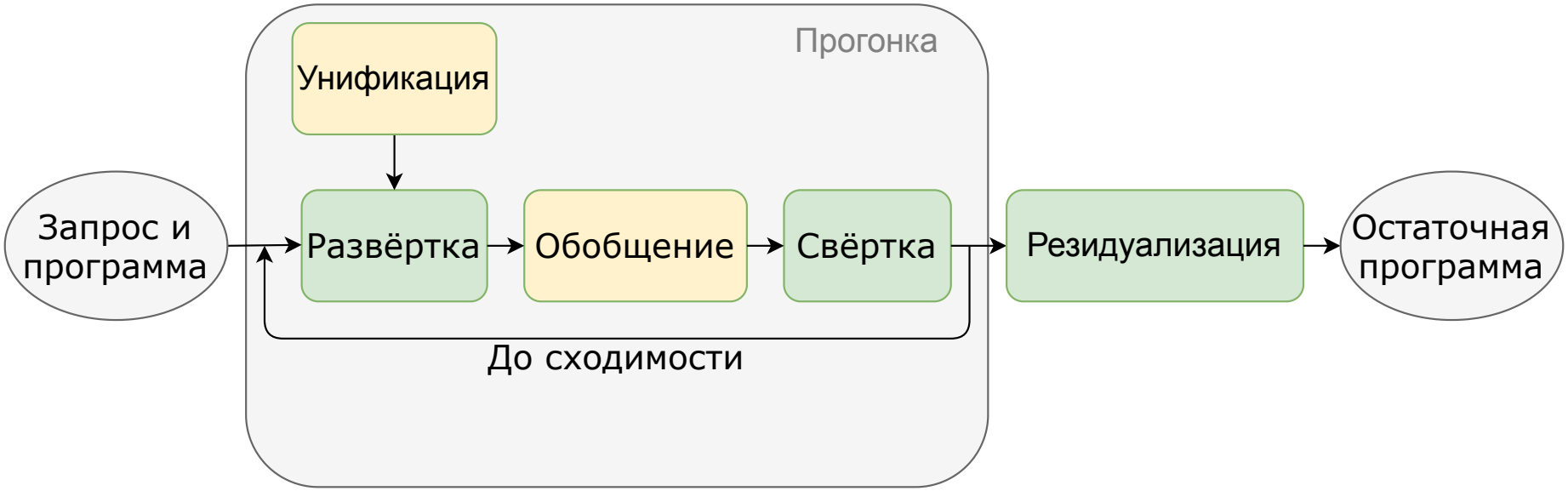

Рис. 1: Схема алгоритма суперкомпиляции

— библиотека по специализации miniKanren с дополнениями

— собственная разработка

Особенности шага развёртки для miniKanren

Развёртка определяет шаг символьного вычисления в суперкомпиляторе, на котором порождается множество возможных состояний программы.

Значимые отличия

- Несколько возможных видов развёртки.
- Допускается переупорядочивание элементов выражения.

Результаты задачи

- Реализован базовый алгоритм суперкомпиляции.
  - Развёртка рассматривает все возможные состояния.
  - Используемый алгоритм обобщения основан на алгоритме для конъюнктивной частичной дедукции, для которого доказана терминируемость.
- Разработан и реализован алгоритм построения оптимизированной программы по графу суперкомпиляции.

Улучшение суперкомпиляции для miniKanren

Проблемы

- Повторение символьных вычислений из-за стратегии свёртки.
- Классическое использование обобщения может приводить к избыточным вычислениям.
  - Существует техника обобщения, описанная в статьях[9].
  - Придумана специфичная для miniKanren техника обощения.
- Тривиальная стратегия вычисления порождает слишком много ветвей исполнения.
- В используемой реализации miniKanren нет способа эффективно сообщить, что можно прервать вычисление.

[9] *"Introduction to Supercompilation"*, Sørensen и Glück, 1998

Результаты задачи

- Применены подходы по улучшению алгоритма суперкомпиляции.
  - Добавлено кэширование.
  - Реализованы модификации обобщения.
  - Проанализированы и реализованы допустимые стратегии вычисления.
- Расширение библиотеки для специализации неравенствами.
- Расширение суперкомпилятора, при котором учитывается “негативная” информация.

Тестирование

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.

Сценарий тестирования:

1. Суперкомпиляция тестовой программы.
2. Трансляция остаточной программы в OCanren.
3. Замер времени исполенения.
4. Сравнение времени исполнения с оригинальной программой и реализациями CPD.

[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, сводящихся к указаной форме.

Результаты тестирования

Сравнение улучшений

	Вариации суперкомпиляторов					
Стратегии развёртки	*Б.С.*	*М.1*	*М.2*	*М.3*	*М.4*	*М.5*
Full	-	-	0.078	0.062	-	-
Full-non-rec	0.137	0.040	0.093	0.042	0.069	**0.040**
Seq	0.086	0.082	0.066	0.049	**0.050**	**0.041**
Non-rec	0.043	**0.031**	0.063	0.044	0.055	0.046
Rec	**0.037**	0.034	**0.045**	0.040	0.051	0.049
Min	**0.037**	0.039	0.049	0.041	0.054	0.045
Max	0.068	0.070	0.067	**0.036**	0.062	0.071
First	0.104	0.100	0.110	0.095	0.137	0.073

Рис. 2: Запуск logint для генерации формул с двумя переменными, секунды.

Результаты тестирования

Базовый суперкомпилятор

Параметр	*Оригинал*	*ECCE*	*CPD*	**M.C.**
sort	случайный список фиксированной длины			
50	8.42	12.28	13.2	**0.242**
isPath	10 путей			
граф 3	> 300	**1.03**	1.19	1.81
isPath	произвольный путь длины 10			
граф 1	12.51	1.01	1.20	**0.48**
граф 2	> 300s	1.73	2.09	**0.48**
logint	размер подстановки			
0	> 300	0.17	2.7	**0.11**
1		0.09	1.7	**0.07**
lam	термы в нормальной форме			
50 термов	> 300	2.98	0.08	**0.04**

Рис. 3: Результаты сравнения алгоритмов специализации, секунды

Результаты работы

- Реализован и протестирован суперкомпилятор для задачи специализации.
- Применены подходы по улучшению качества суперкомпиляции для задачи специализации.
- Добавлены ограничения неравенства в библиотеку по специализации.
- 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`