Применение метода суперкомпиляции для специализации реляционных программ

Куклина Мария Дмитриевна

Университет ИТМО

Научный руководитель: к.ф-м.н. Близнец Иван Александрович
Научный консультант: Вербицкая Екатерина Андреевна

2020 г.

MASTER OF
SOFTWARE
ENGINEERING

Реляционное программирование

Реляционное программирование

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

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

- $\text{leq}^o(1, 2)$ — проверить, что отношение $1 \leq 2$ выполняется.
- $\text{leq}^o(X, Y)$ — поиск всех значений X и Y, таких что $X \leq Y$.

miniKanren

miniKanren

Встраиваемый предметно-ориентированный язык реляционного программирования, представленный как набор операторов, которые нужно реализовать в хостовом языке[1].

Важное преимущество

miniKanren реализует *полный поиск*: гарантируется, что каждый ответ будет со временем найден.

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

Постановка проблемы

- Реализовывать эффективные программы сложно.
- Производительность запроса сильно зависит от того, значения каких компонент отношения нужно найти.

Специализация

Определение

Автоматизированная техника оптимизации программ, при которой из программы удаляются избыточные вычисления, зависимые от частично известного входа[2].

- Частичная дедукция — класс методов специализации для логический языков, в частности, для Prolog.[3]
- Специализация miniKanren на основе *конъюнктивной частичной дедукции (CPD)*[4]:
  - даёт нестабильные результаты;
  - предоставляет библиотеку для построения специализаторов.

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

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

[4] *"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[5].
- Существует полуавтоматическая суперкомпиляция для Prolog[6].
- Имеются теоретические доводы в пользу автоматической суперкомпиляции для Prolog[7].

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

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

[7] *Turchin's Supercompiler Revisited - An operational theory of positive information propagation*, Sørensen, 1996

Цели и задачи

Цель

Улучшение результатов специализации реляционных программ путём применения метода суперкомпиляции.

Задачи

- Реализовать суперкомпилятор для miniKanren.
- Рассмотреть возможные методы улучшения получившегося суперкомпилятора.
- Протестировать суперкомпилированные программы и сравнить их с получившимися в результате применения CPD и с оригинальными.

Суперкомпиляция для miniKanren

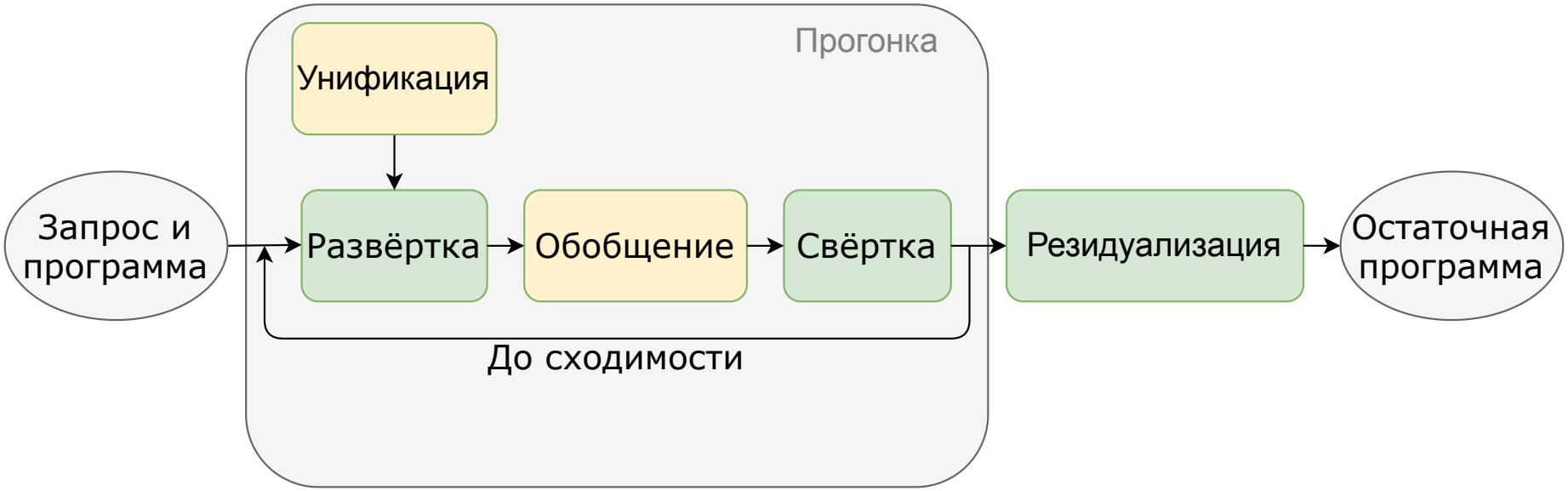

Рис. 1: Схема алгоритма суперкомпиляции

— библиотека по специализации miniKanren с дополнениями

— собственная разработка

Особенности miniKanren для реализации развёртки

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

Значимые особенности

- Существует несколько возможных способов реализации развёртки.
- Допускается переупорядочивание элементов выражения.

Результаты задачи

- Реализован базовый алгоритм суперкомпиляции.
- Разработан и реализован алгоритм построения оптимизированной программы по графу суперкомпиляции.

Улучшение суперкомпиляции для miniKanren

Проблемы

- Стратегия свёртки приводит к повторению символьных вычислений.
- Классическое использование обобщения может приводить к избыточным вычислениям.
  - Существует техника обобщения, описанная в статьях[8].
  - 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.

[8] *"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 неравенствами библиотеки для специализации.
- Расширение суперкомпилятора, учитывающее информацию о неравенствах.

Тестирование

Реализация miniKanren: проект OCanren[9]

Реализация CPD для miniKanren: проект uKanren_transformations[10]

Реализация CPD для Prolog: проект ECCE[11]

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 результатами применения CPD.

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

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

[11] 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, редуцирующиеся к указаной форме.

Сравнение улучшений

- Были рассмотрены 5 модификаций базового алгоритма с 8 стратегиями развертки.
  - 1 модификация описана в статьях.
  - 4 модификации предложены мной.
- Все модификации улучшают работу оригинального суперкомпилятора.
- Систематически модификация, описанная в статьях, давала результат лучше или не сильно хуже, чем остальные модификации.
- По результатам экспериментального исследования была выявлена лучшая стратения развёртки.

Результаты тестирования

Сравнение суперкомпиляторов с существующими решениями

Параметр	*Оригинал*	*ECCE*	*CPD*	**Б.С.**	**М.С.**
sort	случайный список фиксированной длины				
50	8.42	12.28	13.2	**0.239**	0.242
isPath	10 путей				
граф 3	> 300	**1.03**	1.19	2.43	1.81
isPath	произвольный путь длины 10				
граф 1	12.51	1.01	1.20	1.28	**0.48**
граф 2	> 300s	1.73	2.09	0.85	**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	0.08	**0.04**

Рис. 2: Время исполнения программ для данных специализаторов, секунды

Результаты работы

- Реализован и протестирован суперкомпилятор.
- Применены подходы по улучшению качества суперкомпиляции.
- Добавлены неравенства в библиотеку для специализации.
- Исправлены баги библиотеки для специализации.
- Работа была представлена на воркшопе по трендам логического программирования TEASE-LP'20.
- Ссылка на репозиторий: `https://github.com/RehMaar/uKanren-spec`

Дополнительные слайды

Пример сравнения модификаций суперкомпилятора

	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. 3: Время исполнения запроса к logint для генерации формул с двумя переменными, секунды.