Борьба с расходимостью реляционных программ

Дима Розплохас
Научный руководитель: Дмитрий Юрьевич Булычев

Лаборатория языковых инструментов JetBrains

Реляционное программирование

Программа = отношение: аргументы и результат не различаются. Если зафиксировать значения части из них, при исполнении программы будут искаться всевозможные подходящие значения остальных.

MiniKanren — семейство встраиваемых реляционных языков
OCanren — реляционный язык, встроенный в OCaml

Можно транслировать:
функция в OCaml $\longrightarrow$ отношение в OCanren

Постановка задачи

Проблема: Завершаемость программы может зависеть от порядка перчисления условий. Оптимальный порядок может зависеть от направления вычислений.
Это нарушает декларативность.

Постановка задачи

Проблема: Завершаемость программы может зависеть от порядка перчисления условий. Оптимальный порядок может зависеть от направления вычислений.
Это нарушает декларативность.

Цель: изменить алгоритм исполнения реляционных программ, так чтобы расширить класс завершающихся программ.
Идея — будем менять код программы в процессе её исполнения.

Задачи

1. Придумать расширение алгоритма исполнения реляционных программ
2. Реализовать расширенный алгоритм и проверить его работоспособность
3. Доказать хорошие свойства расширенного алгоритма

Описание расширения

Нужна проверка на расходимость с односторонней ошибкой (проверка сработала $\Rightarrow$ исполнение программы гарантированно не завершится).

Наша проверка: если в рекурсивном вызове отношения предоставляется меньше информации об аргументах, то выполнение программы не завершится.

Если в процессе выполнения срабатывает проверка, бросается исключение.

Описание расширения

При проходе вверх по стеку вызовов мы пробуем другие порядки перечисления условий и перезапускаем вычисления.
Если все испытанные порядки дают расходящуюся подпрограмму, исключение идёт выше.

Трудность: нужно сохранить ленивый интерфейс результата исходного алгоритма — ответы выдаются по мере нахождения и только по одному разу.
Решение: для каждого выданного ответа будем помнить путь, которым он был найден, и перехватывать его, когда он будет найден снова при перезапуске.

Реализация и апробация

Особенность реализации: переход от shallow embedding к deep embedding. Требует изменения синтаксиса.

Набор тестовых отношений, расходимость которых устраняется: функции работы со списками, сортировка, двоичная арифметика и арифметика Пеано.

Свойства расширенного алгоритма

Корректность

Множества ответов, получаемые при обычном и расширенном способах вычисления совпадают.

Сохранение завершаемости

Если программа завершалась при обычном способе вычисления, она завершится при расширенном.

Оптимальность для идеальной проверки

Если существует оптимальный порядок перечисления условий в программе (порядок, при котором она завершается), а при любом неоптимальном порядке вычисления заканчиваются исключением, то программа завершится при расширенном способе вычисления.

Результаты

- Работающий прототип расширенного алгоритма исполнения реляционных программ, сохраняющего ленивый интерфейс результата
- Набор простых программ с устраненной расходимостью
- Доказательство хороших свойств расширенного алгоритма

Результаты

- Работающий прототип расширенного алгоритма исполнения реляционных программ, сохраняющего ленивый интерфейс результата
- Набор простых программ с устраненной расходимостью
- Доказательство хороших свойств расширенного алгоритма

в этом семестре

Код: https://github.com/rozplokhas/OCanren/tree/deep