УДК 004.4'422

РЕАЛИЗАЦИЯ АНАЛИЗАТОРА И ОПТИМИЗАТОРА СРЕДНЕГО УРОВНЯ В КОМПИЛЯТОРЕ ЯЗЫКА ВЫСОКОГО УРОВНЯ

Щецова А.Ю. (Национальный исследовательский университет ИТМО)
Научный руководитель – к.ф.-м.н. Недоря А.Е.
(Huawei RRI, SPb OS Lab)

Цель работы – разработка набора анализирующих проходов и medium-level оптимизаций, улучшающих качество статического анализа кода и его эффективность, для экспериментального языка высокого уровня. В работе исследуются оптимизации на основе уже существующего AST с построением дополнительных информационных структур.

Введение.

Большинство современных компиляторов используют для генерации кода LLVM и GCC, что существенно сокращает скорость работы компиляторов и повышает их надежность а также эффективность программ за счет широкого набора оптимизаций. В то же время, низкоуровневая оптимизация (на уровне языково-независимого внутреннего представления) во многих случаях не оптимальна, так как на этом уровне уже потеряна часть информации о исходных типах, в том числе о классах, и об операторах управления.

Для увеличения эффективности современные компиляторы добавляют еще один уровень оптимизаций (ML – medium-level), для которого используется дополнительное представление (medium-level IR). Такие оптимизаторы реализованы в компиляторах Rust (MIR) и Swift (SIL). Большая работа в этом направлении ведется в LLVM (ML IR). Принципиальной особенностью оптимизаций среднего уровня является существенная привязка их к семантике языка программирования и модели выполнения, именно эта привязка позволяет сделать дополнительные оптимизации. Одновременно, это особенность не позволяет использовать напрямую решения, сделанные в других компиляторах.

Перед нами стоит задача добавить ML-оптимизатор (MLO) в существующий экспериментальный компилятор, а для этого

- выбрать набор оптимизаций,
- определить набор необходимой для оптимизаций информации,
- разработать набор анализирующих проходов,
- реализовать набор оптимизаций, улучшающих качество статического анализа кода и его эффективность.

Основная часть.

Опуская детали, мы можем рассматривать используемый компилятор как программную систему, состоящую из следующих основных уровней:

1. Front-end, который проверяет синтаксическую и семантическую корректность единицы компиляции и строит AST (Abstract Syntax Tree)
2. Генератор внутреннего представления, который по AST строит промежуточное представление нижнего уровня (например, LLVM IR)
3. Генератор кода, строящий по представлению машинный код (например, используя LLVM или другие инструменты)

При этом экспериментальный компилятор представляет собой «конструктор», в котором есть несколько сменных вариантов 2 и 3 уровней.

Очевидно, что ML-оптимизатор должен встраиваться между 1 и 2 уровнями и на входе получать AST. Есть два варианта того, что MLO передает следующему уровню:

- AST (возможно расширенное), что позволяет использовать существующие генераторы IR без изменений или с небольшими изменениями
- Другое представление, что приводит к необходимости переделки генераторов

В рамках проекта было принято решение использовать способ AST -> AST с построением дополнительных информационных структур.

На первом этапе идет работа над выполнением анализа и оптимизаций:
- Анализ корректного выхода из функций
- Удаление мертвого кода
- Удаление избыточных проверок индекса массива
- Удаление избыточных проверок на null
- Девиртуализация

В дальнейшем список будет расширен.

Выводы.

Решение об использовании MLO в компиляторе будет определятся результатами измерения бенчмарков: время выполнения, размер кода, увеличение времени компиляции. В качестве тестов будет использовано подмножество тестов, аналогичных JMH.

Также будут сделан вывод о пригодности подхода AST -> AST или о необходимости переходить на другое представление для анализа и оптимизаций.

Щецова А.Ю. (автор) Подпись

Недоря А.Е. (научный руководитель) Подпись