Встроенные функции Рефала-5λ

Студент группы ИУ9-81Б: Агазаде Х.Р.

Научный руководитель: Коновалов А.В.

Постановка задачи

- Разработка алгоритма оптимизации встроенных функций в компиляторе.
- Изменение грамматики языка Рефал-5λ и организация поддержки новой синтаксической конструкции, необходимой для реализации оптимизации в компиляторе.
- Оптимизация самоприменимого компилятора с помощью разработанного решения.
- Оценка результатов оптимизации встроенных функций.

Обзор языка

- Рефал-5λ – язык функционального программирования. Он ориентирован на символьные вычисления. Основные конструкции языка – функции.
- Основные виды символьных данных: составные слова, идентификаторы, литеры и числа.
- Функции могут быть описаны через указатель и через объекты замыкания.
- Для описания структур данных используются круглые скобки.
- Последовательность символов и структур данных образуют выражение.
- Переменные описывают некоторое выражение. S-переменные соответствуют символьным данным, e-переменные могут содержать любое выражение.

Оптимизация функции Mu

- Эта функция принимает на вход функцию и аргумент и возвращает значение, которое было получено при вызове функции с данным аргументом.
- Функцию можно определить пятью способами: указать имя функции как идентификатор или строку литер в круглых скобках, использовать арифметический символ в одинарных или двойных кавычках, записать функцию в объекте замыкания или использовать указатель на функцию.
- В библиотеке Рефала требуется использование функции Mu, но её использование не позволяет оптимизировать функции библиотеки, поэтому важно заменять вызовы функции Mu на непосредственные вызовы функций при компиляции.
- Пример: `<Mu ('Add') 1 2>` заменяется на `<Add 1 2>`

Оптимизация функции Add

- Функция вычисляет сумму своих аргументов.
- Аргументы функции могут быть целыми числами со знаком.
- Данная функция оптимизируется компилятором, если её аргументы константны.
- Пример: `<Add '-' 1 2>` заменяется на `1`

Оптимизация функции Chr

- Эта функция сопоставляет каждому своему аргументу литеру из таблицы символов ASCII.
- Аргумент данной функции – последовательность чисел.
- Функция оптимизируется для каждого аргумента отдельно: числа заменяются на результат вычисления функции, переменные заменяются вызовом функции.
- Пример: `<Chr e.x 115>` заменяется на `<Chr e.x> 's'`

Оптимизация функции Implode

- Эта функция распознает префикс максимальной длины, который является идентификатором и возвращает его.
- Оптимизация этой функции происходит, если все её аргументы статичны.
- Пример: `<Implode 'ident#@$'>` заменяется на `ident '#@$'`

Оптимизация функции First

- Эта функция выделяет из выражения префикс и суффикс определённой длины.
- Её аргументы: число и выражение.
- Оптимизация происходит, если первый аргумент статичен, а в выражении нет e-переменных, т.к. в таком случае невозможно на этапе компиляции узнать длину выражения.
- `<First 2 a s.x c>` заменяется на `(a s.x) c`

Оптимизация функции Type

- Эта функция принимает на вход выражение, возвращает тип и подтип его первого символа, а также само выражение.
- Данная функция оптимизируется, если её первый аргумент статичен.
- Пример: `<Type 'a' b>` заменяется на `'Lla' b`.

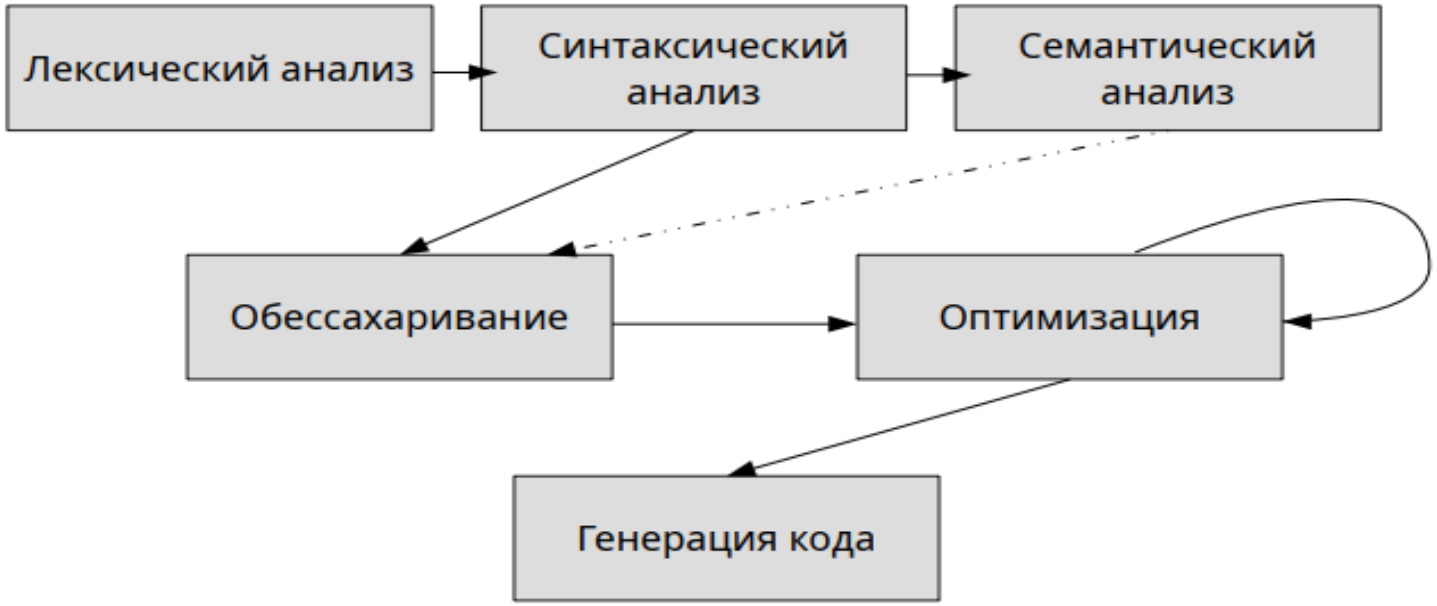
Расширение синтаксиса

- В рамках работы было разработано расширение синтаксиса для применения оптимизации внутренних функций – ключевое слово `$INTRINSIC`
- Если у встроенной функции есть метка `$INTRINSIC,` то её вызовы оптимизируются.
- Оптимизации можно включить, передав специальные флаги компилятору.

Пример:

```
$INTRINSIC Add, First;
```

Компилятор языка Рефал-5λ



Стадия *оптимизации* выполняется несколько раз до тех пор, пока синтаксическое дерево не перестанет меняться, или компилятор не достигнет лимита итераций оптимизации.

Тестирование

Оценка времени работы производилась на разных версиях компилятора:

1. Стандартная версия компилятора.
2. Версия, собранная с использованием функции Mu в библиотеке.
3. Версия, собранная с оптимизацией встроенных функций.
4. Версия, собранная с использованием оптимизированной функции Mu в библиотеке.
5. Версия, собранная с оптимизацией встроенных функций и использованием функции Mu в библиотеке.

Сборка компилятора в каждом случае производилась 13 раз. Для времени компиляции были измерены медиана и значения на границах первого и четвертого квартилей. Также в каждом из случаев было измерено количество шагов рефал-машины.

Тестирование

№ теста	Количество шагов	Время компиляции, с	Время компиляции, I квартиль, с	Время компиляции, IV квартиль, с
1	23506400	35.95	35.67	36.15
2	27082390	37.48	37.34	37.92
3	21471802	34.21	33.82	34.43
4	25012700	36.12	35.77	36.27
5	24391125	35.56	35.51	35.66

Заключение

- В процессе выполнения этой работы были выполнены все поставленные задачи. Был изучен язык Рефал-5λ, устройство его компилятора и его библиотека.
- Компилятор языка был модифицирован: реализована поддержка синтаксиса для оптимизации встроенных функций, произведено тестирование оптимизации.
- Оптимизации были применены внутри компилятора, что дало прирост производительности компилятора в 4.8%. Оптимизация функции Mu не смогла полностью убрать эффект замедления компиляции от внедрения этой функции в библиотеке, но заметно снизила этот эффект. Компилятор с оптимизациями внутренних функций и с использованием Mu ускорился на 1.1%.