Разработка и реализация алгоритмов автоматической сборки мусора в системе LuNA.

Зимняя школа 2020

Исполнитель: Плешков Андрей, НГУ ФИТ 3 курс

Руководитель: Перепёлкин Владислав Александрович

31.01.2020

Проблема освобождения памяти

В данный момент времени, в системе LuNA не работает автоматическая сборка мусора. Одна из возможностей управления памятью это указание того, сколько раз собираются обратиться к этому фрагменту данных, что, безусловно, не так удобно и замедляет процесс разработки.

Идея решения

С помощью уже существующего функционала можно уменьшить масштаб проблемы: во время компиляции вести подсчёт количества использований фрагментов данных и автоматически вставлять правила для удаления фрагментов данных, после того, как к ним закончатся обращения

Для этого считывается уже существующее представление, сохранённое в виде json, после чего тело каждой процедуры разбирается на составляющие и в зависимости от типа выражения происходит подсчёт числа использований фрагмента данных.

Реализация

После завершения разбора программы, к json-представлению программы добавляются новые правила по удалению тех фрагментов, количество использований получилось подсчитать.

В уже существующий код компилятора языка LuNA был встроен модуль, позволяющий анализировать представление кода программы и вставлять рекомендации к удалению фрагментов данных.

Трудности реализации

- Подсчёт количества использования элементов массива: Для человека очевидно, что x[2+2] и x[2*2] это один и тот же элемент, однако в представлении программы это таковым не является. Для решения этой проблемы можно воспользоваться методами символьных вычислений.
- Сложности при обработке условных операторов и циклы.

Тестирование

Для проверки корректности модификаций использовались автоматические тесты LuNA.

Для подтверждения удаления фрагментов данных были сравнены логи запусков программы без и с использованием модуля.

Тестирование

```
| Wait ID<0, 0> -> DF<int>(7) by CF 139867489046352 : 4 | ./src/rts/cf.cpp:94
| Wait ID<0, 0> -> DF<int>(7) by CF 139867489046800 : 8 | ./src/rts/cf.cpp:94
| Invoke CF 139867489046352 : 5 | ./src/rts/rts.cpp:503
| Wait ID<0, 0> -> DF<int>(7) by CF 139867489046352 : 5 | ./src/rts/cf.cpp:94
| Finish CF 139867489046800 : 8 | ./src/rts/rts.cpp:521
| Finish CF 139867489046352 : 5 | ./src/rts/rts.cpp:521
| WORKLOAD + -1 => 1 | /mnt/d/Projects/source/repos/lo3//include/idle_stopper.h:122
| WORKLOAD + -1 => 0 | /mnt/d/Projects/source/repos/lo3//include/idle_stopper.h:122
| WORKLOAD: CF destroyed | ./src/rts/cf.cpp:23
| WORKLOAD: CF destroyed | ./src/rts/cf.cpp:23
```

```
| WORKLOAD + -1 => 2 | /mnt/d/Projects/source/repos/lo3//include/idle_stopper.h:122
| Wait ID<0, 0> -> DF<int>(7) by CF 139670860075280 : 7 | ./src/rts/cf.cpp:94
| WORKLOAD: CF destroyed | ./src/rts/cf.cpp:23
| WORKLOAD + -1 => 1 | /mnt/d/Projects/source/repos/lo3//include/idle_stopper.h:122
| Invoke CF 139670860075280 : 8 | ./src/rts/rts.cpp:503
| Wait ID<0, 0> -> DF<int>(7) by CF 139670860075280 : 8 | ./src/rts/cf.cpp:94
| WORKLOAD: CF destroyed | ./src/rts/cf.cpp:23
| Finish CF 139670860075280 : 8 | ./src/rts/rts.cpp:521
| WORKLOAD + -1 => 0 | /mnt/d/Projects/source/repos/lo3//include/idle_stopper.h:122
| WORKLOAD: CF destroyed | ./src/rts/cf.cpp:23
```

Заключение

Был реализован алгоритм автоматической генерации рекомендаций по удалению фрагментов данных в системе фрагментированного программирования LuNA.

В дальнейшем планируется добавить поддержку массивов, обработку циклов и условных операторов.