Расширение транслятора предметно-ориентированного языка Landau арифметикой сдвоенных чисел двойной точности (double-double)

Языково-ориентированное программирование

- ЯОП представляет собой метод проектирования интерфейса
- Идеален для задач, которые требуют **минимальной нотации** с сохранением **максимальной точности**
- Создаваемые языки называются **предметно-ориентированными языками** (DSL), они удобны для применения в задачах с узкой специализацией
- Racket идеально подходит для ЯОП из-за своей **системы макросов**. Они работают в стиле компилятора, упрощая преобразование кода. Система макросов Racket лучше любой другой.

Преимущества Racket

- Racket был разработан с нуля специально для ЯОП
- Данный язык обладает глубоко проработанной системой макросов, что важно для определения синтаксиса языка
- Racket позволяет компилировать текст в специальные *синтаксические объекты* при написании макросов
- Синтаксические объекты хранят не только строку макроса, но так же связанные с ней метаданные (местоположение в коде, синтаксический контекст и дополнительные свойства), эти данные используются при обмене между макросами
- Сохранение лексического контекста сильно упрощает написание предметно-ориентированных языков, например уменьшается возможность коллизий названий объектов

Landau

Landau - язык для динамических систем с автоматическим дифференцированием.

В языке есть функции, циклы, условия, вещественные числа, массивы.

Landau достаточно полон, для того, чтобы выразить и продифференцировать уравнение любой сложности за минимальные усилия.

Преимущества языка

- Широкие возможности языков общего назначения затрудняют реализацию удобной системы AD, особенно при наличии многомерных функций со многими независимыми переменными.
- Существующие инструменты для дифференцирования функций, такие как MATLAB (ADMAT) или Mathematica (TIDES) часто требуют больших усилий (по сравнению с языками общего назначения) для ввода практической динамической системы большого размера с множеством свободных переменных.

Компоненты реализации Landau

- Транспайлер Landau написан на Racket и состоит из двух основных частей:
- Синтаксический анализатор
  - Разворачивание циклов в последовательности присваиваний
  - Отслеживание зависимости переменных и их производных
  - Проверка программы на корректность
- Генератор кода
  - Генерация ANSI C и Racket кода

Синтаксический анализатор

Lexer:

- Разбивает исходный код на лексемы (токены), следуя правилам языка
- Тем самым производится идентификация токенов, они относятся к одному из классов (идентификатор, оператор и т.д.)

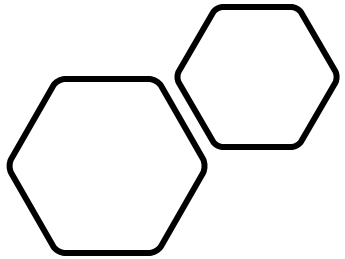
Parser:

- Получает на вход список токенов, и генерирует из них абстрактное синтаксическое дерево
- На данном этапе обрабатывается большинство возможных ошибок в коде

Генератор кода

- Получает на вход AST из синтаксических объектов
- Использует макросы Racket для трансформирования литералов исходного кода без его выполнения
- Узлы дерева преобразуются в соответствии с грамматикой целевого языка (завести необходимые переменные, выставить соответствующие идентификаторы и т.д.)
- В итоге код Landau транслируется в код на соответствующем языке (Racket/ANSI-C)

Double-double



- Число double-double выражается как сумма двух компонент double (большого и маленького числа) x=x_l+x_h.
- Арифметика double-double имеет программную реализацию.

Интеграция double-double

- Описание модулей, в которых происходили изменения в связи с добавлением double-double.

- #target-config.rkt модуль
  описывает парсинг конфигурационного файла config.json, экспортирует глобальный биндинг TARGET, который содержит все бекенд-настройки, в том числе, язык генерации кода и представление действительных чисел.

- Был изменен параметр (target-real-implementation), отвечающий за бэкенд вещественного типа данных, изменен парсинг параметров, добавлены новые ветвления.

- #constant-propagation.rkt модуль с логикой для разрешения констант во время компиляции. Константы должны использовать представление 80-bit если выбран long-double или double-double бекенд.

- В связи с изменением параметра target-real-implementation были добавлены новые ветвления.

- #semantics.rkt модуль с логикой генерации кода вычисления производных. Он использует информацию из синтаксического анализатора и генерирует код в промежуточном представлении, не зависящем от выбранного бэкенда.

- Были добавлены ветвления для операций с double-double и double (сложение/вычитание, умножение/деление).

#metalang.rkt модуль описывает макросы промежуточного представления для всех бэкендов. Каждый добавленный сюда макрос должен быть экспортирован из semantics.rkt.

Написаны новые функции, которые используются в semantics для операций с операндами double и double-double.

- #combinators.rkt модуль описывает функции кодогенерации для ansi-c бэкенда.

- Написаны новые функции и ветвления, которые генерируют ANSI-C код.

- К примеру функция для сложения в случае double-double бэкенда генерирует C-функцию dd_add, а в случае double и long double бэкендов генерирует "+".

Внедрение сгенерированного ANSI-C кода с помощью Landau в проект с задачей N тел

- На языке Landau была написана функция для расчёта состояния системы N тел в следующий момент времени.
- Затем был сгенерирован C код.
- Из библиотеки QD были выбраны и переписаны на ANSI-C нужные функции для работы с double-double.
- Из этого кода была сгенерирована статическая библиотека и интегрирована в проект с задачей N тел.

Вывод

- В Landau был успешно интегрирован тип данных double-double.
- Была написана и применена в стороннем проекте функция.