Блок I. Расширение конвертера формальных языков

Примечание: языки C++ или Рефал.

В принципе, все задачи, кроме линтера (особенно сишного, рефальский можно), можно продолжать на ВКР, но уже не с такими формулировками, а в более исследовательском контексте

Задача I.1. - Преобразование и анализ регулярных выражений с отрицанием

Задача I.2. - DSL для модульного расширения конвертера формальных языков

Примечание: не рекомендуется брать эту задачу тем, кто не был в группе Чиполлино. Очень рекомендуется брать в случае взятия другими студентами задач I.1, I.4 или I.6. Важно! Тут придётся немного освоиться с Рефалом, т.к. модульное расширение должно затрагивать и фронтенд.

Задача I.3. - Генерация автоматов-преобразователей на базе языков переписывания графов

Примечание: разработка вспомогательного модуля

Задача I.4. - Интеграция выражений с обратными ссылками в конвертер

Примечание: не рекомендуется брать эту задачу тем, кто не был в группе Чиполлино. Базируется (но не ограничивается) на ВКР и исследовательской работе Исмагиловой Дарьи.

Задача I.5. - Линтер для Чиполлино (формальное название — позже)

Также не рекомендуется тем, кто не участвовал в разработке конвертера раньше. Написание модуля качества контроля кода и логики (анализ зависимостей, накрытие тестами, и пр) Приветствуется (по согласованию) интеграция с внешними линтерами. Эта задача может разбиваться даже на две, т.к. языка там два: линтер для бэка и линтер для фронта. Но следует учесть, что на ВКР нельзя будет продолжать обе.

Задача I.6. - Преобразование и анализ автоматов со стековой памятью

В этой группе возможно «рабовладельчество», т.е. кооперативная работа с третьекурсниками, проходящими курс ТФЯ, по добровольному согласию всех сторон

Блок II. Теоретические основания формальных языков

Особенность задач в этом блоке: каждая из них, кроме 4 и 5 одновременно, может быть продолжена на ВКР, независимо от других

Задача II.1. - Декомпозиция Родеса-Крона ДКА по алгоритму Гинцбурга

Примечание: задача занята

Задача II.2. - Выявление серий однозначных образцов с помощью анализа моделей в теории строк

В данном случае модели экзистенциальные с равенством: уравнения в словах (т.е. две строки в смешанном алфавите букв и переменных, между которыми стоит знак равенства), построенные на базе образцов. Рекомендуется использовать внешние решатели (SMT-солверы).

Примечание: задача исследовательская, связанная с разработкой эвристик.

Задача II.3. - Построение компактного описания полугрупп в терминах правил переписывания

Примечание: вы должны хорошо помнить теорию про трансформационные моноиды

Задача II.4. - Добавление переписывающих лемм к алгоритму развёртки уравнений по лемме Леви

Алгоритм развёртки уравнений по лемме Леви подразумевает подстановку переменных на базе анализа префиксов уравнений. При этом система уравнений может распадаться на несколько, каждое из которых может рассматриваться также как правило переписывания.

Задача II.5. - Добавление обработки регулярных рестрикций к алгоритму развёртки уравнений по лемме Леви

Продолжение, возможно, следует...

Блок III. Функциональные языки

Задача III.1. - Исследование CPS-преобразований в типизированном лямбда-исчислении

Примечание: только для тех, кто уже знает Хаскелл.

Задача III.2. - Построение системы автоматического вывода термов по типу в базовом Haskell

Язык реализации — желательно функциональный, но вопрос может обсуждаться. Формула Крипке — один из обязательных тестов

Задача III.3. - Использование рекомпрессии для сопоставления с образцом

Рекомпрессия — метод сжатия однородных блоков констант и пар констант с использованием расширенного алфавита. Используется как при сжатии данных, так и в теоретических алгоритмах анализа формальных моделей. Требуется применить ограниченную технику рекомпрессии к задаче сопоставления строки с образцом.

Язык реализации не существен, но лучше обойтись без извращений (не C++)

Задача III.4. - Использование техник быстрого решения ограниченных уравнений для сопоставления с образцом

Язык реализации не существен, но лучше обойтись без извращений (не C++)

Задача III.5. - Использование техники входных форматов для сопоставления с группой образцов

В функциональных языках образцы имеют приоритеты при чтении сверху вниз. В связи с этим возникает задача древесного или графового представления группы образцов, допускающего совместное сопоставление (с экономией откатов).

Язык реализации — желательно функциональный.

Задача III.6. - Анализ симуляций в Рефал-программах

При автоматическом порождении программ часто в результатной программе появляются дублирующие структуры, реализующие один и тот же функционал. Требуется провести анализ симуляций, выявляющий дублирование. Входным языком анализатора по умолчанию предполагается Рефал, язык реализации — по обсуждению.

Продолжение, возможно, следует...