Санкт-Петербургский государственный университет
Кафедра системного программирования

Разработка инфраструктуры для автоматизации проверки задач по курсу «Теория формальных языков»

Николай Алексеевич Пономарев, группа 21.Б10-мм

Научный руководитель: доцент кафедры системного программирования, к.ф.-м.н., Григорьев С. В.

Санкт-Петербург
2024

Введение

- На СПбГУ читается курс по формальным языкам
- Домашние задания требуют реализации нетривиальных алгоритмов
- Тесты обязательны, но не всегда покрывают граничные случаи
- Алгоритмические задачи хорошо проверяются с помощью property-based тестирования

Постановка задачи

Цель: разработать инфраструктуру для автоматизации проверки задач по курсу «Теория формальных языков»

Задачи:

1. Изучить способы автоматического тестирования учебных задач
2. Создать базовую инфраструктуру для автоматического тестирования
3. Произвести проверку автоматических тестов, путём реализации решений
4. Исправить накопленный технический долг
    1. Сменить систему управления зависимостями
    2. Обновить зависимости

Виды автоматических тестов

Автоматические тесты можно характеризовать несколькими способами:

- По видимости
  - открытые
  - закрытые
- По способу реализации
  - ручные
  - property-based
  - смешанные
- По способу сдачи решений
  - Learning Management Systems
  - Pull Request
  - GitHub Classroom

Требования к реализации

- Тесты должны быть открытыми
- Тесты должны быть property-based
- Задачи должны сдаваться с помощью PR
- Имена и сигнатуры требуемых функций и классов известны заранее
- Архитектуру проекта и связь между модулями студенты определяют сами

Используемые технологии

Используемые технологии во многом были зафиксированы

- Python — основной язык программирования
- Pytest — библиотека для тестирования
- CFPQ_Data — библиотека-датасет графов и грамматик
- Pyformlang — библиотека для работы с грамматиками, регулярными выражениями и различными конечными автоматами
- NetworkX — библиотека для работы с графами
- SciPy — библиотека для работы с разреженными матрицами
- ANTLR4 — инструмент для генерации парсеров
- Grammarinator — инструмент генерации слов по грамматике ANTLR4

Инфраструктура решения

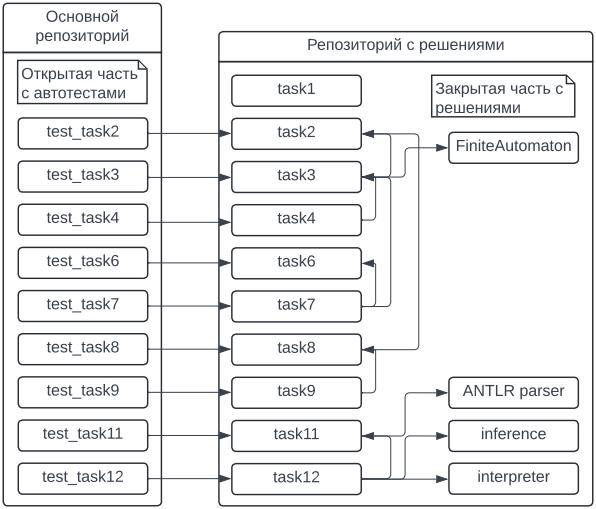
Тесты к нереализованным студентом задачам необходимо пропускать. Это можно сделать с помощью:

- создания отдельных репозиториев для каждой задачи
- конфигурации системы тестирования
- проверки наличия требуемых функций

Условный запуск тестов

```
try:
    from project.taskN import some_func
except ImportError:
    pytestmark = pytest.mark.skip("Task N is not ready to test!")
```

Структура решения



Обновление используемых инструментов

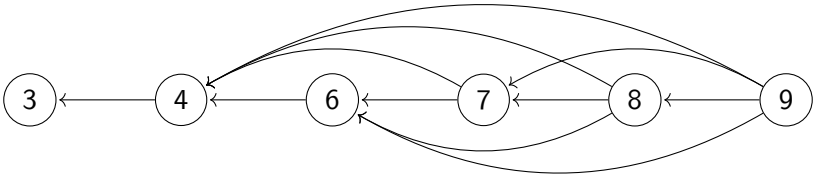
- При реализации потребовалось обновить пакеты
- Получили проблемы с pip
- В семестре обновили зависимости библиотеки CFPQ_Data
- Перешли на Poetry

Апробация

Весной курс читался первокурсникам магистратуры ИТМО. В течение семестра были выявлены следующие проблемы:

- Отсутствие ручных тестов
- Необходимость тестирования всех требуемых функции
- Необходимость предупреждать студентов о том, что решения нужно оптимизировать

Рис.: Зависимость задач друг от друга

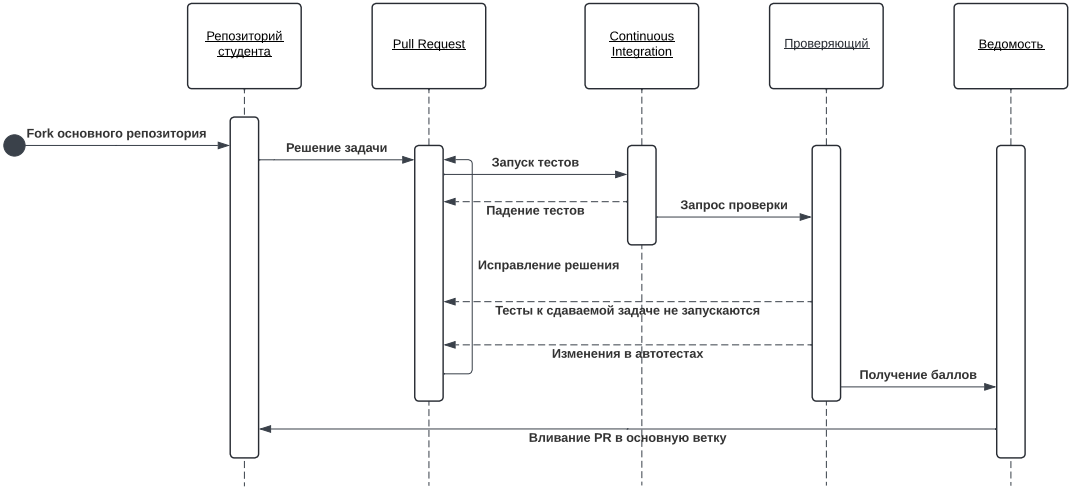


Результаты

1. Изучены способы автоматического тестирования учебных задач
2. Создана базовая инфраструктура для автоматического тестирования
3. Произведена проверка автоматических тестов, путём реализации решений
4. Исправлен накопленный технический долг

Исходный код тестов и инфраструктуры:
`https://github.com/FormalLanguageConstrainedPathQuerying/formal-lang-course`
Имя пользователя: `WoWaster`
Исходный код решений находится в приватном репозитории
Pull Request в проект CFPQ_Data:
`https://github.com/FormalLanguageConstrainedPathQuerying/CFPQ_Data/pull/92`

Ход работы студента



Планы на будущее

Для исправления выявленных недостатков в будущем планируется:

1. добавить ручные тесты
2. добавить тесты на наличие вспомогательных функций
3. добавить ограничение на скорость исполнения алгоритмов