

# Знакомство с интерпретатором LuNAScript

Докладчик: Морозов Р. С.

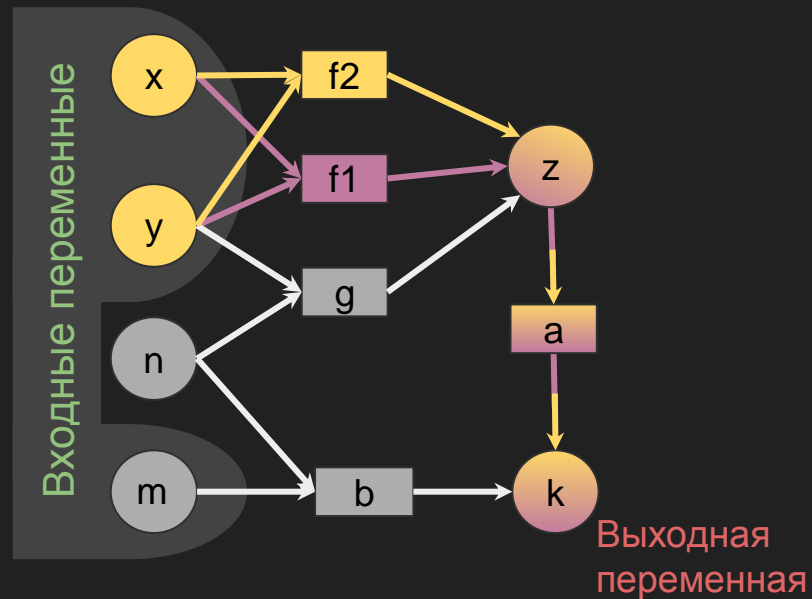
Руководитель: Перепёлкин В. А.

Всероссийская летняя XLII молодежная Школа-конференция  
по параллельному программированию

Новосибирск, 2024

# Проблема

- Известная проблема — автоматическое конструирование параллельных программ
- Вычислительные модели (ВМ) позволяют частично описать предметную область в виде простых отношений



Представление ВМ в  
виде графа

# Подход

- LuNAScript использует текстовый формат описания, что позволяет задавать более сложные отношения с зависимостью от значений переменных

```
x[N] := 10;  
N := 5;  
y := N + x[N];  
demand y;
```

- Планирование не всегда удобно — необходим интерпретатор, вычисляющий значения для определения возможных путей решения

# Цель

Улучшить существующую реализацию интерпретатора LuNAScript.

# Задачи

1. Изучить реализацию и алгоритмы.
2. Проанализировать возможные способы улучшения.
3. Расширить интерпретатор новой функциональностью.

# Обзор интерпретатора

- Интерпретатор LuNAScript (LSI) написан на JavaScript
- Для получения AST используется GNU Bison

## Алгоритм вывода

- Интерпретация заключается в разрешении всех demand-запросов
- Выполняется обход в ширину с вычислением всех потенциально полезных выражений

# Недостатки

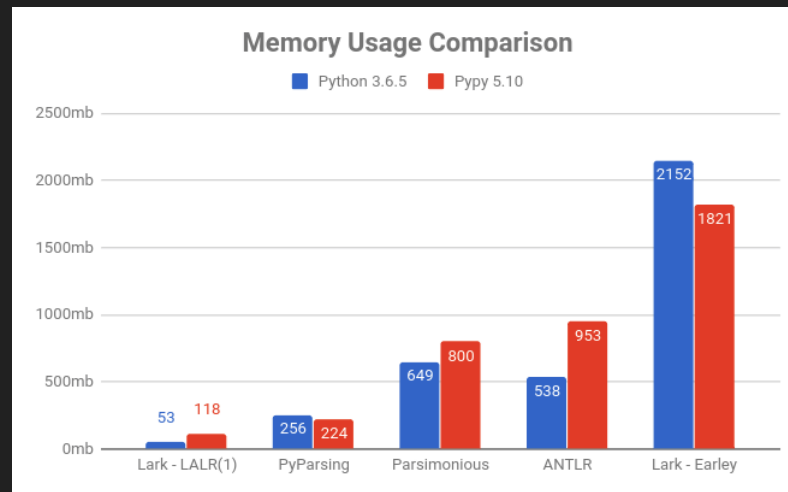
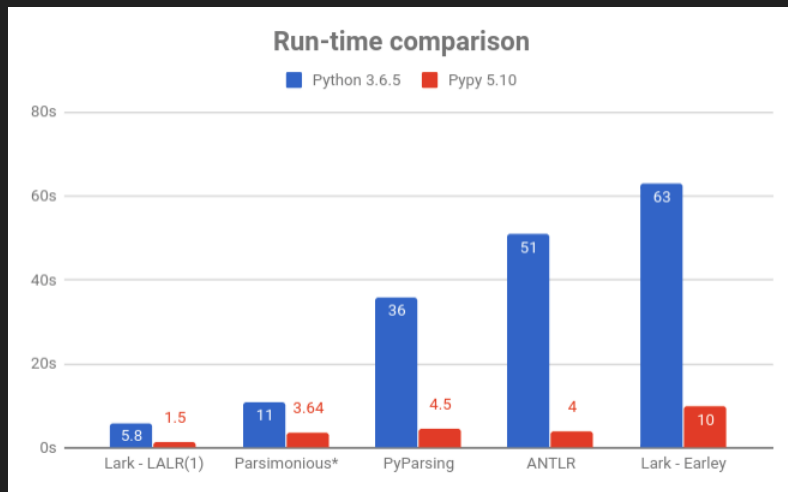
- Синтаксический анализатор GNU Bison доступен только для Linux
- Используется JSON-представление программ
- Полный обход в ширину может выполнять лишнюю работу

# Решение

- Использовать кроссплатформенное решение для парсинга LuNAScript
- Переписать код на Python с использованием подходов ООП.

# Синтаксический анализ

- Библиотека Lark (Python)



# Синтаксический анализ

- Библиотека Lark (Python)

Преимущества:

- Простота использования
- Доступно много алгоритмов парсинга
- Регулярные выражения для базовых токенов (строки, числа, комментарии)
- Настраиваемая обработка синтаксических ошибок

Недостатки:

- Поверхностная документация

# Синтаксический анализ

Результаты:

- Описана иерархия классов AST
- Переписана грамматика языка LuNAScript
- Реализованы методы конвертирования AST в JSON
- Проведено тестирование и выполнено сравнение JSON-представлений с исходной версией

# Расширение для VS Code

Разработано расширение для .Is файлов:

- Подсветка ключевых слов и константных значений
- Автоматическое дополнение парных символов

```
1  #!/usr/bin/env lsi
2  import print as p;
3
4  x[0] := 0;
5
6  forall i
7  |   x[i] := x[i-1] + 1;
8
9  forall i {
10 |   y[i] := x[i]*x[i];
11 }
12
13 N := min(i>=0)(x[i+1]>10);
14 //N := min(i>=0)(y[i+1]>100);
15
16 p(N) as pn
17
18 demand y[N];
19 demand N;
20 demand pn;
```

# Реализация интерпретатора

- Реализована main функция для запуска интерпретатора из командной строки с выбором режима работы.
- Интерпретатор выводит вычисления, использующие операции присваивания (:=) и импортирования готовых функций.
- Не реализована массовость (оператор минимизации и forall)

# Результаты

- Реализован синтаксический разбор LuNAScript через библиотеку Lart.
- Разработано VSCode расширение для разработки на LuNAScript
- Частично реализован интерпретатор на Python
  
- Приобретение понимания работы интерпретатора и недостатков подхода