Разработка компонентов графоориентированного программного каркаса для реализации сложных вычислительных методов

Тришин Илья Вадимович, группа РК6-81Б

МГТУ им. Н.Э. Баумана

Россия, Москва, – 20 июня 2022

Введение

↳ Программное обеспечение для научных вычислений

Введение

↳ Научные системы организации рабочего процесса

Введение

↳ Современные подходы к организации вычислений

Цель

Разработать программные средства для создания и интерпретации графовых описаний вычислительных методов в программном каркасе comsdk.

Задачи

1. Провести сравнение объекта разработки с некоторым аналогичным.
2. Сформировать требования к алгоритму, выполняющему этапы алгоритма по его описанию, составленному по методологии GBSE.
3. Спроектировать структуры данных для описания и представления описаний алгоритмов и их элементов в программном каркасе comsdk.
4. Разработать алгоритм обхода графовых моделей с использованием спроектированных структур данных.
5. Представить интерфейсы или реализации разработанных алгоритмов и структур данных на языке C++.

Описание алгоритма – ориентированный граф. В его вершинах – процессы обработки данных, рёбра – пути данных между процессами.

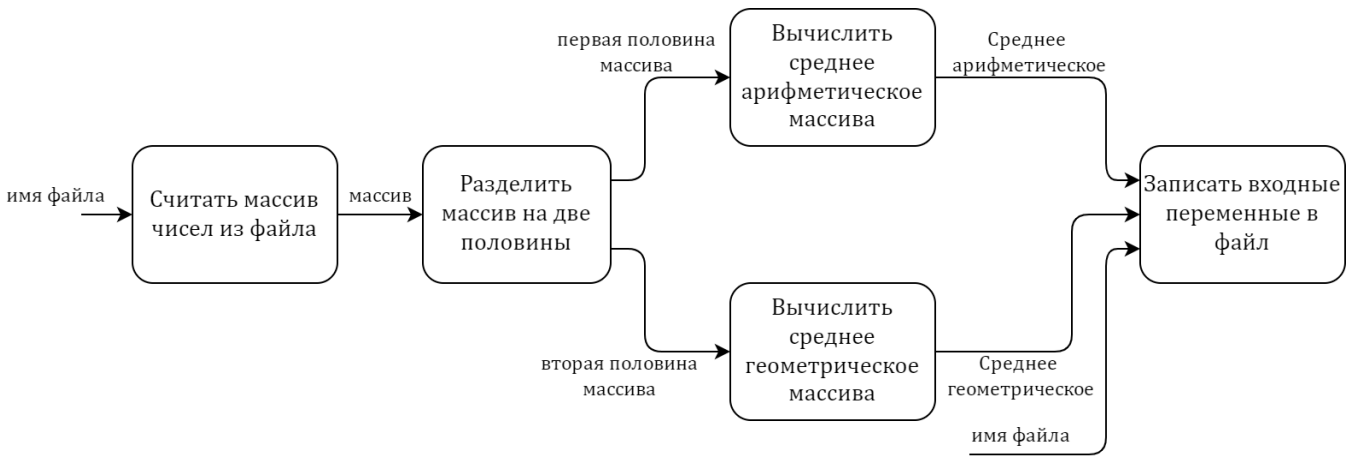


Рис. 1: Пример диаграммы потоков данных, описывающей вычисление среднего арифметического и среднего геометрического двух половин массива целых чисел с последующей записью результатов в файл

Аналитический обзор

↳ pSeven – реализация диаграмм потоков данных

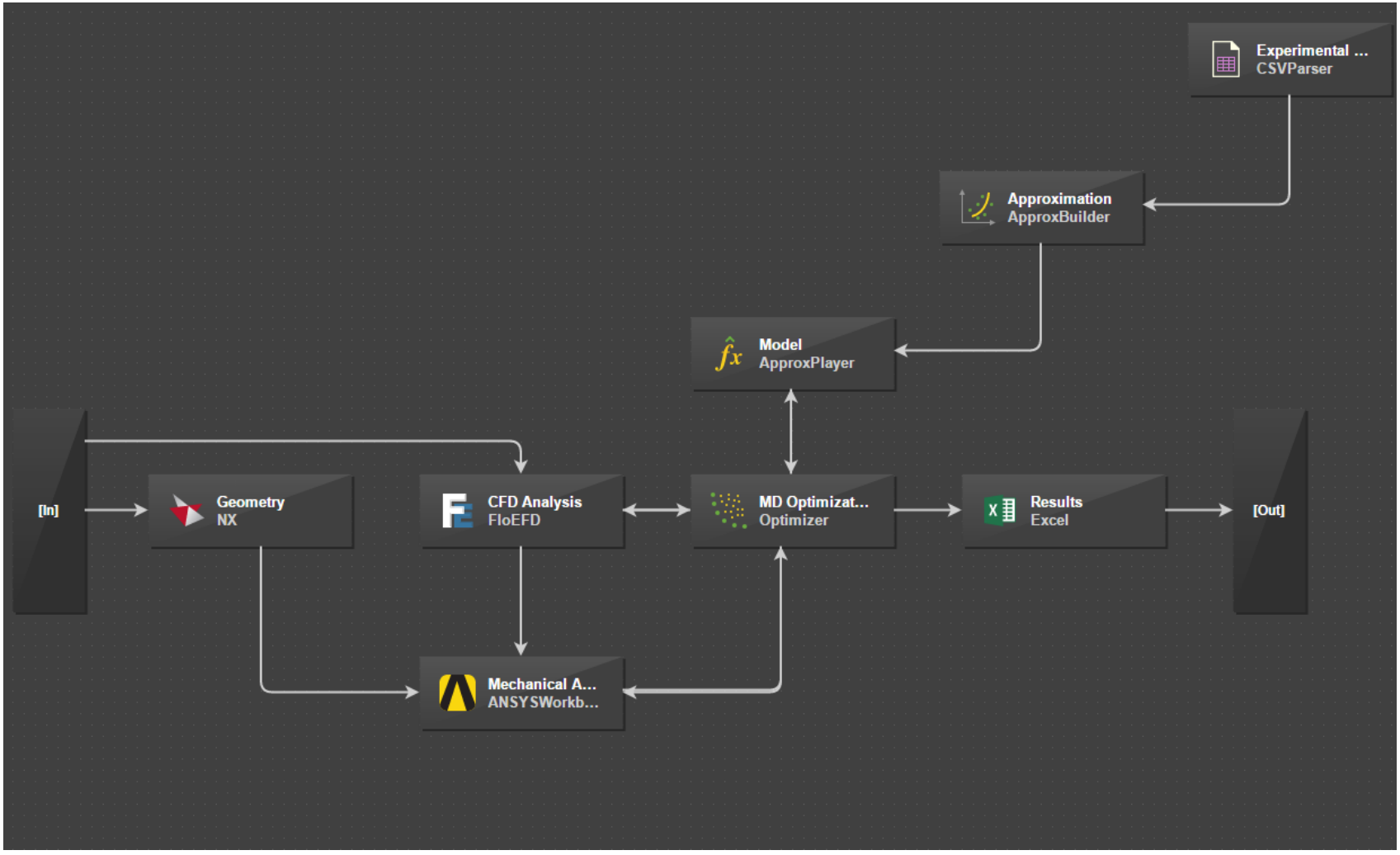
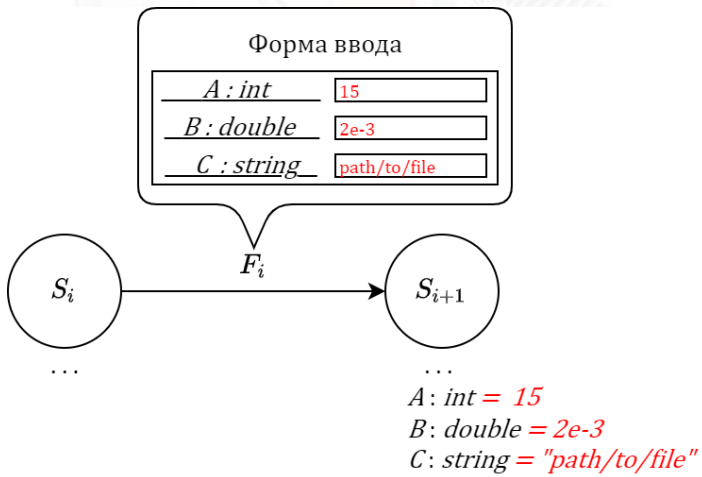


Рис. 2: Графический пользовательский интерфейс pSeven

Основные достоинства comsdk по сравнению с pSeven:

- Нет необходимости указывать входные и выходные данные при описании алгоритма.
- По умолчанию поддерживаются алгоритмы, подразумевающие взаимодействие с пользователем



- Результат применения – компилируемая программа с возможностью запуска на различных платформах.

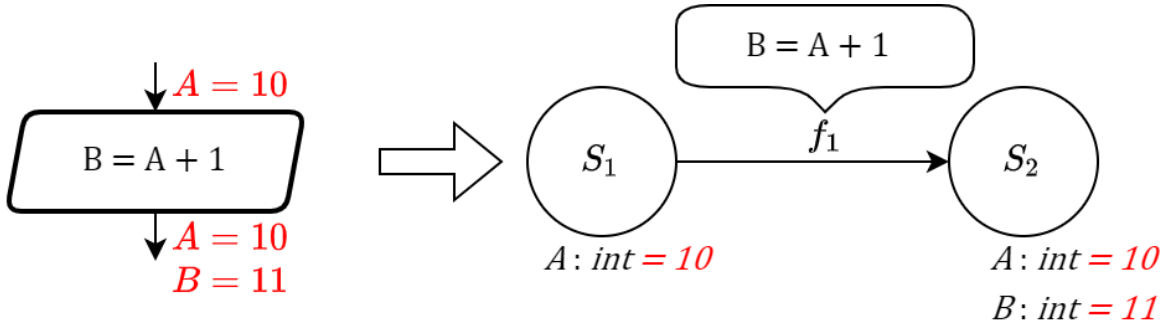
Основные недостатки comsdk по сравнению с pSeven:

- Отсутствие поддержки матричных типов данных.
- Отсутствие средств визуализации результатов расчётов.
- Отсутствие возможности использования при расчётах распределённых вычислительных систем;

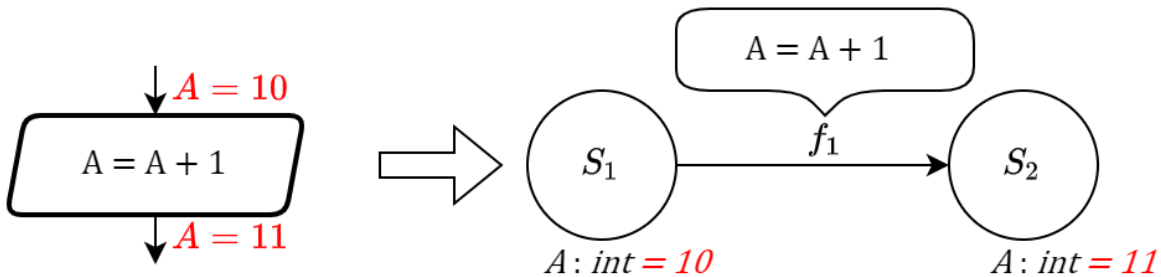
$$\underbrace{S = \left\langle \begin{array}{l} A : int \\ B : double \\ C : string \end{array} \right\rangle}_{\text{Состояние данных}} \qquad \underbrace{D \multimap S = \left\langle \begin{array}{l} A : int = 10 \\ B : double = 2e\text{-}3 \\ C : string = "path/to/file" \end{array} \right\rangle}_{\text{Данные } D \text{ в состоянии } S}$$

Состояние данных определяет, какие переменные какого типа должны быть определены на данном этапе алгоритма.
Данные алгоритма модифицируются по ходу его выполнения.

Постановка задачи
↳ Функции-обработчики



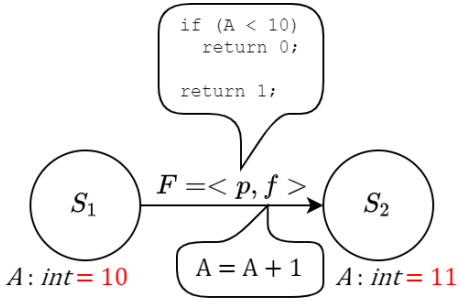
Вариант 1. Функция-обработчик модифицирует состояние данных



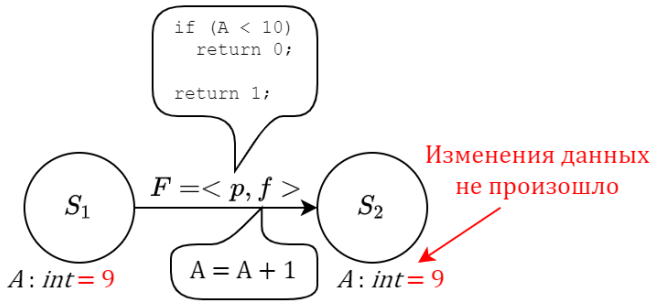
Вариант 2. Функция-обработчик модифицирует только сами данные

Функции-обработчики отвечают за обработку данных и их перевод из одного состояния в другое.

Постановка задачи
↳ Функции-предикаты и функции перехода



Вариант 1. Функция-предикат возвращает 1



Вариант 2. Функция-предикат возвращает 0

Рис. 3: Принцип работы функции-предиката

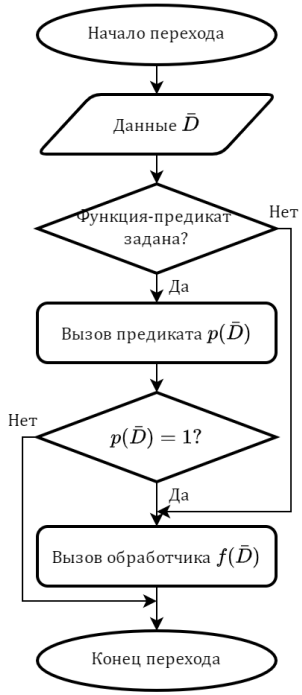


Рис. 4: Блок-схема логики функции перехода

Функции-предикаты отвечают за предварительную проверку данных перед их обработкой.

Функция перехода – составная функция $F =< p, f >$, содержащая в себе функцию-предикат p и функцию-обработчик f.

Функции-селекторы отвечают за проверку условий при ветвлении алгоритма.

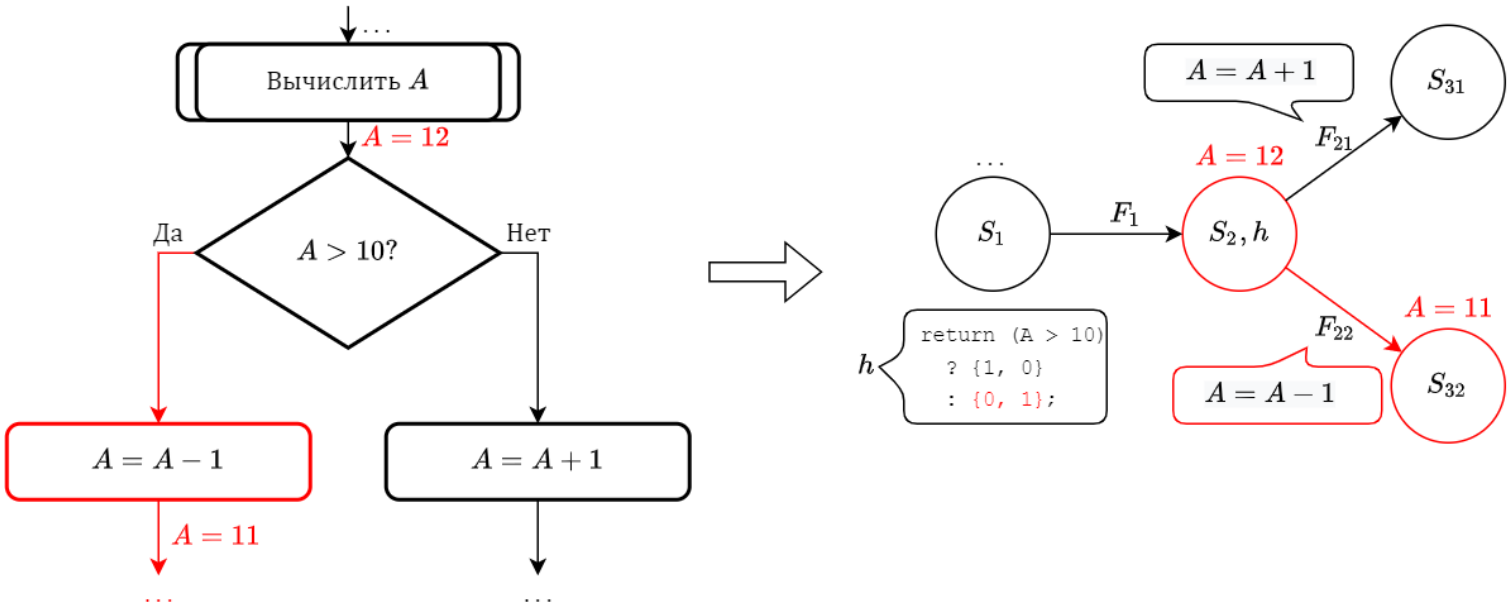


Рис. 5: Принцип работы функций-селекторов. h – функция селектор. Красным показана ветвь алгоритма, которая будет выполнена.

Постановка задачи
↳ Графовая модель

Графовая модель сложного вычислительного метода описывает его логику в виде ориентированного графа, где узлам ставятся в соответствие состояния данных, а рёбрам – функции перехода.

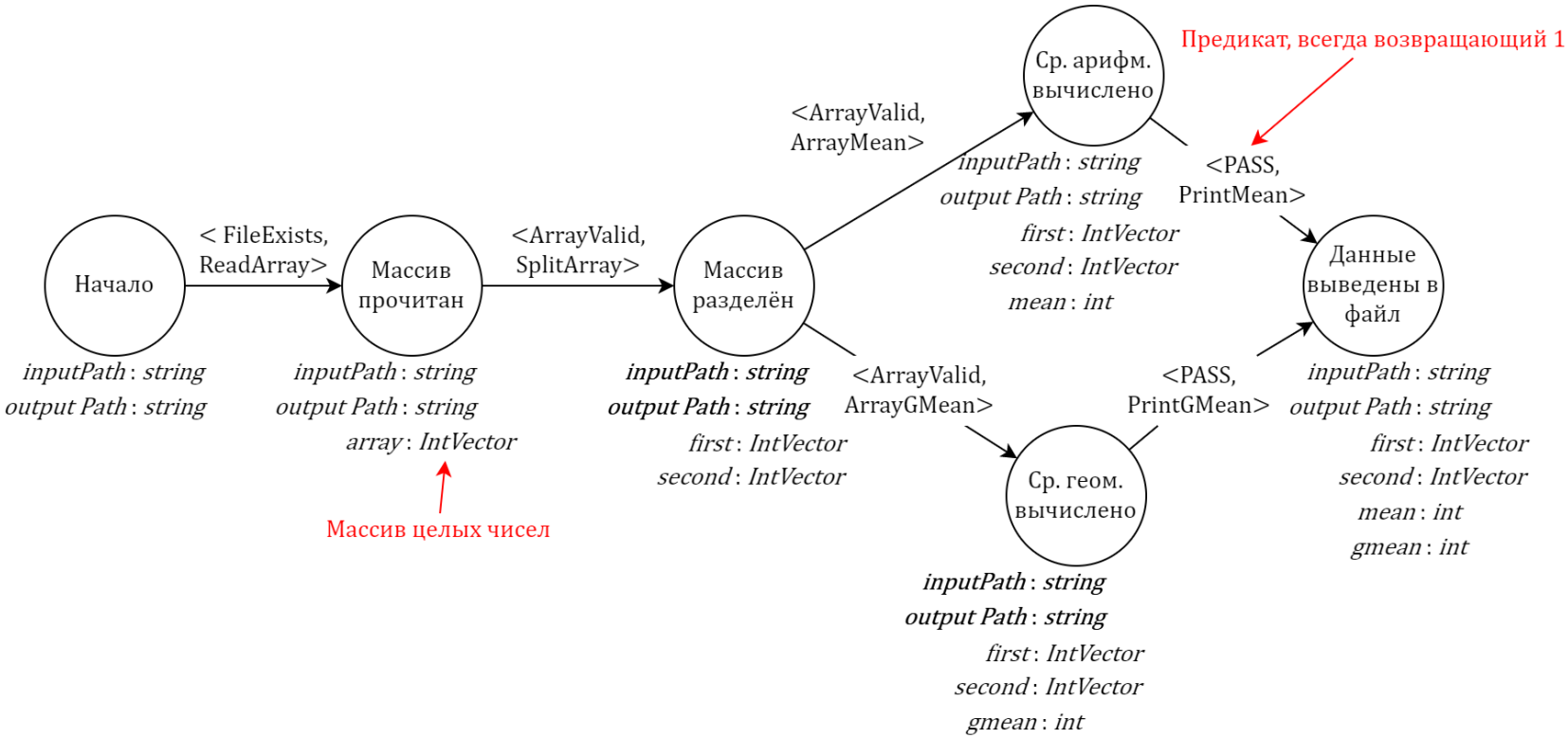
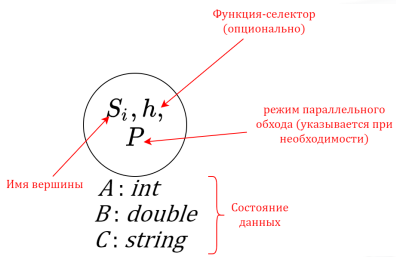


Рис. 6: Пример графовой модели, описывающей вычисление среднего арифметического и среднего геометрического двух половин массива целых чисел с последующей записью результатов в файл

Постановка задачи
↳ Графовая модель



Атрибуты вершины:

1. имя;
2. состояние данных;
3. селектор;
4. режим параллельного обхода исходяших ветвей

Программная реализация
↳ Функциональные структуры данных

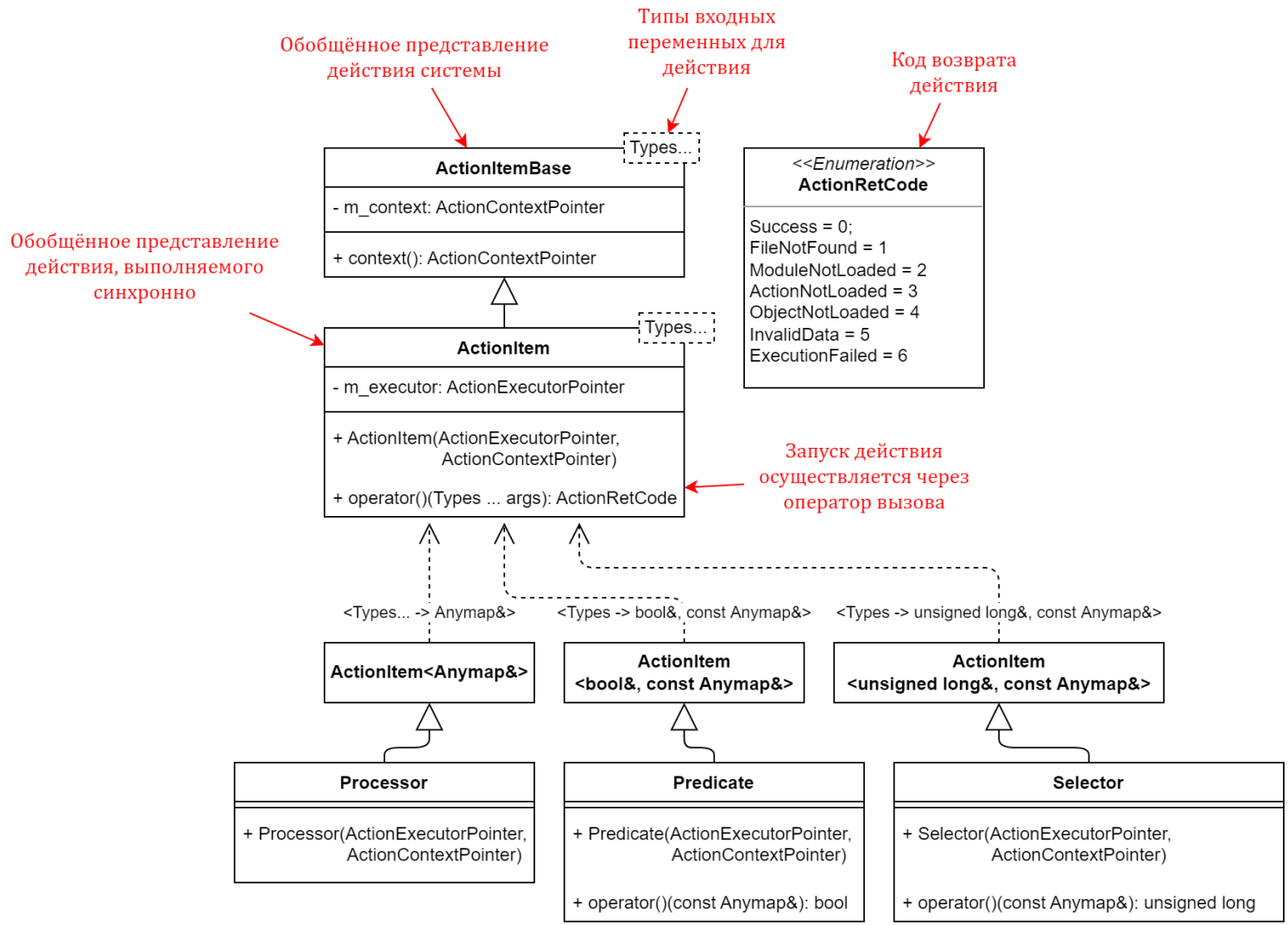


Рис. 7: UML-диаграмма разработанных структур данных, отвечающих за представление функций-предикатов, обработчиков и селекторов

Программная реализация
↳ Информационные структуры данных

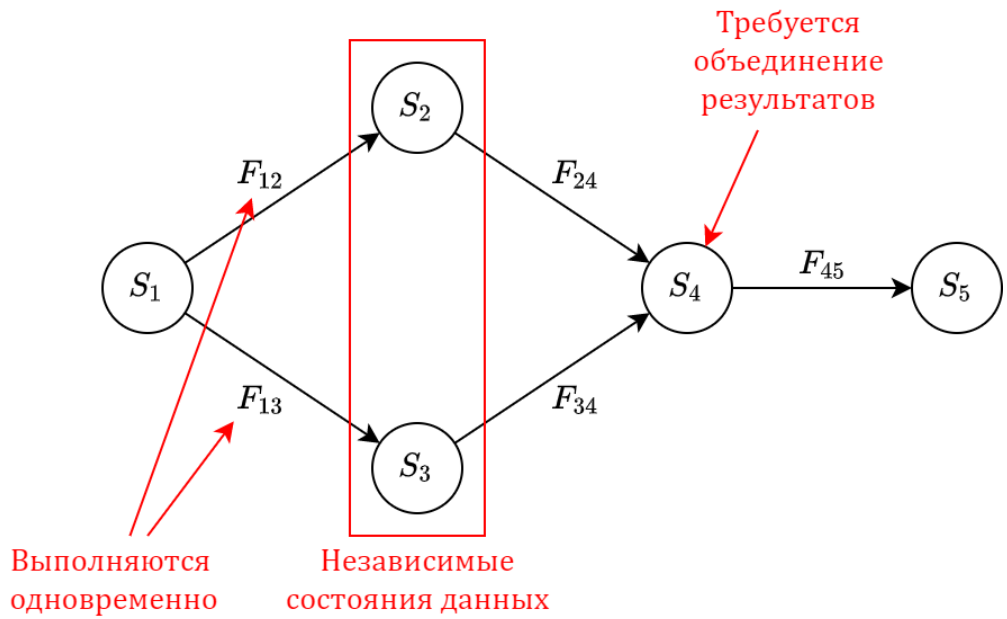


Рис. 8: Пример графовой модели, подразумевающей паралельный обход ветвей

Режим параллельного исполнения в вершине S_1 определяет, какие ресурсы будут задействованы для выполнения ветвей $S_1 \to S_2 \to S_4$ и $S_1 \to S_3 \to S_4$

Программная реализация

↳ Управляющие структуры данных

В результате выполнения работы:

1) расширены функциональные возможности библиотеки comsdk
2) создана новая архитектура классов, позволяющая упростить процесс формирования программного представления графовых моделей;
3) разработаны структуры данных для программного представления графовых моделей алгоритмов и их элементов;
4) был разработан алгоритм, осуществляющий выполнение этапов алгоритма в соответствии с его графовой моделью;

Перспективы развития программного каркаса comsdk включают в себя:

- реализации алгоритма обхода графовой модели с задействованием различных вычислительных ресурсов
- интеграцию средства генерации форм ввода[1];
- разработку средства визуализации обрабатываемых данных;
- разработку средства автоматической документации реализуемых алгоритмов;

[1] Соколов А.П Першин А.Ю. Программный инструментарий для создания подсистем ввода данных при разработке систем инженерного анализа // Программная инженерия. 2017. Т. 8, № 12. С. 543–555.

Спасибо за внимание!