Рословцев Владимир Владимирович
ст. преподаватель НИЯУ МИФИ,
Россия, г. Москва

Сердюков Кирилл Алексеевич
студент НИЯУ МИФИ,
Россия, г. Москва

Корнишев Даниил Олегович
студент НИЯУ МИФИ,
Россия, г. Москва

Самойленко Дарья Олеговна
студент НИЯУ МИФИ,
Россия, г. Москва

Спасский Владислав Михайлович
студент НИЯУ МИФИ,
Россия, г. Москва

Андреев Тимофей Павлович
студент НИЯУ МИФИ,
Россия, г. Москва

РАЗРАБОТКА СИСТЕМЫ ИЗУЧЕНИЯ И ИСПОЛНЕНИЯ П-ИСЧИСЛЕНИЯ

В настоящее время наблюдается развитие большого числа workflow-систем: от специализированных решений для биоинформатики и экологии (Taverna, Pegasus) до широко распространенных платформ построения бизнес-процессов (Amazon Step Functions, Apache Airflow, Microsoft Azure Logic Apps). Многообразие существующих подходов подтверждает актуальность разработки workflow-систем, позволяющих гибко и формально описывать бизнес процессы. В работе за основу взято чистое π-исчисление, разработанное Р. Милнером, которое расширено дополнительными конструкциями: условный оператор, интегрированные λ-термы. Предложенные расширения позволяют сохранить формализм описания взаимодействий и одновременно обеспечить возможность проектирования более сложных и детерминированных конструкций.

Ключевые слова: алгебра процессов, π-исчисление, параллельные вычисления, асинхронные вычисления, аппликативные системы, визуализация вычислений.

Annotation

Currently, a large number of workflow systems are being developed: from specialized solutions for bioinformatics and ecology (Taverna, Pegasus) to widespread platforms for building business processes (Amazon Step Functions, Apache Airflow, Microsoft Azure Logic Apps). The variety of existing approaches confirms the relevance of developing workflow systems that allow flexible and formal description of business processes. The paper is based on the pure π-calculus developed by R. Milner, which is expanded by additional constructions: conditional operator, integrated λ-terms. The proposed extensions make it possible to preserve the formalism of the description of interactions and at the same time provide the possibility of designing more complex and deterministic structures.

Keywords: algebra of processes, pi calculus, parallel computing, asynchronous computing, applicative systems, visualization of computing.

Введение

Современные формальные модели вычислений, основанные на аппликативных парадигмах, таких как λ-исчисление, и процессных моделях, таких как π-исчисление, занимают ключевое место в теории программирования и лежат в основе функциональных, параллельных и распределённых вычислительных систем. Данные модели позволяют строго описывать процессы вычисления, взаимодействие компонентов и передачу данных, однако их абстрактный характер существенно усложняет практическое изучение и применение, особенно в образовательном процессе. Одной из актуальных проблем является отсутствие наглядных и интерактивных инструментов, способных доступным образом отражать семантику аппликативных вычислений и процессных взаимодействий.

Рассмотренная система PiVizTool [1] позволяет построить граф выражения, вершинами которого будут являться возможные состояния в процессе вычисления, а дугами графа – коммуникации процессов.

The Mobility Workbench (MWB) [2] является инструментом анализа процессов π-исчисления, ориентированным на формальную верификацию.

Существующие средства визуализации не позволяют в полной мере продемонстрировать динамику редукций, асинхронное взаимодействие процессов и недетерминированность параллельных вычислений.

Целью данной работы является создание системы для построения, анализа и визуализации комбинационных схем в рамках аппликативной модели вычислений с использованием π-исчисления, системы, поддерживающей изучение π-исчисления, в том числе в автономном режиме. В рамках разрабатываемой системы предполагается реализация формального представления процессов, поддержки расширенного синтаксиса π-исчисления, пошагового выполнения выражений, анимации редукций и механизмов анализа параллельных и асинхронных взаимодействий.

Научная новизна работы заключается в разработке интерактивной среды, объединяющей аппликативную модель λ-исчисления и процессную модель π-исчисления. В рамках системы предложен механизм трансляции λ-выражений в процессы π-исчисления, позволяющий сопоставлять шаги β-редукции с коммуникационными редукциями π-процессов. Это обеспечивает возможность пошаговой визуализации выполнения выражения одновременно в двух вычислительных моделях.

В разделе рассмотрение теоретических основ λ- и π-исчисления и описание трансляции λ-термов в π-выражения дается краткая теоретическая справка по моделям λ- и π-исчисления, приводятся примеры синтаксиса, проводится рассмотрение трансляции λ-термов в π-выражения.

В разделе анализ существующих программных инструментов, поддерживающих моделирование процессов средствами π-исчисления

проводится краткий обзор систем MWB и PiVizTool, сравнительный анализ между существующими системами и разработанной SLPC.

В разделе архитектура системы изучения и исполнения процессов π-исчисления дается краткое описание архитектуры SLPC, приводится пример описания процесса сдачи научной работы научному руководителю в синтаксисе SPLC и исходного π-исчисления и пример работы системы над выражением.

Рассмотрение теоретических основ λ- и π-исчисления и описание трансляции λ-термов в π-выражения

Аппликативные методы вычислений в распределенной среде

Аппликативные системы позволяют идентифицировать части процессов, которые можно выполнить параллельно

λ-исчисление представляет собой формальную систему, разработанную Алонзо Чёрчем в 1930-х годах, которая легла в основу функционального программирования и теории вычислений [3]. В контексте описания вычислительных процессов λ-исчисление предлагает такие ключевые возможности для формализации, как чистота математической модели, возможность строгих доказательств свойств программ, естественное описание высокоуровневых абстракций [4].

В рамках λ-исчисления используются конструкции абстракции ($\lambda x.M$) и аппликации (MN), а вычисление формализуется посредством β-редукции.

Пример выражения [6]:

$$Y = \lambda f.(\lambda x.f(x))(\lambda x.f(x))$$

Использование π-исчисления для явного отражения обмена информацией между независимыми агентами

π-исчисление, разработанное Робином Милнером, представляет собой процессное исчисление, специально предназначенное для описания параллельных и распределенных систем [5, 6].

В рамках π-исчисления процессы определяются как первичные сущности, именованные каналы используются для взаимодействия, новые каналы

создаются динамически $(vx)P$, а параллельная композиция описывается как $P|Q$ [6, 7].

Из особенностей моделирования можно выделить явное представление коммуникации, возможность передачи имен каналов как данных, представления мобильных систем с изменяющейся топологией [7].

π-исчисление позволяет гибко описывать изменяющуюся структуру взаимодействий [8] и параллелизм естественным образом, поддерживать мобильность, то есть изменять структуры взаимодействий во время выполнения. Эти свойства делают π-исчисление удобной моделью для описания взаимодействующих процессов и используются как теоретическая основа SLPC.

К основным конструкциям относятся:

- Оператор репликации $!\,(P)$ для описания бесконечных процессов
- Коммуникация формализуется следующим редукционным правилом [6]:

$$x![y].P \mid x?(z).Q \;\rightarrow\; P \mid Q[y/z]$$

Формализованный синтаксис, используемый в SLPC, задаётся следующим образом:

$$P::=0 \mid x![y].P \mid x?(z).P \mid P \mid Q \mid (vx)P \mid !P \;,$$

где 0 – завершенный процесс.

На основе анализа применимости возможностей аппликативных и процессных вычислений к созданию workflow-системы были определены требования к разрабатываемой платформе: необходимость поддержки расширенного синтаксиса π-исчисления, включая условные выражения, возможность вычисления выражений, построенных с одновременным использованием π и λ–исчисления, реализация пошагового выполнения редукций, режима обучения и разработка наглядного способа отображения асинхронного взаимодействия процессов через каналы связи.

Таким образом, λ-исчисление обеспечивает формализацию вычислений над выражениями, а π-исчисление — формализацию взаимодействия процессов.

Объединение этих моделей требует согласованной операционной семантики, обеспечивающей корректное взаимодействие редукций λ-термов и коммуникационных шагов π-процессов; соответствующий механизм был реализован в разработанной системе.

Трансляции λ-термов в π-выражения

В работе [9] предлагается подход к интерпретации аппликативной модели λ-исчисления в терминах процессной модели π-исчисления. Рассуждая в λ-терминах, есть три базовых конструкта: переменная, абстракция и аппликация.

Введём отображение (трансляцию) λ-термов в процессы π-исчисления:

$$T(M, k),$$

где: M — λ-выражение;

k — канал, в который передаётся результат вычисления выражения.

Для переменной трансляция:

$$T(x, k) = x![k],$$

что интерпретируется как: «возьми значение переменной x и отправь в канал результата k».

Для абстракции трансляция:

$$T(\lambda x.M, k) = (\nu f)(k![f]|!f?(x, r).T(M, r)),$$

что интерпретируется как: «создать новый канал f, представляющий функцию. Канал передаётся во внешнюю среду через k. Далее процесс f выступает в роли сервера функции и принимает пару значений: аргумент x и канал результата r, после чего выполняется трансляция тела функции.».

Для аппликации трансляция:

$$T(MN, k) = (\nu f, a)(T(M, f)|T(N, a)|f![a, k]),$$

что интерпретируется как: «создать два новых канала f и a. Канал f используется для получения результата вычисления функции M, а канал a — для результата вычисления аргумента N. После вычисления аргумент a и канал результата k передаются функции через канал f.»

Пример: $(\lambda x.x)a$.

$$T((\lambda x.x)a, k) = (\nu f, b)(T(\lambda x.x, f)|T(a, b)|f![b, k])$$

Подставим правила трансляции:

$$(vf, b)((vs)(!\, s?\, (x, r).\, x!\, [r])|a!\, [b]|f!\, [b, k])$$

Происходит взаимодействие процессов по каналу f:

$$f!\, [s]|f!\, [b, k] \;=> s!\, [b, k]$$

После редукции получаем:

$$(vf, b)((vs)(!\, s?\, (x, r).\, x!\, [r] \mid s!\, [b, k]) \mid a!\, [b])$$

Происходит взаимодействие по процессу s:

$$s!\, [b, k] \mid !\, s?\, (x, r).\, x!\, [r]$$

Что дают подстановки:

$$x := b, r := k$$

и выполняется тело функции:

$$b!\, [k]$$

Получаем процесс:

$$(vb)(b!\, [k] \mid a!\, [b])$$

который представляет последовательную передачу значения аргумента через промежуточный канал b в канал результата k. Такая передача соответствует результату β-редукции выражения

$$(\lambda x.\, x)a \to a$$

Анализ существующих программных инструментов, поддерживающих моделирование процессов средствами π-исчисления

В процессе формирования требований к System for Learning Pi-Calculus (SLPC), были проанализированы существующие инструменты поддержки π-исчисления: The PiVizTool и The Mobility Work Bench.

The Mobility Workbench

MWB представляет собой инструмент анализа процессов π-исчисления, ориентированный на формальную верификацию. Согласно [2], базовой функциональностью системы является проверка открытой бисимуляции (open bisimilarity).

Дополнительно MWB поддерживает:

- Симуляцию вычислений,

- Проверку эквивалентности процессов,
- Обнаружение взаимных блокировок (deadlock),
- Поиск состояний.

The PiVizTool

PiVizTool является визуализатором π-процессов. Система преобразует текстовое описание процесса в граф переходов, где вершины соответствуют состояниям, а рёбра — действиям.

Инструмент ориентирован на наглядное представление динамики процесса, однако не предоставляет механизмов строгой формальной проверки эквивалентности или свойств системы.

Ключевые отличия SLPC от рассмотренных аналогов, позволяющие удовлетворить требования в анализе комбинационных схем расширенного π-исчисления

Разработанная система SLPC отличается от существующих решений по следующим направлениям:

Режим обучения. Рассмотренные выше инструменты не обладают данной функциональностью: SLPC позволяет пошагово вводить выражение и получать от системы текстовый ответ о корректности введенного выражения, статусе процесса, значениях переменных.

Работа с π- и λ-выражениями. SLPC позволяет вычислять выражения, содержащие λ-термы. Для этого используется другая разработка – интерпретатор КАМ-машины.

Расширенный синтаксис π-исчисления. SLPC позволяет проводить вычисления над выражениями, состоящими из условных конструкций if, then, else.

Встроенный банк выражений. SLPC обладает внутренним набором выражений для использования в образовательных целях.

Рассмотрим инструменты в сравнении друг с другом:

Таблица 1.

Сравнение SLPC с аналогами

Инструмент	**PiVizTool**	**MWB**	**SLPC (System for Learning Pi-Calculus)**
Основная цель	Визуализация поведения процессов	Формальная верификация	Обучение и пошаговая семантика
Тип интерфейса	GUI	CLI	GUI
Проверка эквивалентности	Нет	Да, open bisimilarity	Нет, ориентация на интерпретацию
Поддержка λ-исчисления	Нет	Нет	Да
Расширенный синтаксис	Ограниченный	Близок к стандартному π-исчислению	Да, условные конструкции
Методологическая ориентация	Иллюстративная	Доказательная	Обучающая и интерпретационная

Таким образом, MWB ориентирован на решение задач эквивалентности процессов и формальной верификации протоколов, что предполагает анализ полного пространства состояний. PiVizTool ориентирован на графическое отображение поведения системы. SLPC занимает иную позицию: не осуществляет автоматическую проверку эквивалентности и не строит глобальное пространство состояний, а реализует локальную операционную семантику с возможностью пошагового анализа одновременно в двух моделях – это достижимо посредством трансляции λ-термов в процессы π-исчисления.

Архитектура системы изучения и исполнения процессов π-исчисления

Проектирование SLPC основывается на принципах модульности, расширяемости и разделения ответственности между компонентами.

Архитектура SLPC предусматривает поддержку двух режимов работы: автоматического и обучающего.

Архитектура системы исполнения процессов π-исчисления: ядро исполнения процессов, обеспечивающее формальное выполнение выражений π-исчисления, управление параллельными процессами, каналами и окружением выполнения; серверный интерфейс взаимодействия, предоставляющий доступ к функциональности системы через API и обеспечивающий управление жизненным циклом сессий исполнения; учебный модуль, реализующий пошаговое выполнение, контроль корректности действий пользователя и формирование обратной связи.

Ядро исполнения является центральным компонентом системы и отвечает за интерпретацию процессов, координацию параллельного выполнения, управление каналами и сообщениями, сохранение состояния вычислительной среды между шагами выполнения [10].

Параллельные процессы рассматриваются как независимые вычислительные сущности, эволюция которых осуществляется в рамках общего шага выполнения. Такое решение обеспечивает корректное моделирование конкурентных вычислений и фиксацию промежуточных состояний системы.

Архитектура ядра предусматривает сохранение полного состояния выполнения, включая текущее выражение процессов, значения переменных и состояние каналов. Это является необходимым условием для реализации пошагового исполнения и анализа корректности вычислений в учебном режиме.

Учебная направленность системы проявляется в поддержке пошагового режима, при котором каждый шаг исполнения фиксируется и может быть проанализирован пользователем. Это позволяет отчетливо демонстрировать влияние параллелизма, асинхронных взаимодействий и условных конструкций на состояние системы.

Диаграмма классов отражает структуру серверного ядра SLPC. Центральным элементом архитектуры является класс `EnvironmentManager`,

отвечающий за управление окружением выполнения, жизненным циклом процессов и каналов связи.

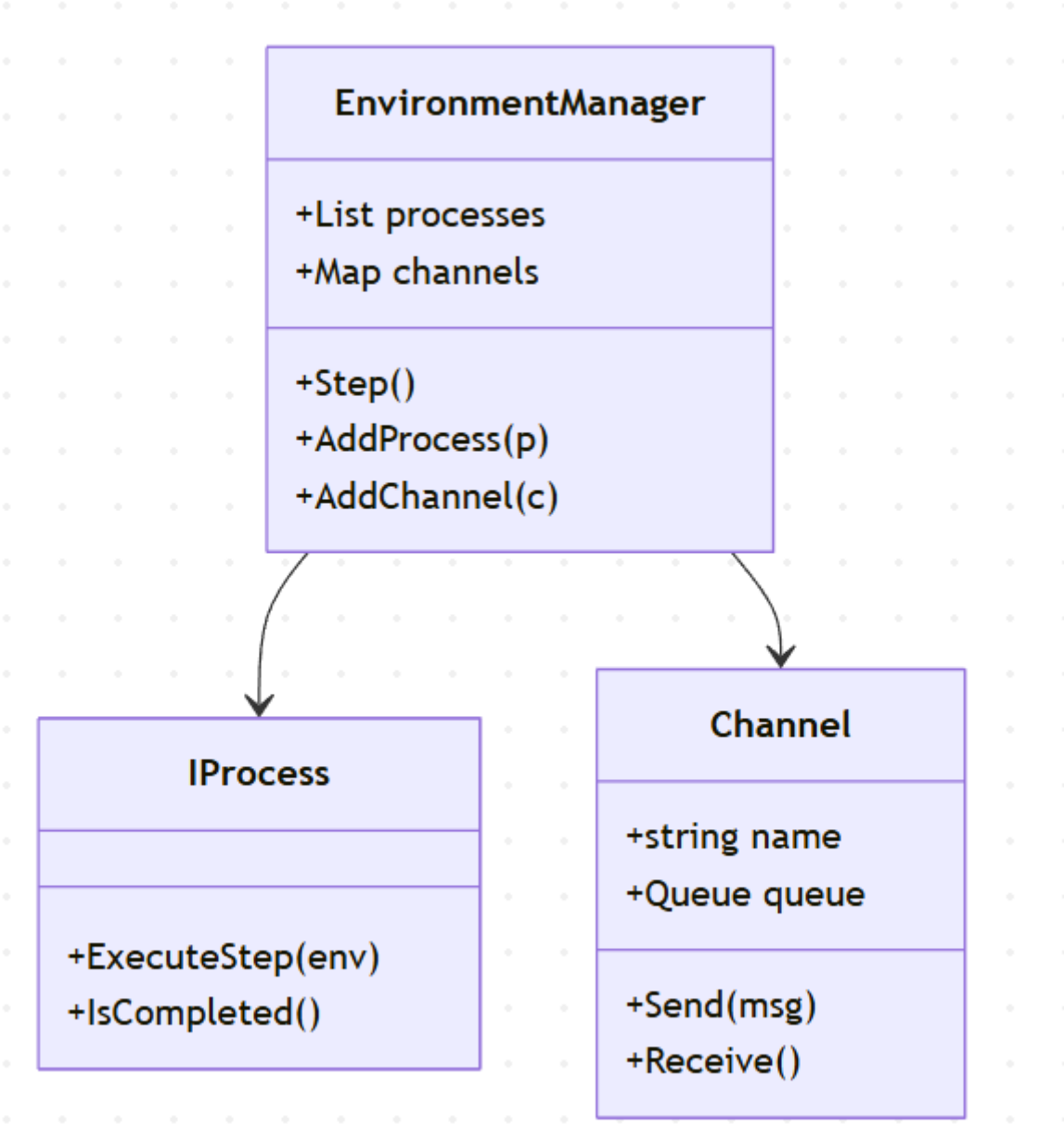

Рисунок 1. Структура серверного ядра SLPC

Все процессы реализуют общий интерфейс `IProcess`, определяющий единый контракт выполнения. Это обеспечивает полиморфизм и расширяемость системы при добавлении новых конструкций языка.

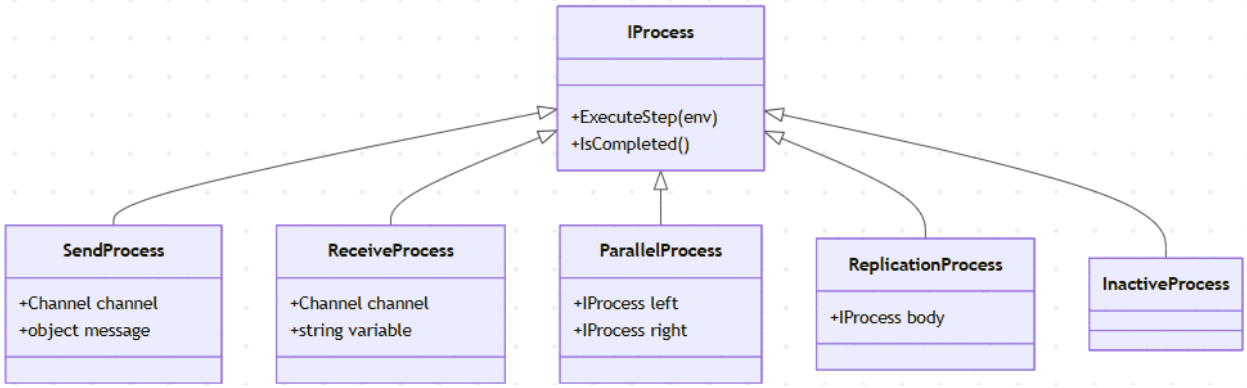

Рисунок 2. Система процессов ядра SLPC

Механизм коммуникации реализован через класс `Channel`, который инкапсулирует очередь сообщений и стратегию взаимодействия.

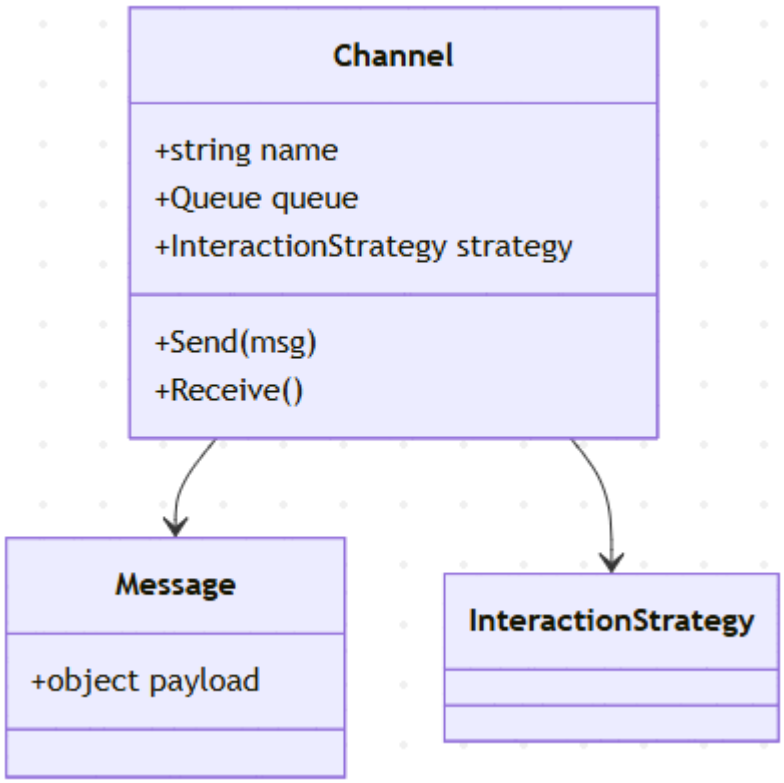

Рисунок 3. Структура канала Channel

Формальное определение конфигурации системы изучения и исполнения пи исчисления вида

$$C = \langle P, \Gamma, \Sigma \rangle ,$$

где: P — множество процессов;

Γ — окружение переменных;

Σ — состояние каналов.

Шаг выполнения определяется отношением перехода:

$$C \to C' ,$$

которое соответствует выполнению одной коммуникации между процессами.

Правило коммуникации:

$$\frac{x\langle v\rangle \in P \quad x(z).Q \in P}{\langle P, \Gamma, \Sigma\rangle \to \langle (P \setminus \{x\langle v\rangle \in P, x(z).Q \in P\}) \cup \{P, Q[v/z]\}, \Gamma, \Sigma\rangle}$$

Данное правило описывает передачу значения v по каналу x. В результате выполнения коммуникации переменная z в процессе Q заменяется на переданное значение v. Таким образом, каждый шаг выполнения системы соответствует одной коммуникационной редукции процессов.

Сравнительное моделирование типового примера с параллельным взаимодействием

Приведем примеры построения выражения в системе и сравним практичность и эффективность расширенного синтаксиса SLPC и стандартного π-исчисления. Синтаксис SLPC:

$$(\nu\ C1, C2, R1, R2, SA)((C1![work].0 \mid C2![work].0 \mid$$
$$R1?(comment1).R2?(comment2).SA![fixed\_work(comment1, comment2)].0) \mid$$
$$C1?(work).if\ is\_correct(work)\ then\ R1![work]\ else\ R1![correct(work)].0 \mid$$
$$C2?(work).if\ is\_correct(work)\ then\ R2![work]\ else\ R2![correct(work)].0 \mid$$
$$SA?(fixed\_work).0)$$

Синтаксис оригинального π-исчисления:

$$(\nu\ C1, C2, R1, R2, SA)((C1![work].0 \mid C2![work].0 \mid$$
$$R1?(comment1).R2?(comment2).SA![fixed\_work(comment1, comment2)].0) \mid$$
$$C1?(work).R1![comment1].0 \mid$$
$$C2?(work).R2![comment2].0 \mid$$
$$SA?(fixed\_work).0)$$

В обоих случаях создали процессы: C1 – для отправки студентом работы рецензенту 1, C2 – для отправки студентом работы рецензенту 2, R1 – для отправки отрецензированной работы студенту рецензентом 1, R2 – для отправки отрецензированной работы студенту рецензентом 2, SA – для отправки исправленной работы научному руководителю.

Одной из ключевых задач практической части работы являлась реализация механизма пошагового исполнения процессов π-исчисления. Данный механизм ориентирован на использование системы в учебном процессе и позволяет пользователю анализировать поведение параллельных и асинхронных вычислений на уровне отдельных шагов.

Пошаговое исполнение реализовано за счёт явного разделения логики вычисления на элементарные переходы состояний. Каждый процесс выполняется не целиком, а поэтапно, с фиксацией текущего состояния, ожидаемых действий (отправка или приём сообщения) и изменений в окружении выполнения. `EnvironmentManager` управляет очередностью шагов и обеспечивает согласованное выполнение параллельных процессов.

Для поддержки самопроверки студентов в систему заложены механизмы:

- фиксации последовательности выполненных шагов;
- сравнения ожидаемого и фактического поведения процесса;
- анализа корректности взаимодействия между процессами и каналами.

Таким образом, студент может пошагово проследить выполнение модели, сравнить полученный результат с эталонным сценарием и выявить логические ошибки в описании процессов. Реализованный подход способствует более глубокому пониманию семантики π-исчисления и принципов асинхронного взаимодействия.

Ниже представлен скриншот, содержащий полный цикл работы с выражением в системе:

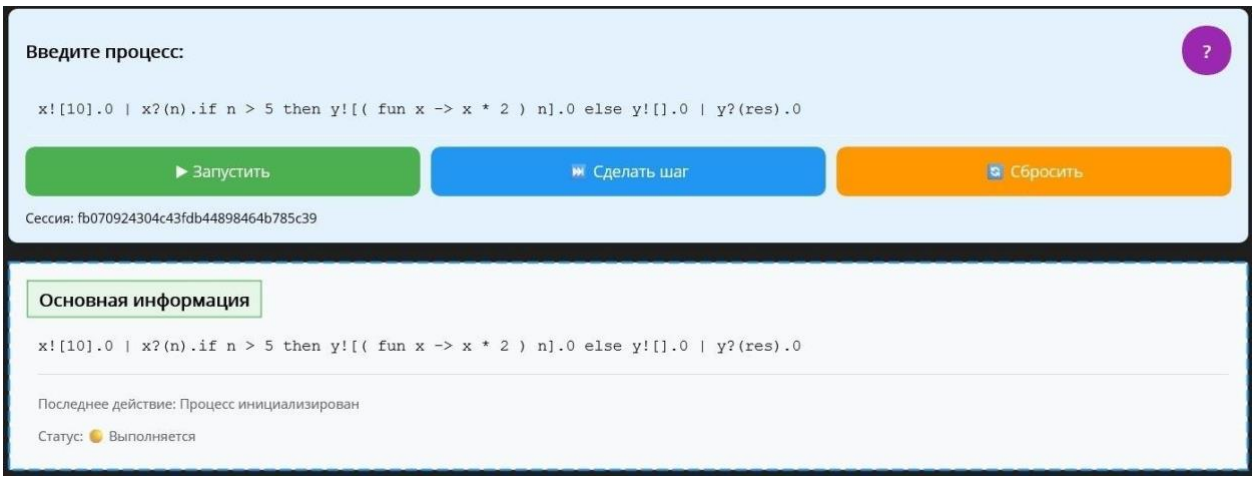

Рисунок 4. Демонстрация интерфейса системы

Ограничения и проблемы реализации системы в основном связаны с решением проектировать приложение как настольное: сложность внесения изменений, сложность поддержки и мониторинга, зависимость от внешней системы, которая занимается обработкой λ-выражений.

Вопрос масштабирования для настольной системы не актуален, т.к. все вычисления происходят непосредственно на устройстве установленной программы.

Поскольку π-исчисление допускает недетерминированный выбор коммуникаций, порядок редукций может различаться. В разработанной системе этот эффект используется для демонстрации различных возможных сценариев выполнения процессов.

Список литературы:

1. PiVizTool — A Tool for Visualizing π-Calculus Processes [Электронный ресурс]. — Режим доступа: https://frapu.de/bpm/piviztool.html (дата обращения: 07.04.2026).
2. Victor B., Moller F. The Mobility Workbench — A Tool for the π-Calculus // Proceedings of the 6th International Conference on Computer Aided Verification (CAV'94). — Berlin: Springer-Verlag, 1994. — Vol. 818. — P. 428–440. — (Lecture Notes in Computer Science). — DOI: 10.1007/3-540-58179-0_72.

3. Church A. The Calculi of Lambda-Conversion. — Princeton: Princeton University Press, 1941. — 221 p.
4. Wadler P. The Essence of Functional Programming // Proceedings of the 19th ACM Symposium on Principles of Programming Languages (POPL). — San Francisco, 1992. — P. 1–14.
5. Milner R. Communicating and Mobile Systems: the π-Calculus. — Cambridge: Cambridge University Press, 1999. — 253 p.
6. Milner R. A Calculus of Communicating Systems. — Berlin: Springer-Verlag, 1980. — 211 p.
7. Milner R. Communication and Concurrency. — Englewood Cliffs: Prentice Hall, 1989. — 221 p.
8. Sangiorgi D. Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms. — PhD Thesis. — University of Edinburgh, 1992. — 186 p.
9. Milner R. Functions as Processes // Mathematical Structures in Computer Science. — 1992. — Vol. 2, No. 2. — P. 119–141.
10. Рословцев В.В., Кононов В.М., Николайчик А.Е., Эргашев О.И. Реализация программного каркаса для реляционно-аппликативной вычислительной среды // Scientific WorldJournal. — 2017. — Вып. 15, т. 1. — Минск: ЧП «Ёлнать». — С. 85–93.

List of literature:

1. PiVizTool — A Tool for Visualizing π-Calculus Processes [Electronic resource]. — Access mode: https://frapu.de/bpm/piviztool.html (date of access: 04/07/2026).
2. Victor B., Moller F. The Mobility Workbench — A Tool for the π-Calculus // Proceedings of the 6th International Conference on Computer Aided Verification (CAV'94). — Berlin: Springer-Verlag, 1994. — Vol. 818. — P. 428–440. — (Lecture Notes in Computer Science). — DOI: 10.1007/3-540-58179-0_72.
3. Church A. The Calculi of Lambda-Conversion. — Princeton: Princeton University Press, 1941. — 221 p.

4. Wadler P. The Essence of Functional Programming // Proceedings of the 19th ACM Symposium on Principles of Programming Languages (POPL). — San Francisco, 1992. — P. 1–14.
5. Milner R. Communicating and Mobile Systems: the π-Calculus. — Cambridge: Cambridge University Press, 1999. — 253 p.
6. Milner R. A Calculus of Communicating Systems. — Berlin: Springer-Verlag, 1980. — 211 p.
7. Milner R. Communication and Concurrency. — Englewood Cliffs: Prentice Hall, 1989. — 221 p.
8. Sangiorgi D. Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms. — PhD Thesis. — University of Edinburgh, 1992. — 186 p.
9. Milner R. Functions as Processes // Mathematical Structures in Computer Science. — 1992. — Vol. 2, No. 2. — P. 119–141.
10. Roslovtsev V.V., Kononov V.M., Nikolaichik A.E., Ergashev O.I. Implementation of a software framework for a relational-applicative computing environment // Scientific WorldJournal. — 2017. — Issue 15, vol. 1. — Minsk: State of emergency "Eln". — pp. 85-93.
11.