Санкт-Петербургский государственный университет
Кафедра системного программирования

Разработка транслятора модельного функционального языка в Interaction Nets

Пономарев Николай Алексеевич

Научный руководитель: доцент кафедры системного программирования, к. ф.-м. н., Григорьев С. В.
Рецензент: инженер-исследователь, Лаборатория YADRO СПбГУ, Косарев Д. С.

Санкт-Петербург
2025

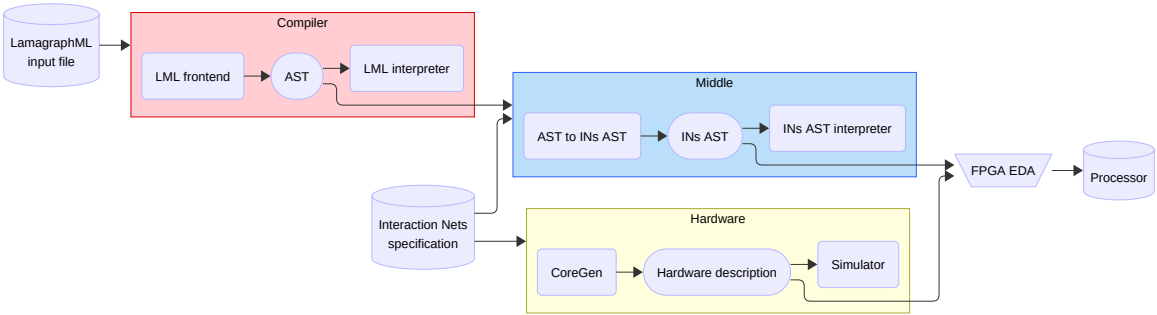
Введение

- Задачи искусственного интеллекта и анализа графов влекут за собой нерегулярный параллелизм
- Традиционные архитектуры плохо справляются с нерегулярным параллелизмом $\Longrightarrow$ требуются специализированные ускорители
- Interaction Nets — модель вычислений, которой естествен нерегулярный параллелизм
- Существуют программные реализации Interaction Nets, однако попыток реализовать ускоритель на её основе пока не предпринималось

Lamagraph

Проект Lamagraph исследует возможности по разработке

- параметризуемого многоядерного сопроцессора на основе Interaction Nets
- ML-подобного функционального языка для программирования сопроцессора



Состояние проекта

Проект на первом этапе: разработка минимальной инфраструктуры для создания ускорителей на основе Interaction Nets

На данном этапе от проекта ожидается следующее

- Приоритизация получения полнофункционального прототипа, содержащего все компоненты
- Использование единого стека технологий: Clash $\Longrightarrow$ Haskell
- Возможность анализа результатов каждого этапа работы

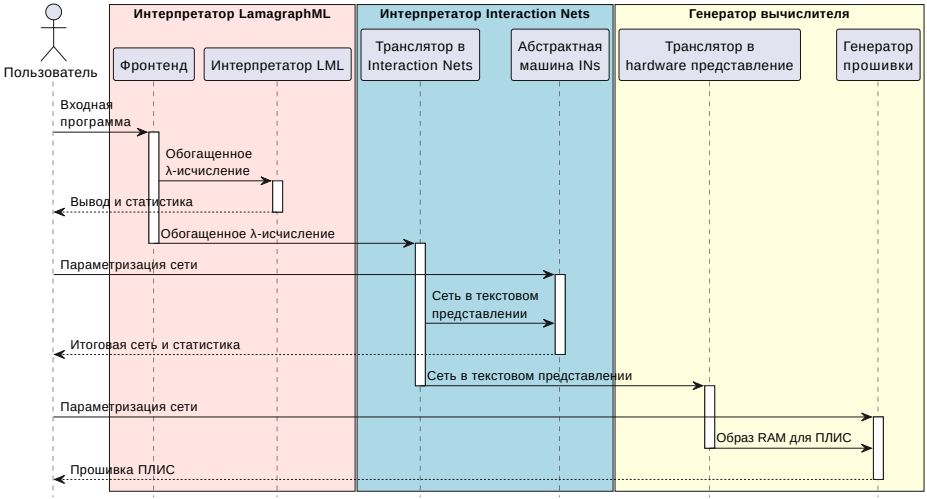
Постановка задачи

Целью работы является разработка транслятора модельного функционального языка в Interaction Nets

Задачи:

1. Реализовать интерпретатор модельного ML-подобного языка
   - Фронтенд транслятора
   - Интерпретатор обогащенного λ-исчисления
2. Реализовать транслятор λ-исчисления в Interaction Nets
3. Реализовать интерпретатор Interaction Nets, поддерживающий сбор метрик исполнения

Реализация


Пользователь
Интерпретатор LamagraphML
Интерпретатор Interaction Nets
Генератор вычислителя
Фронтенд
Интерпретатор LML
Транслятор в Interaction Nets
Абстрактная машина INs
Транслятор в hardware представление
Генератор прошивки
Входная программа
Обогащенное λ-исчисление
Вывод и статистика
Обогащенное λ-исчисление
Параметризация сети
Сеть в текстовом представлении
Итоговая сеть и статистика
Сеть в текстовом представлении
Параметризация сети
Образ RAM для ПЛИС
Прошивка ПЛИС

Фронтенд и интерпретатор LamagraphML

- Используется ML-подобный синтаксис, для представления AST используется паттерн Trees That Grow
- Парсер реализован с помощью связки Alex и Happy с применением property-based тестов
- Используется система типов Хиндли-Милнера
- Для упрощения дальнейших преобразований используется промежуточное представление — обогащенное λ-исчисление
- Реализован интерпретатор на основе замыканий, на данный момент он наследует стратегию языка реализации — call-by-need

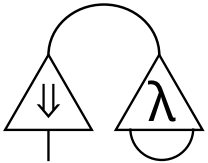
Транслятор λ-исчисления в Interaction Nets

- Существует не одна схема трансляции λ-исчисления в Interaction Nets
- Используем схему, реализующую стратегию вычислений call-by-value
- Выяснилось, что схемы трансляции разрабатываются только для чистого λ-исчисления
  - Существующие расширения не подходят в нашем случае
  - Реализована поддержка только чистого λ-исчисления
- Расширения схемы трансляции — предмет дальнейшей работы

Интерпретатор Interaction Nets

- Стандартное представление Interaction Nets — графовое
- Работать с графами в функциональных языках сложно $\Longrightarrow$ используем альтернативное текстовое представление
- Для него существует абстрактная машина, она и была реализована

Графовое представление:



Текстовое представление

$$\langle x \mid \Downarrow (x) \bowtie \lambda(y, y) \rangle$$

Метрики исполнения

Получаемые метрики позволяют отвечать на вопросы

- Сколько операций потребовалось для вычислений?
- Какое ускорение можно получить при параллельном исполнении программы?
- Сколько «ядер» ускорителя может использовать программа?

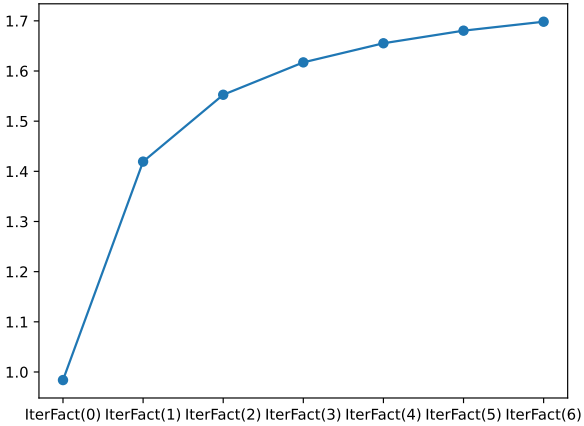


Рис.: Оценка ускорения при использовании параллельности на итеративном факториале

Результаты

В рамках выпускной квалификационной работы был создан программный стек, состоящий из

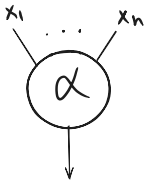
1. интерпретатора модельного ML-подобного языка со стратегий call-by-need
2. транслятора чистого λ-исчисления в Interaction Nets в стратегии call-by-value
3. интерпретатора Interaction Nets на основе абстрактной машины с возможностью сбора метрик исполнения

Исходный код находится в репозитории:
`https://github.com/Lamagraph/interaction-nets-in-fpga`
Имя коммитера: `WoWaster`

Результаты работы были представлены на конференции «MAT-MEX. НАУКА 2025»

Формальное определение Interaction Nets I/II

- Σ — множество символов
- Помеченная символом из Σ вершина — *агент*
- Связи между агентами — *провода*
- Места соединения агентов проводами — *порты*
- Каждый агент имеет арность ar
- Если $\alpha \in \Sigma$ и $\mathrm{ar}(\alpha) = n \in \mathbb{N}$, то у α имеется $n + 1$ портов: n *дополнительных* и один выделенный — *главный*.



Формальное определение Interaction Nets II/II

- *Сеть* — неориентированный граф с символами из Σ в его вершинах
- Ребра соединяют порты вершин, в каждый порт приходит не более одного ребра
- Порт не соединенный ни с одним ребром — *свободный*, множество таких портов — *интерфейс*
- Пара агентов $(\alpha, \beta) \in \Sigma \times \Sigma$, соединенных своими главными портами, — *активная пара*
- Правило $((\alpha, \beta) \Longrightarrow N)$ заменяет активную пару (α, β) на сеть N.
- Для каждой пары агентов существует не более одного правила редукции, при этом в процессе редукции интерфейс сохраняется

Пример Interaction Nets

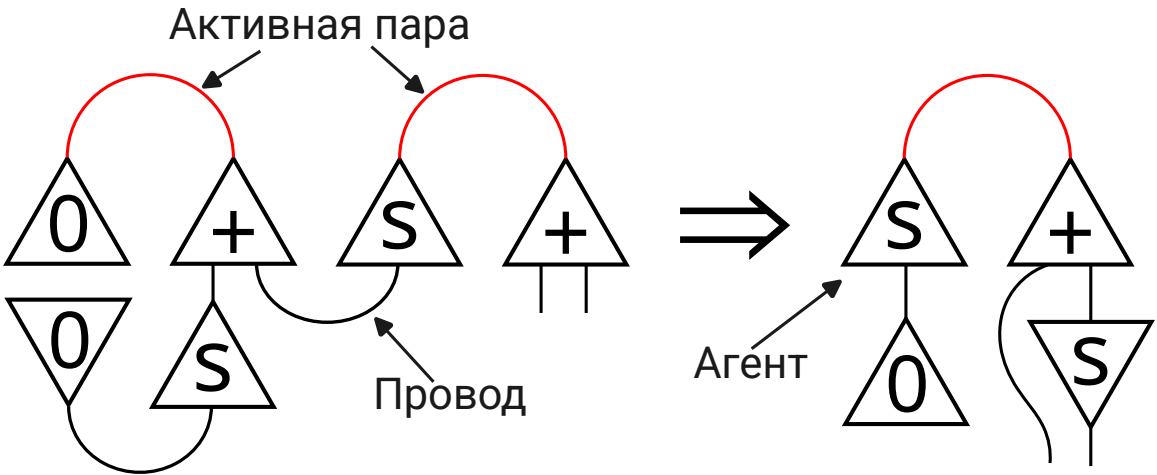
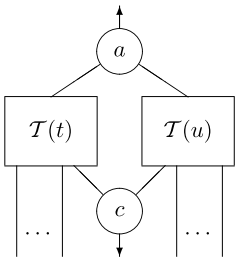


Схема трансляции[1]

Пусть $\mathcal{T}(\cdot)$ — трансляция

Переменные Переменные представляются проводами, поскольку рассматриваются только замкнутые термы

Применение Для применения $\mathcal{T}(t\ u)$ генерируется агент a, а к его портам подключаются результаты $\mathcal{T}(t)$ и $\mathcal{T}(u)$. Если множество свободных переменных t и u не пусто, то для общих переменных генерируются агенты-дупликаторы c



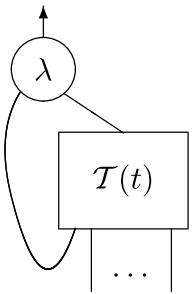
[1]По материалам Sinot F.-R. Call-by-Name and Call-by-Value as Token-Passing Interaction Nets // Typed Lambda Calculi and Applications Lecture Notes in Computer Science. / под ред. P. Urzyczyn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. С. 386–400.

Схема трансляции[1]

Пусть $\mathcal{T}(\cdot)$ — трансляция

Переменные Переменные представляются проводами, поскольку рассматриваются только замкнутые термы

Абстракция Для абстракции $\mathcal{T}(\lambda x.t)$ генерируется агент λ, правый порт которого связывается с $\mathcal{T}(t)$, а левый с проводом, соответствующим x



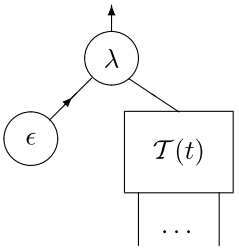
[1] По материалам Sinot F.-R. Call-by-Name and Call-by-Value as Token-Passing Interaction Nets // Typed Lambda Calculi and Applications Lecture Notes in Computer Science. / под ред. P. Urzyczyn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. С. 386–400.

Схема трансляции[1]

Пусть $\mathcal{T}(\cdot)$ — трансляция

Переменные Переменные представляются проводами, поскольку рассматриваются только замкнутые термы

Абстракция Для абстракции $\mathcal{T}(\lambda x.t)$ генерируется агент λ, правый порт которого связывается с $\mathcal{T}(t)$, если x не содержится в t, то генерируется агент-уничтожитель ε



[1] По материалам Sinot F.-R. Call-by-Name and Call-by-Value as Token-Passing Interaction Nets // Typed Lambda Calculi and Applications Lecture Notes in Computer Science. / под ред. P. Urzyczyn. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. С. 386–400.