Выпускная квалификационная работа

Разработка синтаксиса dataflow языка программирования для решения задач в разностных схемах

Студент: Зайченков П.О., 818 группа,
Научный руководитель: к.ф.-м.н. Нейман-заде М.И.

Московский Физико-Технический Институт

Москва, 2012

Цель работы

- Разработка синтаксиса языка программирования для решения задач в разностных схемах
- Использование dataflow парадигмы: описание зависимостей между данными в терминах потоков

Особенности языка

- Сокращение времени написания задачи за счет приближения синтаксиса языка математической нотацией к предметной области
- Поддержка параллельности

Концепции языка
Математическая нотация

- Математическая нотация естественна для dataflow систем
- Использование текстового процессора LaTeX для визуализации математических концепций

Программа на Eq

$$heat(T, X, h, \tau, a)\colon \mathbb{Z}, \mathbb{Z}, \mathbb{R}, \mathbb{R}, \mathbb{R} \rightarrow ([0, \ldots] \Rightarrow \mathbb{R})$$

$$u \in ([0, X] \Rightarrow \mathbb{R})$$

$$u_j^{[0]} \mid \forall j \leftarrow sin(\pi \cdot j \cdot h)$$

$$u_j^{[\mathbf{l}]} \leftarrow \begin{cases} j = 0 \cup j = X & 0 \\ \textbf{otherwise} & \frac{a^2 \cdot \tau}{h^2} \cdot (u_{j-1}^{[\mathbf{l}-1]} + u_{j+1}^{[\mathbf{l}-1]}) + (1 - \frac{2 \cdot a^2 \cdot \tau}{h^2}) \cdot u_j^{[\mathbf{l}-1]} \end{cases}$$

$$\textbf{return } (u^{[T-1]})$$

Концепции языка
Макропроцессор

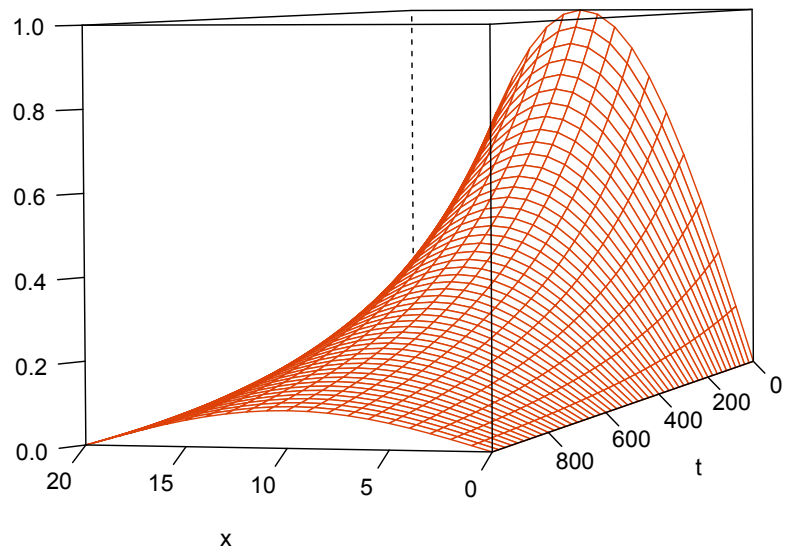
- Дополнение синтаксиса языка произвольными конструкциями для обеспечения совместимости с множество библиотек к LaTeX
- Замена препроцессора LaTeX
- Проверка выражений на основе анализа грамматических правил

Пример	LaTeX	Замена
$\lvert -5 \rvert$	`| \expr |`	`\call{abs}{\expr{1}}`
$\int_0^5 f(x)\mathrm{d}x$	`\int_\expr^\expr \expr dx`	`\call{int}{\expr{1}, \expr{2}, \expr{3}}`

Пример
Уравнение теплопроводности

Формулировка задачи

$$
\begin{aligned}
4u_t &= u_{xx}, 0 \le x \le 2, t > 0 && (1)\\
u(0,t) &= 0 && (2)\\
u(2,t) &= 0 && (3)\\
u(x,0) &= \sin(\pi x) = f(x) && (4)
\end{aligned}
$$



Разностная схема

$$\frac{u_m^p - u_m^{p-1}}{\tau} - a^2 \frac{u_{m-1}^{p-1} - 2u_m^{p-1} + u_{m+1}^{p-1}}{h^2} = 0, \quad a = \frac{1}{2} \qquad (5)$$

Пример
Решение задачи теплопроводности

Язык C

```c
double* heat(unsigned T, unsigned X,
             double h, double tau, double a)
{
  int i, j;
  double alpha = a * a * tau / (h * h);
  double* new_u = (double*)
                  malloc (sizeof (double) * X);
  double* u = (double*)
              malloc (sizeof (double) * X);
  for (i = 0; i < X; i++)
    u[i] = sin(M_PI * i * h);
  for (i = 1; i < T; i++)
    {
      double* tmp;
      for (j = 0; j < X; j++)
        {
          if (j == 0 || j == X - 1)
            new_u[j] = 0;
          else
            new_u[j] = alpha * (u[j-1] + u[j+1])
                     + (1 - 2 * alpha) * u[j];
        }
      tmp = u, u = new_u, new_u = tmp;
    }
  free (new_u);
  return u;
}
```

Язык Eq

$heat(T, X, h, \tau, a)\colon \mathbb{Z}, \mathbb{Z}, \mathbb{R}, \mathbb{R}, \mathbb{R} \to ([0, \ldots] \Rightarrow \mathbb{R})$

$u \in ([0, X] \Rightarrow \mathbb{R})$

$$u_j^{[0]} \mid \forall j \leftarrow sin(\pi \cdot j \cdot h)$$

$$u_j^{[\mathbf{l}]} \leftarrow \begin{cases} j = 0 \cup j = X & 0 \\ \textbf{otherwise} & \frac{a^2 \cdot \tau}{h^2} \cdot (u_{j-1}^{[\mathbf{l}-1]} + u_{j+1}^{[\mathbf{l}-1]}) + (1 - \frac{2 \cdot a^2 \cdot \tau}{h^2}) \cdot u_j^{[\mathbf{l}-1]} \end{cases}$$

return $(u^{[T-1]})$

Пример
Решение задачи теплопроводности

Представление программы на Eq в текстовом виде

```
\begin{eqcode}{heat}{T, X, h, \tau, a}{\type{Z}, \type{Z}, \type{R},
   \type{R}, \type{R}}{\arraytype{\ldots}{R}}
 u \in \arraytype{X}{R} \lend
 u^{[0]}_j | \forall j \gets \call{sin}{\pi \cdot j \cdot h} \lend
 u^{[\iter]}_j \gets
   \begin{cases}
     j = 0 \cup j = X & 0 \lend
     \otherwise & \frac{a^2 \cdot \tau}{h^2} \cdot (u^{[\iter-1]}_{j-1} +
     u^{[\iter - 1]}_{j+1}) + (1 - \frac{2 \cdot a^2 \cdot \tau}{h^2}) \cdot
     u^{[\iter-1]}_j \lend
   \end{cases} \lend
 \return {u^{[T - 1]}} \lend
\end{eqcode}
```

Поддержка параллельности
Разделение параллельных и зависимых операций

1: $u(\partial\Omega) \leftarrow g$
2: **for** $t \leftarrow 1$ **to** T **do**
3: **for** $x \leftarrow 1$ **to** X **do**
4: $u(x,t) \leftarrow$
$f(u, x, t, t-1, \ldots, t-k)$
5: **end for**
6: **end for**

g –граничные/начальные условия

Особенности:

- Вычисления описываются рекуррентным соотношением по t
- Вычисления независимы по x

Поддержка параллельности
Декларативный подход

- Программа представляет собой последовательность деклараций, из которых можно построить граф зависимостей по данным
- Итерационная зависимость выражается рекуррентным соотношением
- Если отсутствует связь между операциями в графе зависимостей, то операции могут быть исполнены параллельно

$$u_j^{[\mathbf{l}]} \quad \leftarrow \begin{cases} j = 0 \cup j = X & 0 \\ \textbf{otherwise} & \frac{a^2 \cdot \tau}{h^2} \cdot (u_{j-1}^{[\mathbf{l}-1]} + u_{j+1}^{[\mathbf{l}-1]}) + (1 - \frac{2 \cdot a^2 \cdot \tau}{h^2}) \cdot u_j^{[\mathbf{l}-1]} \end{cases}$$

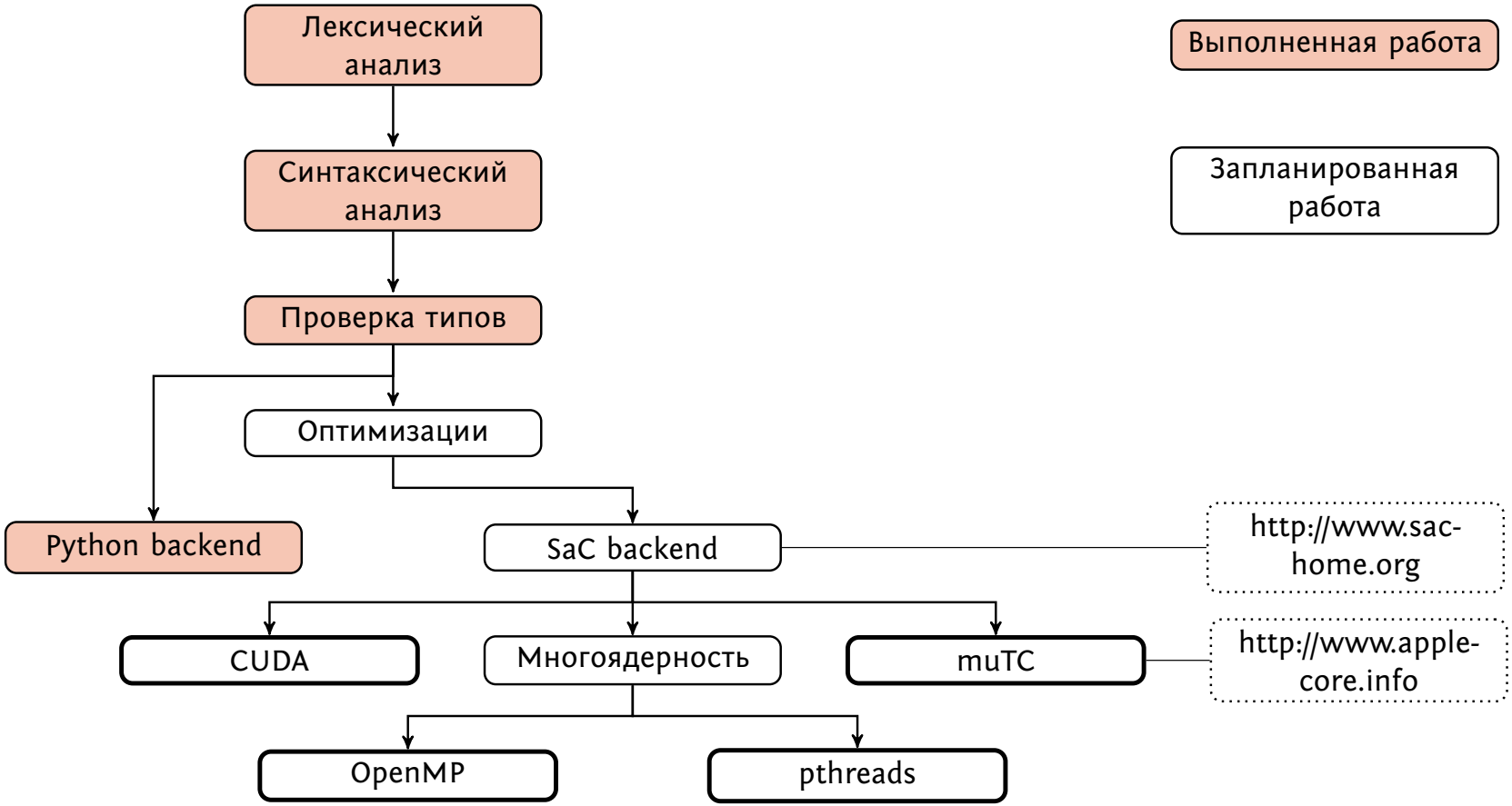
Поддержка параллельности
Синхронные dataflow потоки

- Независимые рекуррентные соотношения могут вычисляться параллельно
- Рекуррентные соотношения порождают потоки

Потоковый функционал

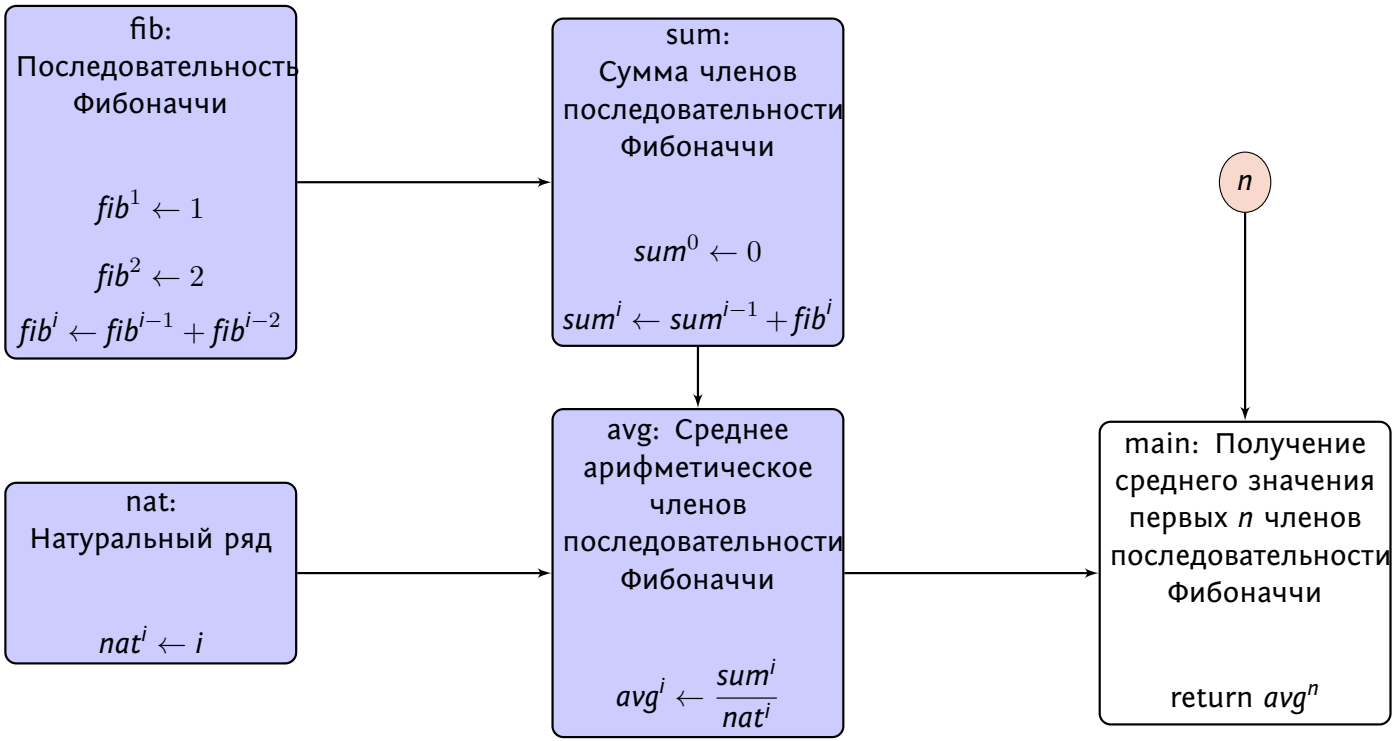
- Источник потоков в виде рекуррентных соотношений
- Активация потоковой схемы через обращение к элементу потока
- Трансформаторы — чистые функции над потоками (значение функции зависит только от своих аргументов)

Структура компилятора



Задача

Нахождение арифметического среднего первых n чисел последовательности Фибоначчи.



Реализация языка и компилятора
Пример

Промежуточное представление

- Программа хранится в виде абстрактного синтаксического дерева
- Рекуррентные соотношения хранятся в отдельной таблице
- Каждая потоковая переменная содержит ссылку на описание рекуррентного соотношения

Реализация языка и компилятора
Пример

Реализация на Eq

$sum(fib)\colon \overline{\mathbb{Z}} \to \overline{\mathbb{Z}}$
$s \in \overline{\mathbb{Z}}$
$s^{[0]} \leftarrow 0$
$s^{[\mathbf{l}]} \leftarrow s^{[\mathbf{l}-1]} + fib^{[\mathbf{l}]}$
return (s)

$avg(sum)\colon \overline{\mathbb{Z}} \to \overline{\mathbb{R}}$
$n \in \overline{\mathbb{Z}}$
$n^{[\mathbf{l}]} \leftarrow \mathbf{l}$
$avg \in \overline{\mathbb{R}}$
$avg^{[\mathbf{l}]} \leftarrow \frac{sum^{[\mathbf{l}]}}{n^{[\mathbf{l}]}}$
return (avg)

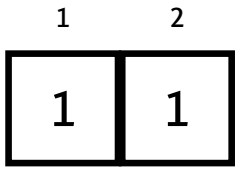
$\mu()\colon \to \mathbb{R}$
$fib \in \overline{\mathbb{Z}}$
$fib^{[1]} \leftarrow 1$
$fib^{[2]} \leftarrow 1$
$fib^{[\mathbf{l}]} \leftarrow fib^{[\mathbf{l}-1]} + fib^{[\mathbf{l}-2]}$
$res \in \overline{\mathbb{R}}$
$res \leftarrow avg(sum(fib))$
return $(res^{[5]})$

Реализация рекуррентного соотношения

Рекуррентное соотношение

$$fib^{[1]} \leftarrow 1$$
$$fib^{[2]} \leftarrow 1$$
$$fib^{[\mathbf{I}]} \leftarrow fib^{[\mathbf{I}-1]} + fib^{[\mathbf{I}-2]}$$

- Текущее окно:



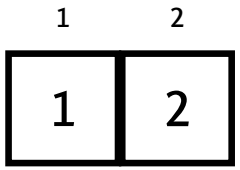
- Вычисление 3 элемента:
  $f^3 = f^2 + f^1 = 2$

Алгоритм вычисления рекуррентного соотношения

Рекуррентное соотношение

$$fib^{[1]} \leftarrow 1$$
$$fib^{[2]} \leftarrow 1$$
$$fib^{[\mathbf{l}]} \leftarrow fib^{[\mathbf{l}-1]} + fib^{[\mathbf{l}-2]}$$

- Текущее окно:



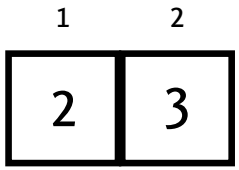
- Вычисление 4 элемента:
  $f^4 = f^3 + f^2 = 3$

Алгоритм вычисления рекуррентного соотношения

Рекуррентное соотношение

$$fib^{[1]} \leftarrow 1$$
$$fib^{[2]} \leftarrow 1$$
$$fib^{[\mathbf{l}]} \leftarrow fib^{[\mathbf{l}-1]} + fib^{[\mathbf{l}-2]}$$

- Текущее окно:



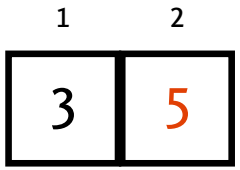
- Вычисление 5 элемента:
$f^5 = f^4 + f^3 = 5$

Алгоритм вычисления рекуррентного соотношения

Рекуррентное соотношение

$$fib^{[1]} \leftarrow 1$$
$$fib^{[2]} \leftarrow 1$$
$$fib^{[\mathbf{l}]} \leftarrow fib^{[\mathbf{l}-1]} + fib^{[\mathbf{l}-2]}$$

- Текущее окно:



Реализация языка и компилятора

Пример

Сгенерированный код на Python

```
class recur_mu_fib:
  def __init__(self, local_vars): # stream constructor
    self.window = [local_vars['mu_fib_1'], local_vars['mu_fib_2']] # sliding window initial values
    self.size = len(self.window)
    self.locals = local_vars
  def generate(self, _iter): # stream generator
    start = 1 # initial case
    i = start # iterator variable
    window = self.window # local window
    while i <= _iter:
      if i < start + self.size:
        yield window[i - start] # if i less than window size, then return a base case
      else:
        yield window[-1] # else return the last value
      if i - start >= self.size-1:
        # calculate new value using recurrent formula
        new = (window[-1] + window[(self.size - 2)])
        window = list (deque(window).rotate(-1)) # window shift routine
        window[-1] = new
      i += 1
def __main():
  mu_fib_1 = 1  # stream initial values
  mu_fib_2 = 1
  mu_fib = recur_mu_fib(locals()) # stream is constructed as a new object
  mu_res = avg(sum(mu_fib)) # we pass a stream object to a 'avg' function
  return (last_value (_mu_res.generate(5))) # generate 5th stream value
```

Результаты

- ✓ Разработан прототип языка, включающий следующие концепции:
  - Математическая нотация
  - Макропроцессор, расширяющий синтаксис языка
  - Разделение параллельных и зависимых операций
  - Рекуррентные соотношения
  - Синхронные потоки
- ✓ Описан синтаксис языка в виде формы Бэкуса-Наура
- ✓ Разработан front-end компилятора
- ✓ Написан backend на Python

Дальнейшая работа

- Интеграция с языком SaC
- Написание тестов и измерение производительности

Спасибо за внимание

https://github.com/zayac/eq