Язык программирования Eq

Введение

В данном докладе мы представляем дизайн и концепцию языка Eq, который предназначен для решения вычислительных задач. Eq основан на простом и важном наблюдении – любая вычислительная задача может быть переформулирована как рекуррентное соотношение над параллельными по данным операциям. Неформально это можно пояснить так: любая вычислительная задача состоит из независимых друг от друга вычислений и итеративного процесса. Традиционно в императивных языках итеративный процесс записывается в виде циклов. Мы полагаем, что это представление имеет недостатки, поэтому в языке Eq мы используем другой подход, учитывающий специфику вычислительных задач.

Структурные особенности вычислительной программы

В данном разделе иллюстрируются особенности программ, в которых реализованы вычислительные методы. В языке Eq мы используем особый подход по отношению к подобным задачам.

В качестве примера рассмотрим классическую вычислительную задачу – краевую задачу для уравнения теплопроводности.

Краевая задача для уравнения теплопроводности

$$\frac{\partial u}{\partial t} = \mu \frac{\partial^2 u}{\partial x^2} \quad (\mu > 0) \text{ при } 0 < t < T,\ 0 < x < 1, \tag{1}$$

$$u(0, x) = \phi(x) \tag{2}$$

$$u(t, 0) = \psi_1(t) \tag{3}$$

$$u(t, 1) = \psi_2(t) \tag{4}$$

Решение задачи – непрерывная и гладкая функция $u(t, x)$ на области $t \in (0, T), x \in (0, 1)$. При решении этой задачи на компьютере используется *метод сетки*. Он заключается в том, что на область определения функции $u(t, x)$ накладывается сетка, в узлах которой ищется решение (Рис. 1). Таким образом решение краевой задачи для уравнения теплопроводности представимо в виде матрицы или двумерного массива. Значения функции считаются с использованием *разностной схемы*. Разностная схема задается шаблоном, в который входят несколько узлов сетки (Рис. 2). Мы используем четыре соседних узла, образующие перевернутую букву "T". Значения функции в этих точках связаны соотношением, которое позволяет по трем заданным значениям восстановить четвертое.

Из начального условия (2) нам известны значения функции в узлах при $t = 0$. Таким образом можно использовать значения функции в узлах разностой схемы на нижнем слое при $i = 0$ для подсчета внутренних точек сетки при $i = 1$. Приведем без доказательства выражение в общем виде для u_j^{i+1} через $u_{j-1}^i, u_j^i, u_{j-1}^i$:

$$u_j^{i+1} = u_j^i + \alpha(u_{j-1}^i - 2u_j^i + u_{j+1}^i), \quad \text{где } \alpha \text{ – некоторая константа.} \tag{5}$$

После вычисления узлов u_j^1 возможно посчитать значения в узлах u_j^2, используя полученные данные. Данный процесс прекращается, когда посчитаны значения функции в узлах самого верхнего слоя. После этого можно гарантировать, что значения во всех узлах сетки посчитаны, и эти значения *апроксимируют* поведение искомой функции на всей области определения.

Основываясь на приведенном решении заметим, что

– **Вычисления на каждом этапе производятся независимо.** Пусть j принимает целые значения $[0, n-1]$, где n – количество узлов в каждом слое. Тогда имеется $n-2$ независимых друг от друга операций.

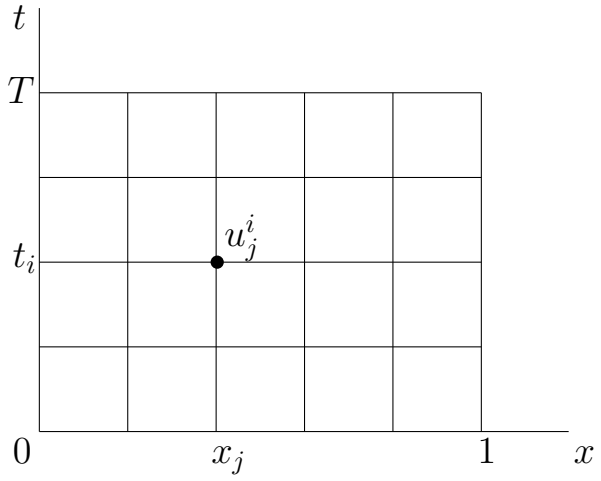

Рис. 1. Сетка, заданная на области $t \in (0,T), x \in (0,1)$, в узлах которой считаются значения функции $u(t,x)$

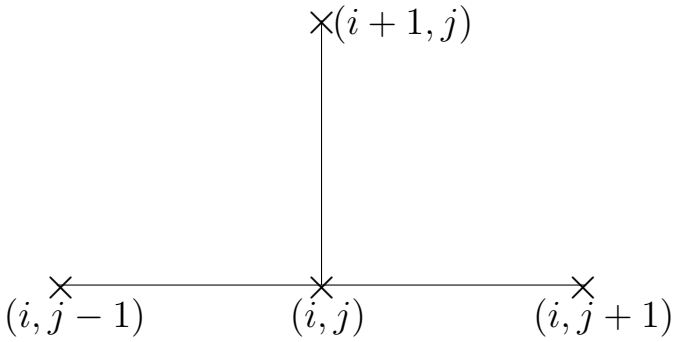

Рис. 2. Разностная схема для вычисления краевой задачи для уравнения теплопроводности.

– **Вычисления представляют собой итеративный процесс по индексу** i. Из выражения (5) следует, что u_i^j при фиксированном i могут быть вычислены независимо, а u^i зависит только от u^{i-1}. Так формируется итеративны процесс. Он прекращается, когда найдены значения в узлах $u(T,x),\ 0 < x < 1$, что гарантирует наличие посчитанных значений во всех слоях. Эти значения апроксимируют искомую функцию на заданной области определения.

Для языка Eq эти принципы являются базовыми. Нами рассмотрен один пример решения вычислительной задачи на компьютере, но вышеуказанные особенности распостраняются на множество различных задач.

Концепция языка Eq

Основная задача языка Eq – сохранение естественной формулировки задачи. Формула (5) лаконично описывает процесс (1). В языке Eq эта формула сохраняет свой вид.

Для начала покажем, как выглядит реализация итеративного процесса на языке программирования Fortran 90.

```
1  allocate ( u(1:x_num,1:t_num) )
2  call u0 ( x_min, x_max, t_min, x_num, x, u(1:x_num,1) )
3  do j = 2, t_num
4    call ua ( x_min, x_max, t_min, t(j-1), u(1,j) )
5    u(2:x_num-1,j) = u(2:x_num-1,j-1) &
```

```
6                     + k * (       u(1:x_num-2,j-1) &
7                           - 2 * u(2:x_num-1,j-1) &
8                           +     u(3:x_num,  j-1) ) &
9                       / x_delt / x_delt
10    call ub ( x_min, x_max, t_min, t(j-1), u(x_num,j) )
11  end do
```

В императивном языке этот алгоритм будет описываться с помощью циклов. Алгоритм полностью соответсвует итеративному процессу (5), но в нем отсутствуют важные детали, имеющиеся в формуле. В выражении (5) верхний и нижний индекс имеют разное назначение, а поэтому сознательно разделены. Индексы i и j в программе пишутся рядом, и используются для адресации элементов массива. В программе не содержится информации о том, что вычисления независимы и производятся параллельно. Также отсутствует явная рекуррентная зависимость по i. В языке Eq мы избавляемся от традиционных циклов и вводим разделение параллельных и зависимых операций.

Возможен функциональный подход к решению задачи. Приведем пример решения задачи на языке Haskell:

```
1  u :: Double -> [Double]
2  u 0 = [0.84, 0.91, 0.14, -0.76, -0.96]
3  u i = l ++ [u(i-1)!!j + (u(i-1)!!(j-1) - 2*u(i-1)!!j + u(i-1)!!(j+1)) | j
       <- [1..3]] ++ r
```

В отличие от реализации на императивном языке здесь представлена формула (5) в том виде, в котором она записывается "на бумаге". Более того, код удовлетворяет вышеописанным требованиям:

- u^i есть функция от u^{i-1}, и только,
- итеративный процесс не зависит от j.

Концепция рекуррентных соотношений над параллельными операциями здесь реализована. Однако основная проблема состоит в экспоненциальном росте вызова функций (Рис. 3).

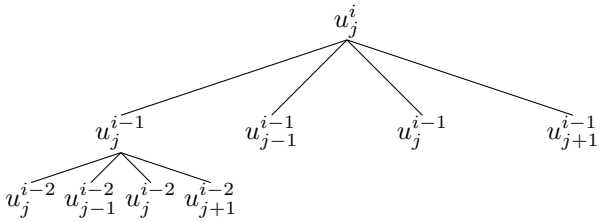

Рис. 3. Фрагмент дерева вызовов функций для формулы (5).

Из-за того, что посчитанные значения не запоминаются, их на каждом этапе приходится вычислять заново. Это ведет к экспоненциальному возрастанию вычислительной сложности и стека.

В языке Eq объединены императивный и функциональный подходы. Решение задачи выглядит так:

$$u^{[0]} = \begin{pmatrix} 0.84 \\ 0.91 \\ 0.14 \\ -0.76 \\ -0.96 \end{pmatrix}$$

$$u_j^{[i]} = \begin{cases} \phi_i & j = 0 \\ \psi_i & j = 4 \\ u_j^{[i-1]} + \alpha \cdot (u_{j-1}^{[i-1]} + 2 \cdot u_j^{[i-1]} + u_{j+1}^{[i-1]}) & 1 \le j \le 3 \end{cases}$$

В качестве синтаксиса используется стандарт для верстки научных публикаций – LaTeX. Это позволяет записать в программе формулу (5) в неизменном виде. В Eq циклы не используются вообще. Вместо этого мы вводим понятие *верхнего индекса*. Формула, в которой используется верхний индекс, должна быть записана таким образом, чтобы выражение с индексом, стоящее в левой части формулы, выражалось через выражение в правой части. Этого достаточно для представления итеративной последовательности. Зависимость будет рекуррентной, если значение функции выражается через собственные значения предыдущих итераций. Нижний индекс используется для обращения к элементам массива. Необходимо явно задать выражение для каждого из элементов. Тогда вычисления значений элементов могут выполняться независимо.

В отличие от функционального подхода экспоненциальный рост вызова функций невозможен в случае с Eq, потому что вычисляется значение элемента массива, а не функции. В этом случае посчитанные значения сохраняются и повторного вычисления не происходит.

Описанный подход позволяет избежать недостатков, которые имеют императивные и функциональные языки. Он обобщается на большой класс вычислительных задач для которых способ записи решения (5) естественен.

Выводы

В языке Eq мы объединяем два подхода к программированию: функциональный и императивный с целью оптимального решения вычислительных задач. В рамках этой идеи мы вводим разделение индексов для массивов данных. Верхний индекс существует в единственном экземпляре и описывает итеративный процесс над массивом. Обращение к элементам массива и запись в них выполняется с помощью нижних индексов. Количество нижних индексов ограничено только размерностью самого массива. Операции над элементами массива или подмассивами выполняются параллельно. Независимое выполнение осуществимо, так как в случае каких-либо зависимостей, они решаются построением итеративного процесса с помощью операций над верхним индексом.

В основе синтаксиса для языка лежит стандарт для текстового описания LaTeX. Это позволяет разделять индексы естественным образом. Более того, мы получаем возможность использовать явно многие математические обозначения в самом тексте программы.