LECTURE 5: POLYHEDRAL MODEL (2)

ALPHA:

A functional data-parallel language based on Polyhedra

Outline

- Motivation & Introduction
- ALPHA Syntax and Semantics (of Domains)
- Substitution & Normalization
- Change of Basis
- Executing, Parallelizing and Compiling ALPHA

Recap: Polyhedral Model

- Exact data-flow analysis: ACL's ⇒ (Polyhedral) SAREs
- Polyhedral SAREs ⇒ SAREs+Reductions (*)
- Determine schedule and allocation
- Transform the SARE
- Generate Code

Why not skip the first two steps?

Programming with SAREs+Reductions

- Advantage: Very high level of programming, functional, amenable to formal manipulation, verification, etc.
- Disadvantage: Restrictive, inefficient, . . .

High Level Programming (examples)

The n-point convolution of a sequence of samples $x_0, x_1, \ldots$ with the weights $w_0 \ldots w_{n-1}$ produces the sequence $y_0, y_1, \ldots$ given by the following equation (assuming that $x_k = 0$, for $-n < k < 0$).

$$y_i = \sum_j w_j x_{i-j} \tag{1}$$

Can this equation **be the program** for convolution?

Forward Substitution

Given an $n \times n$ lower triangular matrix, L, and an n-vector, b, solve the equation $Lx = b$.

By definition,

$$\begin{aligned} b_i &= \sum_{j=1}^{n} L_{i,j} x_j = \sum_{j=1}^{i} L_{i,j} x_j \\ &= \begin{cases} i = 1 & \Rightarrow \quad L_{1,1} x_1 \\ i > 1 & \Rightarrow \quad L_{i,i} x_i + \sum_{j=1}^{i-1} L_{i,j} x_j \end{cases} \end{aligned}$$

Hence,

$$x_i = \begin{cases} i = 1 & \Rightarrow \quad b_1 / L_{1,1} \\ i > 1 & \Rightarrow \quad \frac{b_i - \sum_{j=1}^{i-1} L_{i,j}}{L_{i,i}} \end{cases}$$

LU Decomposition

Given an $n \times n$ square matrix, A, find its factorization into $A = LU$, where L is (strictly) lower-triangular (with unit diagonal) and U is upper-triangular

By definition,

$$\begin{aligned} A_{i,j} &= \sum_{k=1}^{n} L_{i,k} U_{k,j} = \sum_{k=1}^{\min(i,j)} L_{i,k} U_{k,j} = \begin{cases} i \le j & \Rightarrow \quad \sum_{k=1}^{i} L_{i,k} U_{k,j} \\ j < i & \Rightarrow \quad \sum_{k=1}^{j} L_{i,k} U_{k,j} \end{cases} \\ &= \begin{cases} 1 = i \le j & \Rightarrow \quad L_{i,i} U_{i,j} \\ 1 < i \le j & \Rightarrow \quad L_{i,i} U_{i,j} + \sum_{k=1}^{i-1} L_{i,k} U_{k,j} \\ 1 = j < i & \Rightarrow \quad L_{i,j} U_{j,j} \\ 1 < j < i & \Rightarrow \quad L_{i,j} U_{j,j} + \sum_{k=1}^{j-1} L_{i,k} U_{k,j} \end{cases} \end{aligned}$$

LU Decomposition

Hence,

$$U_{i,j} = \begin{cases} 1 = i \le j & \Rightarrow \quad A_{1,j} \\ 1 < i \le j & \Rightarrow \quad A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} U_{k,j} \end{cases}$$

$$L_{i,j} = \begin{cases} 1 = j < i & \Rightarrow \quad A_{i,j} / U_{j,j} \\ 1 < j < i & \Rightarrow \quad \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} U_{k,j} \right) / U_{j,j} \end{cases}$$

Cholesky Factorization

Given an $n \times n$ symmetric positive semi-definite matrix, A, factorize it into $A = LL^T$, where L is lower-triangular.

Definition (of positive semi-definite): $x \neq 0 \Rightarrow x^t A x \geq 0$

Important property: Principal submatrices of positive semi-definite matrices are also positive semi-definite

Theorem: If A is positive semi-definite, there exists a unique lower-triangular matrix L (with positive diagonal elements) such that $A = LL^T$

Proof: By induction:

- If $A = [a]$ then $L = [\sqrt{a}]$

Cholesky Factorization

- Let the theorem hold for matrices of order $n-1$ and let $A' = \begin{pmatrix} A & a \\ a^T & \alpha \end{pmatrix}$ be a symmetric positive semi-definite matrix of order n (A, of order n, is itself positive semi-definite).

We seek to factorize A' as $A' = L'L'^T$; $L' = \begin{pmatrix} L & 0 \\ l^T & \lambda \end{pmatrix}$. So, $A' = L'L'^T$ can be written as $\begin{pmatrix} A & a \\ a^T & \alpha \end{pmatrix} = \begin{pmatrix} L & 0 \\ l^T & \lambda \end{pmatrix} \begin{pmatrix} L^T & l \\ 0 & \lambda \end{pmatrix}$. Hence

$$\begin{aligned} A &= LL^T \\ a &= Ll \\ a^T &= l^T L^T \\ \alpha &= l^T l + \lambda^2 \end{aligned}$$

Cholesky Factorization Equations

$$L_{i,j} = \begin{cases} i = j = 1 & \Rightarrow \sqrt{A_{i,j}} \\ 1 = j < i \leq n & \Rightarrow A_{i,j}/L_{j,j} \\ 1 < j < i \leq n & \Rightarrow \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k} Lj,k\right)/L_{j,j} \\ 1 < j = i \leq n & \Rightarrow \sqrt{a_{i,i} - \sum_{k=1}^{i-1} l_{i,k}^2} \end{cases}$$

Why can't we let these equations be the program?

Thank God for Alpha