Meta-Algorithms
the precursor of

Conformal Computing

Joint work of
D. J Rosenkrantz, L. R. Mullin, and
H. B. Hunt III
Department of Computer Science
College of Computing and Information
UAlbany, SUNY
Targeted Areas of Scientific Computing

- DSP, integral transform analysis, quantum computing, computation quantum mechanics (ab-initio first principles computations/simulations).
  - *Simple* algebraic semantics, are centered around compositions of linear or meta-linear operators and hence are *array-based*.
  - Regular, often predictable memory access patterns.
  - *Simple* control structure.

- compositions of linear and/or meta-linear operators
Scientific Programming
Standard “general high-level programming” approach

- scalarization prior to compiler analysis/code optimization/code generation

- analysis/code optimization in terms of data-flow analysis and loop transformations applied to scalarized code.

- tiling used for cache optimizations.

- Moore’s Law, but in contrast there is Proebsting’s Law

  Proebsting’s Law: Compiler advances double computing power only every 18 years.

- But most scientific programs, especially in the identified above, do a relatively small number of basic operations. However, they do compose these operations.
Scientific Libraries

- **BLAS**: Linear Algebra

- **ATLAS**: adaptable to parallel and distributed architectures
  - must run experiments
  - non-deterministic

- **FFTW**: just the FFT
  - must run experiments

- **Linpack, Scalapack, etc.**:
  - just linear algebra polyalgorithms
  - no compositions
Many problems arising in high-performance parallel/distributed computing may not have a single best algorithm. Rather, the efficient solution of such problems in different computing environments and for different ranges of problem sizes may require a collection of related algorithm each tuned to particular computing environments and range of problem sizes.

Polyalgorithm → meta-algorithm

is

Conformal Computing

For us meta-algorithms consist of non only code but also a program design methodology and a supporting programming environment. We envisioned a meta-algorithm environment used for its development and support a s a non-disjoint union of the following:
• a suitably focused range of related algebraic problems and problem specifications. **Here problem specifications are in terms of simple regular combinations/compositions of high-level operators especially linear and/or meta-linear operators**, and hence, **matrices**.

• transformations, based upon array calculus and an associated **index calculus**. **Here’s where MoA and the ψ-calculus come in**

• analysis and optimization of the high-level specification **prior** to scalarization. **The scalarization is to be driven by this analysis and optimization:** **Intentional Optimizations**

• a suitable message processing/shared memory mechanism, such as **MPI** or **OpenMP**.

• a script-based uniform methodology for exploring the efficiency of alternative implementations.

• In several **proof-of-concept** studies we **initiated the development of a theoretical framework for the transformation and optimization of algorithm/programs that utilize operations and functions operating on entire arrays or array sections (rather than on the individual array elements).**
Monolithic Array Operations

- To date, this framework is based upon the systematic of MoA and the ψ-calculus, that whenever possible uses arrays and array algebra to abstractly model
  - processor topologies
  - levels of cache
  - sets of communication patterns
  - data layouts and distributions.

Applications

FFT, DSP, Quantum Algorithms, Quantum Mechanical Simulations

ab-initio simulations of quantum mechanical effect
An Example of just how simple monolithic-level array programs can be.

**Input:** \( x \in \mathcal{C}, \ n = 2^t \ (t \geq 1, \text{an integer}) \).

**Output:** FFT of \( x \)

\[
x \leftarrow P_n x
\]

for \( q = 1 \) to \( t \)
begin

\[
L \leftarrow 2^q \\
r \leftarrow \frac{n}{L} \\
x_{L\times r} \leftarrow B_L x_{L\times r}
\]

end

- \( P_n \) is an \( n \times n \) permutation matrix.
- Let \( L_* = \frac{L}{2} \) and \( \omega_L \) be a primitive \( L^{th} \) root of 1.

\[
B_L = \begin{bmatrix} I_{L_*} & \Omega_{L_*} \\ I_{L_*} & -\Omega_{L_*} \end{bmatrix}
\]

where \( \Omega_{L_*} \) is a diagonal matrix with values \( 1, \omega_L, \omega_L^2, ..., \omega_L^{L_*-1} \) along the diagonal.
Supporting Facts

- HPF distributions are definable simply in terms of reshape-transpose.
- Tiling can be simulated using reshape-transpose.
- Use of the ψ-calculus eliminates unnecessary intermediate result materializations in monolithic array formulas.
- We’ve also address monolithic BASIC BLOCK level materializations. (To appear in ACM’s TOPLAS).
Combinatorial Optimization Problem

- Cuts down radially the sizes of various combinatorial search spaces arising in program analysis, optimizations, and code generation.
  - E.g. radically decreasing loop complexity/depth of nesting of loops.
  - Enables intentional optimizations → transformations.
  - Supports optimizations on compositions of basic high-level operators/operations.

- By adding computation time as a new array dimension, enables new alternative computation strategies.

**SIMPLE
BUT SURPRISINGLY POWERFUL**
• Likely compatibility with *meta-compiling*, *template meta-programming*

\[ C = A \times B \rightarrow \text{Translate to } \psi \text{ expressions.} \]

\[ \psi \text{ expressions } \rightarrow \text{optimized } \psi \text{ expressions} \]

**Optimize \(\psi\) Expressions \(\equiv \psi\) Reduction.**

• \(\psi\) reduction: Through the \(\psi\)-calculus definitions of operations using shapes, indices are composed to produce a *generic normal form*. These are linear transformations, i.e. index compositions,

**The \(\psi\) Calculus.**

• Loop Nests. These are the realizations of the normal form.

• Object Code. With optimized loop nests a traditional compiler can then produce the object for each level of the loop nests. That is, message passing, shared memory, cache levels, etc.

**A Possible Transformation Process**

Related Work

- *Old* work on efficient compilation of APL programs: P. S. Abrams


- K. Kennedy’s compiler group, HPF stencils: Roth, Kennedy, ...

- L. Snyder’s compiler project, Factor-Join approach: Snyder, etc.
Moral

Sometimes one **size** DOES fit **ALL**

- extensible syntax
- syntax-directed compiling of array programs