SPIRAL
www.spiral.net

SPIRAL: Past, Present, and Future

Jeremy Johnson
Department of Computer Science
Drexel University

Drexel

Autotuning Workshop 2007

SPIRAL
www.spiral.net

Outline

- SPIRAL past (Algebraic foundations and algebra of programs)
- SPIRAL present (Algorithm generator, optimizer, and adapter)
  - www.spiral.net
  - See Franz Franchetti's talk for details
- SPIRAL future (What is a library?)
  - Symmetric FFTs
  - OL (Operator Language) [See Franz Franchetti's talk]

This work was supported by NSF through awards 0234293, 0325687 and by DARPA through the Department of Interior grant NBCH1050009.

Drexel

Outline

- **SPIRAL past (Algebraic foundations and algebra of programs)**
  - **Themes**
  - **People**
  - **Projects**
- **SPIRAL present (Algorithm generator, optimizer, and adapter)**
- **SPIRAL future (What is a library?)**

Themes

- **Algebra and Algorithms**
  - Algebraic foundations of transform algorithms
- **Algebra and Programs**
  - Algebraic constructions of programs
  - Algebraic program transformations
- **Tensor Product**
  - Organizing tool
  - Architectural interpretation
  - Implementation guide
- **Algorithm and Code Generation**
  - Formula generation
  - Formula manipulation
  - Platform adaptation via search of algorithm/implementation space

People

- **Shmuel Winograd, Louis Auslander, Ephraim Feig**
  - ***Multiplicative Complexity of the Fourier Transform*** (1976-84)
  - ***Abelian semi-simple algebras and algorithms for the discrete Fourier transform*****,** 1984
- **Louis Auslander and Richard Tolimieri**
  - ***Is computing with the finite Fourier transform pure or applied mathematics?*****,** 1979
- **Marshall Pease**
  - ***An adaptation of the Fast Fourier Transform for Parallel Processing***, J. ACM, 15(2):252–264, April 1968.
- **C. Temperton**
  - ***Self-sorting mixed-radix Fast Fourier Transforms*****.** J. Comput. Phys., 52(1):1–23, October 1983.
- **Ramesh Agarwal and James Cooley**
  - ***New algorithms for digital convolution*****, 1977.**
  - ***Fourier Transform and Convolution Subroutines for. the IBM 3090 Vector Facility*****, 1986.**

Projects

- **Center For Large Scale Computation (Auslander, Gertner, Tolimieri, R. Johnson)**
  - **Cray X-MP, VAX, AT&T DSP32, Sun SPARCstation**
  - J. R. Johnson, R. W Johnson, D. Rodriguez, and R. Tolimieri**, *A methodology for designing, modifying, and implementing Fourier Transform algorithms on various architectures***,** Circuits Systems Signal Process., (9)4: 449-500, 1990.
  - R. Tolimieri, M. An, and C. Lu**, *Algorithms for Discrete Fourier Transforms and Convolution***,** Springer, 2nd edition, 1997.
  - **Crystallographic FFTs**
    - L. Auslander, R. W. Johnson, and M. Vulis**. *Evaluating finite Fourier transforms that respect group symmetries*.** Acta Cryst., A44:467–478,1988.
    - M. An, J. W. Cooley, and R. Tolimieri**. *Factorization method for crystallographic Fourier transforms*.** Adv. in Appl. Math., 11:358–371, 1990.
    - L. Auslander, J. Johnson, R. W. Johnson**, *An Equivariant Fast Fourier Transform Algorithm***, 1996.

Projects

- **EXTENT (1994 – DARPA James Crowley),** Dai, D. L., Gupta, S. K., Kaushik, S. D., Lu, J. H., Singh, R. V., Huang, C. H., Sadayappan, P., Johnson, J. R., Johnson, R. W.
- **TPL (1996 – DARPA Louis Auslander) J. Johnson and R. W. Johnson**
  - **Demo IBM Symposium for S. Winograd**
  - **DARPA ACMP program review**
  - **MDFT Classification theorems (data flow)**
- **FFTW (1997) Matteo Frigo and Steven G. Johnson**
- **SPIRAL (1998 – DARPA Louis Auslander, Anna Tsao, Dennis Healy)**
  - José Moura, Jeremy Johnson, Robert Johnson, David Padua, Markus Püschel, Victor Prasanna, Manuela Veloso
  - Franz Franchetti, James Hoe, Maria Garzaran
  - Anthony Breitzman (Drexel), Aca Gacic (CMU), Pinit Kumhom (Drexel), Xiaoming Li (UIUC), Gang Ren (UIUC), Bryan Singer(CMU), Jianxin Xiong (UIUC)
  - James Brodman (CS, UIUC), Srinivas Chellappa (ECE, CMU), Sungchul Han (ECE, CMU), Frédéric de Mesmay (ECE, CMU), Peter Milder (ECE, CMU), Aliaksei Sandryhaila (ECE, CMU), Marek Telgarsky (ECE, CMU), Yevgen Voronenko (ECE, CMU), Roland Wunderlich (ECE, CMU), Xu Xu (MCS, Drexel)

Algebra and Algorithms

- S. Winograd, ***On computing the discrete Fourier transform***, Mathematics of Computation, vol. 32, pp. 175-199, 1978.
- S. Winograd, ***Arithmetic Complexity of Computation***, Siam, 1980. of the discrete Fourier transform, Advances in Applied Mathematics, vol. 5, pp. 87-109, 1984.
- L. Auslander and R. Tolimieri, ***Is computing with the finite Fourier transform pure or applied mathematics?***, Bull. Amer. Math. Soc. (N.S.) 1 (1979), 847–897.
- L. Auslander, E. Feig, and S. Winograd, ***Abelian semi-simple algebras and algorithms for the discrete Fourier transform***, Advances in Applied Mathematics, vol. 5, pp. 31-55, 1984.
- L. Auslander, E. Feig and S.Winograd, ***The multiplicative complexity of the discrete Fourier transform***, Adv. in Appl. Math. (5), 31-55, 1984.
- R. C. Agarwal and J. W. Cooley. ***New algorithms for digital convolution***. IEEE Trans. Acoust.
- Speech and Signal Proc., 25:392–410, 1977.
- L. Auslander, J. R. Johnson, and R. W. Johnson, ***MultidimensionalCooley-Tukey algorithms revisited***, Adv. Appl. Math. 17 (1996), 477–519.
- T. W. Cairns, ***On the fast Fourier transform on finite abelian groups***, IEEE Trans. on Computers, vol. C-21, pp. 569-571, 1971.
- Th. Beth, ***Verfahren der Schnellen Fouriertransformation [Methods for the Fast Fourier Transform]***, Teubner, 1984.
- M. Clausen and U. Baum, ***Fast Fourier Transforms***, Wissshaftsverlag, Manheim, Germany, 1993.
- P. Diaconis and D. Rockmore, ***Efficient computation of the Fourier transform on finite groups***, J. Amer. Math. Soc. (3)297-332, 1990.
- Markus Püschel ***Konstruktive Darstellungstheorie und Algorithmengenerierung***, Ph.D. Thesis Computer Science, University of Karlsruhe, Germany 1998 (advisor Prof. Dr. T. Beth, 135 pages); also in ***English:Constructive Representation Theory and Fast Discrete Signal Transforms,***Technical Report Drexel-MCS-1999-1, Drexel University, Philadelphia, 1999 (141 pages)

Algebra and Programs

- Johnson, H. W. and Burrus, C. S., ***The design of optimal DFT algorithms using dynamic programming***, IEEE Transactions on Acoustics, Speech, and Signal Processing 31: 378–387, 1983.
- J. R. Johnson, R. W Johnson, D. Rodriguez, and R. Tolimieri, ***A methodology for designing, modifying, and implementing Fourier Transform algorithms on various architectures***, Circuits Systems Signal Process., (9)4: 449-500, 1990.
- I. Selesnick and C. Burrus. ***Automatic generation of prime length FFT programs***, IEEETransactions on Signal Processing, 44:14–24, 1996.
- Matteo Frigo, ***A Fast Fourier Transform Compiler***, Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '99), Atlanta, Georgia, May 1999.
- Matteo Frigo and Steven G. Johnson, ***The Design and Implementation of FFTW3***, Proceedings of the IEEE 93 (2), 216–231 (2005)
- Anthony F. Breitzman, ***Automatic Derivation and Implementation of Fast Convolution Algorithms***, PhD thesis, Computer Science, Drexel University, 2003
- Anthony F. Breitzman and Jeremy Johnson, ***Automatic Derivation and Implementation of Fast Convolution Algorithms***, Journal of Symbolic Computation 2004, Special Issue on Computer Algebra and Signal Processing, Vol. 37, No. 2, pp. 157-186.
- Aca Gacic, ***Automatic Implementation and Platform Adaptation of Discrete Filtering and Wavelet Algorithms***, Ph.D. thesis, Electrical and Computer Engineering, Carnegie Mellon University, 2004

Tensor Product, Algorithms, and Architectures

- M. C. Pease, ***An adaptation of the Fast Fourier Transform for Parallel Processing***, J. ACM, 15(2):252–264, April 1968.
- C. Temperton, ***Self-sorting mixed-radix Fast Fourier Transforms***. J. Comput. Phys., 52(1):1–23, October 1983.
- J. R. Johnson, R. W Johnson, D. Rodriguez, and R. Tolimieri, ***A methodology for designing, modifying, and implementing Fourier Transform algorithms on various architectures***, Circuits Systems Signal Process., (9)4: 449-500, 1990.
- R. W. Johnson, C.-H. Huang, and J. R. Johnson, ***Multilinear algebra and parallel programming***, The Journal of Supercomputing, 5: 189-217, 1991.
- J. R. Johnson and R. W. Johnson, Programming Schemata for Tensor Products, 1992.
- Dai, D. L., Gupta, S. K., Kaushik, S. D., Lu, J. H., Singh, R. V., Huang, C. H., Sadayappan, P., and Johnson, R. W. 1994. ***EXTENT: a portable programming environment for designing and implementing high-performance block recursive algorithms***. In Proceedings of the 1994 ACM/IEEE Conference on Supercomputing (Washington, D.C., November 14 - 18, 1994). Supercomputing '94. ACM Press, New York, NY, 49-58.
- R. Tolimieri, M. An, and C. Lu, ***Algorithms for Discrete Fourier Transforms and Convolution***, Springer, 2nd edition, 1997.
- C. Van Loan, Computational Framework of the Fast Fourier Transform, Siam, 1992.
- J. R. Johnson and R. W. Johnson, ***Automatic Generation and Implementation of FFT Algorithms***, Proc. SIAM Conference of Parallel Processing for Scientific Computing, 1999.
- N. P. Pitsianis, ***The Kronecker Product in Approximation and Fast Transform Generation***, PhD Thesis, Department of Computer Science, Cornell University, Jan. 1997

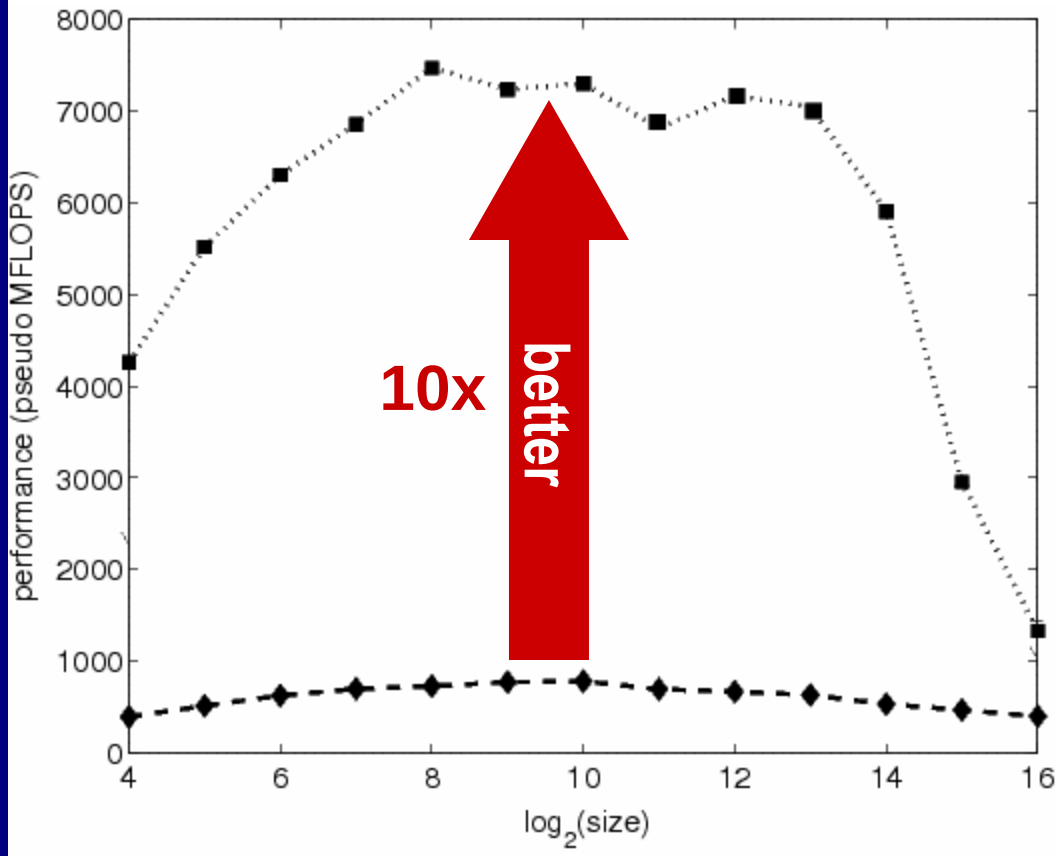
SPIRAL References

- M. Püschel, J. Moura, J. Johnson, D. Padua, M. Veloso, B. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. W. Johnson, N. Rizzolo: ***SPIRAL: Code Generation for DSP Transforms***, Proc. of the IEEE, Special Issue on "Program Generation, Optimization, and Adaptation," Vol. 93, No. 2, pages 232–275, 2005.
- Markus Püschel, Bryan Singer, Jianxin Xiong, José Moura, Jeremy Johnson, David Padua, Manuela Veloso, and Robert W. Johnson, ***SPIRAL: A Generator for Platform-Adapted Libraries of Signal Processing Algorithms***, Journal of High Performance Computing and Applications, special issue on Automatic Performance Tuning, 18(1), 2004, pp. 21-45.
- J. Xiong, J. Johnson, R. Johnson, and D. Padua. ***SPL: A Language and Compiler for DSP Algorithms***. In Proc. PLDI, pages 298–308, 2001.
- Bryan Singer and Manuela Veloso, ***Automating the Modeling and Optimization of the Performance of Signal Transforms***, IEEE Transactions on Signal Processing, Vol. 50, No. 8, 2002, pp. 2003-2014.
- Bryan Singer and Manuela Veloso, ***Learning to Construct Fast Signal Processing Implementations***, Journal of Machine Learning Research, 2002, special issue on the Eighteenth International Conference on Machine Learning (ICML 2001), Vol. 3, pp. 887-919
- F. Franchetti, Y. Voronenko, M. Püschel: ***Loop Merging for Signal Transforms***, Proc. Programming Language Design and Implementation (PLDI) 2005, pages 315–326, 2005.
- F. Franchetti, Y. Voronenko, and M. Püschel, ***A Rewriting System for the Vectorization of Signal Transforms***, VecPar, 2006.
- F. Franchetti, Y. Voronenko, and M. Püschel, ***FFT Program Generation for Shared Memory: SMP and Multicore***, Proc. Supercomputing (SC) 2006.
- Andreas Bonelli, Franz Franchetti, Juergen Lorenz, Markus Püschel, Christoph W. Ueberhuber, ***Automatic Performance Optimization of the Discrete Fourier Transform on Distributed Memory Computers***, Proc. International Symposium on Parellel and Distributed Processing and Applications (ISPA) 2006

Outline

- SPIRAL past (Algebraic foundations and algebra of programs)
- **SPIRAL present**
  - **The Challenge**
  - **DSP algorithms – Symbolic Matrix Factorizations**
  - **Rewrite Rules**
  - **SPIRAL system**
  - **E.G. Architecture adaptation – Vectorization**
  - **Web interface (another view of a library)**
- SPIRAL future (What is a library?)

Challenge #1 (FFT Code Generation)



best available implementation (FFTW, Intel IPP, Spiral)

roughly the same operations count

reasonable implementation (Numerical recipes, GNU scientific library)

- Platforms are complex and change fast
- Best code is platform-dependent
- Optimization at compiler level: difficult, structure lost

DSP Algorithms: E.G. 4-point DFT

$$x \mapsto y = M \cdot x$$

input vector (signal)

output vector (signal)

transform = matrix

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Fourier transform — Diagonal matrix (twiddles) — Input vector

$$y = (\mathrm{DFT}_2 \otimes \mathrm{I}_2) \cdot \mathrm{T}^4_2 \cdot (\mathrm{I}_2 \otimes \mathrm{DFT}_2) \cdot \mathrm{L}^4_2 \, x$$

Output vector — Kronecker product — Identity — Permutation

DSP Algorithms: Transforms & Breakdown Rules

$$\mathbf{DFT}_n = [\omega_n^{k\ell}]_{0\le k,\ell<n}$$

$$\mathbf{DCT\text{-}2}_n = \left[\cos(k(2\ell+1)\pi/2n)\right]_{0\le k,\ell<n},$$

$$\mathbf{DCT\text{-}4}_n = \left[\cos((2k+1)(2\ell+1)\pi/4n)\right]_{0\le k,\ell<n},$$

$$\mathbf{IMDCT}_n = \left[\cos((2k+1)(2\ell+1+n)\pi/4n)\right]_{0\le k<2n,0\le\ell<n},$$

$$\mathbf{DFT}_n \rightarrow \begin{cases} (\mathbf{DFT}_k \otimes \mathrm{I}_m)\, \mathrm{T}^n_m (\mathrm{I}_k \otimes \mathbf{DFT}_m)\, \mathrm{L}^n_k & n = km \\ P_n(\mathbf{DFT}_k \otimes \mathbf{DFT}_m)Q_n & n = km,\ \gcd(k,m) = 1 \\ R_n^T(\mathrm{I}_1 \oplus \mathbf{DFT}_{n-1})D_n(\mathrm{I}_1 \oplus \mathbf{DFT}_{n-1})R_n & p \text{ prime} \\ \mathrm{F}_2 & n = 2 \end{cases}$$

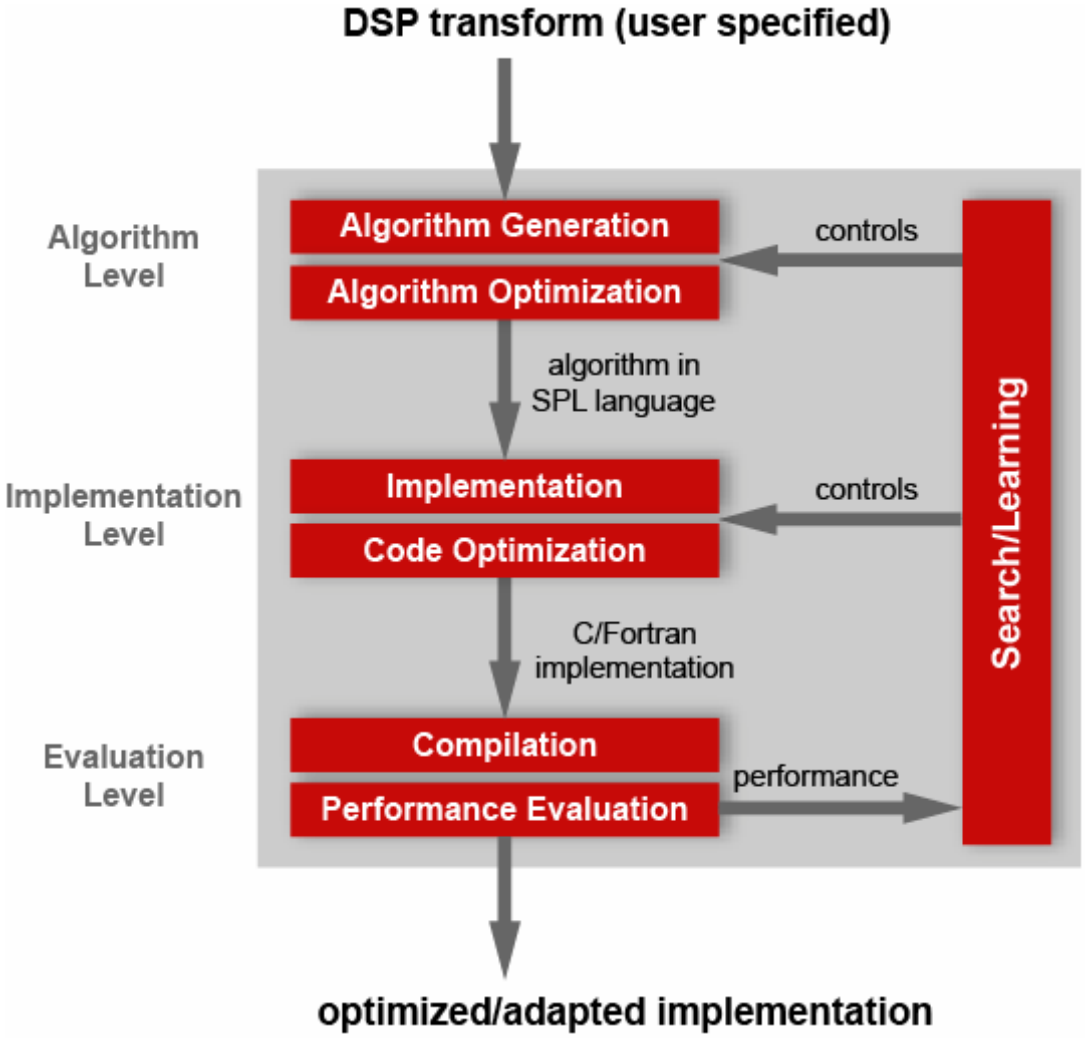
$$\mathbf{DCT\text{-}4}_n \rightarrow \begin{cases} S_n \mathbf{DCT\text{-}2}_n\, \mathrm{diag}_{0\le k<n}(1/(2\cos((2k+1)\pi/4n))) & \\ \mathrm{J}_2\, \mathrm{R}_{13\pi/8} & n = 2 \end{cases}$$

$$\mathbf{DCT\text{-}2}_2 \rightarrow \mathrm{diag}(1, 1/\sqrt{2})\, \mathrm{F}_2$$

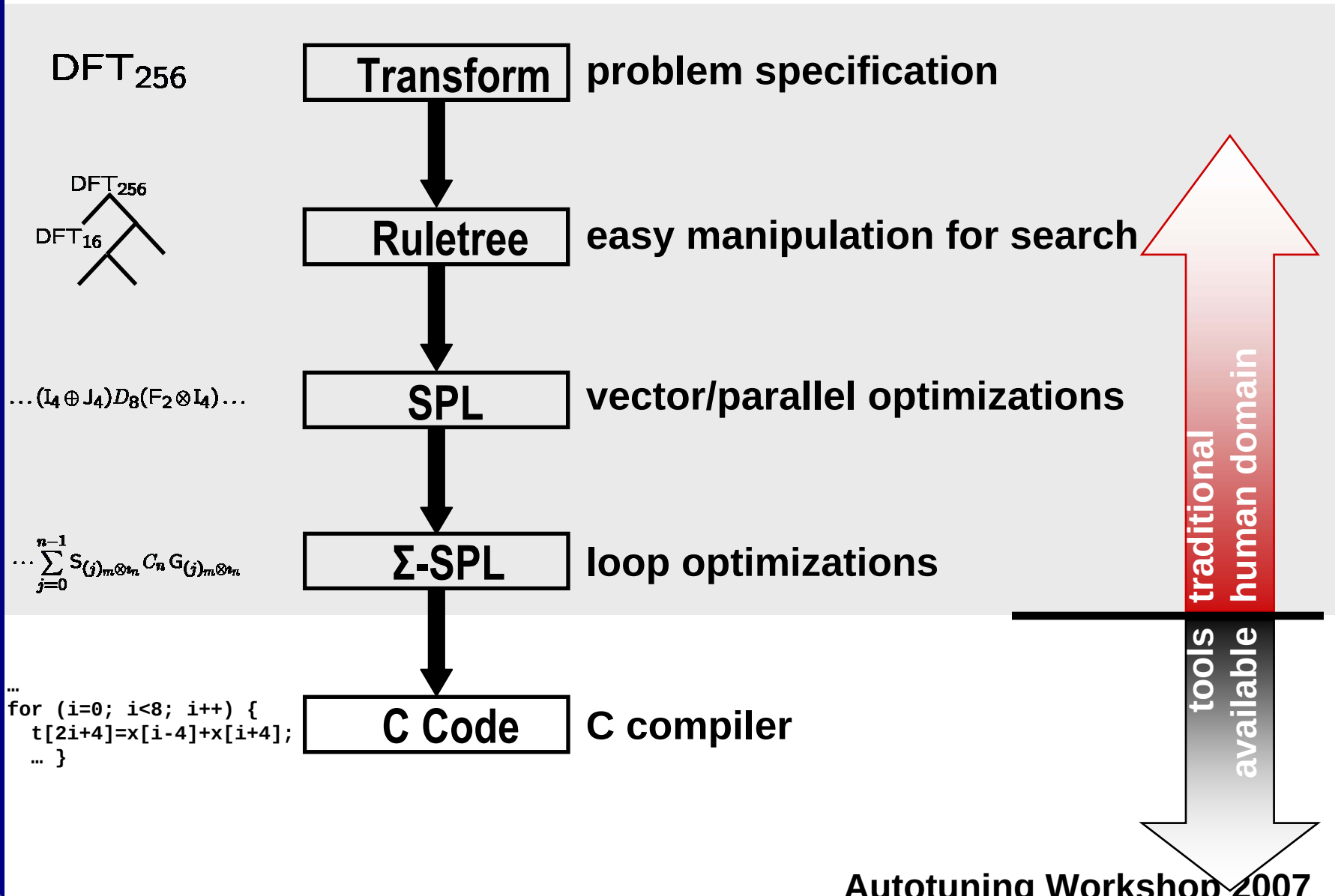
$$\mathbf{IMDCT}_{2m} \rightarrow (\mathrm{J}_m \oplus \mathrm{I}_m \oplus \mathrm{I}_m \oplus \mathrm{J}_m)\left(\left(\begin{bmatrix}1\\-1\end{bmatrix} \otimes \mathrm{I}_m\right) \oplus \left(\begin{bmatrix}-1\\-1\end{bmatrix} \otimes \mathrm{I}_m\right)\right) \mathrm{J}_{2m}\, \mathbf{DCT\text{-}4}_{2m}$$

SPIRAL
www.spiral.net

SPIRAL: Architecture

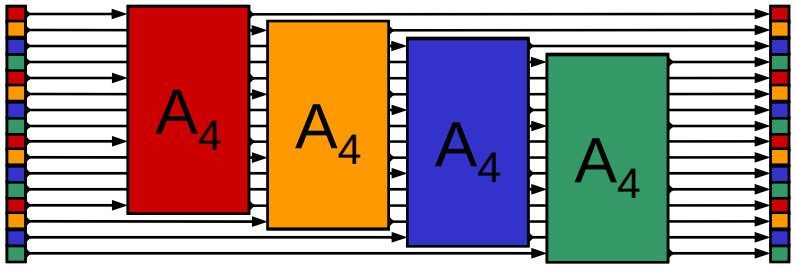


SPIRAL: Abstraction Levels



Vectorization using the Tensor Product

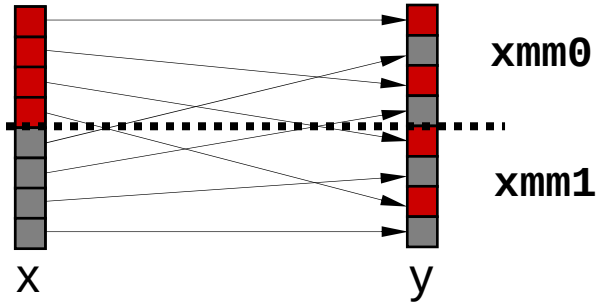
$$y = \left(A \otimes \mathrm{I}_\nu \right) x$$



Characteristics: independent operation on contiguous blocks

Problematic Construct:
permutations must be done in registers

$$y = \mathrm{L}_4^8 x$$



Vectorization Rules

$$\underbrace{(\overline{A})}_{\mathrm{vec}(\nu)} \rightarrow \overbrace{(\underbrace{A}_{\mathrm{vec}(\nu)})}^{\nu} \qquad \overbrace{A \otimes_\nu \mathrm{I}_\nu}^{\nu} \rightarrow (\mathrm{I}_{n/\nu} \otimes \underbrace{\mathsf{L}^{2\nu}_{\nu}}_{\mathrm{vec}(\nu)})(\overline{A} \otimes_\nu \mathrm{I}_\nu)$$

$$\overbrace{\overrightarrow{AB}}^{\nu} \rightarrow \overbrace{\overleftarrow{A}}^{\nu}\overbrace{\overrightarrow{B}}^{\nu} \qquad \overline{(\underbrace{\mathsf{L}^{\nu^2}_{\nu}}_{\mathrm{vec}(\nu)})}^{\nu} \rightarrow (\mathsf{L}^{2\nu}_{\nu} \otimes_\nu \mathrm{I}_\nu)(\mathrm{I}_2 \otimes \underbrace{\mathsf{L}^{\nu^2}_{\nu}}_{\mathrm{vec}(\nu)})(\mathsf{L}^{2\nu}_{2} \otimes_\nu \mathrm{I}_\nu)$$

$$\overbrace{\mathrm{I}_m \otimes A}^{\nu} \rightarrow \mathrm{I}_m \otimes \overbrace{A}^{\nu} \qquad \overbrace{AB}^{\nu} \rightarrow \overbrace{A}^{\nu}\overbrace{B}^{\nu}$$

$$\underbrace{A \otimes \mathrm{I}_m}_{\mathrm{vec}(\nu)} \rightarrow \left(A \otimes \mathrm{I}_{m/\nu}\right) \otimes_\nu \mathrm{I}_\nu$$

$$\underbrace{\mathsf{L}^{n\nu}_{\nu}}_{\mathrm{vec}(\nu)} \rightarrow \left(\mathsf{L}^{n}_{\nu} \otimes_\nu \mathrm{I}_\nu\right)\left(\mathrm{I}_{n/\nu} \otimes \underbrace{\mathsf{L}^{\nu^2}_{\nu}}_{\mathrm{vec}(\nu)}\right)$$

$$\underbrace{\left(\mathrm{I}_m \otimes A\right)\mathsf{L}^{mn}_{m}}_{\mathrm{vec}(\nu)} \rightarrow \left(\mathrm{I}_{m/\nu} \otimes \underbrace{\mathsf{L}^{n\nu}_{\nu}}_{\mathrm{vec}(\nu)}\left(A \otimes_\nu \mathrm{I}_\nu\right)\right)\left(\mathsf{L}^{mn/\nu}_{m/\nu} \otimes_\nu \mathrm{I}_\nu\right)$$

Vectorization by Rewriting

$$\underbrace{\overline{\mathrm{DFT}_{mn}}}_{\mathrm{vec}(\nu)} \rightarrow \overleftarrow{\underbrace{(\overline{\mathrm{DFT}_m \otimes \mathrm{I}_n})}_{\mathrm{vec}(\nu)}}^{\nu} \overline{\underbrace{(\mathrm{T}^{mn}_n)}_{\mathrm{vec}(\nu)}}^{\nu} \overrightarrow{\underbrace{(\mathrm{I}_m \otimes \mathrm{DFT}_n)\,\mathrm{L}^{mn}_m}_{\mathrm{vec}(\nu)}}^{\nu}$$

$$\cdots$$

$$\rightarrow (\mathrm{I}_{mn/\nu} \otimes \underbrace{\mathrm{L}^{2\nu}_{\nu}}_{\mathrm{vec}(\nu)})(\overline{\mathrm{DFT}_m \otimes \mathrm{I}_{n/\nu}} \otimes_\nu \mathrm{I}_\nu)\overline{(\underbrace{\mathrm{T}^{mn}_n}_{\mathrm{vec}(\nu)})}^{\nu}$$

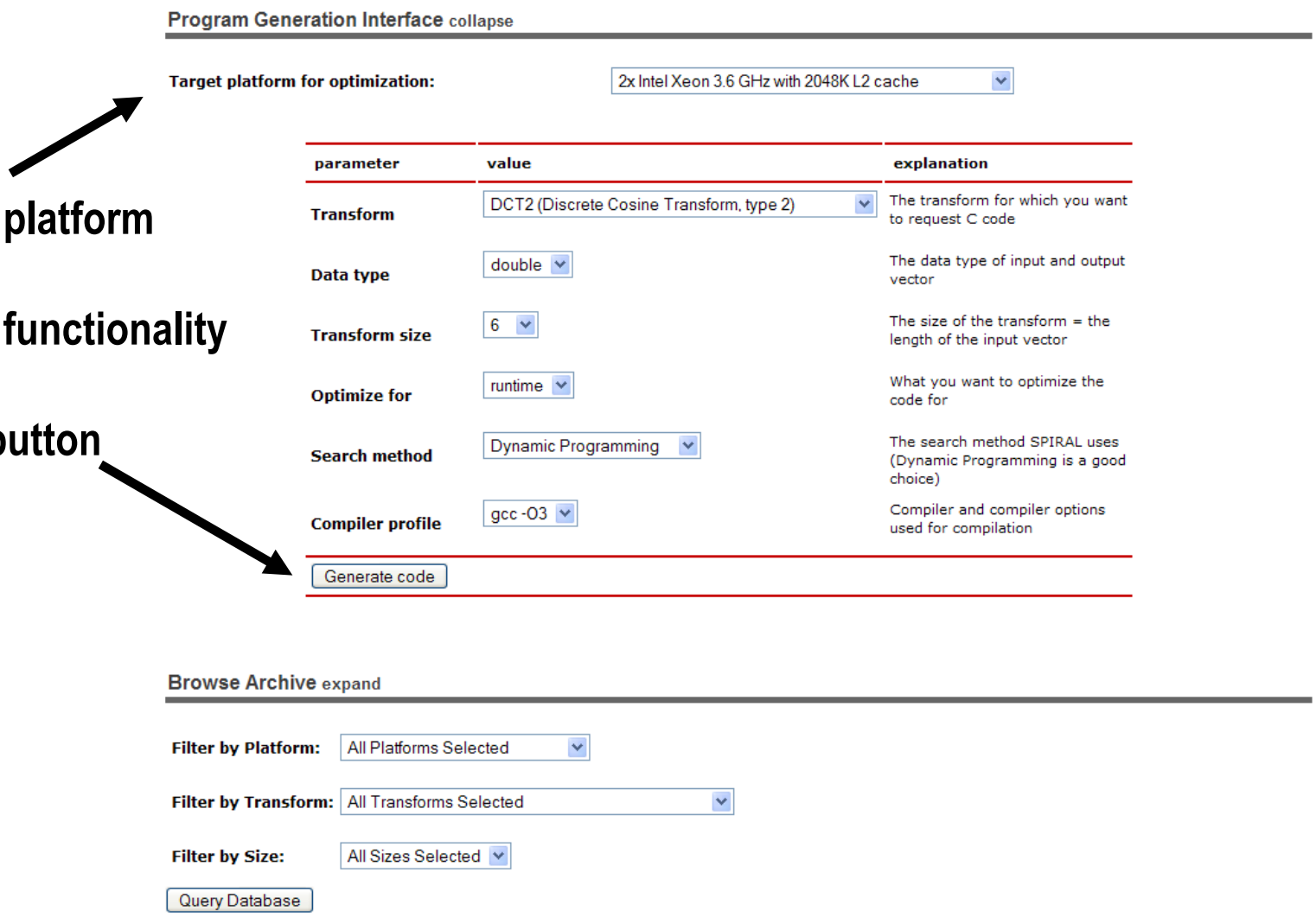
$$\Big(\mathrm{I}_{m/\nu} \otimes (\overline{\mathrm{L}^n_\nu} \otimes_\nu \mathrm{I}_\nu)(\mathrm{I}_{n/\nu} \otimes (\mathrm{L}^{2\nu}_\nu \otimes_\nu \mathrm{I}_\nu)(\mathrm{I}_2 \otimes \underbrace{\mathrm{L}^{\nu^2}_\nu}_{\mathrm{vec}(\nu)})(\mathrm{L}^{2\nu}_2 \otimes_\nu \mathrm{I}_\nu))(\overline{\mathrm{DFT}_n} \otimes_\nu \mathrm{I}_\nu)\Big)$$

$$\Big((\mathrm{L}^{mn}_m \otimes \mathrm{I}_2) \otimes_\nu \mathrm{I}_\nu\Big)(\mathrm{I}_{mn/\nu} \otimes \underbrace{\mathrm{L}^{2\nu}_2}_{\mathrm{vec}(\nu)})$$

Vectorized constructs
Base cases

Spiral Web Interface @spiral.net (beta version)

1. Select platform
2. Select functionality
3. Push button



Outline

- SPIRAL past (Algebraic foundations and algebra of programs)
- SPIRAL present (Algorithm generator, optimizer, and adapter)
- **SPIRAL future (What is a library?)**
  - **Symmetric FFTs**
  - **E.G. Real FFT**
  - **Higher level of abstraction ⇒ Matrix formulation ⇒ SPIRAL**
  - **Generate algorithms as needed to satisfy given properties**
  - **Generalized symmetric DFT and equivariant FFT**

Outline

- Spiral

- Symmetric FFTs
  - Real DFT
  - Real FFT
  - Abstract DFT
  - Abstract FFT
  - Symmetric functions
  - Equivariant FFT algorithm
  - Examples

The Cooley-Tukey Factorization

- Let $N = RS$
  - Divide and Conquer Algorithm

$$y(b_1 S + b_0) = \sum_{a_1} \omega^{S a_1 b_1} \left(\omega^{a_1 b_0} \sum_{a_0} \omega^{R a_0 b_0} x(a_0 R + a_1) \right)$$

  - Matrix Factorization

$$F_N = (F_R \otimes I_S) T_S^N (I_R \otimes F_S) L_R^N$$

  - Computing Time

$$T(N) = RT(S) + ST(R) + \theta(N)$$

$$T(N) = 2T(N/2) + \theta(N) = \theta(N \log N), N = 2^k$$

Real FFT

- **DFT of real vector is conjugate-even:** $y(N-b) = \overline{y(b)}$
- $y = F_{16}x$

$$y(1) = \overline{y(15)}, y(2) = \overline{y(14)}, y(3) = \overline{y(13)}, y(4) = \overline{y(12)},$$
$$y(5) = \overline{y(11)}, y(6) = \overline{y(10)}, y(7) = \overline{y(9)}$$

- **Only need to compute 9 elements of 16 and y(0) and y(8) are real**

Real FFT with Cooley-Tukey

- **Permute** $t_1 = L_4^{16} x$
- $t_2 = (I_4 \otimes F_4) t_1$
  - **Four recursive calls (all have real inputs)**
  - **Use real FFT for recursive calls**
- **Multiplying twiddle factor** $t_3 = T_4^{16} t_2$
  - $t_3(0), t_3(4), t_3(8), t_3(12)$ **are real**
- $y = (F_4 \otimes I_4) t_3$
  - **Four recursive calls**

Four Recursive Calls

$$\begin{pmatrix} y(0) \\ y(4) \\ y(8) \\ y(12) \end{pmatrix} = F_4 \begin{pmatrix} t_3(0) \\ t_3(4) \\ t_3(8) \\ t_3(12) \end{pmatrix} \quad y(4) = \overline{y(12)}$$

Real FFT

Computations saved

$$\begin{pmatrix} y(2) \\ y(6) \\ y(10) \\ y(14) \end{pmatrix} = F_4 \begin{pmatrix} t_3(2) \\ t_3(6) \\ t_3(10) \\ t_3(14) \end{pmatrix} \quad \begin{matrix} y(2) = \overline{y(14)} \\ y(6) = \overline{y(10)} \end{matrix}$$

Computations saved

$$\begin{pmatrix} y(1) \\ y(5) \\ y(9) \\ y(13) \end{pmatrix} = F_4 \begin{pmatrix} t_3(1) \\ t_3(5) \\ t_3(9) \\ t_3(13) \end{pmatrix}$$

$$\begin{pmatrix} y(3) \\ y(7) \\ y(11) \\ y(15) \end{pmatrix} = F_4 \begin{pmatrix} t_3(3) \\ t_3(7) \\ t_3(11) \\ t_3(15) \end{pmatrix} \quad \begin{matrix} y(3) = \overline{y(13)} \\ y(7) = \overline{y(9)} \\ y(11) = \overline{y(5)} \\ y(15) = \overline{y(1)} \end{matrix}$$

Some recursive calls form pairs under conjugate-even

Symmetries inside other recursive calls

Challenge #2 Symmetric FFTs

- Generalize real FFT to other symmetries
  - Avoid certain recursive calls
  - Recursive calls have induced symmetry (may not be same type as original symmetry)
- Apply to multi-dimensional FFTs
  - Multi-dimensional DFTs correspond to DFT of finite Abelian group, A, denoted by F(A)
  - Symmetries obtained from group actions on indexing set A or more generally L(A), the functions on A.
  - Abstract formulation of FFT for F(A) obtained for subgroups B < A
  - Compatible with symmetries provided B chosen appropriately
- Abstract algorithm which must be instantiated and translated into a concrete algorithm, code, and optimized

DFT of Finite Abelian Groups

- **Finite Abelian group *A***
- **Dual group** $\hat{A} = \{\hat{a} \mid \hat{a} : A \to T\}$
  - **The set of homomorphisms, called characters, from *A* to *T* (the unit circle)**
  - **Isomorphic to *A***
- **DFT of *A*, F(A): L(A) $\to$ L($\hat{A}$)**

$$\hat{f}(\hat{a}) = \sum_{a \in A} f(a) < a, \hat{a} >$$

$$\hat{a} \in \hat{A}, < a, \hat{a} > = \hat{a}(a)$$

FFT of Finite Abelian Groups

- **Subgroup** $B < A$
  - **Quotient group**
    $$C = A / B$$
  - **Coset representatives**
    $$\xi : C \to A, a = b + \xi(c)$$
- **Annihilator** $\hat{C} < \hat{A}$
  - **Subgroup of** $\hat{A}$ **whose elements vanishes on B**
    $$\hat{C} = \{\hat{a} \in \hat{A} | < b, \hat{a} >= 1, b \in B\}$$
  - **Quotient group**
    $$\hat{B} = \hat{A} / \hat{C}$$
  - **Coset representatives**
    $$\hat{\eta} : \hat{B} \to \hat{A}, \hat{a} = \hat{c} + \hat{\eta}(\hat{b})$$

$$\hat{f}(\hat{a}) = \sum_{c \in C} < c, \hat{c} > (< \xi(c), \hat{\eta}(\hat{b}) > \sum_{b \in B} f_{\xi(c)}(b) < b, \hat{b} >)$$

$$F_A = Q_\eta (F_C \otimes I_{|B|}) T_{(\eta,\xi)} (I_{|C|} \otimes F_B) P_\xi$$

Divide and Conquer Algorithm

- **Partition section function**

$$f_{\xi(c)}(b) = f(b + \xi(c))$$

- **Compute** $|C| \;\; F(B)s$

$$\hat{f}_{\xi(c)} = F(B) f_{\xi(c)}$$

- **Multiply twiddle factor**

$$h_{\hat{\eta}(\hat{b})}(\hat{c}) = < \xi(c), \hat{\eta}(\hat{b}) > \hat{f}_{\xi(c)}(\hat{b})$$

- **Compute** $|B| \;\; F(C)s$ **and restore** $\hat{f}$

$$\hat{h}_{\hat{\eta}(\hat{b})} = F(C) h_{\hat{\eta}(\hat{b})}$$

$$\hat{f}(\hat{\eta}(\hat{b}) + \hat{c}) = \hat{h}_{\hat{\eta}(\hat{b})}(\hat{c})$$

G-invariant Functions and the DFT

- $\theta \in$ Aut(A)
  - $\theta f(a) = f(\theta(a))$
  - G < Aut(A), $f \in L_G(A) \Rightarrow \theta f = f$ or $f(\theta(a)) = f(a)$
  - Extend symmetries to S(A) = Aut(A) n N(A)
- Symmetric DFT
  - *DFT of G-invariant functions $F_G(A)$*
  - *$F_G(A)$ maps G-invariant functions to $\hat{G}$-invariant functions*

Equivariant FFT

- **Symmetric DFT**
  - ***DFT of G-invariant functions $F_G(A)$ – restriction to $L_G(A)$***
  - ***$F_G(A)$ maps G-invariant functions to $\hat{G}$-invariant functions***
  - **Reduced DFT matrix obtained by adding columns and deleting rows**
- **Divide and conquer algorithm**
  - **Reduce computation to an asymmetric unit**
  - **Subgroup *B<A* is invariant under *G (G acts on cosets)***
  - **Remove redundant recursive calls**
  - **Induced symmetry for recursive calls**
  - **In base case, reduced DFT matrix**

L. Auslander, J. Johnson and R. W. Johnson, An equivariant fast Fourier transform Algorithm, 1996.

Divide and Conquer Algorithm

- **Partition section function for G-invariant B**

$$f_{\xi(c)}(b) = f(b + \xi(c))$$

- **Compute symmetric F(B) for asymmetric unit C/α(G)**

$$\hat{f}_{\xi(c)} = F_{G_c}(B) f_{\xi(c)}$$

- **Multiply twiddle factor**
  - **Create** $h_{\hat{\eta}(\hat{b})} \in L_{G_{\hat{b}}}(C)$ **for asymmetric unit** $\hat{B} / \alpha(\hat{G})$
- **Compute symmetric F(C) for asymmetric unit** $\hat{B} / \alpha(\hat{G})$

$$\hat{h}_{\hat{\eta}(\hat{b})} = F_{G_{\hat{b}}}(C) h_{\hat{\eta}(\hat{b})}$$

Matrix Interpretation of Equivariant Algorithm

$$\hat{f} = P_2 S_{\hat{G}}(\hat{C})(\bigoplus_{\hat{b}\in\hat{B}/\gamma(\hat{G})} F_{\hat{G}_{\hat{b}}}(\hat{C}))L_{\hat{G}}(\hat{C})TS_G(B)(\bigoplus_{c\in C/\gamma(G)} F_{G_c}(B))L_G(B)P_1 f$$

where G_c and $\hat{G}_{\hat{b}}$ are the induced symmetries, $L_G(B)$ and $L_{\hat{G}}(\hat{C})$ are the gather matrices, $S_G(B)$ and $S_{\hat{G}}(\hat{C})$ are the scatter matrices, T is the twiddle factor, P_1 and P_2 are permutations.

$$F_G(A) = P_{\hat{G}}(\bigoplus_{\hat{b}\in\hat{B}/\gamma(\hat{G})} F_{\hat{G}_{\hat{b}}}(\hat{C}))T_G(\bigoplus_{c\in C/\gamma(G)} F_{G_c}(B))P_G,$$

where T_G is a generalized twiddle factor (monomial matrix), P_G and $P_{\hat{G}}$ are monomial matrices.

DFT of $A=Z_3 \times Z_3$

$$
\begin{aligned}
\hat{f} &= F(A)f \\
&= (F_3 \otimes F_3)f \\
&= \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & \omega_3 & \omega_3^2 & 1 & \omega_3 & \omega_3^2 & 1 & \omega_3 & \omega_3^2 \\
1 & \omega_3^2 & \omega_3 & 1 & \omega_3^2 & \omega_3 & 1 & \omega_3^2 & \omega_3 \\
1 & 1 & 1 & \omega_3 & \omega_3 & \omega_3 & \omega_3^2 & \omega_3^2 & \omega_3^2 \\
1 & \omega_3 & \omega_3^2 & \omega_3 & \omega_3^2 & 1 & \omega_3^2 & 1 & \omega_3 \\
1 & \omega_3^2 & \omega_3 & \omega_3 & 1 & \omega_3^2 & \omega_3^2 & \omega_3 & 1 \\
1 & 1 & 1 & \omega_3^2 & \omega_3^2 & \omega_3^2 & \omega_3 & \omega_3 & \omega_3 \\
1 & \omega_3 & \omega_3^2 & \omega_3^2 & 1 & \omega_3 & \omega_3 & \omega_3^2 & 1 \\
1 & \omega_3^2 & \omega_3 & \omega_3^2 & \omega_3 & 1 & \omega_3 & 1 & \omega_3^2
\end{pmatrix} f
\end{aligned}
$$

- $B < A = \{ (0,0),(0,1),(0,2) \}$
- $F(A) = L^9_3 (I_3 \otimes F_3)\, L^9_3 (I_3 \otimes F_3)$

Example Reduced DFT Matrix

$$G = \left\langle \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \right\rangle \quad \{(0,0)\},\ \{(0,1),(0,2)\},\ \{(1,0),(2,0)\},\ \{(1,1),(2,2)\},\ \{(1,2),(2,1)\}$$

$$\begin{pmatrix} a+2b+2c+2d+2e \\ a-b+2c-d-e \\ a-b+2c-d-e \\ a+2b-c-d-e \\ a-b-c-d+2e \\ a-b-c+2d-e \\ a+2b-c-d-e \\ a-b-c+2d-e \\ a-b-c-d+2e \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega_3 & \omega_3^2 & 1 & \omega_3 & \omega_3^2 & 1 & \omega_3 & \omega_3^2 \\ 1 & \omega_3^2 & \omega_3 & 1 & \omega_3^2 & \omega_3 & 1 & \omega_3^2 & \omega_3 \\ 1 & 1 & 1 & \omega_3 & \omega_3 & \omega_3 & \omega_3^2 & \omega_3^2 & \omega_3^2 \\ 1 & \omega_3 & \omega_3^2 & \omega_3 & \omega_3^2 & 1 & \omega_3^2 & 1 & \omega_3 \\ 1 & \omega_3^2 & \omega_3 & \omega_3 & 1 & \omega_3^2 & \omega_3^2 & \omega_3 & 1 \\ 1 & 1 & 1 & \omega_3^2 & \omega_3^2 & \omega_3^2 & \omega_3 & \omega_3 & \omega_3 \\ 1 & \omega_3 & \omega_3^2 & \omega_3^2 & 1 & \omega_3 & \omega_3 & \omega_3^2 & 1 \\ 1 & \omega_3^2 & \omega_3 & \omega_3^2 & \omega_3 & 1 & \omega_3 & 1 & \omega_3^2 \end{pmatrix} \begin{pmatrix} a \\ b \\ b \\ c \\ d \\ e \\ c \\ e \\ d \end{pmatrix}$$

$$\begin{pmatrix} a+2b+2c+2d+2e \\ a-b+2c-d-e \\ a+2b-c-d-e \\ a-b-c-d+2e \\ a-b-c+2d-e \end{pmatrix} = \begin{pmatrix} 1 & 2 & 2 & 2 & 2 \\ 1 & -1 & 2 & -1 & -1 \\ 1 & 2 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & 2 \\ 1 & -1 & -1 & 2 & -1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \end{pmatrix}$$

Equivariant Factorization

$$\begin{aligned}\hat{f} &= F(A)f \\ &= P_3 \begin{pmatrix} \hat{S}_1 & \\ & \hat{S}_{2,3} \end{pmatrix} \begin{pmatrix} F_{\hat{G}1} & \\ & F_3 \end{pmatrix} \begin{pmatrix} \hat{G}_1 & \\ & \hat{G}_{2,3} \end{pmatrix} P_2 T \\ & \quad \begin{pmatrix} S_1 & \\ & S_{2,3} \end{pmatrix} \begin{pmatrix} F_{G1} & \\ & F_3 \end{pmatrix} \begin{pmatrix} G_1 & \\ & G_{2,3} \end{pmatrix} P_1 f\end{aligned}$$

$$\hat{S}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}, \hat{S}_{2,3} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, F_{\hat{G}1} = \begin{pmatrix} 1 & 2 \\ 1 & -1 \end{pmatrix}$$

$$F_{G1} = \begin{pmatrix} 1 & 2 \\ 1 & -1 \end{pmatrix}, G_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, G_{2,3} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Symmetric Factorization Example

$$F_G(A) = \begin{pmatrix} 1 & 2 & 2 & 2 & 2 \\ 1 & -1 & 2 & -1 & -1 \\ 1 & 2 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & 2 \\ 1 & -1 & -1 & 2 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & \omega_3 & \omega_3^2 \\ 0 & 0 & 1 & \omega_3^2 & \omega_3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & \omega_3 & \omega_3^2 \\ 0 & 0 & 1 & \omega_3^2 & \omega_3 \end{pmatrix}$$

Generation of Symmetric FFT Algorithms

- Given A, B < A, G < S(A), (more generally chain of subgroups) compute
  - Symmetric DFT: $F_G(A)$
  - Symmetric CT Factorization
- Implemented using Gap and Spiral
  - Compute C=A/B along with coset reps
  - Compute annihilator
  - Compute asymmetric unit
  - Compute permutations and twiddle factor
  - Compute induced symmetries and reduced DFTs
  - Return as symbolic matrix factorization

J. Johnson and X. Xu, Generating Symmetric DFTs and Equivariant FFT Algorithms, To appear Proc. ISSAC 2007.

SPIRAL
www.spiral.net

Conclusion

- SPIRAL
  - Encodes mathematical knowledge as rewrite rules
  - Encodes architectural knowledge through tags
  - Uses mathematical knowledge to derive (and verify) algorithms and optimize their implementation
  - Uses search (and learning) to adapt algorithms to different platforms
  - Generate algorithms as needed
  - Extensible through the addition of new transforms and new rules
- What next?
  - Generate new algorithms as needed
  - Higher level of abstraction which when reduced to “linear algebra” provides efficient implementation through SPIRAL
  - Generalize beyond linear transforms
- What is a library?
  - Encodes algorithmic knowledge
  - Provides mechanism to derive, tune, adapt, and verify algorithms/implementations

Drexel