How to automatically extract high-level semantics from scientific kernels?

Carnegie Mellon University

Semantics Lifting for Scientific Kernels


Naifeng Zhang


Franz Franchetti

Spiral
Software/Hardware Generation for Performance

```c
#include <math.h>
#include <stdlib.h>
#define M_PI 3.14159265358979323846
void fft_recursive(double* data, int n) {
    if (n <= 1) return;
    // Allocate temporary storage for half-size FFTs.
    double* even = (double*)malloc(n * sizeof(double));
    double* odd = (double*)malloc(n * sizeof(double));
    for (int i = 0; i < n / 2; ++i) {
        even[2 * i] = data[4 * i];
        even[2 * i + 1] = data[4 * i + 1];
        odd[2 * i] = data[4 * i + 2];
        odd[2 * i + 1] = data[4 * i + 3];
    }
    // Recursively compute FFTs.
    fft_recursive(even, n / 2);
    fft_recursive(odd, n / 2);
    for (int i = 0; i < n / 2; ++i) {
        double theta = -2.0 * M_PI * i / n;
        double wr = cos(theta);
        double wi = sin(theta);
        // Twiddle factor multiplication.
        double real = odd[2 * i] * wr - odd[2 * i + 1] * wi;
        double imag = odd[2 * i] * wi + odd[2 * i + 1] * wr;
        data[2 * i] = even[2 * i] + real;
        data[2 * i + 1] = even[2 * i + 1] + imag;
        data[2 * (i + n / 2)] = even[2 * i] - real;
        data[2 * (i + n / 2) + 1] = even[2 * i + 1] - imag;
    }
    // Cleanup.
    free(even);
    free(odd);
}
```

1 Source code

GPT-generated fast Fourier transform in C

Parsing

```
loop(_i, _n/2, decl([_theta, _wr, _wi, _real, _imag], chain(
    assign(_theta, neg(V(2.0))*M_PI() * idiv(_i,_n)),
    assign(_wr, fcall("cos", _theta)),
    assign(_wi, fcall("sin", _theta)),
    assign(_real, nth(_odd, V(2)*_i) * _wr - nth(_odd, V(2)*_i+V(1)) * _wi),
    assign(_imag, nth(_odd, V(2)*_i) * _wi + nth(_odd, V(2)*_i+V(1)) * _wr),
    assign(nth(_d, V(2)*_i), nth(_even, V(2)*_i) + _real),
    assign(nth(_d, V(2)*_i+V(1)), nth(_even, V(2)*_i+V(1)) + _imag),
    assign(nth(_d, V(2)*brackets(_i + idiv(_n, V(2)))),
        nth(_even, V(2)*_i) - _real),
    assign(nth(_d, V(2)*brackets(_i + idiv(_n, V(2)))+V(1)),
        nth(_even, V(2)*_i+V(1)) - _imag))))
```

2 icode

IR I: Low-level internal code

Key Insight

3 Σ-SPL $\sum_{j=0}^{k-1} \mathrm{S}_{\iota_n \otimes (j)_k} A_n \mathrm{G}_{\iota_n \otimes (j)_k}$

IR II: Loop-level program structure (PLDI '05)

Recovering algebraic operators

$A_n \otimes \mathrm{I}_k$ **4 SPL** (Signal processing language)

IR III: Algorithm formalization (PLDI '01)

Imposing a normal form

Combing SPL objects

$$\left(\mathrm{DFT}_2 \otimes \mathrm{I}_{n/2}\right) \mathrm{T}^n_{n/2} \left(\mathrm{I}_2 \otimes \mathrm{DFT}_{n/2}\right) \mathrm{L}^n_2$$

Knowledge base lookup

DFT_n

5 Specification

Discrete Fourier transform

Prototype

```
$ spiral-lift fft.c

    [icode -> Sigma-SPL] ... done
    [Sigma-SPL -> SPL] ... done
    [SPL -> Spec.] ... success

    Recovered specification: RC(DFT(n))
    Recovered algorithm: Recursive Cooley-Tukey FFT
```

Verification

Input Variations	Lifting Output	Outcome
Original (Listing 1)	RC(DFT(n))	Verified
Reordered independent instructions	RC(DFT(n))	Verified
Semantically equivalent indexing	RC(DFT(n))	Verified
Incorrect index access	No SPL match	Bug detected

I: Identity
L: Permutation
T: Twiddle
⊗: Tensor product

Example:

$\mathrm{I}_2 \otimes A = \begin{bmatrix} A & \\ & A \end{bmatrix}$