Uninterpreted Functions: Their use in Code Transformation

Catherine Olschanowsky
Mary Hall
Michelle Strout
CHiLL-I/E Team, Collaborators, and Funding

Mary Hall (University of Utah)
Michelle Strout (University of Arizona)
Catherine Olschanowsky (Boise State Univ.)
Mahdi Soltan Mohammadi (Univ. of Arizona)
Payal Nandi (University of Utah)
Eddie Davis (Boise State University)
Wei He (University of Arizona)
Jongsoo Park, Hongbo Rong, Raj Barik (Intel)
Anand Venkat (Intel, PhD in 2016 at Utah)

This work was supported in part by NSF grant CCF-1563732.
I/E Transformations

**Inspectors**: Traverse index arrays at runtime collecting information and generating new index arrays.

**Executors**: Execute the original computation using the information and/or index arrays produced.

![Diagram of I/E Transformations]

- **CHiLL-I/E**: Compositions of Loop and Data Transformations
- **CUDA-CHiLL**: CHiLL compiler
- **Sparse Polyhedral Framework**
- **Inspector/Executor API**
- **Inspector 1** (e.g., index set splitting)
  - **Explicit Functions**
- **Inspector 2** (e.g., compact-and-pad)
  - **Explicit Functions**
- **Inspector K**
- **Composed Inspector**
- **Index Arrays**
- **Executor** (Transformed Irregular Computation)

---

**Programmer-Defined Functions**
Effective at Improving Performance

Wavefront parallelism [Mirchandaney 88] [Rauchwerger 98] [Zhuang 09] [Park 14]

Distributed memory parallelism [Saltz 91] [Basumallik 09] [Ravishankar 12]

Automatic dense-to-sparse data transformation [Mateev 00] [Pugh 98] [Arnold 10]

Data and iteration reordering of parallel and reduction loops for improved data locality [Ding 99] [Mitchell 99] [Mellor-Crummey 01] [Han 06]

Sparse tiling for aggregating across loops [Douglas 00] (Strout 01) [Mohiyuddin 09] (Krieger 13)
Fast, Compiler Generated I/E\s Require a Common Abstraction

Common Among
• The Inspector
• The Executor
• The Loop Transformation Framework

Uninterpreted Function @ compile time
Explicit Functions @ runtime
Transformation Framework for Sparse Codes

**Problem:** need to modify current approaches to ...
Express inspector-executor transformations (Cathie)
Perform data dependence analysis (Michelle)
Express sparse data transformations (Mary)

**Approach:** Uninterpreted functions to represent
Non-affine loop bounds
Memory accesses
Run-time reordering functions
Run-time groupings
Sparse Polyhedral Framework (SPF)

Loop transformation framework built on the polyhedral model

Uses *uninterpreted functions* to represent index arrays

Enables the *composition of inspector-executor transformations*

Exposes opportunities for compiler to *simplify* indirect array accesses
SPF: Uninterpreted Functions Represent Index Arrays

Uninterpreted Function

\[
y = A \times x
\]

Iteration space (CSR)
\[
I = \{ [i, k] | 0 \leq i < n \land rowptr(i) \leq k < rowptr(i + 1) \}
\]
SPF: Representing Inspector-Executor Transformations with Uninterpreted Functions

Coalesce Transformation

\[ T = \{ [i, k] \rightarrow [k'] | k' = c(i, k) \land 0 \leq k' < NNZ \} \]

\[ NNZ = \text{count}(I) \]
\[ c = \text{order}(I) \]

Old Iterators as Function of New Iterator

\[ i = c^{-1}(k')[0] \]
\[ k = c^{-1}(k')[1] \]
Stop by Eddie’s poster to learn about his proposed IR