IEGen

Automatically Generating Inspectors and Executors

Alan LaMielle

Colorado State University
Tiling Meeting

November 5, 2008

Overview

1. The Polyhedral Framework
   - Representation
   - Transformation
   - Generation
   - Scope
2. The Sparse Polyhedral Framework
   - Representation
   - Transformation
   - Generation
   - Scope
   - Why SPF?
3. IEGen

An example polyhedral computation

An example polyhedral computation

```
    for(i=1; i<=n-1; i++)                    1
    {                                        2
S1:   fx[i-1]+=x[i-1]-x[i];                  3
S2:   fx[i]  +=x[i]  -x[i-1];                4
    }                                        5
```

An example polyhedral computation

```
affine example {N|1<N}                                    1
given (double X  {i|0<=i<=N-1}; )                         2
returns (double FX {i|0<=i<=N-1}; )                       3
through                                                   4
   FX[i] =                                                5
     case                                                 6
        {|i=0}:      X[i]-X[i+1];                         7
        {|0<i<N-1}: X[i]-X[i-1]+                          8
                     X[i]-X[i+1];                         9
        {|i=N-1}:    X[i]-X[i-1];                         10
     esac;                                                11
.                                                         12
```

Iteration space:

- Constraint Representation: $\begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} i \end{bmatrix} \geq \begin{bmatrix} 1 \\ n-1 \end{bmatrix}$
- Vertex/Ray Representation: $(1), (n-1)$
- Set/Relation Syntax: $\{[i] : 1 \leq i \leq n-1\}$

Data spaces of `x` and `fx`:

- Constraint Representation: $\begin{bmatrix} 1 \\ -1 \end{bmatrix} \begin{bmatrix} i \end{bmatrix} \geq \begin{bmatrix} 0 \\ n-1 \end{bmatrix}$
- Vertex/Ray Representation: $(0), (n-1)$
- Set/Relation Syntax: $\{[i] : 0 \leq i \leq n-1\}$

Accesses:

S1's first access of x:

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} i \\ i' \end{bmatrix} \geq \begin{bmatrix} 1 \\ n-1 \\ 1 \\ n-1 \\ -1 \\ 1 \end{bmatrix}$$

Other accesses are similar...

- Unimodular Transformation Framework:

$$[-1]\,[i] = [-i]$$

- Kelly-Pugh Transformation Framework:

$$\{[i] \rightarrow [i'] : i' = n - 1 - i\}$$

Code Generators:

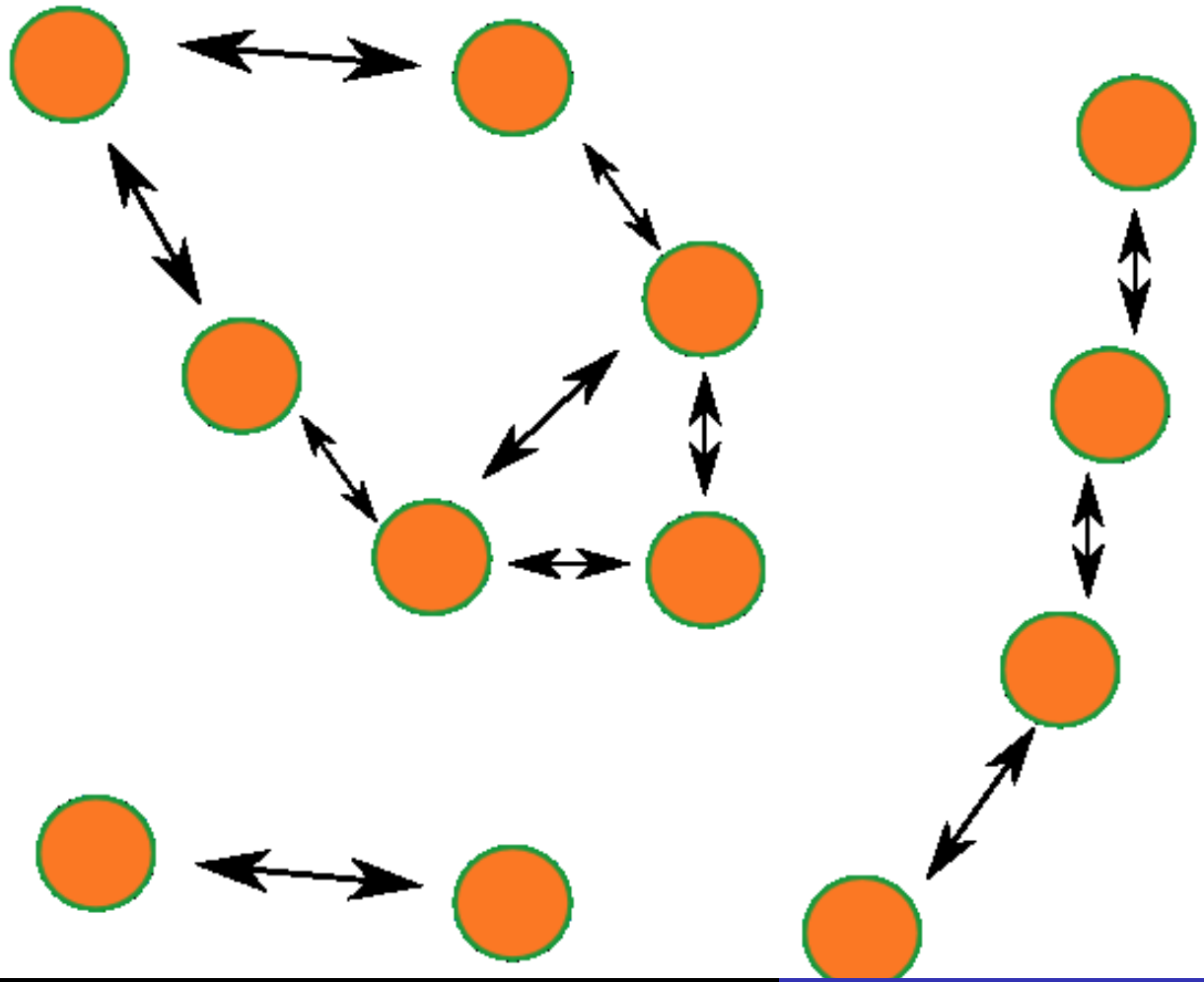
- CLooG
- Omega
- TLOG
- HiTLoG

Common libraries:

- Polylib
- PIP

Dwarf	Support
Dense Linear Algebra	✓
Sparse Linear Algebra	✗
Spectral Methods	✓
N-Body Methods	✗
Structured Grids	✓
Unstructured Grids	✗
Monte Carlo	✓
Combinational Logic	✗
Graph Traversal	✗
Dynamic Programming	✓
Backtrack and Branch & Bound	✗
Construct Graphical Methods	✗
Finite State Machines	✗

An example sparse computation



An example sparse computation

```
    for(i=0; i<n_inter; i++){                1
S1:   fx[inter1[i]]+=                        2
              x[inter1[i]]-                  3
              x[inter2[i]];                  4
S2:   fx[inter2[i]]+=                        5
              x[inter1[i]]-                  6
              x[inter2[i]];                  7
    }                                        8
```

The following needs to be specified:

- Symbolic Constants
- Data Spaces
- Computation inputs/outputs
- Statements with iteration spaces and scattering functions
- Access Relations
- Data dependences
- *Index Arrays (Uninterpreted Functions)*

Three types acting on data spaces and iteration spaces:

	Data Space	Iteration Space
Permutation	CPack	LexMin
Projection	?	?
Embedding	Smashing	Tiling

Specify the relation from the original data/iteration space to the new data/iteration space.

Omega?
Other reordering specific Inspector/Executor generators.

Dwarf	Support
Dense Linear Algebra	✓
Sparse Linear Algebra	✓
Spectral Methods	✓
N-Body Methods	✓
Structured Grids	✓
Unstructured Grids	✓
Monte Carlo	✓
Combinational Logic	✗
Graph Traversal	✗
Dynamic Programming	✓
Backtrack and Branch & Bound	✗
Construct Graphical Methods	✗
Finite State Machines	✗

Supports far more application domains
Still relies on similar theory
Can utilize existing tools for code generation

Demo time!

How do fuzzy dependences in Alphabets compare with UFSs in the SPF?