

Extensions to the Code Generator for Alpha

Inderaj S Bains
IRISA, Rennes

Overview

- Introduction
 - What CodeGen does
 - Where does CodeGen fit in
- Previous Work
- Extensions Proposed

Where Does Code Generator fit in :

What Code Generator does :



Where Does Code Generator fit in :

What Code Generator does :



- Why is it needed :
 - ★ Checking correctness of program

Previous Work

Previous Work

- `writeC()` in C by Doran Wilde
 - ★ Code \Rightarrow Demand Driven

Previous Work

- `writeC()` in `C` by Doran Wilde
 - ★ Code \Rightarrow Demand Driven
- `CodeGen()` in `Mathematica` by Fabien Quillere
 - ★ Code \Rightarrow Respecting Schedule

Previous Work

- `writeC()` in `C` by Doran Wilde
 - ★ Code \Rightarrow Demand Driven
- `CodeGen()` in `Mathematica` by Fabien Quillere
 - ★ Code \Rightarrow Respecting Schedule
- Areas untouched by `CodeGen()`
 - ★ `writeC()` functionality
 - ★ Reductions
 - ★ Subsystems

Functionality of writeC in CodeGen

Functionality of writeC in CodeGen

- Motivations

- ★ People **used to**

- ★ **Single Package**

- ★ Easier **Maintainence** and **Extension**

Functionality of writeC in CodeGen

- Motivations

- ★ People used to
- ★ Single Package
- ★ Easier Maintenance and Extension

- Handling

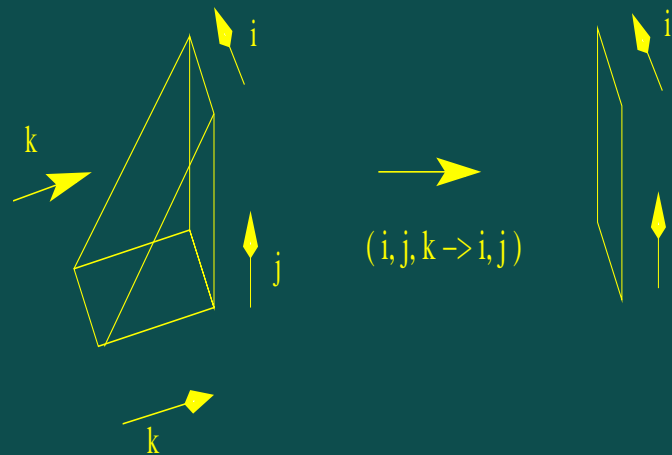
- ★ CodeGen called `Without Schedule` or option `noSched` set
- ★ different function called internally in same file

Handling Reductions

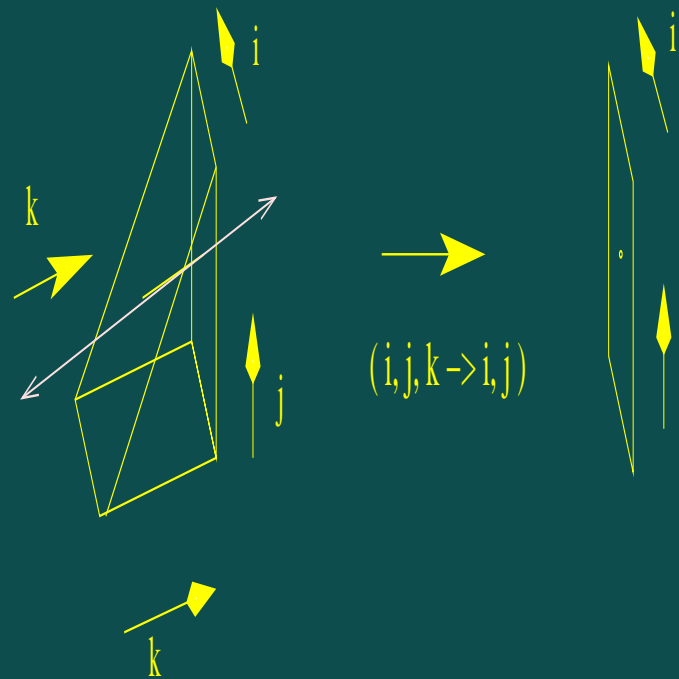
- What is a reduction :
 - `reduce(op , affine ,expr)`
 - $B[i, j] = A[i, j-1] - \text{reduce}(+, (i, j, k \rightarrow i, j), Y)$

Handling Reductions

- What is a reduction :
 - `reduce(op , affine ,expr)`
 - $B[i, j] = A[i, j-1] - \text{reduce}(+, (i, j, k \rightarrow i, j), Y)$
- physically...



- How will it be handled:
 - We take **preimage** of point on expr domain by the **affine function**, **intersect** with the domain of expr and **generate a loop** for the resultant **polyhedra**



Handling Subsystems

Handling Subsystems

- What are subsystems :
 - ★ $\{\text{domain}\}$ use `sub-name[] (inputs) returns (outputs)`;
 - ★ $\{i, j \mid 1 \leq i \leq j \leq N\}$ use `Lxb[] (L , B) returns(X)`;

Handling Subsystems

- What are subsystems :
 - ★ $\{\text{domain}\}$ use sub-name[] (inputs) returns (outputs);
 - ★ $\{i, j \mid 1 \leq i \leq j \leq N\}$ use Lxb[] (L , B) returns(X);
- How can they be handled :
 - ★ To be handled as function calls

Handling Subsystems

- What are subsystems :
 - ★ $\{\text{domain}\}$ use sub-name[] (inputs) returns (outputs);
 - ★ $\{i, j \mid 1 \leq i \leq j \leq N\}$ use Lxb[] (L , B) returns(X);
- How can they be handled :
 - ★ To be handled as function calls
- For testing subsystems I have:
 - ★ Equivalent blocked and un-bloocked programs
 - ★ C programs which generate input,feed in different scanning orders and compare outputs

Handling Subsystems

- What are subsystems :
 - ★ $\{\text{domain}\}$ use sub-name[] (inputs) returns (outputs);
 - ★ $\{i, j \mid 1 \leq i \leq j \leq N\}$ use Lxb[] (L , B) returns(X);
- How can they be handled :
 - ★ To be handled as function calls
- For testing subsystems I have:
 - ★ Equivalent blocked and un-bloocked programs
 - ★ C programs which generate input,feed in different scanning orders and compare outputs
- Usefulness to the Test Bench :

Conclusion

- We Have seen how :
 - **Single package** will be made
 - **writeC()** functionality will be provided
 - **Reductions** and **Subsystems** will be handled