**Alpha, a Language for Parallel Hardware Arrays**  
Mauras 1989 PhD. thesis

Sanjay Rajopadhye  
Colorado State University

**Outline**

- Introduction
- Alpha syntax
- (Denotational semantics)
  - Domains of expressions
  - Context domains
What is Alpha?

- Functional (equational) language
- Declarative
- Based on Systems of Afine Recurrence Equations on polyhedral domains
- with Reductions (& subsystems)
- A data-parallel language

The Essence of Data Parallelism

- **Collections** of elementary data objects
  - sets, bags, arrays
- **Pointwise** Poperations
- **Alignment** of different collections
- **Conditional** operations (on parts of) collections
- **Reductions/Scans** with associative/commutative operators
Recap of Alpha (essence)

A program **Name decls eqns** consists of

- **Name**
  
  ```
  system Name domain;
  ```

- **Declarations:** of Input \((I)\), Output \((O)\), Local \((L)\) variable
  
  ```
  type Vars domain;
  ```

- **Equations**
  
  ```
  Var = Expr
  ```

Alpha syntax

- **Domains:** \{IdxList | ConstrntList\}
- **Expressions:**
  
  - Constant | Var \(V | C\)
  - Point-wise ops: \(Exp \text{ op } Exp\)
  - Case:
    
    ```
    case Exp; ... Exp; esac;
    ```
  - Restrict:
    
    ```
    Dom: Exp
    ```
  - Dependence:
    
    ```
    f@Exp | Exp.f
    ```
  - Reduce:
    
    ```
    reduce(op, f, Exp)
    ```
A system denotes a function from input variables to output variables

An equation denotes a “rule” specifying how a variable is evaluated

An expression denotes a mapping from indices (points in its domain) to values (i.e., multidimensional array/table of values)

Expression semantics (denotational)

Expressions are functions from domains to values. Semantics: two components: domain and a “meaning” function

Domain: set of points where expression is defined.
- “bounds checking” guarantees that NO out-of-bounds exception will be thrown
- Rules to construct domains are straightforward
Domains of Alpha Expressions

- Bottom up construction
  - Domain of a node in the AST is determined by that (those) of its child(ren)
- Simple rules involving properties of polyhedral sets
  - Intersection
  - Union
  - Image (by an affine function)
  - Pre-image (by an affine function)

Constant/Variable (leaf nodes)

- Domain of a constant, $c$ is $\mathbb{Z}^0$, the unique zero-dimensional polyhedron
- The domain of a variable $\text{Var}$ is the domain of its declaration
Pointwise operators

- Domain of $\text{op}(\text{exp}_1, \text{exp}_2)$
- $\mathcal{D}(\text{exp})$ is the intersection of the domains of its children (same for k-ary operators)

$$\mathcal{D}(\text{Exp}) = \mathcal{D}(\text{exp}_1) \cap \mathcal{D}(\text{exp}_2)$$

Restriction operator

- Domain of $\text{Dom:exp}_1$ is the intersection of the domains of its two components

$$\mathcal{D}(\text{Exp}) = \text{Dom} \cap \mathcal{D}(\text{exp}_1)$$
Case operator

- Domain of `case ... exp_i; ... esac` is the disjoint union of the domains of its children

\[ \mathcal{D}(\text{Exp}) = \bigcup_{\text{disj}} \mathcal{D}(\text{exp}_i) \]

Exp =

Dependence

- Domain of `f@exp` is the pre-image of the domain of its child subexpression by the dep function

\[ \mathcal{D}(\text{Exp}) = \text{PreImage}(\mathcal{D}(\text{exp}_1), f) \]

Exp =

\[ f \]

\[ \mathcal{D}(\text{Exp}) = \text{PreImage}(\mathcal{D}(\text{exp}_1), f) \]
Reductions

- Domain of \( \text{reduce}(\text{op}, f, \text{exp}_1) \) is the image of the domain of its child subexpression by the function, \( f \).

\[
\mathcal{D}(\text{Exp}) = \text{Image}(\mathcal{D}(\text{exp}_1), f)
\]

Dependences and Reductions

What does it mean and where is it defined?

- \( f@\text{exp} \) is an expression
  - whose value at \( z \) is the same as that of \( \text{exp} \) at \( f(z) \)
  - and domain is \( f^{-1}(D_{\text{expr}}) \)

- \( \text{reduce}(\text{op}, f, \text{exp}) \) is an expression
  - whose domain is \( f(D_{\text{expr}}) \)
  - and value at \( z \) is obtained by applying \( \text{op} \) to the values of \( \text{exp} \) at all points in \( f^{-1}(z) \)
Example

affine fir \{N|N>0\}
input real w \{i|N>=i>0\};
output real y \{i|i>=0\};
through
\[ y = \text{reduce}(+, (i,j->i), (i,j->j)@w * (i,j->i-j)@y); \]

Example domains

input real w \{i|N>=i>0\}; output real y \{i|i>=0\};
eqns: \[ y = \text{reduce}(+, (i,j->i), w.(i,j->j) * y.(i,j->i-j)); \]

w \(i,j\rightarrow j\)
\(w.(i,j\rightarrow j)\times(i,j\rightarrow i-j)\)
y \(i,j\rightarrow i-j\)
Context domain

- The context domain of a node in the AST is the set of points where it needs be evaluated in order to compute all the values of the LHS variable.
- Deduced "top-down" from the parent's context domain.
  - Based on the type of the parent node using grammar rules.
  - Convention: $exp_i$ is (usually) the node whose context domain we want, $exp$ is the parent.

Topmost node (equation)

- Expression whose parent node is an equation
  - Context domain of $exp$ in the equation

$$Var = exp;$$

is the domain of declaration of $Var$.
Parent is pointwise operator

- $\text{op}(\ldots, \exp_1, \ldots)$
- The context domain of $\exp_1$ is the intersection of $C(\text{Exp})$, and $D(\exp_1)$

$$\text{Exp} = \text{op} \quad C(\exp_1) = D(\exp_1) \cap C(\text{Exp})$$

Parent is restriction operator

- $\text{Dom: } \exp_1$
- Context domain of $\exp_1$ is the intersection of the domains $\text{Dom}$, and $C(\text{Exp})$

$$\text{Exp} = \quad C(\exp_1) = C(\text{Exp}) \cap \text{Dom}$$
Parent is a dependence

- \text{dep} \ @ \ exp
- Context domain of exp is image of the context domain of its parent by \text{dep}

\[ C(\text{exp}_1) = \text{Image}(C(\text{Exp}), \text{dep}) \cap D(\text{exp}_1) \]

Colorado State University
Parent is a reduction

- reduce(op, f, exp₁)
- Context domain of exp₁ is the pre-image of the context domain of its parent by the function, f.

\[ C(\text{exp}_1) = \text{PreImage}(C(\text{Exp}), f) \cap D(\text{exp}_1) \]