

High-Performance Embedded Systems-on-a-Chip

Sanjay Rajopadhye
Computer Science, Colorado State
University

Lecture 10: Alpha

Outline

- Introduction
- ALPHA Syntax
- (Denotational) Semantics
- Substitution, Normalization, & ...
- Change of Basis

Introduction

- Functional (equational, single-assignment, ...)
- Declarative
- Polyhedral Domains
- Systems of Affine Recurrences
- Reductions
- **Data-Parallel**

The Essence of Data Parallelism

- **Collections** of elementary data objects
- **Pointwise Operations**
- **Alignment** of different collections
- **Conditional** operations on (parts of) collections
- **Reductions/Scans** of associative and commutative operators

ALPHA Syntax (BNF)

system $\langle \text{Name} \rangle$: $\langle \text{Domain} \rangle$
 ([$\langle \text{Var} \rangle^*$: $\langle \text{Domain} \rangle$ of $\langle \text{Type} \rangle$]*)

returns
 ([$\langle \text{Var} \rangle^*$: $\langle \text{Domain} \rangle$ of $\langle \text{Type} \rangle$]*)

var
 [$\langle \text{Var} \rangle^*$: $\langle \text{Domain} \rangle$ of $\langle \text{Type} \rangle$]*

let
 [$\langle \text{Var} \rangle = \langle \text{Expression} \rangle$]*

tel

Alpha Domains

- Dual Representation of polyhedra: finite number of **inequalities (constraints)**, or finite number of **generators**
- Alpha (external) syntax:
$$\{ \langle \text{Idx} \rangle * \mid [\langle \text{IdExpr} \rangle \geq \langle \text{IdExpr} \rangle] * \}$$
- Internally, both representations are maintained

Alpha Expression Syntax

- $\langle \text{Var} \rangle \mid \langle \text{Const} \rangle$ Atomic
- $\langle \text{Expression} \rangle \text{op} \langle \text{Expression} \rangle$ Pointwise Ops
- $\langle \text{Domain} \rangle : \langle \text{Expression} \rangle$ Restriction
- `case` [$\langle \text{Expression} \rangle$]* `esac` Case
- $\langle \text{Expression} \rangle . \langle \text{Dep} \rangle$ Affine Dependency
 where $\langle \text{Dep} \rangle$ is $(\langle \text{Idx} \rangle * - > \langle \text{IdExpr} \rangle *)$
- `reduce` (op, $\langle \text{Dep} \rangle$, $\langle \text{Expression} \rangle$) Reduction

Example (Fibonacci)

```

system FibSys : {N | 1<=N} () returns(F : integer );
var    Fib :    {i | 1 <= i <= N} of integer
let
  Fib =
    case
      {i | i<=2} : 1.(i -> ) ;
      {i | i>=3} : Fib.(i -> i-1) + Fib.(i -> i-2) ;
    esac ;
  F =  Fib.( ->N) ;
tel ;

```

Outline

- Introduction
- ALPHA Syntax
- (Denotational) Semantics
- Substitution, Normalization, & ...
- Change of Basis

Semantics of ALPHA Expressions

- Each expression denotes a function from a domain, \mathcal{D} to a $\langle \text{Type} \rangle$ (real, integer or boolean)
- The Semantic function is compositional, and straightforward (eg, $A + B$ denotes the sum of A and B) ...
- Domains too are straightforward

Semantics of ALPHA Expressions

- Each expression denotes a function from a domain, \mathcal{D} to a $\langle \text{Type} \rangle$ (real, integer or boolean)
- The Semantic function is compositional, and straightforward (eg, $A + B$ denotes the sum of A and B) ...
- Domains too are straightforward
- ... except for **affine dependency** and **reductions**

Domains of ALPHA Expressions

- consts, vars

Domains of ALPHA Expressions

- consts, vars

$$\mathcal{D}(c) = \mathcal{Z}^0$$

$$\mathcal{D}(V) = \text{Dom}(V)$$

Domains of ALPHA Expressions

- consts, vars

$$\mathcal{D}(c) = \mathcal{Z}^0$$

$$\mathcal{D}(V) = \text{Dom}(V)$$

- ops

Domains of ALPHA Expressions

- consts, vars

$$\mathcal{D}(c) = \mathcal{Z}^0$$

$$\mathcal{D}(V) = \text{Dom}(V)$$

- ops

$$\mathcal{D}(e_1 \text{ op } e_2) = \mathcal{D}(e_1) \cap \mathcal{D}(e_2)$$

Domains of ALPHA Expressions

- consts, vars

$$\mathcal{D}(c) = \mathcal{Z}^0$$

$$\mathcal{D}(V) = \text{Dom}(V)$$

- ops

$$\mathcal{D}(e_1 \text{ op } e_2) = \mathcal{D}(e_1) \cap \mathcal{D}(e_2)$$

- restriction

Domains of ALPHA Expressions

- consts, vars

$$\mathcal{D}(c) = \mathcal{Z}^0$$

$$\mathcal{D}(V) = \text{Dom}(V)$$

- ops

$$\mathcal{D}(e_1 \text{ op } e_2) = \mathcal{D}(e_1) \cap \mathcal{D}(e_2)$$

- restriction

$$\mathcal{D}(D : e) = D \cap \mathcal{D}(e)$$

Domains of ALPHA Expressions

- consts, vars

$$\mathcal{D}(c) = \mathcal{Z}^0$$

$$\mathcal{D}(V) = \text{Dom}(V)$$

- ops

$$\mathcal{D}(e_1 \text{ op } e_2) = \mathcal{D}(e_1) \cap \mathcal{D}(e_2)$$

- restriction

$$\mathcal{D}(D : e) = D \cap \mathcal{D}(e)$$

- case

Domains of ALPHA Expressions

- consts, vars

$$\mathcal{D}(c) = \mathcal{Z}^0$$

$$\mathcal{D}(V) = \text{Dom}(V)$$

- ops

$$\mathcal{D}(e_1 \text{ op } e_2) = \mathcal{D}(e_1) \cap \mathcal{D}(e_2)$$

- restriction

$$\mathcal{D}(D : e) = D \cap \mathcal{D}(e)$$

- case

$$\mathcal{D}(\text{case } e_1; \dots; e_n; \text{esac}) = \bigsqcup_{i=1}^n \mathcal{D}(e_i)$$

Affine Dependency

- Q: What does the expression $e.(z \rightarrow f(z))$ mean?

Affine Dependency

- Q: What does the expression $e.(z \rightarrow f(z))$ mean?
- A: An expression whose value at z is the same as the value of e at $f(z)$

Affine Dependency

- Q: What does the expression $e.(z \rightarrow f(z))$ mean?
- A: An expression whose value at z is the same as the value of e at $f(z)$, for example:

$$Y = X.(i, j \rightarrow i)$$

Affine Dependency

- Q: What does the expression $e.(z \rightarrow f(z))$ mean?
- A: An expression whose value at z is the same as the value of e at $f(z)$, for example:

$$Y = X.(i, j \rightarrow i) \text{ or } Y[i, j] = X[i]$$

Affine Dependency

- Q: What does the expression $e.(z \rightarrow f(z))$ mean?
- A: An expression whose value at z is the same as the value of e at $f(z)$, for example:

$$Y = X.(i, j \rightarrow i) \text{ or } Y[i, j] = X[i]$$

- Where is it defined?

Affine Dependency

- Q: What does the expression $e.(z \rightarrow f(z))$ mean?
- A: An expression whose value at z is the same as the value of e at $f(z)$, for example:

$$Y = X.(i, j \rightarrow i) \text{ or } Y[i, j] = X[i]$$

- Where is it defined?

$$\mathcal{D}(e.(z \rightarrow f(z))) = f^{-1}(\mathcal{D}(e))$$

Reductions

- Q: What does $\text{reduce}(\oplus, (z \rightarrow f(z)), e)$ mean (and where is it defined)?

Reductions

- Q: What does $\text{reduce}(\oplus, (z \rightarrow f(z)), e)$ mean (and where is it defined)?
- A1: Its **domain** is the **projection** of $\mathcal{D}(e)$, the domain of e , by f :

Reductions

- Q: What does $\text{reduce}(\oplus, (z \rightarrow f(z)), e)$ mean (and where is it defined)?
- A1: Its **domain** is the **projection** of $\mathcal{D}(e)$, the domain of e , by f :

$$\mathcal{D}[\text{reduce}(\oplus, (z \rightarrow f(z)), e)] = f(\mathcal{D}(e))$$

Reductions

- Q: What does $\text{reduce}(\oplus, (z \rightarrow f(z)), e)$ mean (and where is it defined)?
- A1: Its **domain** is the **projection** of $\mathcal{D}(e)$, the domain of e , by f :

$$\mathcal{D}[\text{reduce}(\oplus, (z \rightarrow f(z)), e)] = f(\mathcal{D}(e))$$

- A2: Its **value** at some z' in this domain is

Reductions

- Q: What does $\text{reduce}(\oplus, (z \rightarrow f(z)), e)$ mean (and where is it defined)?
- A1: Its **domain** is the **projection** of $\mathcal{D}(e)$, the domain of e , by f :

$$\mathcal{D}[\text{reduce}(\oplus, (z \rightarrow f(z)), e)] = f(\mathcal{D}(e))$$

- A2: Its **value** at some z' in this domain is the result of applying \oplus

Reductions

- Q: What does $\text{reduce}(\oplus, (z \rightarrow f(z)), e)$ mean (and where is it defined)?
- A1: Its **domain** is the **projection** of $\mathcal{D}(e)$, the domain of e , by f :

$$\mathcal{D}[\text{reduce}(\oplus, (z \rightarrow f(z)), e)] = f(\mathcal{D}(e))$$

- A2: Its **value** at some z' in this domain is the result of applying \oplus to the values of e

Reductions

- Q: What does $\text{reduce}(\oplus, (z \rightarrow f(z)), e)$ mean (and where is it defined)?
- A1: Its **domain** is the **projection** of $\mathcal{D}(e)$, the domain of e , by f :

$$\mathcal{D}[\text{reduce}(\oplus, (z \rightarrow f(z)), e)] = f(\mathcal{D}(e))$$

- A2: Its **value** at some z' in this domain is the result of applying \oplus to the values of e at all points that are mapped to z' by f

Outline

- Introduction
- ALPHA Syntax
- (Denotational) Semantics
- Substitution, Normalization, & . . .
- Change of Basis

Substitution

```
substituteInDef[Fib, Fib, 1]; show[]
```

```
system FibSys : {N | 1<=N} () returns(F : integer );
var      Fib : {i | 1 <= i <= N} of integer
let
Fib =
  case
  {i | i<=2} : 1.(i -> ) ;
  {i | i>=3} : case
                {i | i<=2}:1.(i->);
                {i | 3<=i}:Fib.(i->i-1) + Fib.(i->i-2);
              esac .(i -> i-1) +Fib.(i -> i-2) ;
  esac ;
F = Fib.( ->N) ;
tel ;
```

Normalization: syntax of a normal expression

- $\langle \text{BasicExpr} \rangle ::= \langle \text{Var} \rangle \mid \langle \text{Const} \rangle \mid \langle \text{Var} \rangle . \langle \text{Dep} \rangle \mid \langle \text{Const} \rangle . \langle \text{Dep} \rangle$

Normalization: syntax of a normal expression

- $\langle \text{BasicExpr} \rangle ::= \langle \text{Var} \rangle \mid \langle \text{Const} \rangle \mid \langle \text{Var} \rangle . \langle \text{Dep} \rangle \mid \langle \text{Const} \rangle . \langle \text{Dep} \rangle$
- $\langle \text{InnerExpr} \rangle ::= \langle \text{BasicExpr} \rangle \mid \langle \text{BasicExpr} \rangle \text{op} \langle \text{InnerExpr} \rangle$

Normalization: syntax of a normal expression

- $\langle \text{BasicExpr} \rangle ::= \langle \text{Var} \rangle \mid \langle \text{Const} \rangle \mid \langle \text{Var} \rangle . \langle \text{Dep} \rangle \mid \langle \text{Const} \rangle . \langle \text{Dep} \rangle$
- $\langle \text{InnerExpr} \rangle ::= \langle \text{BasicExpr} \rangle \mid \langle \text{BasicExpr} \rangle \text{op} \langle \text{InnerExpr} \rangle$
- $\langle \text{NormalExpr} \rangle ::=$
 - $\langle \text{InnerExpr} \rangle \mid \langle \text{Domain} \rangle : \langle \text{InnerExpr} \rangle \mid$
 - `case [$\langle \text{InnerExpr} \rangle$] * esac`
 - `case [$\langle \text{Domain} \rangle : \langle \text{InnerExpr} \rangle$] * esac`

Normalization Rules (subset)

No.	Replace	By	Comments
1	$e.f$	e	if $f(z) = z$
2	$(e_1 \oplus e_2).f$	$(e_1.f) \oplus (e_2.f)$	
3	$e \oplus$ case $e_1;$ \vdots $e_n;$ esac	case $e \oplus e_1;$ \vdots $e \oplus e_n;$ esac	

Some More Rules (subset)

No.	Replace	By	Comments
4	$(D : e_1) \oplus e_2$	$D : (e_1 \oplus e_2)$	
5	$e_1 \oplus (D : e_2)$	$D : (e_1 \oplus e_2)$	
6	$(e.f_1).f_2$	$e.f$	$f = f_1 \circ f_2$
7	$D_1 : (D_2 : e)$	$D : E$	$D = D_1 \cap D_2$
8	$(D : e).f$	$D' : (e.f)$	$D' = f^{-1}(D)$

Normalization, example

```

                                normalize[]; show[]

system FibSys : {N | 1<=N} () returns(F : integer );
var    Fib :    {i | 1 <= i <= N} of integer
let
  Fib =
    case
      {i | i<=2} : 1.(i -> ) ;
      {i | i=3}  : 1.(i ->) + Fib.(i -> i-2) ;
      {i | i>=4} : Fib.(i -> i-2) + Fib.(i -> i-3)
                    + Fib.(i -> i-2) ;
    esac ;
  F =  Fib.( ->N) ;
tel ;

```