High-Performance Embedded Systems-on-a-Chip Lecture 11: Alpha (contd)

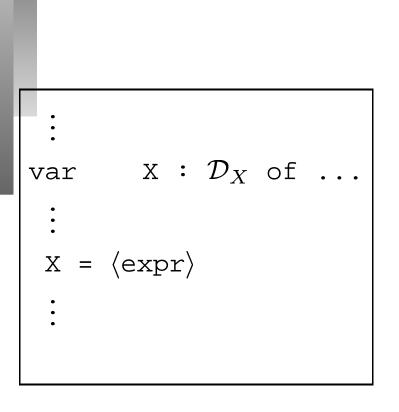
Sanjay Rajopadhye

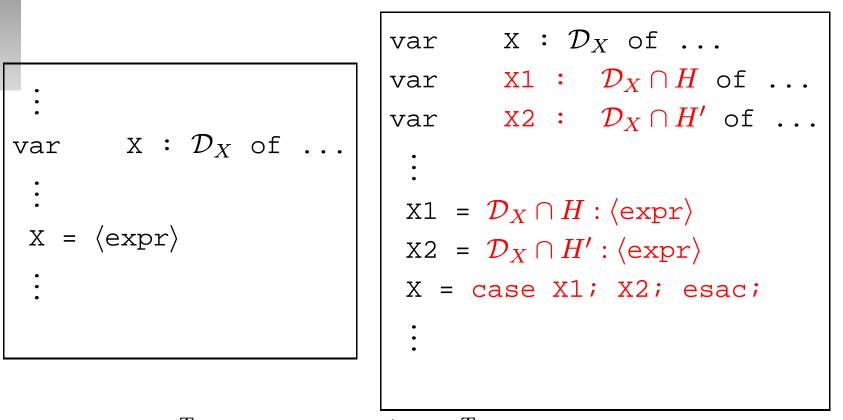
Computer Science, Colorado State University

Outline

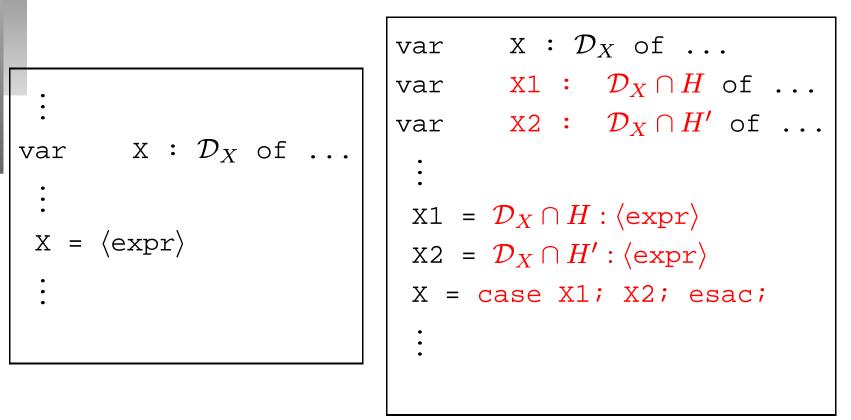
- Other Transformations:
 - Cut
 - Merge
 - Analyze
 - Simplify
 - AddLocal
- Array Notation
- Change of Basis

"Cut" the domain of X with $a^T z = a_0$





where $H \equiv a^T z \ge a_0$, and $H' \equiv a^T z < a_0$ are the two halfspaces defined by $a^T z = a_0$.



where $H \equiv a^T z \ge a_0$, and $H' \equiv a^T z < a_0$ are the two halfspaces defined by $a^T z = a_0$. Next, substituteInDef for **all uses** of X, and then eliminate X

Still More Transformations

- Cut
- Merge
- Analyze
- Simplify
- AddLocal

Array Notation

In general, a normalized equation has the form:

Array Notation

In general, a normalized equation has the form:

The number of index variables in

- any $\langle \text{Dep} \rangle$ to the left of the ightarrow
- any (Domain) to the left of the |
- and in the (Domain) of X

Array Notation

In general, a normalized equation has the form:

The number of index variables in

- any $\langle \text{Dep} \rangle$ to the left of the ightarrow
- any (Domain) to the left of the |
- and in the $\langle \text{Domain} \rangle$ of X

are all equal

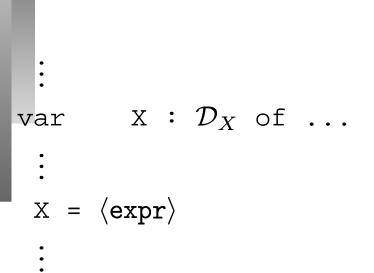
- rename all indices to be the same
- move them to the left of the equation
- drop them from the rhs (wherever they occur)

- rename all indices to be the same
- move them to the left of the equation
- drop them from the rhs (wherever they occur)
- Add (syntactic) sugar, shake well and serve!

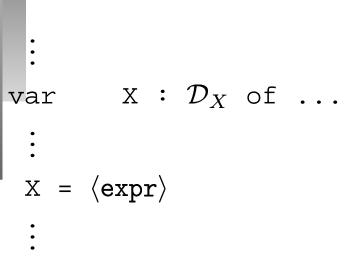
Outline

- Introduction
- ALPHA Syntax ALPHA
- (Denotational) Semantics
- Substitution, Normalization, & ...
- Change of Basis (oh no, not again)

Consider the ALPHA program



Consider the ALPHA program



and affine functions \mathcal{T}' and \mathcal{T} such that $\mathcal{T}'\circ\mathcal{T}=\mathrm{I},$ i.e., $\mathcal{T}'(\mathcal{T}(z))=z$

- Every occurrence of X (on the rhs of any equation) can be replaced by X.T'.T, without affecting the semantics.
- Introduce a new variable

$$\mathtt{X}' = \mathtt{X}.\mathcal{T}' = \langle \mathtt{expr}
angle.\mathcal{T}'$$

- Its domain is $\operatorname{Pre}(\mathcal{D}_X, \mathcal{T}')$
- Replace every occurrence of the subexpression X.T' in the program by X'
- Since X is no longer used in the program, drop it, and then rename the X' to be X

Replace

- the domain \mathcal{D}_X of x by $\mathsf{Pre}(\mathcal{D}_X, \mathcal{T})$
- all occurrences of \underline{x} (on any rhs) by $\underline{x}.\mathcal{T}$
- compose a *T*['] dependence at the end of the entire rhs of the equation of x.

changeOfBasis["Fib.(i -> -i)", "i"]; show[]

```
changeOfBasis["Fib.(i -> -i)", "i"]; show[]
system FibSys : \{N \mid 1 \le N\} ()
returns (F : integer);
var Fib : {i -N<=i<=-1} of integer;
let
Fib = case
         {i|i<=2} : 1.(i->);
         {i|i>=3} : Fib.(i->-i).(i->i-1)
                   + Fib.(i->-i).(i->i-2);
         esac.(i->-i);
F = Fib.(i - > -i).(->N);
tel;
```

Normalize again

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

- \mathcal{T} is unimodular (simplest case)
- \mathcal{T} is not square (but admits an integral left inverse),

- \mathcal{T} is unimodular (simplest case)
- *T* is not square (but admits an integral left inverse), eg.
 alignment (i → i, i)

- T is unimodular (simplest case)
- *T* is not square (but admits an integral left inverse), eg. alignment (i → i, i)
- Can you do even better? What about a transformation $(\mathtt{i}, \mathtt{j} \rightarrow \mathtt{i}, \mathtt{0})$

- T is unimodular (simplest case)
- *T* is not square (but admits an integral left inverse), eg.
 alignment (i → i, i)
- Can you do even better? What about a transformation

 (i, j → i, 0) applied to a variable whose domain is
 really a line-segment, "embedded" in a 2-D index
 space

- \mathcal{T} is unimodular (simplest case)
- *T* is not square (but admits an integral left inverse), eg.
 alignment (i → i, i)
- Can you do even better? What about a transformation

 (i, j → i, 0) applied to a variable whose domain is
 really a line-segment, "embedded" in a 2-D index
 space T must admit an integral left inverse, T' in the
 context of the variable X, i.e.,

$$\forall z \in \mathcal{D}_X, \ \mathcal{T}'(\mathcal{T}(z)) = z$$

- ALPHA domains (finite unions of polyhedra) constitute an abstract data type (ADT), closed under:
 - Intersection
 - Union (of finitely many members of the ADT)
 - Preimage by (arbitrary) affine functions (the class of dependences in ALPHA).
 - Image by unimodular affine functions (the class of transformations used for changes of bases).
- The class of transformations (unimodular affine functions) is a subset of the class of dependences
- the class of dependences is closed under composition