Array Iterators

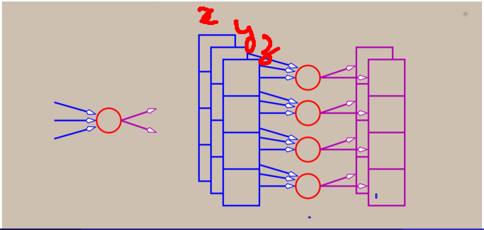
Objective: To manipulate arrays iteratively.

- Given vectors $A : int^n$ and $B : int^n$, add them up to give vector $C : int^n$. Use map.
- Given array $A : int^n$, find sum of its elements. Use fold.
  We can also compute $\Sigma(A[i]^2)$ and use this to find standard deviation.
- Mapfold combines the map and the fold.

Map

- Example: Adding two 3-dimentional vectors `a,b:real^3` to get `c:real^3`.
  Method: Use $+$ pointwise on every index putting the result in the output array.
  [4,9,-10]
- `c = map<<3>>(+) ([1,3,5],[4,3,-2])` gives `[5,6,3]`
- In general `map<<n>>(F)` $(x_1, ..., x_m)$ returns $(y_1, ..., y_k)$
  Here $F : (t_1 \times \ldots \times t_m) \rightarrow (t'_1 \times \ldots \times t'_k)$.
  Also, $x_i : t_i\hat{}n$ for $1 \leq i \leq m$ and $y_j : t'_j\hat{}n$ for $1 \leq j \leq k$.
- Expression `map<<n>>(F)` has type
  $(t_1\hat{}n \times \ldots \times t_m\hat{}n) \rightarrow (t'_1\hat{}n \times \ldots \times t'_k\hat{}n)$.

map<<4>>(F)
(x, y, z)
→ (u, v)

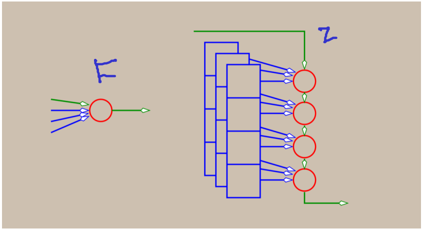


Here
m = 3 k = 2
n = 4

Fold

- Example: Finding sum of array of 4 elements `a:int^4` to get `c:int`.
  Method: Use + pointwise on every index accumulating the sum.
- `c = fold<<4>>(+) ([1,3,5,7],0)` gives 16
- In general `fold<<n>>(F)` $(x_1, \ldots, x_m, z)$ returns y
  Here $F : (t_1 \times \ldots \times t_m \times t) \rightarrow t$.
  Also, $x_i : t_i\hat{}n$ for $1 \leq i \leq m$ and $z, y : t$.
- Expression `fold<<n>>(F)` has type $(t_1\hat{}n \times \ldots \times t_m\hat{}n \times t) \rightarrow (t)$.

$m=3$



0
1 → + → 1
3 → + → 4
5 → + → 9
7 → + → 16

Example

a[0]*b[0] + a[1]*b[1] + ... + a[n-1]*b[n-1]

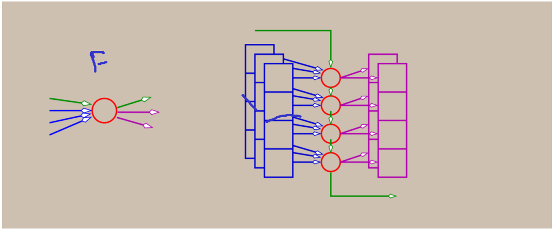
z[0] z[1]

Find dot product of two *n*-dimentional vectors.

```
node dotproduct<<n:int>>(a:real^n; b:real^n) returns (c:real)
var z:real^n
let
   z = map<<n>>(*)(a,b);
   c = fold<<n>>(+)(z,0);
tel
```

Mapfold

- Example: Adding two 3-dimentional vectors `a,b:real^3` AND getting their dot-product `c:real^3; dot:real`
- ```
  node F(x,y,accin:real) returns (z,accout:real)
  let z=x+y; accout = accin + (x*y); tel
  ```
- `c = mapfold<<3>>(F) ([1,3,5],[4,3,-2],0)` gives `[5,6,3], ~~1~~3`
- In general `mapfold<<n>>(F)` $(x_1, ..., x_m, init)$ returns $(y_1, ..., y_k, acc)$
  Here $F : (t_1 \times \ldots \times t_m \times t) \to t'_1 \times \ldots \times t'_k \times t$.
  Also, $x_i : t_i\hat{\ }n$ for $1 \le i \le m$ and $y_j : t'_j\hat{\ }n$ for $1 \le j \le k$ with $init, acc : t$.



indx

0 F(1,4,0)
=5,4

1 F(3,3,4)
=6,13

2 =F(5,-2,13)
=3,3
  ```

Summary

- Features for writing large and complex programs.
- Records, Arrays, Array slices, Global types and constants, Parameterized nodes.
- Array iterators: map, fold and mapfold.