Target Mapping

To generate scheduled code from an Alphabets program, the code generator in AlphaZ needs some information about (i) the schedule, the (ii) processor allocation, (iii) the memory allocation and (iv) a tiling. Some of these are optional. The target mapping is a unified manner to specify these. This tutorial will cover how to specify the first three: the Space-Time mapping (i and ii) and the Memory mapping (iii).

Note that in AlphaZ the code generation is distinct from the modules that choose the

Space-Time Mapping

For each variable in the system, space-time map is the transformation which specifies ***Schedule*** and ***Processor Allocation***. ***Statement ordering*** can be specified explicitly (using an API) or as extra dimensions in the space-time map. If the statement ordering is included in space-time map, then its required that it should be same(dimension) accross all the variables.

Usage

For the Alphabets program for matrix product.

```
affine matrix_product {P, Q, R|P>0 && Q>0 && R>0}
       input  float A {i,k| 0<=i<P && 0<=k<Q};
              float B {k,j| 0<=k<Q && 0<=j<R};
       output float C {i,j,k| 0<=i<P && 0<=j<R && k==Q};
local
   float temp_C {i,j,k|0<=i<P && 0<=j<R && 0<=k<=Q};
let
   temp_C[i,j,k] = case
                      {|k>0}  : temp_C[i,j,k-1] + A[i,k-1]*B[k-1,j];
                      {|k==0} : 0;
                   esac;
   C      = temp_C;
.
```

Each variable in the affine system can be given a space-time map.

```
 setSpaceTimeMap( Program program, String system, String var,  String stMap)
```

An example for space-time mapping for the above program.

```
setSpaceTimeMap(prog, system, "temp_C",  "(i,j,k->i,j,k)");
setSpaceTimeMap(prog, system, "C",  "(i,j,k->i,j,k+1)");
```

since we admit interleaved schedules and processor allocation, each dimension in the space-time map can be set as “sequential” “parallel” or “ordering”. By default, every dimension is set as

sequential, but we can specify the parallel dimension using

```
 setParallel(  Program program , String system , String orderingPrefix,
String dims )
```

This command sets the dimensions (dims) with ordering prefix “orderingPrefix” to be parallel. We can also specify the odering dimension using

```
 setOrderingDimensions(Program program, String system, String dims)
```

Space-Time mapping with statement ordering and “outer parallel” for the above matrix product.

```
setSpaceTimeMap(prog, system, "temp_C",  "(i,j,k->i,j,k)");
setSpaceTimeMap(prog, system, "C",  "(i,j,k->i,j,k)");
# Set up the first and second dimensions (dimensions start with 0) with no
ordering prefix to be parallel
setParallel(prog, system, "", "0,1");

setStatementOrdering(prog, system, "temp_C", "C");
```

```
# This is an alternative space-time map for the matrix multiplication with
ordering dimensions
setSpaceTimeMap(prog, system, "temp_C",  "(i,j,k->0,i,j,0,k)");
setSpaceTimeMap(prog, system, "C",  "(i,j,k->0,i,j,1,k)");

# Set the first and forth dimension to be ordering dimension (dimension
starts with 0)
setOrderingDimensions(prog, system, "0,3");
# Set the first dimension with ordering prefix 0 to be parallel (the i
dimension)
setParallel(prog, system, "0", "0");
```

Memory Mapping

Memory map is an affine function mapping iteration points in the domain the variables in the affine system to memory location. By default each variable will have an identity function as memory map. To set memory map for a variable, *setMemoryMap* command should be used.

```
 setMemoryMap( Program program, String system, String var, String
memorySpace,  String memoryMap)
```