Change of Basis

Change of basis is a useful transformation in a lot of aspect. This page shows how to use command CoB to apply change of basis on a program.

Usage

Here is the usage for CoB:

```
CoB(Program program, String systemName, String varName, String function)
```

The meaning for each parameters are:

- program: the program you are processing.
- systemName: the name for the system you want to solve.
- varName: the name for the variable you want to apply change of basis on.
- function: change of basis function.

Example

The following example does a matrix multiplication, C = A*B.

```
affine matrix_product {P, Q, R|P>0 && Q>0 && R>0}
given  float A {i,k| 0<=i<P && 0<=k<Q};
       float B {k,j| 0<=k<Q && 0<=j<R};
returns  float C {i,j,k| 0<=i<P && 0<=j<R && k==Q};
using // Using an accumulator locally
   float temp_C {i,j,k|0<=i<P && 0<=j<R && 0<=k<=Q};
through
temp_C = case
         {i,j,k|k-1>= 0} : ((i,j,k->i,j,k-1)@temp_C + ((i,j,k->i,k-1)@A *
(i,j,k->k-1,j)@B));
         {i,j,k|k== 0} : (i,j,k->)@0;
         esac;
C = (i,j,k->i,j,k)@temp_C;
.
```

Now we are going to apply change of basis on variable “temp_C” using function (i,j,k→i,j,k+1). We can apply the transformation with the following code:

```
prog = ReadAlphabets("matrix_product.ab");
system = "matrix_product";
CoB(prog, system, "temp_C", "(i,j,k->i,j,k+1)");
```

The result for this transformation is shown below:

```
affine matrix_product {P, Q, R|P>0 && Q>0 && R>0}
given  float A {i,k| 0<=i<P && 0<=k<Q};
       float B {k,j| 0<=k<Q && 0<=j<R};
returns  float C {i,j,k| 0<=i<P && 0<=j<R && k==Q};
using // Using an accumulator locally
   float temp_C {i,j,k|0<=i<P && 0<=j<R && 1<=k<=Q+1};
through
temp_C = (i,j,k->i,j,k-1)@case
                          {i,j,k|k-1>= 0} :
((i,j,k->i,j,k-1)@(i,j,k->i,j,k+1)@temp_C + ((i,j,k->i,k-1)@A *
(i,j,k->k-1,j)@B));
                          {i,j,k|k== 0} : (i,j,k->)@0;
                          esac;
C = (i,j,k->i,j,k)@(i,j,k->i,j,k+1)@temp_C;
.
```

We applied the change of basis function (i,j,k→i,j,k+1) on variable “temp_C”, which means we shifted the domain of “temp_C” along k direction by one unit. We can see that the domain for “temp_C” is changed from:

```
float temp_C {i,j,k|0<=i<P && 0<=j<R && 0<=k<=Q};
```

to

```
float temp_C {i,j,k|0<=i<P && 0<=j<R && 1<=k<=Q+1};
```

And all the access functions for temp_C is composed with function (i,j,k→i,j,k+1). For example, the access function in the computation of C is changed from

```
C = (i,j,k->i,j,k)@temp_C;
```

to

```
C = (i,j,k->i,j,k)@(i,j,k->i,j,k+1)@temp_C;
```