SubSystem in Alpha

In this tutorial, we will present how to write structured alpha programs with subsystems, and we will present the associated transformations.

Syntax of Use Equation (without extension domain)

Let us assume that we want to compute the mean of the values of a vector. It is feasible through the following Alpha system:

```
affine mean {N | N>0}
input
    float A {k | 0<=k<N};
output
    float C {|};
local
    float temp {|};
let
    temp = reduce(+, [k], A[k]);
    C = temp / N;
.
```

However, let us assume that you already have another Alpha system which computes the sum of the elements of a vector. It is possible to use this affine system (instead of rewriting its equation in the main system), by calling it through a “use equation”:

```
affine sum {P| P>0}  // Computes the sum of the elements of a vector of size
P
input
    float vect {i | 0<=i<P };
output
    float Res;
let
    Res = reduce(+, [k], vect[k]);
.

affine mean {N | N>0}
input
    float A {k | 0<=k<N};
output
    float C {|};
local
    float temp {|};
let
    use sum[N] (A) returns (temp);        // Compute "temp" using the system
```

```
"sum"
    C = temp / N;
.
```

The system “mean” is calling the system “sum” (which is called a subsystem). The subsystem is called with the parameter “N” and the input “A”. After doing its computation, the result of “sum” will be stored inside the local variable “temp”.

In general, the syntax of a use equation is the following:

```
use subsystem_name[list of parameters] (list of input expressions) returns
(list of output variables);
```

If your subsystem have several parameters/inputs/outputs, you have to provide them in the order in which they are declared.

Extension domain

(incoming)

Transformations involving subsystems

(incoming, after the previous incoming)