Normalize

Sometimes, the expressions in an alphabets program is not written in the normal form as it is defined, which makes the program hard to read and understand. Nomalize is used to normalize the expressions in a program to the defined normal form. This page shows how to use Normalize to normalize a program.

Usage

The usage for the command Normalize is

```
Normalize(Program program);
```

The parameter program is the program you want to solve.

Normalization Rules

Normalize takes a program and applies a set of normalization rules on it. Some of the basic rules are shown below:

1. <latex> $e.f \Rightarrow e$, if $f(z)=z$</latex>
2. <latex> $(e_{1} \oplus e_{2}).f \Rightarrow (e_{1}.f) \oplus (e_{2}.f)$ </latex>
3. <latex> $(D:e_{1})\oplus e_{2} \Rightarrow D:(e_{1} \oplus e_{2})$</latex>
4. <latex> $e_{1}\oplus (D: e_{2}) \Rightarrow D:(e_{2} \oplus e_{2})$</latex>
5. <latex> $(e\dot f_{1}). f_{2} \Rightarrow e . f$, where $f = f_{1} o f_{2}$</latex>
6. <latex> $D_{1}:(D_{2}:e) \Rightarrow D:e$, where $D=D_{1} \cap D_{2}$ </latex>
7. <latex> $(D:e). f \Rightarrow D':e$, where $D' = f^{-1}(D)$</latex>

Example

Here we present an example and explain how the normalization rules work.

```
affine RestrictExpr {N | N > 1}
given
    int A {i | 0 <= i < 3N};
returns
    int C {i | 0 <= i < N};
    int D {i, j| 0 <= i < N && 0 <= j < N};
through
    C[i] = {i| 0 <= i < N}: ({i|0 <= i < 2N}:A[i]);
    D[i,j] = (i, j -> j, i)@({i,j|0 <= i < N}:A[i]);
.
```

In the above example, C simply copies the first N values of A, and D[i,j] = A[i]. Based on analysis, the normalization can be applied with the following code:

```
prog = ReadAlphabets("IdentityFunc.ab");
Normalize(prog);
```

The above code reads the program “IdentityFunc.ab” using command ReadAlphabets and applies normalization on the program. The following normalization rules can be applied to the program:

- In the computation for C, expression A[i] is equivalent to A.(i -> i), f is an identity function, rule number one is satisfied. So A.(i -> i) => A.
- In the computation for C, expression {i| 0 <= i < N}: ({i|0 <= i < 2N}:A[i]) matches rule number 6, where D1 = {i| 0 <= i < N}, D2={i|0 <= i < 2N}. D=D1 ∩ D1 ={i|0 <= i < N}, the expression is changed to {i|0 <= i < N}:A.
- In the computation for D, expression (i, j -> j, i)@({|0 <= i < N\}:A[i]) matches rule number 7, where D = {i,j |0 <= i < N }, f=(i, j -> j, i). D'=f^(-1)(D)={i,j|0 <= j < N}, the expression is changed to {i,j|0 <= j < N}:A[i,j];

The result for the above program after normalization is:

```
affine RestrictExpr {N | N > 1}
given
    int A {i | 0 <= i < 3N};
returns
    int C {i | 0 <= i < N};
    int D {i, j| 0 <= i < N && 0 <= j < N};
through
C = {i| 0 <= i < N}:A;
D = {i,j| 0 <= j < N} : A[i,j];
```

More Example

Here we give an example about Fibonacci:

```
affine Fib {N | N > 1}
given
returns
    int f;
using
    int fib {i | i >= 0 && -i >= -N};
through
    fib =case
         {i |i <= 1}: 1;
         {i |i >= 2}: (i -> i-1)@(case
                    {i |i <= 1}: 1;
                    {i |i >= 2}: (i -> i-1)@fib + (i -> i-2)@fib;
                     esac)
                  + (i -> i-2)@(case
                    {i |i <= 1}: 1;
                    {i |i >= 2}: (i -> i-1)@fib + (i -> i-2)@fib;
                           esac);
         esac;
```

```
    f = (->N)@fib;
.
```

In this example, the variable fib in the definition is substituted by its definition, which makes the program looks complicated. We do normalization on the above program with the following commands:

```
prog = ReadAlphabets("Fib.ab");
Normalize(prog);
```

The result program after normalization is

```
affine Fib {N | N > 1}
given
returns
    int f;
using
    int fib {i | i >= 0 && -i >= -N};
through
fib = case
      {i|-i+1>= 0} : (i->)@1;
      {i|i-2== 0} : ((i->)@1 + (i->)@1);
      {i|i-3== 0} : (((i->i-2)@fib + (i->i-3)@fib) + (i->)@1);
      {i|i-4>= 0} : (((i->i-2)@fib + (i->i-3)@fib) + ((i->i-3)@fib +
(i->i-4)@fib));
      esac;
f = (->N)@fib;
.
```

It is clear that the normalized program is much easier to read and understand.