lambda and VERTSECT

Johan G. F. Belinfante
2004 January 7

```
In[1]:= << goedel53.06b; << tools.m

        :Package Title: goedel53.06b      2004 January 6 at 2:55 p.m.

        It is now:  2004 Jan 8 at 13:5

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 January 3

        weightlimit = 40
```

introduction

The function corresponding to a unary functor **F[x]** is **lambda[x, F[x]]**. In general, it is faster to use the equivalent construction **VERTSECT[reify[x, F[x]]]**. Compare, for example:

```
In[2]:= lambda[x, rotate[x]] // Timing

Out[2]= {6.88 Second, IMAGE[ROT]}

In[3]:= VERTSECT[reify[x, rotate[x]]] // Timing

Out[3]= {0.06 Second, IMAGE[ROT]}
```

Since **VERTSECT[x]** is always a proper class when **x** is a set, the function corresponding to this functor is empty. To remedy this, it is useful to work with the function **VS** that corresponds to the restriction of **VERTSECT[x]** to the domain of **x**.

```
In[4]:= image[VS, singleton[x]]

Out[4]= intersection[image[V, singleton[x]], singleton[composite[VERTSECT[x], id[domain[x]]]]]
```

Currently neither **lambda[x, composite[VERTSECT[x], id[domain[x]]]** or the reification counterpart currently yields a simple expression. In this notebook, new rewrite rules are derived to permit the simulated **lambda** expression to simplify to **VS**.

simulated lambda rules

The expression curently obtained by applying **reify** involves the rotation−invariant function **RIF**.

```
In[5]:= reify[x, composite[VERTSECT[x], id[domain[x]]]]

Out[5]= union[composite[inverse[E], COMPOSE, intersection[composite[inverse[FIRST], VS],
           composite[inverse[SECOND], IMAGE[DUP], IMAGE[FIRST]]]], composite[SWAP, RIF,
          intersection[composite[inverse[FIRST], RIGHT[0], complement[inverse[E]]],
           composite[inverse[SECOND], inverse[E], IMAGE[DUP]]], IMAGE[FIRST]]]
```

The following new rewrite rule eliminates **RIF**.

```
In[6]:= composite[RIF, intersection[composite[inverse[FIRST], x],
           composite[inverse[SECOND], DUP, y]]] // VSNormality

Out[6]= composite[RIF,
          intersection[composite[inverse[FIRST], x], composite[inverse[SECOND], DUP, y]]] ==
         composite[SWAP, intersection[x, composite[inverse[FIRST], y]]]

In[7]:= composite[RIF,
          intersection[composite[inverse[FIRST], x_], composite[inverse[SECOND], DUP, y_]]] :=
         composite[SWAP, intersection[x, composite[inverse[FIRST], y]]]
```

For the current application, one actually needs a slightly different, but related rule:

```
In[8]:= composite[RIF, intersection[composite[inverse[FIRST], x],
           composite[inverse[SECOND], inverse[E], IMAGE[DUP]]]] // VSNormality

Out[8]= composite[RIF, intersection[composite[inverse[FIRST], x],
           composite[inverse[SECOND], inverse[E], IMAGE[DUP]]]] ==
         composite[SWAP, intersection[x, composite[inverse[FIRST], inverse[E]]]]

In[9]:= composite[RIF, intersection[composite[inverse[FIRST], x_],
           composite[inverse[SECOND], inverse[E], IMAGE[DUP]]]] :=
         composite[SWAP, intersection[x, composite[inverse[FIRST], inverse[E]]]]
```

At this stage, one finds:

```
In[10]:= reify[x, composite[VERTSECT[x], id[domain[x]]]]

Out[10]= union[composite[intersection[composite[inverse[FIRST], inverse[E]],
            composite[RIGHT[0], complement[inverse[E]]]], IMAGE[FIRST]],
          composite[inverse[E], COMPOSE, intersection[composite[inverse[FIRST], VS],
            composite[inverse[SECOND], IMAGE[DUP], IMAGE[FIRST]]]]]
```

The next step is to identify a part of this as a fancy way to write the empty set. For this, the following general rule can be used:

```
In[11]:= intersection[composite[inverse[FIRST], x], composite[RIGHT[y], z]] // RelnNormality

Out[11]= intersection[composite[inverse[FIRST], x], composite[RIGHT[y], z]] ==
          composite[RIGHT[y], intersection[x, z]]

In[12]:= intersection[composite[inverse[FIRST], x_], composite[RIGHT[y_], z_]] :=
          composite[RIGHT[y], intersection[x, z]]
```

This further simplifies the **reify** expession to the following:

```
In[13]:= reify[x, composite[VERTSECT[x], id[domain[x]]]]

Out[13]= composite[inverse[E], COMPOSE, intersection[composite[inverse[FIRST], VS],
           composite[inverse[SECOND], IMAGE[DUP], IMAGE[FIRST]]]]
```

The final step needed to simplify this is a rule that says that restricting the restriction **composite[VERTSECT[x], id[domain[x]]]** to **domain[x]** cuts out nothing. This rule can be derived as follows:

```
In[14]:= Map[VERTSECT,
          SubstTest[reify, x, composite[y, VERTSECT[x]], y -> id[complement[singleton[0]]]]]

Out[14]= composite[COMPOSE, intersection[composite[inverse[FIRST], VS],
            composite[inverse[SECOND], IMAGE[DUP], IMAGE[FIRST]]]] == VS

In[15]:= composite[COMPOSE, intersection[composite[inverse[FIRST], VS],
            composite[inverse[SECOND], IMAGE[DUP], IMAGE[FIRST]]]] := VS
```

This achieves our goal of simplifying the simulated **lambda** expression.

```
In[16]:= VERTSECT[reify[x, composite[VERTSECT[x], id[domain[x]]]]]

Out[16]= VS
```

lambda for the unrestricted VERTSECT functor

The function corresponding to **VERTSECT[x]** is empty, just as for the related function **IMAGE[x]**, to which it is related. Again, some new rules are needed to see that this is the case. At this stage one finds:

```
In[17]:= VERTSECT[reify[x, VERTSECT[x]]]

Out[17]= composite[CUP, intersection[composite[inverse[FIRST], VS], composite[inverse[SECOND],
            VERTSECT[composite[RIGHT[0], complement[inverse[E]]]], IMAGE[FIRST]]]]
```

Part of this expression is the empty set in disguise:

```
In[18]:= VERTSECT[composite[RIGHT[0], complement[inverse[E]]]] // RelnNormality

Out[18]= VERTSECT[composite[RIGHT[0], complement[inverse[E]]]] == 0

In[19]:= VERTSECT[composite[RIGHT[0], complement[inverse[E]]]] := 0
```

With this rule in place, one does find the expected result:

```
In[20]:= VERTSECT[reify[x, VERTSECT[x]]]

Out[20]= 0
```

appendix

The rule used above to simplify an expression involving **RIGHT** has a **LEFT** counterpart. No application for this is contemplated, but it will also be added as a permanent new rule.

```
In[21]:= intersection[composite[inverse[SECOND], x], composite[LEFT[y], z]] // RelnNormality

Out[21]= intersection[composite[inverse[SECOND], x], composite[LEFT[y], z]] ==
          composite[LEFT[y], intersection[x, z]]
```

```
In[22]:= intersection[composite[inverse[SECOND], x_], composite[LEFT[y_], z_]] :=
          composite[LEFT[y], intersection[x, z]]
```

comment

In general **VERTSECT[reify[x, f[x]]]** works faster and gives better results than does **lambda[x, f[x]]** for unary functors. For binary functors, **lambda[pair[x,y], f[x,y]]** can be replaced with the equivalent construction **composite[-VERTSECT[reify[x, f[first[x], second[x]]]], id[cart[V, V]]]**. For example:

```
In[25]:= lambda[pair[x, y], composite[x, y]] // Timing

Out[25]= {14.55 Second, COMPOSE}

In[26]:= composite[VERTSECT[reify[x, composite[first[x], second[x]]]], id[cart[V, V]]] //
          Timing

Out[26]= {0.48 Second, COMPOSE}
```