restrictions of rotate[E]

Johan G. F. Belinfante
2010 January 14

```
In[1]:= SetDirectory["l:"]; << goedel.10jan13a;<< tools.m
```

```
:Package Title: goedel.10jan13a              2010 January 13 at 1:20 p.m.

It is now:  2010 Jan 14 at 9:55

Loading Simplification Rules

TOOLS.M                                    Revised 2010 January 7

weightlimit = 40
```

summary

The rotated membership relation **rotate[E]** itself is not a function, but some of its restrictions are:

```
In[31]:= FUNCTION[composite[rotate[E], id[cart[x, V]]]]
Out[31]= subclass[image[IMAGE[id[cart[V, V]]], x], FUNS]
```

In this notebook, a conditional rewrite rule is derived that automatically transforms certain restrictions of **rotate[E]** into expressions that are more easily recognized to be functions.

a general simplification rule

Theorem. Simplification rule for restrictions of **rotate[x]**.

```
In[7]:= Assoc[rotate[x], id[cart[V, V]], id[y]]
Out[7]= composite[rotate[x], id[composite[Id, y]]] == composite[rotate[x], id[y]]

In[8]:= composite[rotate[x_], id[composite[Id, y_]]] := composite[rotate[x], id[y]]
```

restrictions of funpart[rotate[E]]

In this section some rewrite rules are derived for the function part of the rotated membership relation **E**.

```
In[9]:= funpart[rotate[E]]
Out[9]= composite[inverse[SINGLETON], IMG, cross[Id, SINGLETON]]
```

Theorem. A simplification rule.

```
In[10]:= Assoc[rotate[E], cross[FUNPART, Id], cross[id[FUNS], Id]]

Out[10]= composite[rotate[E], id[cart[FUNS, V]]] ==
          composite[inverse[SINGLETON], IMG, cross[id[FUNS], SINGLETON]]

In[11]:= % /. Equal → SetDelayed
```

Temporary Lemma.

```
In[12]:= Assoc[rotate[E], id[cart[FUNS, V]], id[x]]

Out[12]= composite[rotate[E], id[composite[x, id[FUNS]]]] ==
          composite[inverse[SINGLETON], IMG, cross[id[FUNS], SINGLETON], id[x]]

In[13]:= composite[rotate[E], id[composite[x_, id[FUNS]]]] :=
          composite[inverse[SINGLETON], IMG, cross[id[FUNS], SINGLETON], id[x]]
```

Theorem. When **domain[x] ⊂ FUNS**, there is an explicit formula for **rotate[E] ∘ id[x]** which is easily recognized to be a function.

```
In[14]:= SubstTest[implies, equal[u, v], equal[composite[t, u], composite[t, v]],
           {t → rotate[E], u → id[composite[Id, x]], v → id[composite[x, id[FUNS]]]}] // Reverse

Out[14]= or[equal[composite[rotate[E], id[x]],
             composite[inverse[SINGLETON], IMG, cross[id[FUNS], SINGLETON], id[x]]],
           not[subclass[domain[x], FUNS]]] == True

In[15]:= or[equal[composite[rotate[E], id[x_]],
              composite[inverse[SINGLETON], IMG, cross[id[FUNS], SINGLETON], id[x_]]],
            not[subclass[domain[x_], FUNS]]] := True
```

Theorem. The above formula can be made into a conditional rewrite rule.

```
In[16]:= implies[subclass[domain[x], FUNS], equal[composite[rotate[E], id[x]],
           composite[inverse[SINGLETON], IMG, cross[id[FUNS], SINGLETON], id[x]]]]

Out[16]= True

In[17]:= composite[rotate[E], id[x_]] :=
          composite[inverse[SINGLETON], IMG, cross[id[FUNS], SINGLETON], id[x]] /;
           subclass[domain[x], FUNS]
```

An application:

```
In[18]:= rotate[inverse[reify[x, binop[x]]]]

Out[18]= composite[inverse[SINGLETON], IMG, cross[id[BINOPS], SINGLETON]]
```

Corollary. If **domain[x] ⊂ FUNS**, then **rotate[E] ∘ id[x]** is a function.

In[19]:=
```
Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
   {p1 -> subclass[domain[x], FUNS], p2 -> equal[composite[rotate[E], id[x]],
     composite[inverse[SINGLETON], IMG, cross[id[FUNS], SINGLETON], id[x]]],
    p3 → FUNCTION[composite[rotate[E], id[x]]]}]] // Reverse
```

Out[19]=
```
or[FUNCTION[composite[rotate[E], id[x]]], not[subclass[domain[x], FUNS]]] == True
```

In[20]:=
```
or[FUNCTION[composite[rotate[E], id[x_]]], not[subclass[domain[x_], FUNS]]] := True
```