unions of commuting transitive relations

Johan G. F. Belinfante and Lee Martie
2006 June 27

```
In[1]:= SetDirectory["l:"]; << goedel82.22a; << tools.m;

        :Package Title: goedel82.22a         2006 June 22 at 10:30 p.m.

        It is now:  2006 Jun 27 at 15:55

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 June 21

        weightlimit = 40
```

summary

If **x** and **y** are transitive relations which commute, then **union[x, y, composite[x, y]]** is transitive.

a looping problem

The first attempts toward deriving the theorem in question were unsuccessful due to a looping problem, which needs to be addressed. The following two rules, taken together, cause looping for the input **subclass[composite[trv[x],trv[x],y], composite[trv[x],y]]**.

```
In[2]:= subclass[composite[trv[x], y], y]

Out[2]= subclass[composite[x, y], y]

In[3]:= composite[x, trv[x]]

Out[3]= composite[trv[x], trv[x]]
```

To derive a new rule to fix this problem, the first rule is temporarily removed, and will later be restored.

```
In[4]:= subclass[composite[trv[x_], y_], y_] =.
```

The following rule does not prevent the looping problem, and will be removed:

```
In[5]:= subclass[composite[u___, trv[x_], trv[x_], v___], composite[u___, trv[x_], v___]] =.
```

The removed rule is not really needed because conditional rewrite rules suffice:

```
In[6]:= subclass[composite[u, trv[x], trv[x], v], composite[u, trv[x], v]]
Out[6]= True
```

The looping problem only surfaces when **u** is absent. The conditional rules appear to cover this case:

```
In[7]:= subclass[composite[trv[x], trv[x], y], composite[trv[x], y]]
Out[7]= True
```

Despite the appearance it is not needed, the following rule will be crucial when the removed rule is restored:

```
In[8]:= subclass[composite[trv[x_], trv[x_], y_], composite[trv[x_], y_]] := True
```

A similar rule is needed with the factors reversed:

```
In[9]:= subclass[composite[x, trv[y], trv[y]], composite[x, trv[y]]]
Out[9]= True
```

```
In[10]:= subclass[composite[x_, trv[y_], trv[y_]], composite[x_, trv[y_]]] := True
```

---

restoring the removed rule

The removed rule is restored by repeating the same derivation that was used in March 2004.

```
In[11]:= SubstTest[subclass, range[iterate[u, v]], v, {u -> cross[Id, x], v -> composite[Id, y]}]
Out[11]= subclass[composite[trv[x], y], y] == subclass[composite[x, y], y]
```

```
In[12]:= subclass[composite[trv[x_], y_], y_] := subclass[composite[x, y], y]
```

A similar rule holds when the order of the factors is reversed:

```
In[13]:= SubstTest[subclass, range[iterate[u, v]],
          v, {u -> cross[inverse[y], Id], v -> composite[Id, x]}]
Out[13]= subclass[composite[x, trv[y]], x] == subclass[composite[x, y], x]
```

```
In[14]:= subclass[composite[x_, trv[y_]], x_] := subclass[composite[x, y], x]
```

Specializing to the case that **y** is **x** yields:

```
In[15]:= SubstTest[subclass, composite[trv[x], y], y, y -> x]
Out[15]= subclass[composite[trv[x], trv[x]], x] == TRANSITIVE[composite[Id, x]]
```

```
In[16]:= subclass[composite[trv[x_], trv[x_]], x_] := TRANSITIVE[composite[Id, x]]
```

the main theorem

Lemma 1.

```
In[17]:= Map[implies[subclass[x, cart[V, V]], #] &, SubstTest[implies,
           subclass[u, v], subclass[u, union[v, y]], {u → composite[x, x], v → x}]]

Out[17]= or[not[TRANSITIVE[x]], subclass[composite[x, x], union[x, y]]] == True

In[18]:= or[not[TRANSITIVE[x_]], subclass[composite[x_, x_], union[x_, y_]]] := True
```

Lemma 2.

```
In[19]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> commute[trv[x], trv[y]], p2 → TRANSITIVE[composite[trv[x], trv[y]]],
            p3 -> subclass[composite[trv[x], trv[y], trv[x], trv[y]],
              union[composite[trv[x], trv[y]], trv[x], trv[y]]]}]]

Out[19]= or[not[equal[composite[trv[x], trv[y]], composite[trv[y], trv[x]]]],
           subclass[composite[trv[x], trv[y], trv[x], trv[y]],
            union[composite[trv[x], trv[y]], trv[x], trv[y]]]] == True

In[20]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[21]:= (Map[implies[commute[trv[x], trv[y]], #] &, SubstTest[subclass, composite[w, w],
              w, w -> union[trv[x], trv[y], composite[trv[x], trv[y]]]]] //
            MapNotNot // Reverse) /. z → composite[trv[x], trv[y]]

Out[21]= or[not[equal[composite[trv[x], trv[y]], composite[trv[y], trv[x]]]],
           TRANSITIVE[union[composite[trv[x], trv[y]], trv[x], trv[y]]]] == True

In[22]:= or[not[equal[composite[trv[x_], trv[y_]], composite[trv[y_], trv[x_]]]],
           TRANSITIVE[union[composite[trv[x_], trv[y_]], trv[x_], trv[y_]]]] := True
```

A wrapper-free restatement is easily derived:

```
In[23]:= SubstTest[implies, and[equal[x, trv[u]], equal[y, trv[v]], commute[x, y]],
           TRANSITIVE[union[x, y, composite[x, y]]], {u → x, v → y}]

Out[23]= or[not[equal[composite[x, y], composite[y, x]]], not[TRANSITIVE[x]],
           not[TRANSITIVE[y]], TRANSITIVE[union[x, y, composite[x, y]]]] == True

In[24]:= or[not[equal[composite[x_, y_], composite[y_, x_]]], not[TRANSITIVE[x_]],
           not[TRANSITIVE[y_]], TRANSITIVE[union[composite[x_, y_], x_, y_]]] := True
```