transitive and equivalence relations

Johan G. F. Belinfante
2004 April 4

```
In[1]:= << goedel56.01a; << tools.m;

        :Package Title: goedel56.01a          2004 April 1 at 9:10 p.m.

        It is now:  2004 Apr 5 at 13:20

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 March 29

        weightlimit = 40
```

summary

If **x** is transitive relation, then **intersection[x, inverse[x]]** is an equivalence relation. This result and a variable−free version of it are derived in this notebook.

derivation

The following temporary rewrite rule is convenient, but it is somewhat dangerous in that it could lead to looping in certain situations. The problem is that the (wrapped) definition of **EQUIVALENCE** is in terms of **TRANSITIVE**.

```
In[2]:= SubstTest[and, SYMMETRIC[y], TRANSITIVE[y], y -> intersection[x, inverse[x]]]

Out[2]= TRANSITIVE[intersection[x, inverse[x]]] == EQUIVALENCE[intersection[x, inverse[x]]]

In[3]:= TRANSITIVE[intersection[x_, inverse[x_]]] := EQUIVALENCE[intersection[x, inverse[x]]]
```

Lemma:

```
In[4]:= equiv[and[TRANSITIVE[x], TRANSITIVE[composite[Id, x]]], TRANSITIVE[x]]

Out[4]= True

In[5]:= and[TRANSITIVE[x_], TRANSITIVE[composite[Id, x_]]] := TRANSITIVE[x]
```

Main theorem:

```
In[6]:= SubstTest[implies, and[TRANSITIVE[x], TRANSITIVE[y]],
          TRANSITIVE[intersection[x, y]], y -> inverse[x]] // MapNotNot

Out[6]= or[EQUIVALENCE[intersection[x, inverse[x]]], not[TRANSITIVE[x]]] == True

In[7]:= or[EQUIVALENCE[intersection[x_, inverse[x_]]], not[TRANSITIVE[x_]]] := True
```

emininating the variables

The following function figures in the variable−free formulation of the results found in the preceding section:

```
In[8]:= lambda[x, intersection[x, inverse[x]]]

Out[8]= CORE[SYM]

In[9]:= Map[equal[V, #] &, union[complement[TRV], image[inverse[CORE[SYM]], EQV]] // Normality]

Out[9]= subclass[image[CORE[SYM], TRV], EQV] == True

In[10]:= % /. Equal -> SetDelayed
```

Lemma.

```
In[11]:= SubstTest[image, funpart[x], fix[funpart[x]], x -> CORE[SYM]]

Out[11]= image[CORE[SYM], SYM] == SYM

In[12]:= image[CORE[SYM], SYM] := SYM

In[13]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
             {u -> EQV, v -> SYM, w -> CORE[SYM]}]

Out[13]= subclass[image[CORE[SYM], EQV], SYM] == True

In[14]:= % /. Equal -> SetDelayed

In[15]:= Map[subclass[EQV, #] &,
           SubstTest[image, funpart[x], fix[funpart[x]], x -> composite[id[TRV], CORE[SYM]]]]

Out[15]= subclass[EQV, image[CORE[SYM], EQV]] == True

In[16]:= % /. Equal -> SetDelayed

In[17]:= SubstTest[implies, subclass[u, v],
           subclass[image[w, u], image[w, v]], {u -> EQV, v -> TRV, w -> CORE[SYM]}]

Out[17]= subclass[image[CORE[SYM], EQV], image[CORE[SYM], TRV]] == True

In[18]:= % /. Equal -> SetDelayed

In[19]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
             {u -> EQV, v -> image[CORE[SYM], EQV], w -> image[CORE[SYM], TRV]}]

Out[19]= subclass[EQV, image[CORE[SYM], TRV]] == True

In[20]:= % /. Equal -> SetDelayed
```

Variable−free reformulation of the main theorem:

```
In[21]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> image[CORE[SYM], TRV], v -> EQV}]

Out[21]= True == equal[EQV, image[CORE[SYM], TRV]]
```

In[22]:= **image[CORE[SYM], TRV] := EQV**