binary closure under non-disjoint unions

Johan G. F. Belinfante
2014 March 21

```
In[1]:= SetDirectory["l:"]; << goedel.14mar20a
```

```
:Package Title: goedel.14mar20a                    2014 March 20 at 4:15 p.m.

Loading takes about seventeen minutes, half that time due to builtin pauses.

It is now:  2014 Mar 21 at 15:30

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40

Loading completed.

It is now:  2014 Mar 21 at 15:47
```

summary

If a collection **x** of sets is closed under non-disjoint unions, then **inverse[E] ∘ id[x] ∘ E** is an equivalence relation.

general results

Theorem.

```
In[2]:= Map[implies[TRANSITIVE[composite[inverse[x], x]], #] &,
         SubstTest[and, SYMMETRIC[t], TRANSITIVE[t], t -> composite[inverse[x], x]]]
```

```
Out[2]= or[EQUIVALENCE[composite[inverse[x], x]],
          not[TRANSITIVE[composite[inverse[x], x]]]] == True
```

```
In[3]:= or[EQUIVALENCE[composite[inverse[x_], x_]],
          not[TRANSITIVE[composite[inverse[x_], x_]]]] := True
```

Corollary.

```
In[4]:= SubstTest[or, EQUIVALENCE[composite[inverse[t], t]],
          not[TRANSITIVE[composite[inverse[t], t]]], t → composite[id[y], x]] // Reverse
```

```
Out[4]= or[EQUIVALENCE[composite[inverse[x], id[y], x]],
          not[TRANSITIVE[composite[inverse[x], id[y], x]]]] == True
```

```
In[5]:= or[EQUIVALENCE[composite[inverse[x_], id[y_], x_]],
          not[TRANSITIVE[composite[inverse[x_], id[y_], x_]]]] := True
```

binary closure

Theorem. Simplification rule.

```
In[6]:= SubstTest[fix, composite[S, IMAGE[t], CART, DUP], t → composite[x, id[y]]] // Reverse

Out[6]= fix[composite[S, IMAGE[x], IMAGE[id[y]], CART, DUP]] == binclosed[composite[x, id[y]]]

In[7]:= fix[composite[S, IMAGE[x_], IMAGE[id[y_]], CART, DUP]] := binclosed[composite[x, id[y]]]
```

Observation.

```
In[8]:= member[t, binclosed[composite[x, id[y]]]]

Out[8]= and[member[t, V], subclass[image[x, composite[id[t], y, id[t]]], t]]
```

binary closed under non-disjoint unions

A class **x** is **binary closed under non-disjoint unions** if the following condition holds.

```
In[9]:= assert[forall[u, v,
           implies[and[member[u, x], member[v, x]], or[disjoint[u, v], member[union[u, v], x]]]]]

Out[9]= subclass[image[CUP, composite[id[x], E, inverse[E], id[x]]], x]
```

Observations.

```
In[10]:= class[x, subclass[image[CUP, composite[id[x], E, inverse[E], id[x]]], x]]

Out[10]= binclosed[composite[CUP, id[composite[E, inverse[E]]]]]
```

Theorem.

```
In[11]:= SubstTest[implies, and[member[r, s], subclass[s, t]], member[r, t], {r → pair[u, v],
             s -> composite[id[x], E, inverse[E], id[x]], t → image[inverse[CUP], x]}] // Reverse

Out[11]= or[equal[0, intersection[u, v]],
           member[union[u, v], x], not[member[u, x]], not[member[v, x]],
           not[subclass[image[CUP, composite[id[x], E, inverse[E], id[x]]], x]]] == True

In[12]:= or[equal[0, intersection[u_, v_]],
           member[union[u_, v_], x_], not[member[u_, x_]], not[member[v_, x_]],
           not[subclass[image[CUP, composite[id[x_], E, inverse[E], id[x_]]], x_]]] := True
```

inverse[E] ∘ CART ∘ inverse[CUP]

Theorem.

```
In[13]:= composite[cross[inverse[E], inverse[E]], inverse[CUP]] // RelnNormality
Out[13]= composite[inverse[E], CART, inverse[CUP]] == composite[inverse[E], CART, DUP]

In[14]:= composite[inverse[E], CART, inverse[CUP]] := composite[inverse[E], CART, DUP]
```

Corollary.

```
In[15]:= composite[CUP, inverse[CART], E] // DoubleInverse
Out[15]= composite[CUP, inverse[CART], E] == composite[S, PAIRSET]

In[16]:= composite[CUP, inverse[CART], E] := composite[S, PAIRSET]
```

Theorem.

```
In[17]:= IminComp[CUP, composite[inverse[CART], E], x] // Reverse
Out[17]= composite[inverse[E], image[inverse[CUP], x], E] == composite[inverse[E], id[x], E]

In[18]:= composite[inverse[E], image[inverse[CUP], x_], E] := composite[inverse[E], id[x], E]
```

Theorem.

```
In[19]:= ImageComp[CUP, composite[inverse[CART], E], x] // Reverse
Out[19]= image[CUP, composite[E, x, inverse[E]]] == complement[cliques[complement[x]]]

In[20]:= image[CUP, composite[E, x_, inverse[E]]] := complement[cliques[complement[x]]]
```

Theorem.

```
In[21]:= SubstTest[implies, subclass[u, v],
           subclass[image[t, u], image[t, v]], {t → cross[inverse[E], inverse[E]],
            u -> composite[id[x], E, inverse[E], id[x]], v -> image[inverse[CUP], x]}] // Reverse
Out[21]= or[not[subclass[image[CUP, composite[id[x], E, inverse[E], id[x]]], x]],
           TRANSITIVE[composite[inverse[E], id[x], E]]] == True

In[22]:= or[not[subclass[image[CUP, composite[id[x_], E, inverse[E], id[x_]]], x_]],
           TRANSITIVE[composite[inverse[E], id[x_], E]]] := True
```

Corollary. If **x** is closed under non-disjoint unions, then **inverse[E] ∘ id[x] ∘ E** is an equivalence relation.

```
In[23]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
           {p1 -> subclass[image[CUP, composite[id[x], E, inverse[E], id[x]]], x],
            p2 -> TRANSITIVE[composite[inverse[E], id[x], E]],
            p3 → EQUIVALENCE[composite[inverse[E], id[x], E]]}]] // Reverse

Out[23]= or[EQUIVALENCE[composite[inverse[E], id[x], E]],
           not[subclass[image[CUP, composite[id[x], E, inverse[E], id[x]]], x]]] == True

In[24]:= or[EQUIVALENCE[composite[inverse[E], id[x_], E]],
           not[subclass[image[CUP, composite[id[x_], E, inverse[E], id[x_]]], x_]]] := True
```

eliminating the variable x

Lemma.

```
In[25]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[implies[member[x, u], member[x, v]]],
           {u → binclosed[composite[CUP, id[composite[E, inverse[E]]]]], v → image[
              inverse[IMAGE[DUP]], image[inverse[IMAGE[CART]], image[inverse[BIGCUP], EQV]]]}]]

Out[25]= subclass[image[BIGCUP, image[IMAGE[CART], image[IMAGE[DUP],
             binclosed[composite[CUP, id[composite[E, inverse[E]]]]]]], EQV]] == True

In[26]:= % /. Equal → SetDelayed
```

An inclusion in the opposite direction also holds, as will now be shown.

Lemma.

```
In[27]:= Map[composite[Id, complement[#]] &, symdif[composite[id[ptn[x]], Di, id[ptn[x]]],
            composite[id[ptn[x]], DISJOINT, id[ptn[x]]]] // complement // RelnNormality]

Out[27]= composite[id[ptn[x]], intersection[Di,
           composite[inverse[E], IMAGE[composite[id[ptn[x]], E]]], id[ptn[x]]] == 0

In[28]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[29]:= SubstTest[empty, symdif[u, v], {u -> composite[id[ptn[x]], Di, id[ptn[x]]],
           v -> composite[id[ptn[x]], DISJOINT, id[ptn[x]]]}]

Out[29]= equal[composite[id[ptn[x]], Di, id[ptn[x]]],
           composite[id[ptn[x]], DISJOINT, id[ptn[x]]]] == True

In[30]:= composite[id[ptn[x_]], DISJOINT, id[ptn[x_]]] := composite[id[ptn[x]], Di, id[ptn[x]]]
```

Lemma.

```
In[31]:= Map[composite[id[ptn[x]], complement[#]] &,
           SubstTest[complement, composite[id[t], E, inverse[E], id[t]], t → ptn[x]]] // Reverse

Out[31]= composite[id[ptn[x]], inverse[IMAGE[composite[id[ptn[x]], E]], E] == id[ptn[x]]

In[32]:= composite[id[ptn[x_]], inverse[IMAGE[composite[id[ptn[x_]], E]], E] := id[ptn[x]]
```

Theorem.

```
In[33]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[implies[member[ptn[x], V], member[ptn[x], t]]],
           t -> binclosed[composite[CUP, id[composite[E, inverse[E]]]]]]]

Out[33]= subclass[PARTNS, binclosed[composite[CUP, id[composite[E, inverse[E]]]]]] == True

In[34]:= subclass[PARTNS, binclosed[composite[CUP, id[composite[E, inverse[E]]]]]] := True
```

Lemma.

```
In[35]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → composite[BIGCUP, IMAGE[CART], IMAGE[DUP]], u → PARTNS,
            v -> binclosed[composite[CUP, id[composite[E, inverse[E]]]]]}] // Reverse

Out[35]= subclass[EQV, image[BIGCUP, image[IMAGE[CART], image[IMAGE[DUP],
             binclosed[composite[CUP, id[composite[E, inverse[E]]]]]]]] == True

In[36]:= % /. Equal → SetDelayed
```

Theorem. An equation.

```
In[37]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[BIGCUP, image[IMAGE[CART], image[IMAGE[DUP],
               binclosed[composite[CUP, id[composite[E, inverse[E]]]]]]]], v -> EQV}]

Out[37]= equal[EQV, image[BIGCUP, image[IMAGE[CART], image[IMAGE[DUP],
             binclosed[composite[CUP, id[composite[E, inverse[E]]]]]]]] == True

In[38]:= image[BIGCUP, image[IMAGE[CART],
            image[IMAGE[DUP], binclosed[composite[CUP, id[composite[E, inverse[E]]]]]]]] := EQV
```