equivalence with respect to an ideal

Johan G. F. Belinfante
2011 February 14

```
In[1]:= SetDirectory["l:"]; << goedel.11feb11a

        :Package Title: goedel.11feb11a                    2011 February 11 at 2:40 p.m.

        It is now:  2011 Feb 14 at 8:1

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

If a class **x** is binary closed under unions and hereditary in the sense that every subset of a member is a member, then the condition that the symmetric difference of two sets belongs to **x** is an equivalence relation. The following special case of this is already available in the **GOEDEL** program:

```
In[2]:= EQUIVALENCE[image[inverse[SYMDIF], FINITE]]

Out[2]= True
```

comments

The conditions on **x** are two of several properties commonly used to define an ideal in set theory. (There are also related concepts in ring theory and lattice theory, about which nothing will be said here.) Chapter 11 in the following reference is about filters and ultrafilters.

```
In[3]:= "Karel Hrbacek and Thomas Jech, Introduction to Set Theory, Third Edition, Revised
          and Expanded, Marcel Dekker, Inc., New York, 1999.  ISBN-0-8247-7915-0";
```

The dual concept of ideal is on page 202. These authors define a set **i** to be an **ideal** on a nonempty set **s** if the following conditions hold: **image[CUP, i × i] = i, image[inverse[S], i] = i , 0 ∈ i, s ∉ i** and **U[i] ⊂ s**. For the theorem to be derived in this notebook only the first two conditions on **i** are needed, sethood is not required, and one can dispense with **s** altogether.

a more concise equational formulation

The two conditions of being hereditary and binary closed under unions can be combined into a single equation.

Theorem.

```
In[4]:= equal[image[inverse[CUP], x], cart[x, x]] // AssertTest // Reverse

Out[4]= and[subclass[image[CUP, cart[x, x]], x], subclass[image[inverse[S], x], x]] ==
         equal[cart[x, x], image[inverse[CUP], x]]

In[5]:= and[subclass[image[CUP, cart[x_, x_]], x_], subclass[image[inverse[S], x_], x_]] :=
         equal[cart[x, x], image[inverse[CUP], x]]
```

Corollary.

```
In[6]:= or[not[subclass[image[inverse[S], x], x]], not[subclass[image[CUP, cart[x, x]], x]],
          equal[cart[x, x], image[inverse[CUP], x]]] // NotNotTest

Out[6]= or[equal[cart[x, x], image[inverse[CUP], x]], not[subclass[image[CUP, cart[x, x]], x]],
          not[subclass[image[inverse[S], x], x]]] == True

In[7]:= or[equal[cart[x_, x_], image[inverse[CUP], x_]],
          not[subclass[image[CUP, cart[x_, x_]], x_]],
          not[subclass[image[inverse[S], x_], x_]]] := True
```

In the opposite direction, one has the following.

Theorem.

```
In[8]:= SubstTest[implies, equal[u, v], equal[domain[u], domain[v]],
          {u → cart[x, x], v → image[inverse[CUP], x]}] // Reverse

Out[8]= or[equal[x, image[inverse[S], x]], not[equal[cart[x, x], image[inverse[CUP], x]]]] == True

In[9]:= or[equal[x_, image[inverse[S], x_]],
          not[equal[cart[x_, x_], image[inverse[CUP], x_]]]] := True
```

The following theorem asserts that the condition $\mathbf{0 \in x}$ is automatically satisfied when $\mathbf{x}$ is not empty.

Theorem.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p2],
           not[implies[p1, p3]], {p1 -> equal[cart[x, x], image[inverse[CUP], x]],
            p2 → equal[x, image[inverse[S], x]], p3 -> or[equal[0, x], member[0, x]]}]] // Reverse

Out[10]= or[equal[0, x], member[0, x], not[equal[cart[x, x], image[inverse[CUP], x]]]] == True

In[11]:= or[equal[0, x_], member[0, x_],
           not[equal[cart[x_, x_], image[inverse[CUP], x_]]]] := True
```

the main theorem

The following theorem needed shortly restates a result recently derived using the **GOEDEL** program.

Theorem. If **x** is hereditary and binary closed under unions, then **image[inverse[DIF], x]** is transitive.

```
In[12]:= implies[equal[cart[x, x], image[inverse[CUP], x]],
           TRANSITIVE[image[inverse[DIF], x]]] // AssertTest

Out[12]= or[not[equal[cart[x, x], image[inverse[CUP], x]]],
           TRANSITIVE[image[inverse[DIF], x]]] == True

In[13]:= or[not[equal[cart[x_, x_], image[inverse[CUP], x_]]],
           TRANSITIVE[image[inverse[DIF], x_]]] := True
```

Lemma.

```
In[14]:= IminComp[CUP, composite[cross[DIF, flip[DIF]], DUP], x] // Reverse

Out[14]= fix[composite[inverse[DIF], image[inverse[CUP], x], DIF, SWAP]] ==
          image[inverse[SYMDIF], x]

In[15]:= fix[composite[inverse[DIF], image[inverse[CUP], x_], DIF, SWAP]] :=
          image[inverse[SYMDIF], x]
```

Theorem. If **x** is hereditary and binary closed under unions, then **image[inverse[SYMDIF], x]** is the intersection of the relation **image[inverse[DIF], x]** with its inverse.

```
In[16]:= SubstTest[implies, equal[u, v], equal[image[t, u], image[t, v]],
           {t -> intersection[composite[inverse[DIF], SECOND], composite[SWAP,
               inverse[DIF], FIRST]], u → cart[x, x], v → image[inverse[CUP], x]}] // Reverse

Out[16]= or[equal[image[inverse[SYMDIF], x],
            intersection[image[inverse[DIF], x], inverse[image[inverse[DIF], x]]]],
           not[equal[cart[x, x], image[inverse[CUP], x]]]] == True

In[17]:= or[equal[image[inverse[SYMDIF], x_],
            intersection[image[inverse[DIF], x_], inverse[image[inverse[DIF], x_]]]],
           not[equal[cart[x_, x_], image[inverse[CUP], x_]]]] := True
```

Theorem. If **x** is hereditary and binary closed under unions, then **image[inverse[SYMDIF], x]** is an equivalence relation.

```
In[18]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[and[p2, p3], p4],
            not[implies[p1, p4]], {p1 -> equal[cart[x, x], image[inverse[CUP], x]],
             p2 -> TRANSITIVE[image[inverse[DIF], x]], p3 -> equal[image[inverse[SYMDIF], x],
               intersection[image[inverse[DIF], x], inverse[image[inverse[DIF], x]]]],
             p4 → EQUIVALENCE[image[inverse[SYMDIF], x]]}]] // Reverse

Out[18]= or[EQUIVALENCE[image[inverse[SYMDIF], x]],
           not[equal[cart[x, x], image[inverse[CUP], x]]]] == True
```

```
In[19]:= or[EQUIVALENCE[image[inverse[SYMDIF], x_]],
            not[equal[cart[x_, x_], image[inverse[CUP], x_]]]] := True
```

Corollary.

```
In[20]:= or[EQUIVALENCE[image[inverse[SYMDIF], x]], not[subclass[image[CUP, cart[x, x]], x]],
            not[subclass[image[inverse[S], x], x]]] // NotNotTest

Out[20]= or[EQUIVALENCE[image[inverse[SYMDIF], x]], not[subclass[image[CUP, cart[x, x]], x]],
            not[subclass[image[inverse[S], x], x]]] == True
```

```
In[21]:= or[EQUIVALENCE[image[inverse[SYMDIF], x_]],
            not[subclass[image[CUP, cart[x_, x_]], x_]],
            not[subclass[image[inverse[S], x_], x_]]] := True
```

---

a slight generalization

One can omit the hypothesis that **x** be hereditary if the conclusion is weakened accordingly.

Lemma.

```
In[23]:= SubstTest[or, EQUIVALENCE[image[inverse[SYMDIF], t]],
            not[subclass[image[CUP, cart[t, t]], t]],
            not[subclass[image[inverse[S], t], t]], t → image[inverse[S], x]] // Reverse

Out[23]= or[EQUIVALENCE[image[inverse[SYMDIF], image[inverse[S], x]]],
            not[subclass[image[CUP, cart[x, x]], image[inverse[S], x]]]] == True
```

```
In[24]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. If **x** is binary closed under unions, then **image[inverse[SYMDIF], image[inverse[S], x]]** is an equivalence relation.

```
In[25]:= Map[not, SubstTest[and, implies[p1, p2],
              not[implies[p1, p3]], {p1 → equal[image[CUP, cart[x, x]], x],
               p2 -> subclass[image[CUP, cart[x, x]], image[inverse[S], x]],
               p3 -> EQUIVALENCE[image[inverse[SYMDIF], image[inverse[S], x]]]}]] // Reverse

Out[25]= or[EQUIVALENCE[image[inverse[SYMDIF], image[inverse[S], x]]],
            not[equal[x, image[CUP, cart[x, x]]]]] == True
```

```
In[26]:= or[EQUIVALENCE[image[inverse[SYMDIF], image[inverse[S], x_]]],
            not[equal[x_, image[CUP, cart[x_, x_]]]]] := True
```

The following corollary provides an example of this.

Corollary. The condition that the symmetric difference of two sets be countable is an equivalence relation.

```
In[48]:= SubstTest[or, EQUIVALENCE[image[inverse[SYMDIF], image[inverse[S], x]]],
           not[equal[x, image[CUP, cart[x, x]]]], x → image[Q, set[omega]]] // Reverse

Out[48]= EQUIVALENCE[image[inverse[SYMDIF], image[Q, succ[omega]]]] == True

In[49]:= EQUIVALENCE[image[inverse[SYMDIF], image[Q, succ[omega]]]] := True
```

Observation.

```
In[50]:= COUNTABLE

Out[50]= image[Q, succ[omega]]
```