equality substitution for EQUIVALENCE

Johan G. F. Belinfante
2004 April 2

```
In[1]:= << goedel55.31a; << tools.m;

        :Package Title: goedel55.31a           2004 March 31 at 11:20 p.m.

        It is now:  2004 Apr 2 at 10:38

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 March 29

        weightlimit = 40
```

summary

A rewrite rule for equality substitution is derived for the predicate **EQUIVALENCE**, and some applications of it are made.

equality substitution

The equality substitution rule is derived from similar rules for **SYMMETRIC** and **TRANSITIVE**.

```
In[2]:= Map[not, SubstTest[and, implies[p2, p3], implies[p2, p4],
          implies[and[p1, p3], p5], implies[and[p1, p4], p6], p1, p2, or[not[p5], not[p6]],
            {p1 -> equal[x, y], p2 -> EQUIVALENCE[y], p3 -> SYMMETRIC[y],
           p4 -> TRANSITIVE[y], p5 -> SYMMETRIC[x], p6 -> TRANSITIVE[x]}]]

Out[2]= or[EQUIVALENCE[x], not[equal[x, y]], not[EQUIVALENCE[y]]] == True

In[3]:= or[EQUIVALENCE[x_], not[equal[x_, y_]], not[EQUIVALENCE[y_]]] := True
```

Theorem EQV−FU

Any function determines an equivalence relation on its domain, two arguments being equivalent if they have the same value. One can derive this rule using a **funpart** wrapper as follows:

```
In[4]:= SubstTest[equal, y, composite[y, inverse[y]],
          y -> composite[inverse[funpart[x]], funpart[x]]] // Reverse

Out[4]= EQUIVALENCE[composite[inverse[funpart[x]], funpart[x]]] == True

In[5]:= (% /. x -> x_) /. Equal -> SetDelayed
```

To remove the funpart wrapper, one needs to make use of equality substitutions. The following lemma is needed:

```
In[6]:= SubstTest[implies, equal[u, v], equal[image[w, u], image[w, v]],
            {u -> cart[x, x], v -> cart[y, y], w -> composite[FIRST, id[cart[V, Id]], TWIST]}]

Out[6]= or[equal[composite[inverse[x], x], composite[inverse[y], y]], not[equal[x, y]]] == True

In[7]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed
```

Replacing **y** with **funpart[x]** yields:

```
In[8]:= SubstTest[implies, equal[x, y],
         equal[composite[inverse[x], x], composite[inverse[y], y]], y -> funpart[x]]

Out[8]= or[equal[composite[inverse[x], x], composite[inverse[funpart[x]], funpart[x]]],
          not[FUNCTION[x]]] == True

In[9]:= (% /. x -> x_) /. Equal -> SetDelayed

In[10]:= SubstTest[implies, and[equal[u, v], EQUIVALENCE[v]], EQUIVALENCE[u],
          {u -> composite[inverse[x], x], v -> composite[inverse[funpart[x]], funpart[x]]}]

Out[10]= or[EQUIVALENCE[composite[inverse[x], x]], not[
            equal[composite[inverse[x], x], composite[inverse[funpart[x]], funpart[x]]]]] == True

In[11]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Theorem **EQV−FU** follows:

```
In[12]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
                {p1 -> FUNCTION[x],
            p2 -> equal[composite[inverse[x], x], composite[inverse[funpart[x]], funpart[x]]],
            p3 -> EQUIVALENCE[composite[inverse[x], x]]}]]

Out[12]= or[EQUIVALENCE[composite[inverse[x], x]], not[FUNCTION[x]]] == True

In[13]:= or[EQUIVALENCE[composite[inverse[x_], x_]], not[FUNCTION[x_]]] := True
```

The following reformulation of the theorem which replaces **EQUIVALENCE** with **TRANSITIVE** is needed for the work in the next section.

```
In[14]:= Map[implies[FUNCTION[x], #] &,
           EQUIVALENCE[composite[inverse[x], x]] // AssertTest] // Reverse

Out[14]= or[not[FUNCTION[x]], TRANSITIVE[composite[inverse[x], x]]] == True

In[15]:= (% /. x -> x_) /. Equal -> SetDelayed
```

a variable−free version of Theorem EQV−FU

A variable−free formulation of this result can be obtained for the case that **x** is a set. A version using inverse images is easiest to derive:

```
In[16]:= Map[equal[V, #] &, union[complement[FUNS], image[
             inverse[composite[COMPOSE, id[INVERSE], inverse[SECOND]]], EQV]] // Renormality]

Out[16]= subclass[FUNS, fix[composite[INVERSE, inverse[image[inverse[COMPOSE], EQV]]]]] == True

In[17]:= % /. Equal -> SetDelayed
```

The following general result can be used to transform this result:

```
In[18]:= SubstTest[implies, subclass[x, v], subclass[image[u, x], image[u, v]],
           {u -> funpart[y], v -> image[inverse[funpart[y]], z]}]

Out[18]= or[not[subclass[x, image[inverse[funpart[y]], z]]],
           subclass[image[funpart[y], x], z]] == True

In[19]:= or[not[subclass[x_, image[inverse[funpart[y_]], z_]]],
           subclass[image[funpart[y_], x_], z_]] := True
```

One can use thus to derive a variable−free formulation that uses images instead of inverse images:

```
In[20]:= SubstTest[implies, subclass[x, image[inverse[funpart[y]], z]],
           subclass[image[funpart[y], x], z],
             {x -> FUNS, y -> composite[COMPOSE, id[INVERSE], inverse[SECOND]], z -> EQV}]

Out[20]= subclass[image[COMPOSE, composite[id[FUNS], INVERSE]], EQV] == True

In[21]:= subclass[image[COMPOSE, composite[id[FUNS], INVERSE]], EQV] := True
```

the only functions that are equivalence relations are identity functions

In this section, another application of the equational substitution rule is presented. It will be shown that the only functions that are also equivalence relations are identity functions. The equational characterization of **EQUIVALENCE** implies this lemma:

```
In[22]:= SubstTest[implies, and[equal[x, y], subclass[y, z]],
           subclass[x, z], {y -> composite[x, inverse[x]], z -> Id}]

Out[22]= or[not[EQUIVALENCE[x]], not[FUNCTION[composite[Id, x]]], subclass[x, Id]] == True

In[23]:= (% /. x -> x_) /. Equal -> SetDelayed
```

The hypothesis that **x** be singlevalued can be replaced with the stronger hypothesis that **x** be a function.

```
In[24]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[and[p2, p3], p4], not[implies[and[p1, p3], p4]],
                 {p1 -> FUNCTION[x], p2 -> FUNCTION[composite[Id, x]],
            p3 -> EQUIVALENCE[x], p4 -> subclass[x, Id]}]]

Out[24]= or[not[EQUIVALENCE[x]], not[FUNCTION[x]], subclass[x, Id]] == True

In[25]:= (% /. x -> x_) /. Equal -> SetDelayed
```

Equality substitution yields:

```
In[26]:= SubstTest[implies, and[equal[x, y], EQUIVALENCE[y]], EQUIVALENCE[x], y -> id[fix[x]]]

Out[26]= or[EQUIVALENCE[x], not[subclass[x, Id]]] == True

In[27]:= or[EQUIVALENCE[x_], not[subclass[x_, Id]]] := True
```

The following logical equivalence is now recognized to be true:

```
In[28]:= equiv[and[FUNCTION[x], EQUIVALENCE[x]], subclass[x, Id]] // not // not
Out[28]= True
```

This can be made into a rewrite rule:

```
In[29]:= and[EQUIVALENCE[x_], FUNCTION[x_]] := subclass[x, Id]
```

There is avariable−free version for the case that **x** is a set.

```
In[30]:= intersection[EQV, FUNS] // Normality
Out[30]= intersection[EQV, FUNS] == P[Id]

In[31]:= intersection[EQV, FUNS] := P[Id]
```