x = BIGCUP o IMAGE[x] o POWER

Johan G. F. Belinfante
2004 October 8

```
In[1]:= SetDirectory["i:"]; << goedel62.07a; << tools.m

        :Package Title: goedel62.07a          2004 October 7 at 11:40 a.m.

        It is now:  2004 Oct 8 at 9:40

        Loading Simplification Rules

        TOOLS.M                      Revised 2004 September 25

        weightlimit = 40
```

---

summary

It is shown in this notebook that the equation **x = composite[BIGCUP, IMAGE[x], POWER]** holds if and only if **x** is a total function which subcommutes with **inverse[S]**.

---

lemma 1

In this section it is shown that the equation implies that **x** subcommutes with **inverse[S]**.

```
In[2]:= SubstTest[implies, and[subcommute[u, w], subcommute[v, w]],
          subcommute[composite[u, v], w],
          {u → IMAGE[thinpart[w]], v → POWER, w → inverse[S]}] /.
         w → composite[inverse[E], funpart[x]]

Out[2]= subclass[composite[BIGCUP, IMAGE[funpart[x]], POWER, inverse[S]],
          composite[inverse[S], BIGCUP, IMAGE[funpart[x]], POWER]] == True

In[3]:= (% /. x → x_) /. Equal → SetDelayed

In[4]:= SubstTest[implies,
         and[equal[u, funpart[x]], equal[v, composite[BIGCUP, IMAGE[u], POWER]]],
         subcommute[v, inverse[S]], {u → x, v → x}]

Out[4]= or[not[equal[x, composite[BIGCUP, IMAGE[x], POWER]]], not[FUNCTION[x]],
          subclass[composite[x, inverse[S]], composite[inverse[S], x]]] == True
```

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

```
In[6]:= Map[not, SubstTest[and, implies[p1, p2],
          implies[and[p1, p2], p3], not[implies[p1, p3]],
          {p1 -> equal[x, composite[BIGCUP, IMAGE[x], POWER]],
           p2 → FUNCTION[x], p3 → subcommute[x, inverse[S]]}]]
```

```
Out[6]= or[not[equal[x, composite[BIGCUP, IMAGE[x], POWER]]],
          subclass[composite[x, inverse[S]], composite[inverse[S], x]]] == True
```

```
In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

---

lemma 2

Here it is shown that the equation implies that **x** is total.

```
In[8]:= SubstTest[implies, equal[x, y], equal[domain[x], domain[y]],
         y -> composite[BIGCUP, IMAGE[x], POWER]]
```

```
Out[8]= or[equal[complement[image[S, complement[domain[VERTSECT[x]]]]], domain[x]],
          not[equal[x, composite[BIGCUP, IMAGE[x], POWER]]]] == True
```

```
In[9]:= (% /. x → x_) /. Equal → SetDelayed
```

```
In[10]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p1, p3], implies[and[p2, p3], p4], not[implies[p1, p4]],
           {p1 -> equal[x, composite[BIGCUP, IMAGE[x], POWER]], p2 → thin[x],
            p3 -> equal[complement[image[S, complement[domain[VERTSECT[x]]]]],
              domain[x]], p4 → equal[V, domain[x]]}]]
```

```
Out[10]= or[equal[V, domain[x]],
           not[equal[x, composite[BIGCUP, IMAGE[x], POWER]]]] == True
```

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

---

lemma 3: a consequence of subcommute[x,inverse[S]]

Weak version:

```
In[12]:= SubstTest[implies, subclass[u, v],
          subclass[composite[w, u], composite[w, v]],
          {u -> composite[x, inverse[S]], v -> composite[inverse[S], x], w → inverse[E]}]
```

```
Out[12]= or[not[subclass[composite[x, inverse[S]], composite[inverse[S], x]]],
           subclass[composite[inverse[E], x, inverse[S]],
            composite[inverse[E], x]]] == True
```

```
In[13]:= (% /. x → x_) /. Equal → SetDelayed
```

Stronger version:

```
In[14]:= implies[subcommute[x, inverse[S]], equal[composite[inverse[E], x, inverse[S]],
           composite[inverse[E], x]]] // AssertTest

Out[14]= or[equal[composite[inverse[E], x], composite[inverse[E], x, inverse[S]]],
          not[subclass[composite[x, inverse[S]], composite[inverse[S], x]]]] == True

In[15]:= or[equal[composite[inverse[E], x_], composite[inverse[E], x_, inverse[S]]],
          not[subclass[composite[x_, inverse[S]], composite[inverse[S], x_]]]] := True
```

---

lemma 4: conditions that imply the equation

To circumvent the following rewrite rule, composition with an extra variable **w** is introduced, and then **w** is replaced by **Id**. Alternatively, one could temporarily remove this rule.

```
In[16]:= equal[VERTSECT[x], VERTSECT[y]]

Out[16]= equal[IMAGE[x], IMAGE[y]]

In[17]:= SubstTest[implies, equal[u, v],
          equal[composite[VERTSECT[u], w], composite[VERTSECT[v], w]],
          {u → composite[inverse[E], funpart[x], inverse[S]],
           v → composite[inverse[E], funpart[x]], w → Id}]

Out[17]= or[equal[composite[BIGCUP, IMAGE[funpart[x]], POWER],
            union[cart[complement[domain[funpart[x]]], singleton[0]], funpart[x]]],
           not[equal[composite[inverse[E], funpart[x]],
             composite[inverse[E], funpart[x], inverse[S]]]]] == True

In[18]:= (% /. x → x_) /. Equal → SetDelayed
```

The wrappers can be removed:

```
In[19]:= SubstTest[implies, and[equal[y, funpart[x]], equal[v, domain[y]],
           equal[composite[inverse[E], y], composite[inverse[E], y, inverse[S]]]],
          equal[composite[BIGCUP, IMAGE[y], POWER],
           union[cart[complement[v], singleton[0]], y]], {y → x, v → V}]

Out[19]= or[equal[x, composite[BIGCUP, IMAGE[x], POWER]], not[equal[V, domain[x]]],
           not[equal[composite[inverse[E], x], composite[inverse[E], x, inverse[S]]]],
           not[FUNCTION[x]]] == True

In[20]:= (% /. x → x_) /. Equal → SetDelayed
```

From lemma 3, one obtains this corollary.

```
In[21]:= Map[not, SubstTest[and, implies[p1, p4],
           implies[and[p2, p3, p4], p5], not[implies[and[p1, p2, p3], p5]],
           {p1 → subcommute[x, inverse[S]], p2 → FUNCTION[x],
            p3 → equal[V, domain[x]], p4 ->
             equal[composite[inverse[E], x], composite[inverse[E], x, inverse[S]]],
            p5 -> equal[x, composite[BIGCUP, IMAGE[x], POWER]]}]]

Out[21]= or[equal[x, composite[BIGCUP, IMAGE[x], POWER]],
           not[equal[V, domain[x]]], not[FUNCTION[x]],
           not[subclass[composite[x, inverse[S]], composite[inverse[S], x]]]] == True

In[22]:= (% /. x → x_) /. Equal → SetDelayed
```

---

the main theorem

Putting together the lemmas, one finds a complete characterization of the solutions of the equation.

```
In[23]:= equiv[equal[x, composite[BIGCUP, IMAGE[x], POWER]], and[FUNCTION[x],
             equal[V, domain[x]], subcommute[x, inverse[S]]]] // not // not

Out[23]= True

In[24]:= equal[x_, composite[BIGCUP, IMAGE[x_], POWER]] := and[equal[V, domain[x]],
           FUNCTION[x], subclass[composite[x, inverse[S]], composite[inverse[S], x]]]
```

---

examples from topology

Most **HULL** functions only subcommute with **S**. but a few also subcommute with **inverse[S]**. The condition for this is:

```
In[25]:= subcommute[HULL[x], inverse[S]] // AssertTest

Out[25]= subclass[composite[HULL[x], inverse[S]], composite[inverse[S], HULL[x]]] ==
          or[equal[0, x], equal[V, image[inverse[S], x]]]

In[26]:= subclass[composite[HULL[x_], inverse[S]], composite[inverse[S], HULL[x_]]] :=
          or[equal[0, x], equal[V, image[inverse[S], x]]]
```

One of these functions is **HULL[TOPS]**. As a consequence, the theorem applies to this case:

```
In[27]:= equal[HULL[TOPS], composite[BIGCUP, IMAGE[HULL[TOPS]], POWER]]

Out[27]= True
```

There is currently no rewrite rule for this case, so we can add one:

```
In[28]:= composite[BIGCUP, IMAGE[HULL[TOPS]], POWER] := HULL[TOPS]
```

A variant of this is the case of Hausdorff spaces. A similar rule holds when one adds the **T2** separation condition.

```
In[29]:= Map[implies[#, equal[V, image[inverse[S], intersection[T2, TOPS]]]] &,
          SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u → range[POWER], v → intersection[T2, TOPS], w → inverse[S]}]]

Out[29]= True == equal[V, image[inverse[S], intersection[T2, TOPS]]]
```

```
In[30]:= image[inverse[S], intersection[T2, TOPS]] := V
```

Corollary.

```
In[31]:= equal[HULL[intersection[T2, TOPS]],
          composite[BIGCUP, IMAGE[HULL[intersection[T2, TOPS]]], POWER]]

Out[31]= True
```

Again, one can add this as a new rewrite rule.

```
In[32]:= composite[BIGCUP, IMAGE[HULL[intersection[T2, TOPS]]], POWER] :=
          HULL[intersection[T2, TOPS]]
```

---

comment

Although **HULL[x]** does not in general subcommute with **inverse[S]**, a weakened version of "the equation" does hold for it generally:

```
In[33]:= composite[BIGCUP, IMAGE[HULL[x]], POWER, id[image[inverse[S], x]]]

Out[33]= HULL[x]
```

From this it is clear that the equation holds when **image[inverse[S],x] = V**.