fix[clock[x]]

Johan G. F. Belinfante
2009 November 5

```
In[1]:= SetDirectory["l:"]; << goedel.09nov04b;<< tools.m

        :Package Title: goedel.09nov04b            2009 November 4 at 1:45 a.m.

        It is now:  2009 Nov 5 at 7:30

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 November 2

        weightlimit = 40
```

summary

An explicit formula for **fix[clock[x]]** is derived. A variable-free restatement of this formula is obtained, as well as an explicit formula for **image[inverse[CLOCK], P[Di]]**.

strategy

It will be shown that **fix[clock[x]]** is empty unless **x = 1**, and when **x = 1**, one has **fix[clock[x]] = 1**. Four cases need to be considered: either **x** is not a natural number, or it is **0**, **1**, or a natural number greater than **1**. The cases **x = 0** and **x = 1** require no work.

the case that x is not a number

Lemma.

```
In[2]:= SubstTest[implies, empty[t], empty[fix[t]], t → clock[x]] // Reverse

Out[2]= or[equal[0, fix[clock[x]]], not[equal[0, nat[x]]]] == True
```

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

Theorem. If **x** is not a natural number, then **fix[clock[x]] = 0**.

```
In[4]:= Map[not,
          SubstTest[and, implies[p1, p2], not[implies[p1, p3]], {p1 → not[member[x, omega]],
            p2 → empty[nat[x]], p3 → empty[fix[clock[x]]]}]] // Reverse
```

```
Out[4]= or[equal[0, fix[clock[x]]], member[x, omega]] == True
```

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

---

the case x is a number and x > 1

For this case, the compound wrapper **succ[U[nat[x]]] = union[nat[x], set[0]]** is useful.

Lemma.

```
In[6]:= or[equal[x, union[nat[x], set[0]]],
          not[member[0, x]], not[member[x, omega]]] // AssertTest
```

```
Out[6]= or[equal[x, union[nat[x], set[0]]], not[member[0, x]], not[member[x, omega]]] == True
```

```
In[7]:= or[equal[x_, union[nat[x_], set[0]]], not[member[0, x_]], not[member[x_, omega]]] := True
```

Theorem.

```
In[8]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
            not[implies[p1, p3]], {p1 → and[member[set[0], x], member[x, omega]],
             p2 → member[0, x], p3 -> equal[x, union[nat[x], set[0]]]}]] // Reverse
```

```
Out[8]= or[equal[x, union[nat[x], set[0]]], not[member[x, omega]], not[member[set[0], x]]] == True
```

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

Lemma.

```
In[10]:= equal[intersection[fix[E], nat[x]], 0]
```

```
Out[10]= True
```

```
In[11]:= intersection[fix[E], nat[x_]] := 0
```

Lemma. Temporary rewrite rule.

```
In[12]:= SubstTest[fix, union[u, v],
           {u → cart[set[nat[x]], set[0]], v -> composite[SUCC, id[nat[x]]]}] // Reverse
```

```
Out[12]= fix[clock[succ[nat[x]]]] == intersection[complement[image[V, nat[x]]], set[0]]
```

```
In[13]:= fix[clock[succ[nat[x_]]]] := intersection[complement[image[V, nat[x]]], set[0]]
```

Lemma.

```
In[14]:= SubstTest[fix, clock[succ[nat[t]]], t → U[nat[x]]] // Reverse

Out[14]= fix[clock[union[nat[x], set[0]]]] ==
          intersection[complement[image[V, intersection[complement[set[0]], nat[x]]]], set[0]]

In[15]:= fix[clock[union[nat[x_], set[0]]]] :=
          intersection[complement[image[V, intersection[complement[set[0]], nat[x]]]], set[0]]
```

Theorem. If **x** is a natural number greater than **1**, then **fix[clock[x]] = 0**.

```
In[16]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
            not[implies[p1, p3]], {p1 → and[member[set[0], x], member[x, omega]],
             p2 -> equal[x, union[nat[x], set[0]]], p3 → empty[fix[clock[x]]]}]] // Reverse

Out[16]= or[equal[0, fix[clock[x]]], not[member[x, omega]], not[member[set[0], x]]] == True

In[17]:= (% /. x → x_) /. Equal→ SetDelayed
```

---

an explicit formula for fix[clock[x]]

Two lemmas are needed to derive the explicit formula for **fix[clock[x]]**.

Lemma. The only possible values of **fix[clock[x]]** are **0** and **1**.

```
In[18]:= Map[not, SubstTest[and, implies[p1, p5], implies[p2, p5], implies[p3, p6],
            implies[and[not[p1], p4], p5], not[or[p5, p6]], {p1 → not[member[x, omega]],
             p2 → equal[0, x], p3 → equal[x, set[0]], p4 → member[set[0], x],
             p5 → empty[fix[clock[x]]], p6 → equal[fix[clock[x]], set[0]]}]] // Reverse

Out[18]= subclass[fix[clock[x]], set[0]] == True

In[19]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma. If **clock[x]** is not irreflexive, then **fix[clock[x]] = x**.

```
In[20]:= Map[not, SubstTest[and, implies[p1, p5], implies[p2, p5], implies[p3, p6],
            implies[and[not[p1], p4], p5], not[or[p5, p6]], {p1 → not[member[x, omega]],
             p2 → equal[0, x], p3 → equal[x, set[0]], p4 → member[set[0], x],
             p5 → empty[fix[clock[x]]], p6 → equal[fix[clock[x]], x]}]] // Reverse

Out[20]= or[equal[0, fix[clock[x]]], equal[x, fix[clock[x]]]] == True

In[21]:= (% /. x → x_) /. Equal→ SetDelayed
```

Main Theorem. An explicit formula for **fix[clock[x]]**.

```
In[22]:= equal[fix[clock[x]], intersection[x,
             complement[image[V, intersection[x, complement[set[0]]]]], set[0]]] // not // not

Out[22]= True
```

```
In[23]:= fix[clock[x_]] :=
          intersection[x, complement[image[V, intersection[x, complement[set[0]]]], set[0]]
```

Corollary. A variable-free restatement of the formula for **fix[clock[x]]**.

```
In[24]:= composite[IMAGE[inverse[DUP]], CLOCK] // FastReifNormality

Out[24]= composite[IMAGE[inverse[DUP]], CLOCK] ==
          union[cart[intersection[omega, complement[set[set[0]]]], set[0]],
           cart[set[set[0]], set[set[0]]]]

In[25]:= composite[IMAGE[inverse[DUP]], CLOCK] :=
          union[cart[intersection[omega, complement[set[set[0]]]], set[0]],
           cart[set[set[0]], set[set[0]]]]
```

corollaries

In this section two corollaries are derived.

The following simple result does not require any new rewrite rule.

```
In[26]:= composite[inverse[E], IMAGE[inverse[DUP]], CLOCK]

Out[26]= cart[set[set[0]], set[0]]
```

Theorem. The possible values of **fix[clock[x]]** are **0** and **1**.

```
In[27]:= ImageComp[FIX, CLOCK, V] // Reverse

Out[27]= image[IMAGE[inverse[DUP]], range[CLOCK]] == succ[set[0]]

In[28]:= image[IMAGE[inverse[DUP]], range[CLOCK]] := succ[set[0]]
```

Theorem. Clocks are irreflexive except when **x = 1**.

```
In[29]:= image[inverse[CLOCK], P[Di]] // Normality

Out[29]= image[inverse[CLOCK], P[Di]] == intersection[omega, complement[set[set[0]]]]

In[30]:= image[inverse[CLOCK], P[Di]] := intersection[omega, complement[set[set[0]]]]
```

a wrapper removal rule

In this section, a rule for removing the compound wrapper **succ[U[nat[x]]] = union[nat[x], set[0]]** is derived.

Lemma.

```
In[37]:= SubstTest[member, succ[t], omega, t → U[nat[x]]] // Reverse
Out[37]= member[union[nat[x], set[0]], omega] == True

In[38]:= member[union[nat[x_], set[0]], omega] := True
```

Lemma.

```
In[41]:= SubstTest[implies, equal[x, nat[t]],
           or[equal[0, x], equal[x, union[nat[x], set[0]]]], t → x] // Reverse
Out[41]= or[equal[0, x], equal[x, union[nat[x], set[0]]], not[member[x, omega]]] == True

In[42]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. Wrapper removal rule.

```
In[43]:= equiv[equal[x, succ[U[nat[x]]]], and[member[x, omega], not[empty[x]]]] // not // not
Out[43]= True

In[45]:= equal[x_, union[nat[x_], set[0]]] := and[member[x, omega], not[equal[0, x]]]
```