CLOCK

Johan G. F. Belinfante
2009 November 2

```
In[1]:= SetDirectory["l:"]; << goedel.09oct30b;<< tools.m

        :Package Title: goedel.09oct30b             2009 October 30 at 4:20 p.m.

        It is now:  2009 Nov 2 at 13:30

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 October 30

        weightlimit = 40
```

summary

The function **CLOCK** which takes any natural number **x** to the one-to-one function **clock[x]** is introduced, and some of its properties are derived.

definition

The function **CLOCK** is defined by the following membership rule.

```
In[2]:= member[x_, CLOCK] := and[equal[clock[first[x]], second[x]], member[first[x], omega]]
```

vertical sections

Lemma.

```
In[3]:= modulo[union[x, image[V, y]]] // FastReifNormality

Out[3]= modulo[union[x, image[V, y]]] == composite[id[complement[image[V, y]]], modulo[x]]

In[4]:= modulo[union[x_, image[V, y_]]] := composite[id[complement[image[V, y]]], modulo[x]]

In[5]:= ApComp[modulo[x], composite[SUCC, id[x]], y] // Reverse

Out[5]= APPLY[clock[x], y] ==
         union[complement[image[V, intersection[x, set[y]]]], natmod[succ[y], x]]

In[6]:= APPLY[clock[x_], y_] :=
         union[complement[image[V, intersection[x, set[y]]]], natmod[succ[y], x]]
```

Lemma.

```
In[7]:= clock[union[x, image[V, y]]] // FastReifNormality
Out[7]= clock[union[x, image[V, y]]] == composite[id[complement[image[V, y]]], clock[x]]

In[8]:= clock[union[x_, image[V, y_]]] := composite[id[complement[image[V, y]]], clock[x]]
```

Lemma.

```
In[9]:= SubstTest[clock, union[x, image[V, t]], t → complement[image[V, y]]] // Reverse
Out[9]= clock[union[x, complement[image[V, y]]]] == composite[id[image[V, y]], clock[x]]

In[10]:= clock[union[x_, complement[image[V, y_]]]] := composite[id[image[V, y]], clock[x]]
```

Theorem. Vertical section rule.

```
In[11]:= image[CLOCK, set[x]] // Normality
Out[11]= image[CLOCK, set[x]] == intersection[image[V, intersection[omega, set[x]]],
           set[composite[id[image[V, set[x]]], clock[x]]]]

In[12]:= image[CLOCK, set[x_]] := intersection[image[V, intersection[omega, set[x]]],
           set[composite[id[image[V, set[x]]], clock[x]]]]
```

FUNCTION rule

Lemma.

```
In[14]:= Map[empty, dif[CLOCK, cart[V, V]] // Normality]
Out[14]= subclass[CLOCK, cart[V, V]] == True

In[15]:= subclass[CLOCK, cart[V, V]] := True
```

Theorem.

```
In[16]:= equal[composite[Id, CLOCK], CLOCK]
Out[16]= True

In[17]:= composite[Id, CLOCK] := CLOCK
```

Theorem.

```
In[18]:= Map[FUNCTION, SubstTest[reify, x, image[t, set[setpart[x]]], t → CLOCK]]
Out[18]= FUNCTION[CLOCK] == True

In[19]:= FUNCTION[CLOCK] := True
```

Corollary. **APPLY** rule.

```
In[20]:= Map[A, SubstTest[image, funpart[t], set[x], t → CLOCK]]

Out[20]= APPLY[CLOCK, x] == union[complement[image[V, intersection[omega, set[x]]]],
           composite[id[image[V, set[x]]], clock[x]]]

In[21]:= APPLY[CLOCK, x_] := union[complement[image[V, intersection[omega, set[x]]]],
           composite[id[image[V, set[x]]], clock[x]]]
```

---

mapping rule

Lemma.

```
In[22]:= Map[not, SubstTest[empty, image[t, set[x]], t → CLOCK]]

Out[22]= member[x, domain[CLOCK]] == member[x, omega]

In[23]:= member[x_, domain[CLOCK]] := member[x, omega]
```

Theorem.

```
In[24]:= domain[CLOCK] // Normality

Out[24]= domain[CLOCK] == omega

In[25]:= domain[CLOCK] := omega
```

Lemma.

```
In[32]:= (member[s, image[inverse[funpart[t]], y]] // AssertTest) /. {s → nat[x], t → CLOCK}

Out[32]= member[nat[x], image[inverse[CLOCK], y]] == member[clock[nat[x]], y]

In[33]:= member[nat[x_], image[inverse[CLOCK], y_]] := member[clock[nat[x]], y]
```

Lemma.

```
In[34]:= Map[equal[V, #] &,
          SubstTest[class, x, member[nat[x], t], t -> image[inverse[CLOCK], BIJ]]]

Out[34]= subclass[omega, image[inverse[CLOCK], BIJ]] == True

In[35]:= % /. Equal→ SetDelayed
```

Theorem.

```
In[36]:= equal[image[inverse[CLOCK], BIJ], omega] // AssertTest

Out[36]= equal[omega, image[inverse[CLOCK], BIJ]] == True

In[37]:= image[inverse[CLOCK], BIJ] := omega
```

Corollary.

```
In[38]:= Map[equal[#, range[CLOCK]] &, ImageComp[CLOCK, inverse[CLOCK], BIJ]]

Out[38]= subclass[range[CLOCK], BIJ] == True

In[39]:= subclass[range[CLOCK], BIJ] := True
```

Lemma.

```
In[40]:= member[CLOCK, V] // AssertTest

Out[40]= member[CLOCK, V] == True

In[41]:= member[CLOCK, V] := True
```

Theorem.

```
In[42]:= member[CLOCK, map[omega, BIJ]] // AssertTest

Out[42]= member[CLOCK, map[omega, BIJ]] == True

In[43]:= member[CLOCK, map[omega, BIJ]] := True
```

---

normalization rule for CLOCK

```
In[44]:= composite[inverse[E], IMAGE[SWAP], CLOCK] // FastReifNormality// Reverse

Out[44]= composite[RIF, id[cart[V, SUCC]],
           intersection[composite[inverse[FIRST], SWAP, inverse[rotate[NATMOD]]],
            composite[inverse[SECOND], inverse[FIRST], inverse[E]]]] ==
          composite[inverse[E], IMAGE[SWAP], CLOCK]

In[45]:= composite[RIF, id[cart[V, SUCC]],
           intersection[composite[inverse[FIRST], SWAP, inverse[rotate[NATMOD]]],
            composite[inverse[SECOND], inverse[FIRST], inverse[E]]]] :=
          composite[inverse[E], IMAGE[SWAP], CLOCK]
```

Corollary.

```
In[46]:= CLOCK // FastReifNormality// Reverse

Out[46]= composite[IMAGE[id[cart[V, V]]], CLOCK] == CLOCK

In[47]:= composite[IMAGE[id[cart[V, V]]], CLOCK] := CLOCK
```

Corollary.

```
In[48]:= Assoc[id[P[cart[V, V]]], IMAGE[id[cart[V, V]]], CLOCK]

Out[48]= composite[id[P[cart[V, V]]], CLOCK] == CLOCK
```

```
In[49]:= composite[id[P[cart[V, V]]], CLOCK] := CLOCK
```

properties of clocks

Theorem.

```
In[50]:= composite[IMAGE[FIRST], CLOCK] // FastReifNormality
Out[50]= composite[IMAGE[FIRST], CLOCK] == id[omega]

In[51]:= composite[IMAGE[FIRST], CLOCK] := id[omega]
```

Theorem.

```
In[52]:= composite[IMAGE[SECOND], CLOCK] // FastReifNormality
Out[52]= composite[IMAGE[SECOND], CLOCK] == id[omega]

In[53]:= composite[IMAGE[SECOND], CLOCK] := id[omega]
```

Theorem.

```
In[54]:= composite[DORA, CLOCK] // FastReifNormality
Out[54]= composite[DORA, CLOCK] == composite[DUP, id[omega]]

In[55]:= composite[DORA, CLOCK] := composite[DUP, id[omega]]
```

empty clocks

Lemma.

```
In[56]:= SubstTest[empty, domain[funpart[t]], t → clock[x]]
Out[56]= equal[0, clock[x]] == equal[0, nat[x]]

In[57]:= equal[0, clock[x_]] := equal[0, nat[x]]
```

Theorem.

```
In[58]:= image[inverse[CLOCK], set[0]] // Normality
Out[58]= image[inverse[CLOCK], set[0]] == set[0]

In[59]:= image[inverse[CLOCK], set[0]] := set[0]
```

Theorem.

In[60]:= **Map[member[0, #] &, ImageComp[CLOCK, inverse[CLOCK], set[0]]]**

Out[60]= member[0, range[CLOCK]] == True

In[61]:= **member[0, range[CLOCK]] := True**

finiteness and other properties

Theorem.

In[78]:= **Map[empty, SubstTest[reify, x, dif[set[clock[x]], t], t -> FINITE]]**

Out[78]= subclass[range[CLOCK], FINITE] == True

In[81]:= **subclass[range[CLOCK], FINITE] := True**

Theorem.

In[84]:= **Map[empty, SubstTest[reify, x, dif[set[clock[x]], t], t → LISTS]]**

Out[84]= subclass[range[CLOCK], LISTS] == True

In[85]:= **subclass[range[CLOCK], LISTS] := True**

Theorem.

In[89]:= **Map[empty, SubstTest[reify, x, dif[set[clock[x]], t], t → UNOPS]]**

Out[89]= subclass[range[CLOCK], UNOPS] == True

In[90]:= **subclass[range[CLOCK], UNOPS] := True**