an explicit formula for LUB[id[ω] ∘ S ∘ id[ω]]

Johan G. F. Belinfante
2010 September 19

```
In[1]:= SetDirectory["l:"]; << goedel.10sep18a

        :Package Title: goedel.10sep18a            2010 September 18 at 8:35 a.m.

        It is now:  2010 Sep 19 at 5:1

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

In this notebook is shown that the **LUB** function for the restriction **id[ω] ∘ S ∘ id[ω]** of the subset relation **S** to the set ω of natural numbers is equal to the restriction of the function **BIGCUP** to the set **image[inverse[S], ω]** of finite subsets of ω. The basic idea of the derivation is that two functions are equal if one is a subclass of the other, and their domains are equal. The fact that the relation **id[ω] ∘ S ∘ id[ω]** is conditionally complete is used to obtain a formula for the domain of its **LUB** function.

the domain of the LUB function

In this section a temporary rule for the domain of the **LUB** function is derived.

Lemma. A simplification rule.

```
In[2]:= equal[intersection[image[inverse[BIGCUP], image[inverse[S], omega]], P[omega]],
          image[inverse[S], omega]] // AssertTest

Out[2]= equal[image[inverse[S], omega],
          intersection[image[inverse[BIGCUP], image[inverse[S], omega]], P[omega]]] == True

In[3]:= intersection[image[inverse[BIGCUP], image[inverse[S], omega]], P[omega]] :=
         image[inverse[S], omega]
```

Since **id[ω] ∘ S ∘ id[ω]** is a well ordering, it is conditionally complete. A lower bound on the domain of **LUB** is obtained using this fact. (Here **CC** is the class of all conditionally complete relations.)

Theorem. A lower bound on the domain of **LUB**.

```
In[4]:= SubstTest[implies, member[x, CC], subclass[domain[UB[x]],
          union[set[0], domain[LUB[x]]]], x → restrict[S, omega, omega]] // Reverse

Out[4]= subclass[image[inverse[S], omega],
          domain[LUB[composite[id[omega], S, id[omega]]]]] == True

In[5]:= % /. Equal→ SetDelayed
```

Not every subset of ω has an upper bound, let alone a least upper bound.

Theorem. An upper bound on the domain of **LUB**.

```
In[6]:= SubstTest[subclass, domain[LUB[x]],
          domain[UB[x]], x -> composite[id[omega], S, id[omega]]] // Reverse

Out[6]= subclass[domain[LUB[composite[id[omega], S, id[omega]]]],
          image[inverse[S], omega]] == True

In[7]:= % /. Equal→ SetDelayed
```

Theorem. An explicit formula for the domain of **LUB**.

```
In[8]:= SubstTest[and, subclass[u, v], subclass[v, u],
         {u -> image[inverse[S], omega], v -> domain[LUB[composite[id[omega], S, id[omega]]]]}]

Out[8]= equal[domain[LUB[composite[id[omega], S, id[omega]]]], image[inverse[S], omega]] == True

In[9]:= domain[LUB[composite[id[omega], S, id[omega]]]] := image[inverse[S], omega]
```

---

an explicit formula for the LUB function

The expression $\omega \bigcap$ **fin[t]** serves as a compound wrapper for a finite set of natural numbers.

Lemma.

```
In[10]:= equal[intersection[omega, U[intersection[omega, x]]], U[intersection[omega, x]]]

Out[10]= True

In[11]:= intersection[omega, U[intersection[omega, x_]]] := U[intersection[omega, x]]
```

Lemma.

```
In[12]:= Map[not, SubstTest[implies, member[t, image[inverse[S], omega]],
           not[equal[U[t], omega]], t → intersection[fin[x], omega]]] // Reverse

Out[12]= equal[omega, U[intersection[omega, fin[x]]]] == False

In[13]:= equal[omega, U[intersection[omega, fin[x_]]]] := False
```

Lemma. A special case of a theorem about ordinals.

```
In[14]:= Map[not, SubstTest[and, implies[p1, p2],
           implies[p2, p3], not[implies[p1, p3]], {p1 → subclass[x, omega],
            p2 → subclass[x, OMEGA], p3 -> subclass[x, succ[U[x]]]}]] // Reverse

Out[14]= or[not[subclass[x, omega]], subclass[x, succ[U[x]]]] == True

In[15]:= or[not[subclass[x_, omega]], subclass[x_, succ[U[x_]]]] := True
```

Corollary. A wrapper version of the preceding theorem.

```
In[16]:= SubstTest[implies, subclass[t, omega],
           subclass[t, succ[U[t]]], t → intersection[x, omega]] // Reverse

Out[16]= subclass[intersection[omega, x], succ[U[intersection[omega, x]]]] == True

In[17]:= subclass[intersection[omega, x_], succ[U[intersection[omega, x_]]]] := True
```

Lemma.

```
In[18]:= SubstTest[implies, member[t, REGULAR],
           member[intersection[t, x], REGULAR], t → omega] // Reverse

Out[18]= member[intersection[omega, x], REGULAR] == True

In[19]:= member[intersection[omega, x_], REGULAR] := True
```

Lemma. A consequence of regularity.

```
In[20]:= SubstTest[member, reg[t], image[inverse[S], reg[t]],
           t -> U[intersection[omega, x]]] // Reverse

Out[20]= member[U[intersection[omega, x]], image[inverse[S], U[intersection[omega, x]]]] == False

In[21]:= member[U[intersection[omega, x_]],
           image[inverse[S], U[intersection[omega, x_]]]] := False
```

Lemma.

```
In[22]:= (SubstTest[implies, and[member[x, V], member[y, lub[z, x]]],
            member[pair[x, y], LUB[z]], {y → U[x], z → restrict[S, omega, omega]}] /.
           x → intersection[omega, fin[t]]) // Reverse

Out[22]= member[pair[intersection[omega, fin[t]], U[intersection[omega, fin[t]]]],
           LUB[composite[id[omega], S, id[omega]]]] == True

In[23]:= (% /. t → t_) /. Equal → SetDelayed
```

Lemma. Removal of the compound wrapper $\omega \bigcap$ **fin[t]**.

```
In[24]:= SubstTest[implies, equal[x, intersection[omega, fin[t]]],
           member[pair[x, U[x]], LUB[composite[id[omega], S, id[omega]]]], t → x] // Reverse

Out[24]= or[member[pair[x, U[x]], LUB[composite[id[omega], S, id[omega]]]],
           not[member[x, FINITE]], not[subclass[x, omega]]] == True
```

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

Lemma. Eliminating variables.

```
In[26]:= Map[composite[Id, complement[#]] &,
          SubstTest[class, pair[x, y], implies[member[pair[x, y], u], member[pair[x, y], v]],
           {u -> composite[BIGCUP, id[image[inverse[S], omega]]],
            v -> LUB[composite[id[omega], S, id[omega]]]}]]

Out[26]= composite[BIGCUP, id[intersection[complement[
             fix[composite[inverse[BIGCUP], LUB[composite[id[omega], S, id[omega]]]]]],
            image[inverse[S], omega]]]] == 0

In[27]:= % /. Equal→ SetDelayed
```

Lemma. An inclusion.

```
In[28]:= SubstTest[empty, dif[u, v], {u -> composite[BIGCUP, id[image[inverse[S], omega]]],
            v -> LUB[composite[id[omega], S, id[omega]]]}]

Out[28]= subclass[composite[BIGCUP, id[image[inverse[S], omega]]],
            LUB[composite[id[omega], S, id[omega]]]] == True

In[29]:= % /. Equal→ SetDelayed
```

Main Theorem. An explicit formula for the **LUB** function of the relation **id[ω] ∘ S ∘ id[ω]**.

```
In[30]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
            equal[u, composite[v, id[domain[u]]]],
            {u -> composite[BIGCUP, id[image[inverse[S], omega]]],
             v -> LUB[composite[id[omega], S, id[omega]]]}] // Reverse

Out[30]= equal[composite[BIGCUP, id[image[inverse[S], omega]]],
            LUB[composite[id[omega], S, id[omega]]]] == True

In[31]:= LUB[composite[id[omega], S, id[omega]]] :=
          composite[BIGCUP, id[image[inverse[S], omega]]]
```