greater and lesser of two natural numbers

Johan G. F. Belinfante
2005 April 28

```
In[1]:= SetDirectory["i:"]; << goedel68.28a; << tools.m

        :Package Title: goedel68.28a         2005 April 28 at 9:15 a.m.

        It is now:  2005 Apr 28 at 12:8

        Loading Simplification Rules

        TOOLS.M                       Revised 2005 April 16

        weightlimit = 40
```

summary

This notebook is concerned with deriving counterparts within Gödel's class theory of a several of the theorems about **max** and **min** proved by Art Quaife many years ago, using William McCune's program **Otter**. By using the natural number wrapper **nat**, the statements of the results closely resemble Quaife's formulas. Since each natural number in Gödel's class theory is the set of all lesser numbers, the lesser and greater of two natural numbers are their intersection and union, respectively. As a consequence there is no need to introduce separate constructors for **max** and **min**. Some of Quaife's theorems become consequences of the laws of Boolean algebra, while the rest require completely different proofs.

```
In[2]:= "Art Quaife, Automated Development of Fundamental Mathematical
          Theories, Kluwer Academic Publishers, 1992.  (See pages 192–194)."

Out[2]= Art Quaife, Automated Development of Fundamental Mathematical
          Theories, Kluwer Academic Publishers, 1992.  (See pages 192–194).
```

min and max are numbers

```
In[3]:= SubstTest[or, member[intersection[u, v], omega],
         not[member[u, omega]], not[member[v, omega]], {u → nat[x], v → nat[y]}]

Out[3]= member[intersection[nat[x], nat[y]], omega] == True
```

```
In[4]:= member[intersection[nat[x_], nat[y_]], omega] := True

In[5]:= SubstTest[or, member[union[u, v], omega], not[member[u, omega]],
         not[member[v, omega]], {u → nat[x], v → nat[y]}]

Out[5]= member[union[nat[x], nat[y]], omega] == True

In[6]:= member[union[nat[x_], nat[y_]], omega] := True
```

---

two subclass rewrite rules

Rule 1.

```
In[7]:= SubstTest[subclass, nat[w], nat[z], w → intersection[nat[x], nat[y]]]

Out[7]= subclass[intersection[nat[x], nat[y]], nat[z]] ==
         or[not[member[nat[z], nat[x]]], not[member[nat[z], nat[y]]]]

In[8]:= subclass[intersection[nat[x_], nat[y_]], nat[z_]] :=
         or[not[member[nat[z], nat[x]]], not[member[nat[z], nat[y]]]]
```

Rule 2.

```
In[9]:= SubstTest[subclass, nat[x], nat[w], w → union[nat[y], nat[z]]]

Out[9]= subclass[nat[x], union[nat[y], nat[z]]] ==
         not[member[union[nat[y], nat[z]], nat[x]]]

In[10]:= subclass[nat[x_], union[nat[y_], nat[z_]]] :=
          not[member[union[nat[y], nat[z]], nat[x]]]
```

Quaife's Theorem (MM5)

Only one of Quaife's four clauses needs to be derived.

```
In[11]:= Map[not, SubstTest[subclass, nat[x], nat[w], w → union[nat[x], nat[y]]]] //
          Reverse

Out[11]= member[union[nat[x], nat[y]], nat[x]] == False

In[12]:= member[union[nat[x_], nat[y_]], nat[x_]] := False
```

Quaife's (MM8)

Lemma.

```
In[13]:= SubstTest[implies, and[equal[nat[x], nat[u]], equal[nat[y], nat[v]]],
          equal[natadd[nat[x], nat[y]], natadd[nat[u], nat[v]]],
          {v → intersection[nat[x], nat[y]], u → union[nat[x], nat[y]]}]

Out[13]= or[equal[natadd[intersection[nat[x], nat[y]], union[nat[x], nat[y]]],
            natadd[nat[x], nat[y]]], member[nat[x], nat[y]]] == True

In[14]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed

Out[14]= 
```

Theorem.

```
In[15]:= Map[not, SubstTest[and, implies[p1, p4], or[p1, p2, p3],
           or[p2, p4], or[p3, p4], not[p4], {p1 → equal[nat[x], nat[y]],
            p2 -> member[nat[x], nat[y]], p3 -> member[nat[y], nat[x]],
            p4 -> equal[natadd[intersection[nat[x], nat[y]], union[nat[x], nat[y]]],
              natadd[nat[x], nat[y]]]}]]

Out[15]= equal[natadd[intersection[nat[x], nat[y]], union[nat[x], nat[y]]],
           natadd[nat[x], nat[y]]] == True

In[16]:= natadd[intersection[nat[x_], nat[y_]], union[nat[x_], nat[y_]]] :=
          natadd[nat[x], nat[y]]
```

Quaife's (MM9)

Only one of Quaife's clauses needs to be derived.

```
In[17]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → nat[y], v → nat[x], w → union[nat[x], nat[z]]}]

Out[17]= or[member[nat[x], nat[y]], not[member[union[nat[x], nat[z]], nat[y]]]] == True

In[18]:= or[member[nat[x_], nat[y_]],
           not[member[union[nat[x_], nat[z_]], nat[y_]]]] := True
```

Quaife's Schematic Theorem (MM10)

Quaife's schematic theorem will be reformulated here as an ordinary theorem within first order logic by replacing Quaife's generic predicate **phi** with an arbitrary class **z**. Statements of the form **phi[x]** are replaced with **member[nat[x], z]**.

```
In[19]:= (implies[and[member[u, z], member[v, z], or[equal[u, w], equal[v, w]]],
            member[w, z]] // NotNotTest) /.
          {u → nat[x], v → nat[y], w → intersection[nat[x], nat[y]]}

Out[19]= or[member[intersection[nat[x], nat[y]], z],
           not[member[nat[x], z]], not[member[nat[y], z]]] == True

In[20]:= or[member[intersection[nat[x_], nat[y_]], z_],
           not[member[nat[x_], z_]], not[member[nat[y_], z_]]] := True

In[21]:= (implies[and[member[u, z], member[v, z], or[equal[u, w], equal[v, w]]],
            member[w, z]] // NotNotTest) /.
          {u → nat[x], v → nat[y], w → union[nat[x], nat[y]]}

Out[21]= or[member[union[nat[x], nat[y]], z],
           not[member[nat[x], z]], not[member[nat[y], z]]] == True

In[22]:= or[member[union[nat[x_], nat[y_]], z_],
           not[member[nat[x_], z_]], not[member[nat[y_], z_]]] := True
```