binclosed[Id]

Johan G. F. Belinfante
2013 October 12

```
In[1]:= SetDirectory["l:"]; << goedel.13oct07a

        :Package Title: goedel.13oct07a                  2013 October 7 at 8:25 a.m.

        Loading takes about seventeen minutes, half that time due to builtin pauses.

        It is now:  2013 Oct 12 at 18:28

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2013 Oct 12 at 18:45
```

summary

Gödel's proof of the consistency of the axiom of choice and of the generalized continuum hypothesis uses an inner model of set theory that is closed under a finite number of constructors commonly called Gödel operations. In the **GOEDEL** program the ordered pair is taken as an additional primitive instead of being defined in terms of unordered pairs as suggested by Kuratowski. One would therefore need to add **pair[x, y]** to Gödel's list of basic constructors. The class **binclosed[Id]** is the class of all sets that are binary closed under the formation of ordered pairs. Rewrite rules for this class are derived in this notebook.

derivation

Theorem. Every set is a subset of a set that is binary closed under the formation of ordered pairs.

```
In[2]:= SubstTest[image, inverse[S], binclosed[thinpart[x]], x → Id] // Reverse

Out[2]= image[inverse[S], binclosed[Id]] == V

In[3]:= image[inverse[S], binclosed[Id]] := V
```

One can be more explicit about this.

Observation. If **x** is a set, then so **hull[binclosed[Id], x]**, and conversely.

```
In[4]:= member[hull[binclosed[Id], x], V]

Out[4]= member[x, V]
```

If **x** is not a set, then **hull[binclosed[Id], x] = V**.

```
In[5]:= equal[V, hull[binclosed[Id], x]]

Out[5]= not[member[x, V]]
```

Observation. The inclusion **x ⊂ hull[binclosed[Id], x]** holds.

```
In[6]:= subclass[x, hull[binclosed[Id], x]]

Out[6]= True
```

Lemma.

```
In[7]:= SubstTest[implies, and[not[empty[x]], subclass[x, binclosed[y]]],
          member[A[x], binclosed[y]], y → Id] // Reverse

Out[7]= or[equal[0, x], not[subclass[x, binclosed[Id]]], subclass[cart[A[x], A[x]], A[x]]] == True

In[8]:= (% /. x → x_) /. Equal → SetDelayed
```

A redundant literal can be removed.

Theorem. Every subclass **x ⊂ binclosed[Id]** satisfies **A[x] × A[x] ⊂ A[x]**.

```
In[9]:= SubstTest[and, implies[p, q], or[p, q], {p → equal[0, x],
          q -> or[not[subclass[x, binclosed[Id]]], subclass[cart[A[x], A[x]], A[x]]]}]

Out[9]= or[not[subclass[x, binclosed[Id]]], subclass[cart[A[x], A[x]], A[x]]] == True

In[10]:= or[not[subclass[x_, binclosed[Id]]], subclass[cart[A[x_], A[x_]], A[x_]]] := True
```

Corollary. The class **y = hull[binclosed[Id], x]** satisfies **y × y ⊂ y**.

```
In[11]:= SubstTest[implies, subclass[t, binclosed[Id]], subclass[cart[A[t], A[t]], A[t]],
           t → intersection[binclosed[Id], image[S, set[x]]]] // Reverse

Out[11]= subclass[cart[hull[binclosed[Id], x], hull[binclosed[Id], x]],
           hull[binclosed[Id], x]] == True

In[12]:= subclass[cart[hull[binclosed[Id], x_], hull[binclosed[Id], x_]],
           hull[binclosed[Id], x_]] := True
```

Theorem. If a set **y** is binary closed under **w** and contains **x,** then **hull[binclosed[w], x] ⊂ y**.

```
In[13]:= Map[implies[member[y, z], #] &, SubstTest[implies, and[member[y, t], subclass[x, y]],
            subclass[hull[t, x], y], t → binclosed[w]]] // Reverse

Out[13]= or[not[member[y, z]], not[subclass[x, y]],
           not[subclass[image[w, cart[y, y]], y]], subclass[hull[binclosed[w], x], y]] == True
```

```
In[14]:= or[not[member[y_, z_]], not[subclass[image[w_, cart[y_, y_]], y_]],
           not[subclass[x_, y_]], subclass[hull[binclosed[w_], x_], y_]] := True
```

Corollary. If $\mathbf{x}$ is a subset of a set $\mathbf{y}$ satisfying $\mathbf{y} \times \mathbf{y} \subset \mathbf{y}$, then **hull[binclosed[Id], x]** $\subset \mathbf{y}$.

```
In[15]:= SubstTest[implies, and[member[y, z], subclass[x, y], subclass[image[w, cart[y, y]], y]],
           subclass[hull[binclosed[w], x], y], w → Id] // Reverse

Out[15]= or[not[member[y, z]], not[subclass[x, y]],
           not[subclass[cart[y, y], y]], subclass[hull[binclosed[Id], x], y]] == True

In[16]:= or[not[member[y_, z_]], not[subclass[cart[y_, y_], y_]],
           not[subclass[x_, y_]], subclass[hull[binclosed[Id], x_], y_]] := True
```

various inclusions and equations

Theorem.

```
In[17]:= SubstTest[implies, subclass[u, v],
           subclass[binclosed[v], binclosed[u]], {u → id[x], v → Id}] // Reverse

Out[17]= subclass[binclosed[Id], binclosed[id[x]]] == True

In[18]:= subclass[binclosed[Id], binclosed[id[x_]]] := True
```

Theorem.

```
In[19]:= SubstTest[binclosed, composite[x, id[cart[V, V]]], x → Id] // Reverse

Out[19]= binclosed[id[cart[V, V]]] == binclosed[Id]

In[20]:= binclosed[id[cart[V, V]]] := binclosed[Id]
```

Theorem. A general result.

```
In[21]:= SubstTest[subclass, intersection[invar[x], binclosed[y]],
           binclosed[composite[x, y]], y → Id] // Reverse

Out[21]= subclass[intersection[binclosed[Id], invar[x]], binclosed[x]] == True

In[22]:= subclass[intersection[binclosed[Id], invar[x_]], binclosed[x_]] := True
```

Theorem.

```
In[23]:= SubstTest[binclosed, composite[t, inverse[DUP]], t → DUP] // Reverse

Out[23]= binclosed[id[Id]] == invar[DUP]

In[24]:= binclosed[id[Id]] := invar[DUP]
```

Corollary.

```
In[25]:= SubstTest[subclass, binclosed[Id], binclosed[id[x]], x → Id] // Reverse

Out[25]= subclass[binclosed[Id], invar[DUP]] == True

In[26]:= subclass[binclosed[Id], invar[DUP]] := True
```

an equation for HULL[binclosed[Id]]

Theorem.

```
In[27]:= SubstTest[composite, BIGCUP,
           IMAGE[HULL[binclosed[thinpart[x]]]], POWER, x → Id] // Reverse

Out[27]= composite[BIGCUP, IMAGE[HULL[binclosed[Id]]], POWER] == HULL[binclosed[Id]]

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

Corollary.

```
In[29]:= ApComp[BIGCUP, composite[IMAGE[HULL[binclosed[Id]]], POWER], setpart[x]]

Out[29]= U[image[HULL[binclosed[Id]], P[setpart[x]]]] == hull[binclosed[Id], setpart[x]]

In[30]:= U[image[HULL[binclosed[Id]], P[setpart[x_]]]] := hull[binclosed[Id], setpart[x]]
```

Corollary.

```
In[31]:= Map[implies[member[x, y], #] &, SubstTest[implies, equal[x, setpart[t]], equal[
             U[image[HULL[binclosed[Id]], P[x]]], hull[binclosed[Id], x]], t → x]] // Reverse

Out[31]= or[equal[hull[binclosed[Id], x], U[image[HULL[binclosed[Id]], P[x]]]],
           not[member[x, y]]] == True

In[32]:= or[equal[hull[binclosed[Id], x_], U[image[HULL[binclosed[Id]], P[x_]]]],
           not[member[x_, y_]]] := True
```