IRC

Johan G. F. Belinfante
2014 February 1

```
In[1]:= SetDirectory["l:"]; << goedel.14jan31a

        :Package Title: goedel.14jan31a                 2014 January 31 at 4:00 p.m.

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

        It is now:  2014 Feb 1 at 10:15

        Loading Simplification Rules

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

        weightlimit = 40

        Loading completed.

        It is now:  2014 Feb 1 at 10:32
```

summary

Rewrite rules involving **IRC** are derived.

BIGCUP rules

Observation.

```
In[2]:= U[image[RC[U[x]], x]]

Out[2]= intersection[complement[A[x]], image[V, set[x]], U[x]]
```

Theorem.

```
In[3]:= composite[BIGCUP, IRC] // FastReifNormality // Reverse

Out[3]= VERTSECT[intersection[composite[complement[inverse[E]], BIGCAP],
          composite[inverse[E], BIGCUP]]] == composite[BIGCUP, IRC]

In[4]:= VERTSECT[intersection[composite[complement[inverse[E]], BIGCAP],
          composite[inverse[E], BIGCUP]]] := composite[BIGCUP, IRC]
```

Corollary.

```
In[5]:= composite[DIF, intersection[composite[inverse[FIRST], BIGCUP],
          composite[inverse[SECOND], BIGCAP]]] // FastReifNormality

Out[5]= composite[DIF, intersection[composite[inverse[FIRST], BIGCUP],
          composite[inverse[SECOND], BIGCAP]]] == composite[BIGCUP, IRC, id[complement[set[0]]]]

In[6]:= composite[DIF, intersection[
          composite[inverse[FIRST], BIGCUP], composite[inverse[SECOND], BIGCAP]]] :=
         composite[BIGCUP, IRC, id[complement[set[0]]]]
```

Restatement.

```
In[7]:= composite[DIF, cross[BIGCUP, BIGCAP], DUP]

Out[7]= composite[BIGCUP, IRC, id[complement[set[0]]]]
```

BIGCAP rules

Observation.

```
In[8]:= A[image[RC[U[x]], x]]

Out[8]= union[complement[image[V, x]], complement[image[V, set[x]]]]
```

Theorem.

```
In[9]:= composite[BIGCAP, IRC] // FastReifNormality

Out[9]= composite[BIGCAP, IRC] == cart[complement[set[0]], set[0]]

In[10]:= composite[BIGCAP, IRC] := cart[complement[set[0]], set[0]]
```

Observation.

```
In[12]:= class[x, equal[A[x], 0]]

Out[12]= image[inverse[BIGCAP], set[0]]
```

Observation.

```
In[13]:= implies[and[member[x, V], equal[A[x], 0]], equal[U[image[RC[U[x]], x]], U[x]]] // not //
          not

Out[13]= True
```

Lemma.

```
In[14]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[implies[and[member[x, V], equal[A[x], 0]],
             equal[U[APPLY[funpart[t], x]], U[x]]]], t → IRC]]

Out[14]= subclass[image[inverse[BIGCAP], set[0]],
           fix[composite[inverse[S], POWER, BIGCUP, IRC]]] == True

In[15]:= % /. Equal → SetDelayed
```

Lemma.

```
In[16]:= dif[composite[BIGCUP, IRC, id[image[inverse[BIGCAP], set[0]]]], BIGCUP] // VSNormality

Out[16]= composite[intersection[composite[BIGCUP, IRC], composite[Di, BIGCUP]],
           id[image[inverse[BIGCAP], set[0]]]] == 0

In[17]:= % /. Equal → SetDelayed
```

Theorem.

```
In[18]:= SubstTest[empty, dif[u, v],
          {u -> composite[BIGCUP, IRC, id[image[inverse[BIGCAP], set[0]]]], v -> BIGCUP}]

Out[18]= subclass[composite[BIGCUP, IRC, id[image[inverse[BIGCAP], set[0]]]], BIGCUP] == True

In[19]:= % /. Equal → SetDelayed
```

Theorem.

```
In[20]:= SubstTest[implies, and[subclass[u, v], FUNCTION[v]],
           equal[u, composite[v, id[domain[u]]]],
           {u -> composite[BIGCUP, IRC, id[image[inverse[BIGCAP], set[0]]]],
            v -> BIGCUP}] // Reverse

Out[20]= equal[composite[BIGCUP, id[image[inverse[BIGCAP], set[0]]]],
           composite[BIGCUP, IRC, id[image[inverse[BIGCAP], set[0]]]]] == True

In[21]:= composite[BIGCUP, IRC, id[image[inverse[BIGCAP], set[0]]]] :=
          composite[BIGCUP, id[image[inverse[BIGCAP], set[0]]]]
```

---

fix[IRC]

Theorem.

```
In[22]:= Map[subclass[#, fix[IRC]] &,
           SubstTest[fix, composite[x, id[y]], {x → IRC, y → range[POWER]}]] // Reverse

Out[22]= subclass[range[POWER], fix[IRC]] == True

In[23]:= subclass[range[POWER], fix[IRC]] := True
```

Theorem. Membership rule for **IRC**.

```
In[24]:= SubstTest[member, y, image[t, set[x]], t → IRC]

Out[24]= member[pair[x, y], IRC] == and[equal[y, image[RC[U[x]], x]], member[x, V], member[y, V]]

In[25]:= member[pair[x_, y_], IRC] :=
          and[equal[y, image[RC[U[x]], x]], member[x, V], member[y, V]]
```

Theorem.

```
In[26]:= member[P[x], fix[IRC]] // AssertTest

Out[26]= member[P[x], fix[IRC]] == member[x, V]

In[27]:= member[P[x_], fix[IRC]] := member[x, V]
```

Theorem.

```
In[28]:= (member[x, fix[t]] // AssertTest) /. {x → 0, t → IRC}

Out[28]= member[0, fix[IRC]] == True

In[29]:= member[0, fix[IRC]] := True
```

Theorem.

```
In[30]:= (member[x, fix[t]] // AssertTest) /. {x → set[0], t → IRC}

Out[30]= member[set[0], fix[IRC]] == True

In[31]:= member[set[0], fix[IRC]] := True
```

Theorem.

```
In[32]:= (member[x, fix[t]] // AssertTest) /. {x → succ[set[0]], t → IRC}

Out[32]= member[succ[set[0]], fix[IRC]] == True

In[33]:= member[succ[set[0]], fix[IRC]] := True
```

Theorem.

```
In[34]:= (member[w, fix[t]] // AssertTest) /. {w → set[0, setpart[x]], t → IRC}

Out[34]= member[set[0, setpart[x]], fix[IRC]] == True

In[35]:= member[set[0, setpart[x_]], fix[IRC]] := True
```

Corollary.

```
In[36]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[member[set[0, setpart[x]], t]], t -> fix[IRC]]]

Out[36]= subclass[image[ADJOIN[set[0]], range[SINGLETON]], fix[IRC]] == True

In[37]:= subclass[image[ADJOIN[set[0]], range[SINGLETON]], fix[IRC]] := True
```

inverse image rules

Theorem.

```
In[38]:= image[inverse[IRC], set[0]] // Normality

Out[38]= image[inverse[IRC], set[0]] == set[0]

In[39]:= image[inverse[IRC], set[0]] := set[0]
```

Theorem.

```
In[40]:= IminComp[BIGCAP, IRC, set[0]] // Reverse

Out[40]= image[inverse[IRC], image[inverse[BIGCAP], set[0]]] == complement[set[0]]

In[41]:= image[inverse[IRC], image[inverse[BIGCAP], set[0]]] := complement[set[0]]
```

range[IRC]

Lemma.

```
In[42]:= member[0, range[IRC]] // AssertTest

Out[42]= member[0, range[IRC]] == True

In[43]:= % /. Equal → SetDelayed
```

Lemma.

```
In[44]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[implies[and[member[x, V], equal[A[x], 0]],
             equal[APPLY[t, APPLY[t, x]], x]]], t → IRC]]

Out[44]= subclass[image[inverse[BIGCAP], set[0]], fix[composite[IRC, IRC]]] == True

In[45]:= % /. Equal → SetDelayed
```

Corollary.

```
In[46]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]],
           subclass[u, w], {u → image[inverse[BIGCAP], set[0]],
            v -> fix[composite[IRC, IRC]], w → range[IRC]}] // Reverse

Out[46]= subclass[image[inverse[BIGCAP], set[0]], range[IRC]] == True

In[47]:= % /. Equal → SetDelayed
```

Lemma.

```
In[49]:= ImageComp[BIGCAP, IRC, V] // Reverse

Out[49]= image[BIGCAP, range[IRC]] == set[0]

In[50]:= % /. Equal → SetDelayed
```

Theorem.

```
In[51]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → union[set[0], image[inverse[BIGCAP], set[0]]], v → range[IRC]}]

Out[51]= equal[range[IRC], union[image[inverse[BIGCAP], set[0]], set[0]]] == True

In[52]:= range[IRC] := union[image[inverse[BIGCAP], set[0]], set[0]]
```

fix[IRC ∘ IRC]

Lemma.

```
In[53]:= SubstTest[subclass, fix[composite[t, t]], range[t], t → IRC] // Reverse

Out[53]= subclass[image[BIGCAP, fix[composite[IRC, IRC]]], set[0]] == True

In[54]:= % /. Equal → SetDelayed
```

Lemma.

```
In[55]:= member[0, fix[composite[IRC, IRC]]] // AssertTest

Out[55]= member[0, fix[composite[IRC, IRC]]] == True

In[56]:= % /. Equal → SetDelayed
```

Theorem.

```
In[57]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → union[set[0], image[inverse[BIGCAP], set[0]]], v → fix[composite[IRC, IRC]]}]

Out[57]= equal[fix[composite[IRC, IRC]], union[image[inverse[BIGCAP], set[0]], set[0]]] == True

In[58]:= fix[composite[IRC, IRC]] := union[image[inverse[BIGCAP], set[0]], set[0]]
```

image[inverse[BIGCAP], {0}]

Lemma.

```
In[60]:= member[0, fix[BIGCAP]] // AssertTest

Out[60]= member[0, fix[BIGCAP]] == False

In[61]:= member[0, fix[BIGCAP]] := False
```

Theorem.

```
In[62]:= ImageComp[IRC, inverse[IRC], image[inverse[BIGCAP], set[0]]] // Reverse

Out[62]= image[IRC, complement[set[0]]] == image[inverse[BIGCAP], set[0]]

In[63]:= image[IRC, complement[set[0]]] := image[inverse[BIGCAP], set[0]]
```

Lemma.

```
In[64]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → IRC, u -> image[inverse[BIGCAP], set[0]], v -> complement[set[0]]}] // Reverse

Out[64]= subclass[image[IRC, image[inverse[BIGCAP], set[0]]],
           image[inverse[BIGCAP], set[0]]] == True

In[65]:= % /. Equal → SetDelayed
```

Lemma.

```
In[66]:= Map[image[#, image[inverse[BIGCAP], set[0]]] &, SubstTest[composite,
            funpart[x], id[fix[funpart[x]]], x → composite[IRC, IRC]]] // Reverse

Out[66]= image[IRC, image[IRC, image[inverse[BIGCAP], set[0]]]] == image[inverse[BIGCAP], set[0]]

In[67]:= % /. Equal → SetDelayed
```

Lemma.

```
In[68]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → IRC, u -> image[IRC, image[inverse[BIGCAP], set[0]]],
            v -> image[inverse[BIGCAP], set[0]]}] // Reverse

Out[68]= subclass[image[inverse[BIGCAP], set[0]],
           image[IRC, image[inverse[BIGCAP], set[0]]]] == True

In[69]:= % /. Equal → SetDelayed
```

Theorem.

```
In[70]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[IRC, image[inverse[BIGCAP], set[0]]], v -> image[inverse[BIGCAP], set[0]]}]

Out[70]= equal[image[IRC, image[inverse[BIGCAP], set[0]]],
           image[inverse[BIGCAP], set[0]]] == True

In[71]:= image[IRC, image[inverse[BIGCAP], set[0]]] := image[inverse[BIGCAP], set[0]]
```

additional rules

Theorem.

```
In[72]:= SubstTest[fix, composite[x, id[y]],
          {x → composite[IRC, IRC], y → complement[P[complement[set[0]]]]}]

Out[72]= intersection[complement[P[complement[set[0]]]], image[inverse[BIGCAP], set[0]]] ==
          complement[P[complement[set[0]]]]

In[73]:= intersection[complement[P[complement[set[0]]]], image[inverse[BIGCAP], set[0]]] :=
          complement[P[complement[set[0]]]]
```

Theorem.

```
In[76]:= SubstTest[composite, funpart[t],
           id[fix[funpart[t]]], t → composite[IRC, IRC]] // Reverse

Out[76]= composite[IRC, IRC, id[union[image[inverse[BIGCAP], set[0]], set[0]]]] ==
          id[union[image[inverse[BIGCAP], set[0]], set[0]]]

In[77]:= composite[IRC, IRC, id[union[image[inverse[BIGCAP], set[0]], set[0]]]] :=
          id[union[image[inverse[BIGCAP], set[0]], set[0]]]
```

Corollary.

```
In[78]:= Assoc[composite[IRC, IRC], id[range[IRC]], IRC]

Out[78]= composite[IRC, IRC, IRC] == IRC

In[79]:= composite[IRC, IRC, IRC] := IRC
```

Theorem.

```
In[88]:= SubstTest[implies, subclass[composite[x, y], Id],
           FUNCTION[composite[inverse[y], id[domain[x]]]],
           {x → IRC, y → composite[IRC, id[range[IRC]]]}] // Reverse

Out[88]= FUNCTION[composite[id[union[image[inverse[BIGCAP], set[0]], set[0]]], inverse[IRC]]] ==
          True

In[89]:= FUNCTION[
           composite[id[union[image[inverse[BIGCAP], set[0]], set[0]]], inverse[IRC]]] := True
```

Corollary.

```
In[91]:= SubstTest[implies, FUNCTION[x], FUNCTION[composite[id[y], x]],
           {x -> composite[id[union[image[inverse[BIGCAP], set[0]], set[0]]], inverse[IRC]],
            y -> image[inverse[BIGCAP], set[0]]}] // Reverse

Out[91]= FUNCTION[composite[id[image[inverse[BIGCAP], set[0]]], inverse[IRC]]] == True

In[92]:= FUNCTION[composite[id[image[inverse[BIGCAP], set[0]]], inverse[IRC]]] := True
```

serendipity

Theorem.

```
In[80]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3], not[implies[p1, p3]],
           {p1 -> subclass[image[x, y], y], p2 -> subclass[image[x, image[x, y]], image[x, y]],
            p3 → subclass[image[x, image[x, y]], y]}]] // Reverse

Out[80]= or[not[subclass[image[x, y], y]], subclass[image[x, image[x, y]], y]] == True

In[81]:= or[not[subclass[image[x_, y_], y_]], subclass[image[x_, image[x_, y_]], y_]] := True
```