special APPLY rules

Johan G. F. Belinfante and Tiffany D. Goble
2003 November 13

```
In[1]:= << goedel52.t16; << tools.m

        :Package Title: goedel52.t16      2003 November 13 at 12:05 midnite

        It is now:  2004 Mar 1 at 12:41

        Loading Simplification Rules

        TOOLS.M                       Revised 2004 February 21

        weightlimit = 40
```

summary

In this notebook special rewrite rules for **APPLY** are derived for certain named functions. The examples are presented in lexicographic order. Many rules simplify when wrapped with **singleton**. Rules for some functions are omited because these will require additional work.

the empty set

```
In[2]:= SubstTest[A, image[z, singleton[x]], z -> 0] // Reverse

Out[2]= APPLY[0, x] == V

In[3]:= APPLY[0, x_] := V
```

ACLOSURE

```
In[4]:= SubstTest[A, image[z, singleton[x]], z -> ACLOSURE] // Reverse

Out[4]= APPLY[ACLOSURE, x] == union[Aclosure[x], complement[image[V, singleton[x]]]]

In[5]:= APPLY[ACLOSURE, x_] := union[Aclosure[x], complement[image[V, singleton[x]]]]
```

ADJOIN[x]

```
In[6]:= SubstTest[A, image[w, singleton[y]], w -> ADJOIN[x]] // Reverse

Out[6]= APPLY[ADJOIN[x], y] ==
         union[x, y, complement[image[V, singleton[x]]], complement[image[V, singleton[y]]]]
```

```
In[7]:= APPLY[ADJOIN[x_], y_] :=
         union[x, y, complement[image[V, singleton[x]]], complement[image[V, singleton[y]]]]
```

BIGCAP

```
In[8]:= SubstTest[A, image[z, singleton[x]], z -> BIGCAP] // Reverse

Out[8]= APPLY[BIGCAP, x] == union[A[x], complement[image[V, singleton[x]]]]

In[9]:= APPLY[BIGCAP, x_] := union[A[x], complement[image[V, singleton[x]]]]
```

BIGCUP

```
In[10]:= SubstTest[A, image[z, singleton[x]], z -> BIGCUP] // Reverse

Out[10]= APPLY[BIGCUP, x] == union[complement[image[V, singleton[x]]], U[x]]

In[11]:= APPLY[BIGCUP, x_] := union[complement[image[V, singleton[x]]], U[x]]
```

CARD

```
In[12]:= SubstTest[A, image[y, singleton[x]], y -> CARD] // Reverse

Out[12]= APPLY[CARD, x] ==
          union[card[x], complement[image[V, intersection[OMEGA, image[Q, singleton[x]]]]]]

In[13]:= APPLY[CARD, x_] :=
          union[card[x], complement[image[V, intersection[OMEGA, image[Q, singleton[x]]]]]]
```

cart

```
In[14]:= SubstTest[A, image[w, singleton[z]], w -> cart[x, y]] // Reverse

Out[14]= APPLY[cart[x, y], z] == union[A[y], complement[image[V, intersection[x, singleton[z]]]]]

In[15]:= APPLY[cart[x_, y_], z_] :=
          union[A[y], complement[image[V, intersection[x, singleton[z]]]]]
```

Di

The relation **Di** is not a function of course, but one can nonetheless derive a rule for it:

```
In[16]:= SubstTest[A, image[w, singleton[x]], w -> Di] // Reverse

Out[16]= APPLY[Di, x] == complement[image[V, singleton[x]]]

In[17]:= APPLY[Di, x_] := complement[image[V, singleton[x]]]
```

DUP

```
In[18]:= SubstTest[A, image[z, singleton[x]], z -> DUP] // Reverse

Out[18]= APPLY[DUP, x] == PAIR[x, x]

In[19]:= APPLY[DUP, x_] := PAIR[x, x]
```

This example brought to light a need for a special normalization rule needed for **PAIR** when its arguments are identical.

```
In[20]:= PAIR[x, x] // Normality // Reverse

Out[20]= union[complement[image[V, singleton[x]]], pair[x, x]] == PAIR[x, x]

In[21]:= union[complement[image[V, singleton[x_]]], pair[x_, x_]] := PAIR[x, x]
```

FUNPART

Some effort is required to get a clean formula for applying **FUNPART**.

```
In[22]:= equal[union[complement[image[V, singleton[x]]],
           complement[image[V, singleton[domain[funpart[x]]]]]],
          complement[image[V, singleton[x]]]]

Out[22]= True

In[23]:= union[complement[image[V, singleton[x_]]],
           complement[image[V, singleton[domain[funpart[x_]]]]]] :=
          complement[image[V, singleton[x]]]

In[24]:= SubstTest[A, image[y, singleton[x]], y -> FUNPART] // Reverse

Out[24]= APPLY[FUNPART, x] == union[complement[image[V, singleton[x]]], funpart[x]]

In[25]:= APPLY[FUNPART, x_] := union[complement[image[V, singleton[x]]], funpart[x]]
```

HC

```
In[26]:= Map[complement[complement[#]] &,
           SubstTest[A, image[y, singleton[x]], y -> HC] // Reverse]

Out[26]= APPLY[HC, x] == union[complement[image[V, singleton[x]]], H[x]]

In[27]:= APPLY[HC, x_] := union[complement[image[V, singleton[x]]], H[x]]
```

Id

```
In[28]:= SubstTest[A, image[z, singleton[x]], z -> Id] // Reverse

Out[28]= APPLY[Id, x] == union[x, complement[image[V, singleton[x]]]]

In[29]:= APPLY[Id, x_] := union[x, complement[image[V, singleton[x]]]]
```

id[x]

```
In[30]:= SubstTest[A, image[z, singleton[y]], z -> id[x]] // Reverse

Out[30]= APPLY[id[x], y] == union[y, complement[image[V, intersection[x, singleton[y]]]]]

In[31]:= APPLY[id[x_], y_] := union[y, complement[image[V, intersection[x, singleton[y]]]]]
```

LAMBHULL

```
In[32]:= SubstTest[A, image[y, singleton[x]], y -> LAMBHULL] // Reverse

Out[32]= APPLY[LAMBHULL, x] == union[complement[image[V, singleton[x]]], HULL[x]]

In[33]:= APPLY[LAMBHULL, x_] := union[complement[image[V, singleton[x]]], HULL[x]]
```

LEFT[x]

```
In[34]:= SubstTest[A, image[z, singleton[y]], z → LEFT[x]] // Reverse

Out[34]= APPLY[LEFT[x], y] == PAIR[x, y]

In[35]:= APPLY[LEFT[x_], y_] := PAIR[x, y]
```

PLUS and plus[x]

The following are of interest for arithmetic:

```
In[36]:= SubstTest[A, image[y, singleton[x]], y -> PLUS] // Reverse

Out[36]= APPLY[PLUS, x] ==
          union[complement[image[V, intersection[omega, singleton[x]]]], plus[x]]

In[37]:= APPLY[PLUS, x_] :=
          union[complement[image[V, intersection[omega, singleton[x]]]], plus[x]]
```

```
In[38]:= SubstTest[A, image[z, singleton[y]], z -> plus[x]] // Reverse

Out[38]= APPLY[plus[x], y] == natadd[x, y]

In[39]:= APPLY[plus[x_], y_] := natadd[x, y]

In[40]:= SubstTest[A, image[z, singleton[y]], z -> inverse[plus[x]]] // Reverse

Out[40]= APPLY[inverse[plus[x]], y] == natsub[y, x]

In[41]:= APPLY[inverse[plus[x_]], y_] := natsub[y, x]
```

POWER

```
In[42]:= SubstTest[A, image[z, singleton[x]], z -> POWER] // Reverse

Out[42]= APPLY[POWER, x] == union[complement[image[V, singleton[x]]], P[x]]

In[43]:= APPLY[POWER, x_] := union[complement[image[V, singleton[x]]], P[x]]
```

RANK

```
In[44]:= SubstTest[A, image[y, singleton[x]], y -> RANK] // Reverse

Out[44]= APPLY[RANK, x] ==
          union[complement[image[V, intersection[REGULAR, singleton[x]]]], rank[x]]

In[45]:= APPLY[RANK, x_] :=
          union[complement[image[V, intersection[REGULAR, singleton[x]]]], rank[x]]
```

RC[x] and RCF

The result for **RC[x]** is simplified by applying a double complement:

```
In[46]:= Map[complement[complement[#]] &,
          SubstTest[A, image[w, singleton[y]], w -> RC[x]] // Reverse]

Out[46]= APPLY[RC[x], y] == union[complement[image[V, singleton[x]]],
            image[V, intersection[y, complement[x]]], intersection[x, complement[y]]]

In[47]:= APPLY[RC[x_], y_] := union[complement[image[V, singleton[x]]],
            image[V, intersection[y, complement[x]]], intersection[x, complement[y]]]

In[48]:= SubstTest[A, image[y, singleton[x]], y -> RCF] // Reverse

Out[48]= APPLY[RCF, x] == union[complement[image[V, singleton[x]]], RC[x]]

In[49]:= APPLY[RCF, x_] := union[complement[image[V, singleton[x]]], RC[x]]
```

RIGHT[x]

```
In[50]:= SubstTest[A, image[z, singleton[y]], z → RIGHT[x]] // Reverse

Out[50]= APPLY[RIGHT[x], y] == PAIR[y, x]

In[51]:= APPLY[RIGHT[x_], y_] := PAIR[y, x]
```

SINGLETON

```
In[52]:= SubstTest[A, image[y, singleton[x]], y -> SINGLETON] // Reverse

Out[52]= APPLY[SINGLETON, x] == union[complement[image[V, singleton[x]]], singleton[x]]

In[53]:= APPLY[SINGLETON, x_] := union[complement[image[V, singleton[x]]], singleton[x]]
```

SUBVAR

```
In[54]:= Map[complement[complement[#]] &,
          SubstTest[A, image[y, singleton[x]], y -> SUBVAR] // Reverse]

Out[54]= APPLY[SUBVAR, x] == union[complement[image[V, singleton[x]]], subvar[x]]

In[55]:= APPLY[SUBVAR, x_] := union[complement[image[V, singleton[x]]], subvar[x]]
```

SUCC

```
In[56]:= SubstTest[A, image[y, singleton[x]], y -> SUCC] // Reverse

Out[56]= APPLY[SUCC, x] == union[complement[image[V, singleton[x]]], succ[x]]

In[57]:= APPLY[SUCC, x_] := union[complement[image[V, singleton[x]]], succ[x]]
```

TC

```
In[58]:= SubstTest[A, image[y, singleton[x]], y -> TC] // Reverse

Out[58]= APPLY[TC, x] == union[complement[image[V, singleton[x]]], tc[x]]

In[59]:= APPLY[TC, x_] := union[complement[image[V, singleton[x]]], tc[x]]
```

UCLOSURE

```
In[60]:= SubstTest[A, image[y, singleton[x]], y -> UCLOSURE] // Reverse

Out[60]= APPLY[UCLOSURE, x] == union[complement[image[V, singleton[x]]], Uclosure[x]]

In[61]:= APPLY[UCLOSURE, x_] := union[complement[image[V, singleton[x]]], Uclosure[x]]
```

V

```
In[62]:= SubstTest[A, image[V, y], {y -> singleton[x]}]

Out[62]= APPLY[V, x] == complement[image[V, singleton[x]]]

In[63]:= APPLY[V, x_] := complement[image[V, singleton[x]]]
```

VS

The case of **VS** needs extensive work.

```
In[64]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3], not[implies[p1, p3]],
              {p1 -> member[x, V], p2 -> member[domain[x], V],
            p3 -> member[intersection[domain[x], y], V]}]]

Out[64]= or[member[intersection[y, domain[x]], V], not[member[x, V]]] == True

In[65]:= (% /. {x -> x_, y -> y_}) /. Equal -> SetDelayed

In[66]:= equal[union[complement[image[V, singleton[x]]],
           complement[image[V, singleton[intersection[domain[x], domain[VERTSECT[x]]]]]]],
             complement[image[V, singleton[x]]]]

Out[66]= True

In[67]:= union[complement[image[V, singleton[x_]]],
           complement[image[V, singleton[intersection[domain[x_], domain[VERTSECT[x_]]]]]] :=
          complement[image[V, singleton[x]]]

In[68]:= equal[union[complement[image[V, singleton[x]]],
           complement[image[V, singleton[image[VERTSECT[x], domain[x]]]]],
             complement[image[V, singleton[x]]]]

Out[68]= True

In[69]:= union[complement[image[V, singleton[x_]]],
           complement[image[V, singleton[image[VERTSECT[x_], domain[x_]]]]] :=
          complement[image[V, singleton[x]]]

In[70]:= SubstTest[A, image[z, singleton[x]], z -> VS] // Reverse

Out[70]= APPLY[VS, x] ==
          union[complement[image[V, singleton[x]]], composite[VERTSECT[x], id[domain[x]]]]
```

```
In[71]:= APPLY[VS, x_] :=
          union[complement[image[V, singleton[x]]], composite[VERTSECT[x], id[domain[x]]]]
```