domain[APPLY[CURRY,x]]

Johan G. F. Belinfante
2006 November 9

```
In[1]:= SetDirectory["l:"]; << goedel87.09b; << tools.m

        :Package Title: goedel87.09b          2006 November 9 at 5:25 p.m.

        It is now:  2006 Nov 9 at 19:50

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 November 5

        weightlimit = 40
```

summary

A formula for **domain[APPLY[CURRY, x]]** is derived.

FUNCTION rule

Theorem. (Variable-free version.)

```
In[2]:= Assoc[FUNPART, VS, composite[IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]]]

Out[2]= composite[FUNPART, CURRY] == CURRY

In[3]:= composite[FUNPART, CURRY] := CURRY
```

Lemma.

```
In[4]:= SubstTest[implies, member[y, V], member[funpart[y], V], y → APPLY[CURRY, x]] // Reverse

Out[4]= or[member[funpart[APPLY[CURRY, x]], V],
          not[member[x, V]], not[subclass[x, cart[cart[V, V], V]]]] == True

In[5]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma. (A temporary simplification rule.)

```
In[6]:= equiv[and[member[x, V], member[funpart[APPLY[CURRY, x]], V],
          subclass[x, cart[cart[V, V], V]]], and[member[x, V], subclass[x, cart[cart[V, V], V]]]]

Out[6]= True
```

```
In[7]:= and[member[x_, V], member[funpart[APPLY[CURRY, x_]], V],
          subclass[x_, cart[cart[V, V], V]]] :=
         and[member[x, V], subclass[x, cart[cart[V, V], V]]]
```

Theorem.

```
In[8]:= Map[FUNCTION[A[#]] &, ImageComp[FUNPART, CURRY, set[x]]]

Out[8]= FUNCTION[APPLY[CURRY, x]] == and[member[x, V], subclass[x, cart[cart[V, V], V]]]

In[9]:= FUNCTION[APPLY[CURRY, x_]] := and[member[x, V], subclass[x, cart[cart[V, V], V]]]
```

empty function

Applying **CURRY** to **0** yields **0**.

```
In[11]:= APPLY[CURRY, 0] // Normality

Out[11]= APPLY[CURRY, 0] == 0

In[12]:= APPLY[CURRY, 0] := 0

In[13]:= SubstTest[APPLY, inverse[CURRY], APPLY[CURRY, x], x → 0] // Reverse

Out[13]= APPLY[inverse[CURRY], 0] == 0

In[14]:= APPLY[inverse[CURRY], 0] := 0
```

Lemma.

```
In[15]:= SubstTest[implies, equal[u, v], equal[APPLY[w, u], APPLY[w, v]],
           {u → 0, v → APPLY[CURRY, x], w → inverse[CURRY]}] // Reverse

Out[15]= or[equal[0, x], not[equal[0, APPLY[CURRY, x]]]] == True

In[16]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[17]:= equiv[equal[0, APPLY[CURRY, x]], equal[0, x]]

Out[17]= True

In[18]:= equal[0, APPLY[CURRY, x_]] := equal[0, x]
```

variable-free results about domains

Lemma.

```
In[19]:= composite[IMAGE[FIRST], IMAGE[ASSOC]] // ReifNormality

Out[19]= composite[IMAGE[FIRST], IMAGE[ASSOC]] == composite[IMAGE[FIRST], IMAGE[FIRST]]

In[20]:= composite[IMAGE[FIRST], IMAGE[ASSOC]] := composite[IMAGE[FIRST], IMAGE[FIRST]]
```

Theorem.

```
In[21]:= Assoc[IMAGE[FIRST], VS, composite[IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]]]

Out[21]= composite[IMAGE[FIRST], CURRY] ==
          composite[IMAGE[FIRST], IMAGE[FIRST], id[P[cart[cart[V, V], V]]]]

In[22]:= composite[IMAGE[FIRST], CURRY] :=
          composite[IMAGE[FIRST], IMAGE[FIRST], id[P[cart[cart[V, V], V]]]]
```

---

temporary lemmas

Lemma.

```
In[29]:= SubstTest[implies, member[w, V],
           not[equal[V, domain[w]]], w → APPLY[CURRY, x]] // Reverse

Out[29]= or[not[equal[V, domain[APPLY[CURRY, x]]]],
           not[member[x, V]], not[subclass[x, cart[cart[V, V], V]]]] == True

In[30]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[31]:= SubstTest[implies, equal[w, V], equal[domain[w], V], w → APPLY[CURRY, x]] // MapNotNot //
          Reverse

Out[31]= or[equal[V, domain[APPLY[CURRY, x]]], subclass[x, cart[cart[V, V], V]]] == True

In[32]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. (Temporary rewrite rule needed in the next section.)

```
In[33]:= equiv[equal[V, domain[APPLY[CURRY, x]]],
            or[not[member[x, V]], not[subclass[x, cart[cart[V, V], V]]]]] // not // not

Out[33]= True

In[34]:= equal[V, domain[APPLY[CURRY, x_]]] :=
          or[not[member[x, V]], not[subclass[x, cart[cart[V, V], V]]]]
```

domain formula

Lemma. (Sethood rule for **image[IMAGE[SWAP], x]**.)

```
In[23]:= image[V, set[image[IMAGE[SWAP], x]]] // Normality

Out[23]= image[V, set[image[IMAGE[SWAP], x]]] ==
          intersection[image[V, set[domain[U[x]]]], image[V, set[range[U[x]]]]]

In[24]:= image[V, set[image[IMAGE[SWAP], x_]]] :=
          intersection[image[V, set[domain[U[x]]]], image[V, set[range[U[x]]]]]
```

Lemma. Using a **setpart** wrapper, one obtains this temporary rewrite rule.

```
In[25]:= Map[domain[A[#]] &,
          ImageComp[composite[VS, IMAGE[ASSOC]], id[P[cart[cart[V, V], V]]], set[setpart[x]]]]

Out[25]= domain[APPLY[CURRY, setpart[x]]] == union[domain[domain[setpart[x]]],
           image[V, intersection[complement[cart[cart[V, V], V]], setpart[x]]]]

In[26]:= domain[APPLY[CURRY, setpart[x_]]] := union[domain[domain[setpart[x]]],
           image[V, intersection[complement[cart[cart[V, V], V]], setpart[x]]]]
```

Removing the **setpart** wrapper yields:

```
In[35]:= SubstTest[implies, equal[x, setpart[z]],
           equal[domain[APPLY[CURRY, x]], union[domain[domain[x]],
             image[V, intersection[complement[cart[cart[V, V], V]], x]]]], z → x] // Reverse

Out[35]= or[equal[domain[APPLY[CURRY, x]], domain[domain[x]]],
           not[member[x, V]], not[subclass[x, cart[cart[V, V], V]]]] == True

In[36]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. (General formula for **domain[APPLY[CURRY, x]]**.)

```
In[37]:= equal[domain[APPLY[CURRY, x]], union[domain[domain[x]], complement[image[V, set[x]]],
           image[V, intersection[complement[cart[cart[V, V], V]], x]]]]

Out[37]= True

In[38]:= domain[APPLY[CURRY, x_]] := union[complement[image[V, set[x]]],
           domain[domain[x]], image[V, intersection[x, complement[cart[cart[V, V], V]]]]]
```