IMAGE[x] ∘ id[P[domain[x]]]

Johan G. F. Belinfante
2013 July 12

```
In[1]:= SetDirectory["l:"]; << goedel.13jul11a

        :Package Title: goedel.13jul11a                    2013 July 11 at 9:00 p.m.

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

        It is now:  2013 Jul 12 at 15:39

        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 Jul 12 at 15:56
```

summary

Rewrite rules for the function **IMAGE[x] ∘ id[P[domain[x]]]** are derived, including some results involving **thinpart[x]**, and an inclusion involving the expression **image[DORA, P[x]]**. As a corollary, an upper bound for the function **IPD** is obtained.

derivation

Lemma.

```
In[2]:= equal[intersection[domain[thinpart[x]], domain[VERTSECT[x]]], domain[thinpart[x]]]

Out[2]= True

In[3]:= intersection[domain[thinpart[x_]], domain[VERTSECT[x_]]] := domain[thinpart[x]]
```

Theorem.

```
In[4]:= Assoc[IMAGE[thinpart[x]], id[P[domain[VERTSECT[x]]]], id[P[domain[x]]]]

Out[4]= composite[IMAGE[thinpart[x]], id[P[domain[thinpart[x]]]]] ==
         composite[IMAGE[x], id[P[domain[x]]]]

In[5]:= composite[IMAGE[thinpart[x_]], id[P[domain[thinpart[x_]]]]] :=
         composite[IMAGE[x], id[P[domain[x]]]]
```

Corollary.

```
In[6]:= Assoc[IMAGE[thinpart[x]],
          id[P[domain[VERTSECT[x]]]], id[P[domain[thinpart[x]]]]] // Reverse

Out[6]= composite[IMAGE[x], id[P[domain[thinpart[x]]]]] == composite[IMAGE[x], id[P[domain[x]]]]

In[7]:= composite[IMAGE[x_], id[P[domain[thinpart[x_]]]]] :=
         composite[IMAGE[x], id[P[domain[x]]]]
```

---

the relation image[DORA,P[x]]

Theorem.

```
In[8]:= SubstTest[implies, subclass[u, v],
          subclass[image[t, u], image[t, v]], {t → DORA, u → RS[x], v → P[x]}] // Reverse

Out[8]= subclass[composite[IMAGE[x], id[P[domain[x]]]], image[DORA, P[x]]] == True

In[9]:= subclass[composite[IMAGE[x_], id[P[domain[x_]]]], image[DORA, P[x_]]] := True
```

Corollary.

```
In[10]:= SubstTest[subclass, composite[IMAGE[t], id[P[domain[t]]]],
           image[DORA, P[t]], t → inverse[x]] // Reverse

Out[10]= subclass[composite[id[P[range[x]]], inverse[IMAGE[inverse[x]]]], image[DORA, P[x]]] ==
          True

In[11]:= subclass[composite[id[P[range[x_]]], inverse[IMAGE[inverse[x_]]]],
           image[DORA, P[x_]]] := True
```

---

an inclusion involving IPD

Lemma.

```
In[12]:= Map[equal[V, #] &, SubstTest[case, subclass[f[x], g[x]],
           {f[x] -> composite[IMAGE[x], id[P[domain[x]]]], g[x] → image[DORA, P[x]]}]]

Out[12]= subclass[P[domain[thinpart[x]]],
           fix[composite[inverse[image[DORA, P[x]]], IMAGE[x]]] == True

In[13]:= (% /. x → x_) /. Equal → SetDelayed

In[14]:= fix[composite[inverse[image[DORA, P[x]]], inverse[S], IMAGE[x]]]

Out[14]= P[domain[thinpart[x]]]
```

Theorem.

```
In[15]:= SubstTest[implies, subclass[u, v],
           subclass[composite[x, u, y], composite[x, v, y]], {u → Id, v → S}] // Reverse
```

```
Out[15]= subclass[composite[x, y], composite[x, S, y]] == True
```

```
In[16]:= subclass[composite[x_, y_], composite[x_, S, y_]] := True
```

Theorem.

```
In[17]:= SubstTest[implies, subclass[u, v],
           subclass[composite[x, u, y], composite[x, v, y]], {u → Id, v → inverse[S]}] // Reverse
```

```
Out[17]= subclass[composite[x, y], composite[x, inverse[S], y]] == True
```

```
In[18]:= subclass[composite[x_, y_], composite[x_, inverse[S], y_]] := True
```

Lemma.

```
In[19]:= SubstTest[subclass, fix[composite[u, v]], fix[composite[u, inverse[S], v]],
           {u -> inverse[image[DORA, P[x]]], v -> IMAGE[x]}] // Reverse
```

```
Out[19]= subclass[U[fix[composite[inverse[image[DORA, P[x]]], IMAGE[x]]]],
           domain[thinpart[x]]] == True
```

```
In[20]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[21]:= SubstTest[and, subclass[u, v], subclass[v, u], {u -> P[domain[thinpart[x]]],
           v -> fix[composite[inverse[image[DORA, P[x]]], IMAGE[x]]]}]
```

```
Out[21]= equal[fix[composite[inverse[image[DORA, P[x]]], IMAGE[x]]],
           P[domain[thinpart[x]]]] == True
```

```
In[22]:= fix[composite[inverse[image[DORA, P[x_]]], IMAGE[x_]]] := P[domain[thinpart[x]]]
```

Lemma.

```
In[23]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[subclass[image[u, set[x]], image[v, set[x]]]],
           {u -> composite[inverse[E], IPD], v -> composite[DORA, inverse[S]]}]]
```

```
Out[23]= subclass[composite[inverse[IPD], E], composite[S, inverse[DORA]]] == True
```

```
In[24]:= % /. Equal → SetDelayed
```

Theorem.

```
In[25]:= SubstTest[subclass, inverse[u], inverse[v],
          {u -> composite[inverse[E], IPD], v -> composite[DORA, inverse[S]]}]
```

```
Out[25]= subclass[composite[inverse[E], IPD], composite[DORA, inverse[S]]] == True
```

```
In[26]:= subclass[composite[inverse[E], IPD], composite[DORA, inverse[S]]] := True
```

Corollary. An upper bound for the function **IPD**.

```
In[27]:= SubstTest[subclass, VERTSECT[u], composite[inverse[S], VERTSECT[v]],
           {u → composite[inverse[E], IPD], v → composite[DORA, inverse[S]]}] // Reverse

Out[27]= subclass[IPD, composite[inverse[S], IMAGE[DORA], POWER]] == True

In[28]:= subclass[IPD, composite[inverse[S], IMAGE[DORA], POWER]] := True
```