range[IMAGE[funpart[x]]]

Johan G. F. Belinfante
2005 November 8

```
In[1]:= SetDirectory["l:"]; << goedel75.06a; << tools.m

        :Package Title: goedel75.06a            2005 November 6 at 9:10 a.m.

        It is now:  2005 Nov 9 at 7:46

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 October 25

        weightlimit = 40
```

summary

The range of **IMAGE[x]** is a power set when **x** is a small function.

a monotonicity theorem

Lemma.

```
In[2]:= SubstTest[implies, equal[x, z],
         equal[range[IMAGE[x]], range[IMAGE[z]]], z → composite[y, id[domain[x]]]]

Out[2]= or[equal[image[IMAGE[y], P[domain[x]]], range[IMAGE[x]]],
          not[equal[x, composite[y, id[domain[x]]]]]] == True

In[3]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. The compound constructor **range[IMAGE[x]]** is monotone when restricted to functions.

```
In[4]:= Map[not, SubstTest[and, implies[and[p1, p2], p3],
          implies[p3, p4], implies[p4, p5], not[implies[and[p1, p2], p5]],
          {p1 → subclass[x, y], p2 → FUNCTION[y], p3 → equal[x, composite[y, id[domain[x]]]],
           p4 → equal[image[IMAGE[y], P[domain[x]]], range[IMAGE[x]]],
           p5 → subclass[range[IMAGE[x]], range[IMAGE[y]]]}]]

Out[4]= or[not[FUNCTION[y]], not[subclass[x, y]],
          subclass[range[IMAGE[x]], range[IMAGE[y]]]] == True

In[5]:= or[not[FUNCTION[y_]], not[subclass[x_, y_]],
          subclass[range[IMAGE[x_]], range[IMAGE[y_]]]] := True
```

domains of subsets of a function

Using the **funpart** wrapper, one can derive this result.

```
In[6]:= SubstTest[image, IMAGE[SECOND], RS[y], y → funpart[x]]

Out[6]= image[IMAGE[SECOND], P[funpart[x]]] == range[IMAGE[funpart[x]]]

In[7]:= image[IMAGE[SECOND], P[funpart[x_]]] := range[IMAGE[funpart[x]]]
```

The wrapper can be removed:

```
In[8]:= SubstTest[implies, equal[x, funpart[y]],
          equal[image[IMAGE[SECOND], P[x]], range[IMAGE[x]]], y → x]

Out[8]= or[equal[image[IMAGE[SECOND], P[x]], range[IMAGE[x]]], not[FUNCTION[x]]] == True

In[9]:= or[equal[image[IMAGE[SECOND], P[x_]], range[IMAGE[x_]]], not[FUNCTION[x_]]] := True
```

Comment. If **x** is not a function, only an inclusion holds:

```
In[10]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u → RS[x], v → P[x], w → IMAGE[SECOND]}]

Out[10]= subclass[range[IMAGE[x]], image[IMAGE[SECOND], P[x]]] == True

In[11]:= subclass[range[IMAGE[x_]], image[IMAGE[SECOND], P[x_]]] := True
```

the case of small functions

For small functions, this corollary is obtained:

```
In[12]:= SubstTest[image, IMAGE[SECOND], P[funpart[y]], y → setpart[funpart[x]]] // Reverse

Out[12]= range[IMAGE[setpart[funpart[x]]]] == P[range[setpart[funpart[x]]]]

In[13]:= range[IMAGE[setpart[funpart[x_]]]] := P[range[setpart[funpart[x]]]]
```

Corollary. The **funpart** and **setpart** wrappers can be interchanged.

```
In[14]:= SubstTest[range, IMAGE[setpart[funpart[y]]], y → setpart[x]]

Out[14]= range[IMAGE[funpart[setpart[x]]]] == P[range[funpart[setpart[x]]]]

In[15]:= range[IMAGE[funpart[setpart[x_]]]] := P[range[funpart[setpart[x]]]]
```

Corollary. The wrappers can be removed altogether.

```
In[16]:= Map[implies[member[x, y], #] &, SubstTest[implies,
           equal[x, setpart[funpart[z]]], equal[P[range[x]], range[IMAGE[x]]], z → x]]

Out[16]= or[equal[P[range[x]], range[IMAGE[x]]], not[FUNCTION[x]], not[member[x, y]]] == True

In[17]:= or[equal[P[range[x_]], range[IMAGE[x_]]],
           not[FUNCTION[x_]], not[member[x_, y_]]] := True
```

Attempts to extend this result to functions in general did not succeed.

---

serendipity

These results are special cases of a general result already in the **GOEDEL** program.

```
In[18]:= ImageComp[IMAGE[FIRST], inverse[S], P[x]]

Out[18]= image[inverse[S], image[IMAGE[FIRST], P[x]]] == image[IMAGE[FIRST], P[x]]

In[19]:= image[inverse[S], image[IMAGE[FIRST], P[x_]]] := image[IMAGE[FIRST], P[x]]

In[20]:= ImageComp[IMAGE[SECOND], inverse[S], P[x]]

Out[20]= image[inverse[S], image[IMAGE[SECOND], P[x]]] == image[IMAGE[SECOND], P[x]]

In[21]:= image[inverse[S], image[IMAGE[SECOND], P[x_]]] := image[IMAGE[SECOND], P[x]]
```