image[FUNPART, CL]

Johan G. F. Belinfante
2009 June 21

```
In[1]:= SetDirectory["l:"]; << goedel.09jun19a; << tools.m

        :Package Title: goedel.09jun19a              2009 June 19 at 3:55 p.m.

        It is now:  2009 Jun 21 at 3:8

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 June 1

        weightlimit = 40
```

summary

Every complete lattice order has a top element, which is the greatest, and hence the unique maximal element of its fixed point set. The **funpart** (function part) of a complete lattice order is its restriction to this top element. A formula for **image[-FUNPART, CL]** is derived from these observations.

greatest => unique maximal

A subset of the fixed-point class of a partial order need not have a greatest element, but if it does, then that element is the unique maximal element. This fact is derived in this section. The **po[x]** wrapper is used to denote a generic partial order.

Lemma. A simplification rule.

```
In[2]:= equal[intersection[complement[subvar[intersection[Di, inverse[po[x]]]]],
          domain[GREATEST[po[x]]], domain[GREATEST[po[x]]]

Out[2]= True

In[3]:= intersection[complement[subvar[intersection[Di, inverse[po[x_]]]]],
          domain[GREATEST[po[x_]]] := domain[GREATEST[po[x]]]
```

Theorem. If a subset has a greatest element, then it has a unique maximal element.

```
In[4]:= SubstTest[subclass, intersection[domain[GREATEST[t]], domain[MAXIMAL[t]]],
          domain[funpart[MAXIMAL[t]]], t → po[x]] // Reverse

Out[4]= subclass[domain[GREATEST[po[x]]], domain[funpart[MAXIMAL[po[x]]]]] == True

In[5]:= subclass[domain[GREATEST[po[x_]]], domain[funpart[MAXIMAL[po[x_]]]]] := True
```

Corollary. The function **GREATEST[po[x]]** is contained in the function **funpart[MAXIMAL[po[x]]]**.

```
In[6]:= SubstTest[subclass, GREATEST[po[x]],
          composite[MAXIMAL[po[x]], id[t]], t → domain[funpart[MAXIMAL[po[x]]]]] // Reverse

Out[6]= subclass[GREATEST[po[x]], funpart[MAXIMAL[po[x]]]] == True

In[7]:= subclass[GREATEST[po[x_]], funpart[MAXIMAL[po[x_]]]] := True
```

Lemma. The fixed point class of **funpart[po[x]]** is a singleton if and only if **funpart[po[x]]** itself is a singleton.

```
In[8]:= SubstTest[member, domain[funpart[t]], range[SINGLETON], t → po[x]] // Reverse

Out[8]= member[fix[funpart[po[x]]], range[SINGLETON]] ==
         member[funpart[po[x]], range[SINGLETON]]

In[9]:= member[fix[funpart[po[x_]]], range[SINGLETON]] :=
         member[funpart[po[x]], range[SINGLETON]]
```

Theorem. If a partial order has a top element, then its funpart is a singleton.

```
In[10]:= Map[implies[member[po[x], y], #] &,
           SubstTest[implies, and[member[t, u], subclass[u, v]], member[t, v], {t → fix[po[x]],
             u → domain[GREATEST[po[x]]], v → domain[funpart[MAXIMAL[po[x]]]]}]] // Reverse

Out[10]= or[equal[0, po[x]], equal[0, ub[po[x], fix[po[x]]]],
           member[funpart[po[x]], range[SINGLETON]], not[member[po[x], y]]] == True

In[11]:= or[equal[0, po[x_]], equal[0, ub[po[x_], fix[po[x_]]]],
           member[funpart[po[x_]], range[SINGLETON]], not[member[po[x_], y_]]] := True
```

application to complete lattices

Theorem. The funpart of a complete lattice order is a singleton.

```
In[12]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p1, p3], not[implies[p1, p4]], {p1 → member[po[x], CL],
             p2 → not[empty[po[x]]], p3 → not[empty[ub[po[x], fix[po[x]]]]],
             p4 -> member[funpart[po[x]], range[SINGLETON]]}]] // Reverse

Out[12]= or[member[funpart[po[x]], range[SINGLETON]], not[member[po[x], CL]]] == True

In[13]:= or[member[funpart[po[x_]], range[SINGLETON]], not[member[po[x_], CL]]] := True
```

The **po[x]** wrapper is removed to prepare for eliminating the variable **x**.

Lemma.

```
In[14]:= SubstTest[implies, equal[x, po[t]], implies[member[x, CL],
             member[funpart[x], range[SINGLETON]]], t → x] // Reverse // MapNotNot

Out[14]= or[member[funpart[x], range[SINGLETON]], not[member[x, CL]]] == True

In[15]:= or[member[funpart[x_], range[SINGLETON]], not[member[x_, CL]]] := True
```

The next step is to eliminate the set variable **x**.

Theorem.

```
In[16]:= Map[equal[V, #] &,
          complement[dif[CL, image[inverse[FUNPART], range[SINGLETON]]]] // Normality]

Out[16]= subclass[image[FUNPART, CL], range[SINGLETON]] == True

In[17]:= % /. Equal → SetDelayed
```

This result can be improved upon.

Lemma.

```
In[18]:= SubstTest[implies, subclass[u, v],
            subclass[image[t, u], image[t, v]], {t → FUNPART, u → CL, v → PO}] // Reverse

Out[18]= subclass[U[image[FUNPART, CL]], Id] == True

In[19]:= % /. Equal → SetDelayed
```

Lemma.

```
In[20]:= SubstTest[subclass, t, intersection[u, v],
            {t -> image[FUNPART, CL], u → P[Id], v → range[SINGLETON]}] // Reverse

Out[20]= subclass[image[FUNPART, CL], image[SINGLETON, Id]] == True

In[21]:= % /. Equal → SetDelayed
```

The reverse inclusion also holds.

Lemma.

```
In[22]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
            {t → FUNPART, u → image[SINGLETON, Id], v → CL}] // Reverse

Out[22]= subclass[image[SINGLETON, Id], image[FUNPART, CL]] == True

In[23]:= % /. Equal → SetDelayed
```

Theorem. The funpart of a complete lattice order can be any singleton identity function.

```
In[24]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> image[FUNPART, CL], v -> image[SINGLETON, Id]}]

Out[24]= equal[image[FUNPART, CL], image[SINGLETON, Id]] == True

In[25]:= image[FUNPART, CL] := image[SINGLETON, Id]
```

an example: the divisibility relation

The divisibility relation for natural numbers is a complete lattice order. Its funpart is the identity function restricted to the single element **0**.

```
In[26]:= funpart[DIV]

Out[26]= cart[set[0], set[0]]
```

The inverse of any complete lattice order is another one. For the inverse of the divisibility relation, the funpart is the identity function restricted to the single element **1 = set[0]**.

```
In[27]:= funpart[inverse[DIV]]

Out[27]= cart[set[set[0]], set[set[0]]]
```

The **oopart** (one-to-one part) is empty.

Corollary.

```
In[29]:= SubstTest[intersection, funpart[t], inverse[funpart[inverse[t]]], t → DIV]

Out[29]= oopart[DIV] == 0

In[30]:= oopart[DIV] := 0
```