definition of the function IPD

Johan G. F. Belinfante
2009 September 3

```
In[1]:= SetDirectory["l:"]; << goedel.09sep02a; << tools.m

        :Package Title: goedel.09sep02a              2009 September 2 at 3:25 p.m.

        It is now:  2009 Sep 3 at 15:42

        Loading Simplification Rules

        TOOLS.M                                    Revised 2009 August 27

        weightlimit = 40
```

summary

The function **IMAGE[x]** maps a set **y** to **image[x, y]** whenever the latter is a set. When **x** is a set, this is the case for every set **y**. That is, the domain of the function **IMAGE[x]** is the proper class **V** when **x** is a set. It follows that the function **IMAGE[x]** itself is also a proper class when **x** is a set. As a consequence, there does not exist any function that takes **x** to **IMAGE[x]**. One can however factor the function **IMAGE[x]** as the product of two functions, **composite[IMAGE[x], id[P[domain[x]]]]** and **IMAGE[id[domain[x]]]**.

```
In[2]:= composite[IMAGE[x], id[P[domain[x]]], IMAGE[id[domain[x]]]]

Out[2]= IMAGE[x]
```

When **x** is a set, the restriction of **IMAGE[x]** to the power set of the domain of **x** is a set. This notebook introduces the function **IPD** that maps any set **x** to this restriction of **IMAGE[x]**. Normalization and other basic properties of **IPD** are derived.

a temporary definition

The following temporary abbreviation will be used in this notebook.

```
In[3]:= f[x_] := composite[IMAGE[x], id[P[domain[x]]]]
```

If **x** is a set, the factorization **IMAGE[x]** = **f[x]** ∘ **IMAGE[id[domain[x]]]** factors the proper class function **IMAGE[x]** as the product of the set function **f[x]** and the proper class function **IMAGE[id[domain[x]]]**.

definition of the function IPD

The function **IPD** can be defined by the following membership rule: **x ∈ IPD** iff **first[x] ∈ V** and **second[x] = f[first[x]]**.

```
In[4]:= member[x_, IPD] := and[member[first[x], V],
          equal[second[x], composite[IMAGE[first[x]], id[P[domain[first[x]]]]]]]
```

Theorem. The class **IPD** is a relation.

```
In[5]:= Map[equal[V, #] &, dif[IPD, cart[V, V]] // complement // Normality]

Out[5]= subclass[IPD, cart[V, V]] == True

In[6]:= subclass[IPD, cart[V, V]] := True
```

Corollary. A useful rewrite rule.

```
In[7]:= equal[composite[Id, IPD], IPD]

Out[7]= True

In[8]:= composite[Id, IPD] := IPD
```

normalization

Theorem. Temporary vertical section formula. (The **setpart** wrapper that occurs here will be eliminated shortly.)

```
In[9]:= image[IPD, set[setpart[x]]] // Normality

Out[9]= image[IPD, set[setpart[x]]] ==
         set[composite[IMAGE[setpart[x]], id[P[domain[setpart[x]]]]]]

In[10]:= image[IPD, set[setpart[x_]]] :=
          set[composite[IMAGE[setpart[x]], id[P[domain[setpart[x]]]]]]
```

Theorem. Normalization rewrite rule.

```
In[11]:= composite[inverse[E], INVERSE, IPD] // FastReifNormality // Reverse

Out[11]= intersection[composite[inverse[SECOND], inverse[S], IMAGE[FIRST]],
           inverse[rotate[composite[IMG, SWAP]]]] == composite[inverse[E], INVERSE, IPD]

In[12]:= intersection[composite[inverse[SECOND], inverse[S], IMAGE[FIRST]],
           inverse[rotate[composite[IMG, SWAP]]]] := composite[inverse[E], INVERSE, IPD]
```

The normalization rewrite rule implies that **IPD** is a function.

Corollary. The class **IPD** is a function.

```
In[13]:= Map[FUNCTION, SubstTest[reify, x, image[t, set[setpart[x]]], t → IPD]]
Out[13]= FUNCTION[IPD] == True

In[14]:= FUNCTION[IPD] := True
```

domain and range

Theorem. The function **IPD** is total.

```
In[15]:= Map[domain, SubstTest[reify, x, image[t, set[setpart[x]]], t → IPD]]
Out[15]= domain[IPD] == V

In[16]:= domain[IPD] := V
```

For the range only upper bounds and some related facts will be derived.

Theorem.

```
In[17]:= Map[member[0, image[IPD, #]] &,
           IminInt[composite[inverse[SECOND], inverse[S], IMAGE[FIRST]],
            inverse[rotate[composite[IMG, SWAP]]], V]] // Reverse
Out[17]= member[0, range[IPD]] == False

In[18]:= member[0, range[IPD]] := False
```

Lemma.

```
In[19]:= Map[subclass[image[IPD, #], P[cart[V, V]]] &,
           IminInt[composite[inverse[SECOND], inverse[S], IMAGE[FIRST]],
            inverse[rotate[composite[IMG, SWAP]]], V]] // Reverse
Out[19]= subclass[U[range[IPD]], cart[V, V]] == True

In[20]:= subclass[U[range[IPD]], cart[V, V]] := True
```

Corollary. Simplification rule.

```
In[21]:= Assoc[id[P[cart[V, V]]], id[range[IPD]], IPD]
Out[21]= composite[id[P[cart[V, V]]], IPD] == IPD

In[22]:= composite[id[P[cart[V, V]]], IPD] := IPD
```

Corollary. Simplification rule.

```
In[23]:= Assoc[IMAGE[id[cart[V, V]]], id[P[cart[V, V]]], IPD]
Out[23]= composite[IMAGE[id[cart[V, V]]], IPD] == IPD
```

```
In[24]:= composite[IMAGE[id[cart[V, V]]], IPD] := IPD
```

Corollary. Simplification rule.

```
In[25]:= Assoc[IMAGE[SWAP], id[P[cart[V, V]]], IPD]
Out[25]= composite[IMAGE[SWAP], IPD] == composite[INVERSE, IPD]

In[26]:= composite[IMAGE[SWAP], IPD] := composite[INVERSE, IPD]
```

FUNPART formula

The fact that **f[x]** is a function implies the following rewrite rule.

Theorem.

```
In[27]:= composite[FUNPART, IPD] // FastReifNormality
Out[27]= composite[FUNPART, IPD] == IPD

In[28]:= composite[FUNPART, IPD] := IPD
```

Corollary. A variable-free statement of the fact that if **x** is a set, then **f[x] ∈ FUNS**.

```
In[29]:= Map[subclass[#, FUNS] &, ImageComp[FUNPART, IPD, V]]
Out[29]= subclass[range[IPD], FUNS] == True

In[30]:= subclass[range[IPD], FUNS] := True
```

Since **f[x]** is a function, one has **VERTSECT[f[x]] ∘ id[domain[f[x]]] = SINGLETON ∘ f[x]**. A variable-free statement of this is given by the following corollary. Note that **IMAGE[x ⊗ SINGLETON] = λx. SINGLETON ∘ x**.

Corollary.

```
In[31]:= Assoc[VS, FUNPART, IPD]
Out[31]= composite[VS, IPD] == composite[IMAGE[cross[Id, SINGLETON]], IPD]

In[32]:= composite[VS, IPD] := composite[IMAGE[cross[Id, SINGLETON]], IPD]
```

APPLY formula

Lemma.

```
In[33]:= ImageComp[id[cart[V, V]], composite[inverse[E], IPD], set[x]] // Reverse
Out[33]= composite[Id, U[image[IPD, set[x]]]] == U[image[IPD, set[x]]]
```

```
In[34]:= % /. Equal → SetDelayed
```

Lemma.

```
In[35]:= Map[inverse, ImageInt[composite[inverse[SECOND], inverse[S], IMAGE[FIRST]],
           inverse[rotate[composite[IMG, SWAP]]], set[x]]]

Out[35]= U[image[IPD, set[x]]] ==
          composite[IMAGE[x], id[intersection[image[V, set[x]], P[domain[x]]]]]

In[36]:= U[image[IPD, set[x_]]] :=
          composite[IMAGE[x], id[intersection[image[V, set[x]], P[domain[x]]]]]
```

Lemma. A more general vertical section formula.

```
In[37]:= SubstTest[image, VERTSECT[t], set[x],
           t -> composite[SWAP, intersection[composite[inverse[SECOND], inverse[S],
               IMAGE[FIRST]], inverse[rotate[composite[IMG, SWAP]]]]] // Reverse

Out[37]= image[IPD, set[x]] == intersection[image[V, set[x]],
           set[composite[IMAGE[x], id[intersection[image[V, set[x]], P[domain[x]]]]]]]

In[38]:= image[IPD, set[x_]] := intersection[image[V, set[x]],
           set[composite[IMAGE[x], id[intersection[image[V, set[x]], P[domain[x]]]]]]]
```

Theorem. A general **APPLY** formula.

```
In[39]:= SubstTest[A, image[t, set[x]], t → IPD]

Out[39]= APPLY[IPD, x] == union[complement[image[V, set[x]]],
           composite[IMAGE[x], id[intersection[image[V, set[x]], P[domain[x]]]]]]

In[40]:= APPLY[IPD, x_] := union[complement[image[V, set[x]]],
           composite[IMAGE[x], id[intersection[image[V, set[x]], P[domain[x]]]]]]
```

The **APPLY** rule simplifies to **APPLY[IPD, x] = f[x]** when **x** is a set. This is most easily seen when **x** is wrapped with **setpart**.

```
In[41]:= APPLY[IPD, setpart[x]]

Out[41]= composite[IMAGE[setpart[x]], id[P[domain[setpart[x]]]]]
```

domain[f[x]]

When **x** is a set, the domain of **f[x]** is the power set of the domain of **x**.

```
In[42]:= domain[f[setpart[x]]]

Out[42]= P[domain[setpart[x]]]
```

A variable-free restatement of this fact is expressed by the following theorem.

Theorem. The function **IPD** maps a set **x** to a function whose domain is the power set of the domain of **x**.

```
In[43]:= composite[IMAGE[FIRST], IPD] // FastReifNormality
Out[43]= composite[IMAGE[FIRST], IPD] == composite[POWER, IMAGE[FIRST]]
```

```
In[44]:= composite[IMAGE[FIRST], IPD] := composite[POWER, IMAGE[FIRST]]
```

Theorem. The class of domains of functions of the form **f[x]** is the class of all power sets.

```
In[45]:= ImageComp[IMAGE[FIRST], IPD, V] // Reverse
Out[45]= image[IMAGE[FIRST], range[IPD]] == range[POWER]
```

```
In[46]:= image[IMAGE[FIRST], range[IPD]] := range[POWER]
```

range[f[x]]

When **x** is a set, the range of **f[x]** is the same as the range of **IMAGE[x]**.

```
In[47]:= range[f[setpart[x]]]
Out[47]= range[IMAGE[setpart[x]]]
```

The following theorem relates the function **IPD** to the function **VERTSECT[composite[IMG,inverse[FIRST]]]** that takes any set **x** to the set **range[IMAGE[x]]**.

Theorem.

```
In[48]:= SubstTest[reify, x, image[t, set[setpart[x]]], t -> composite[IMAGE[SECOND], IPD]]
Out[48]= composite[IMAGE[SECOND], IPD] == VERTSECT[composite[IMG, inverse[FIRST]]]
```

```
In[49]:= composite[IMAGE[SECOND], IPD] := VERTSECT[composite[IMG, inverse[FIRST]]]
```

Corollary. The range of **IPD** is related to the range of the function **λx. range[IMAGE[x]]**.

```
In[50]:= ImageComp[IMAGE[SECOND], IPD, V] // Reverse
Out[50]= image[IMAGE[SECOND], range[IPD]] == range[VERTSECT[composite[IMG, inverse[FIRST]]]]
```

```
In[51]:= image[IMAGE[SECOND], range[IPD]] := range[VERTSECT[composite[IMG, inverse[FIRST]]]]
```

identity functions

The constructor **f** takes the identity function on any class **x** to the identity function on the power class of **x**.

```
In[52]:= f[id[x]]
Out[52]= id[P[x]]
```

Theorem. The function **IPD** maps identity functions to identity functions.

```
In[53]:= composite[IPD, IDP] // FastReifNormality
Out[53]= composite[IPD, IMAGE[DUP]] == composite[IMAGE[DUP], POWER]

In[54]:= composite[IPD, IMAGE[DUP]] := composite[IMAGE[DUP], POWER]
```

---

one-to-one property

The function **f[x]** only depends on the relational part of **x**.

```
In[55]:= f[composite[Id, x]] == f[x]
Out[55]= True
```

A variable-free expression of this fact is given by the following rewrite rule.

Theorem. A simplification rule.

```
In[56]:= SubstTest[reify, x, image[t, set[setpart[x]]],
           t -> composite[IPD, IMAGE[id[cart[V, V]]]]]
Out[56]= composite[IPD, IMAGE[id[cart[V, V]]]] == IPD

In[57]:= composite[IPD, IMAGE[id[cart[V, V]]]] := IPD
```

The function **IPD** is closely related to the function **VS** that maps any set to the set **composite[VERTSECT[x], id[domain[x]]]**. This is an immediate consequence of the fact that **VERTSECT[x] = IMAGE[x] ∘ SINGLETON**.

Theorem.

```
In[58]:= composite[IMAGE[cross[inverse[SINGLETON], Id]], IPD] // FastReifNormality
Out[58]= composite[IMAGE[cross[inverse[SINGLETON], Id]], IPD] == VS

In[59]:= composite[IMAGE[cross[inverse[SINGLETON], Id]], IPD] := VS
```

The function **IPD** is not one-to-one because **f[x]** only depends on the relational part of **x**, but the restriction of **IPD** to relations is one-to-one. The same situation occurs with the vertical section function **VS**, and this fact will be used for the proof.

Lemma.

```
In[60]:= SubstTest[implies, and[subclass[range[y], domain[x]], subclass[composite[x, y], Id]],
           FUNCTION[inverse[y]], {x -> IMAGE[cross[inverse[SINGLETON], Id]],
            y -> composite[IPD, id[P[cart[V, V]]], inverse[VS]]}] // Reverse

Out[60]= FUNCTION[composite[VS, id[P[cart[V, V]]], inverse[IPD]]] == True

In[61]:= % /. Equal→ SetDelayed
```

Theorem. The restriction of **IPD** to relations is one-to-one.

```
In[62]:= SubstTest[implies, and[FUNCTION[x], FUNCTION[y]],
           FUNCTION[composite[x, y]], {x → composite[id[P[cart[V, V]]], inverse[VS]],
            y -> composite[VS, id[P[cart[V, V]]], inverse[IPD]]}] // Reverse

Out[62]= FUNCTION[composite[id[P[cart[V, V]]], inverse[IPD]]] == True

In[63]:= FUNCTION[composite[id[P[cart[V, V]]], inverse[IPD]]] := True
```

Corollary.

```
In[64]:= Map[inverse[composite[inverse[IMAGE[id[cart[V, V]]]], #, IMAGE[id[cart[V, V]]]]] &,
           SubstTest[composite, funpart[t], inverse[funpart[t]],
            t -> composite[id[P[cart[V, V]]], inverse[IPD]]]] // Reverse

Out[64]= composite[inverse[IPD], IPD] ==
          composite[inverse[IMAGE[id[cart[V, V]]]], IMAGE[id[cart[V, V]]]]

In[65]:= composite[inverse[IPD], IPD] :=
          composite[inverse[IMAGE[id[cart[V, V]]]], IMAGE[id[cart[V, V]]]]
```