characteristic functions

Johan G. F. Belinfante
2006 August 22

```
In[1]:= SetDirectory["l:"]; << goedel84.19b; << tools.m

        :Package Title: goedel84.19b       2006 August 19 at 11:45 p.m.

        It is now:  2006 Aug 22 at 19:2

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 August 15

        weightlimit = 40
```

summary

The **characteristic function** or **indicator function** of a subset **u** of a set **x** has the value **1 = set[0]** on **u** and **0** on the relative complement of **u** in **x**. In this notebook the one-to-one correspondence **CHAR[x]** between the subsets of **x** and their characteristic functions is defined and studied.

definition of CHAR[x]

The definition does not require that **x** be a set.

```
In[2]:= member[w_, CHAR[x_]] := and[member[first[w], V], subclass[first[w], x],
          equal[second[w], union[cart[first[w], set[set[0]]], cart[dif[x, first[w]], set[0]]]]]
```

Lemma 1.

```
In[3]:= SubstTest[implies, and[equal[u, v], member[v, V]], member[u, V],
         {u → union[cart[first[w], set[set[0]]], cart[dif[x, first[w]], set[0]]], v -> second[w]}]

Out[3]= or[member[x, V], not[equal[second[w], union[cart[first[w], set[set[0]]],
            cart[intersection[x, complement[first[w]]], set[0]]]]],
          not[member[first[w], V]], subclass[x, first[w]]] == True

In[4]:= (% /. {x → x_, w → w_}) /. Equal → SetDelayed
```

Lemma 2. The literal **subclass[x, first[w]]** can be eliminated from Lemma 1.

```
In[5]:= Map[not, SubstTest[and, implies[and[p1, p2, p3], p4],
          implies[and[p1, p2], not[p4]], and[p1, p2, p3], {p1 → not[member[x, V]],
           p2 -> member[first[w], V], p3 -> equal[second[w], union[cart[first[w], set[set[0]]],
              cart[dif[x, first[w]], set[0]]]], p4 → subclass[x, first[w]]}]]

Out[5]= or[member[x, V],
          not[equal[second[w], union[cart[first[w], set[set[0]]], cart[intersection[
               x, complement[first[w]]], set[0]]]]], not[member[first[w], V]]] == True

In[6]:= (% /. {x → x_, w → w_}) /. Equal → SetDelayed
```

When **x** is not a set, **CHAR[x]** is the empty function.

```
In[7]:= Map[equal[V, #] &,
          SubstTest[class, w, implies[member[w, z], member[x, V]], z → CHAR[x]]] // Reverse

Out[7]= or[equal[0, CHAR[x]], member[x, V]] == True

In[8]:= (% /. x → x_) /. Equal → SetDelayed
```

---

domain

Lemma.

```
In[9]:= domain[VERTSECT[composite[RIGHT[0], id[x], complement[inverse[E]]]]] // Normality

Out[9]= domain[VERTSECT[composite[RIGHT[0], id[x], complement[inverse[E]]]]] == image[V, set[x]]

In[10]:= domain[VERTSECT[composite[RIGHT[0], id[x_], complement[inverse[E]]]]] :=
          image[V, set[x]]
```

Theorem.

```
In[11]:= Map[domain[reify[w, #]] &, (image[CHAR[x], set[v]] // Normality) /. v → setpart[w]]

Out[11]= domain[CHAR[x]] == intersection[image[V, set[x]], P[x]]

In[12]:= domain[CHAR[x_]] := intersection[image[V, set[x]], P[x]]
```

Note that in particular one has:

```
In[13]:= domain[CHAR[setpart[x]]]

Out[13]= P[setpart[x]]
```

Corollary.

```
In[14]:= SubstTest[implies, equal[0, y], equal[0, domain[y]], y → CHAR[x]]

Out[14]= or[not[equal[0, CHAR[x]]], not[member[x, V]]] == True

In[15]:= (% /. x → x_) /. Equal → SetDelayed
```

This yields a rewrite rule:

```
In[16]:= equiv[equal[0, CHAR[x]], not[member[x, V]]]

Out[16]= True

In[17]:= equal[0, CHAR[x_]] := not[member[x, V]]
```

CHAR[x] is a function

Lemma.

```
In[18]:= Map[equal[0, #] &, dif[CHAR[x], cart[V, V]] // Normality]

Out[18]= subclass[CHAR[x], cart[V, V]] == True

In[19]:= subclass[CHAR[x_], cart[V, V]] := True
```

Corollary.

```
In[20]:= equal[composite[Id, CHAR[x]], CHAR[x]]

Out[20]= True

In[21]:= composite[Id, CHAR[x_]] := CHAR[x]

In[22]:= Map[FUNCTION[reify[w, #]] &, (image[CHAR[x], set[v]] // Normality) /. v → setpart[w]]

Out[22]= FUNCTION[CHAR[x]] == True

In[23]:= FUNCTION[CHAR[x_]] := True
```

Sethood corollary.

```
In[24]:= SubstTest[member, domain[funpart[y]], V, y → CHAR[x]] // Reverse

Out[24]= member[CHAR[x], V] == True

In[25]:= member[CHAR[x_], V] := True
```

formulas for CHAR[x]

A formula for **CHAR[x]** can be obtained using **reify**. The formula simplifies when one wraps **x** with **setpart**.

```
In[26]:= Map[equal[CHAR[y], reify[v, #]] &,
           image[CHAR[y], set[v]] // Normality // Reverse] /. y → setpart[x]

Out[26]= equal[CHAR[setpart[x]],
           composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
             composite[inverse[SECOND], IMAGE[RIGHT[0]], RC[setpart[x]]]]]] == True
```

```
In[27]:= (% /. x → x_) /. Equal → SetDelayed
```

The **setpart** wrapper can be eliminated:

```
In[28]:= SubstTest[implies, equal[x, setpart[y]], equal[CHAR[x],
           composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
             composite[inverse[SECOND], IMAGE[RIGHT[0]], RC[x]]]]], y → x]

Out[28]= or[equal[CHAR[x],
           composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
             composite[inverse[SECOND], IMAGE[RIGHT[0]], RC[x]]]]], not[member[x, V]]] == True

In[29]:= (% /. x → x_) /. Equal → SetDelayed
```

The same conclusion can be derived when **x** is not a set:

```
In[30]:= SubstTest[implies, and[equal[0, t], equal[0, z]],
          equal[t, composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
            composite[inverse[SECOND], IMAGE[RIGHT[0]], z]]]], {t → CHAR[x], z → RC[x]}]

Out[30]= or[equal[CHAR[x],
           composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
             composite[inverse[SECOND], IMAGE[RIGHT[0]], RC[x]]]]], member[x, V]] == True

In[31]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining these two cases yields:

```
In[32]:= SubstTest[and, implies[p, q], or[p, q], {p → member[x, V],
           q -> equal[CHAR[x],
             composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
               composite[inverse[SECOND], IMAGE[RIGHT[0]], RC[x]]]]}]

Out[32]= True == equal[CHAR[x],
           composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
             composite[inverse[SECOND], IMAGE[RIGHT[0]], RC[x]]]]

In[33]:= composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
           composite[inverse[SECOND], IMAGE[RIGHT[0]], RC[x_]]]] := CHAR[x]
```

Since **CUP** is symmetric, a similar formula also holds:

```
In[34]:= Assoc[CUP, SWAP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
           composite[inverse[SECOND], IMAGE[RIGHT[0]], RC[x]]]]

Out[34]= composite[CUP, intersection[composite[inverse[SECOND], IMAGE[RIGHT[set[0]]]],
           composite[inverse[FIRST], IMAGE[RIGHT[0]], RC[x]]]] == CHAR[x]

In[35]:= composite[CUP, intersection[composite[inverse[SECOND], IMAGE[RIGHT[set[0]]]],
           composite[inverse[FIRST], IMAGE[RIGHT[0]], RC[x_]]]] := CHAR[x]
```

range

In this section it is shown that the range of **CHAR[x]** is the set of characteristic functions (also called indicator functions).

Lemma.

```
In[36]:= FUNCTION[union[cart[x, set[u]], cart[y, set[v]]]] // AssertTest

Out[36]= FUNCTION[union[cart[x, set[u]], cart[y, set[v]]]] ==
          or[equal[0, intersection[x, y]], equal[u, v], not[member[u, V]], not[member[v, V]]]

In[37]:= FUNCTION[union[cart[x_, set[u_]], cart[y_, set[v_]]]] :=
          or[equal[0, intersection[x, y]], equal[u, v], not[member[u, V]], not[member[v, V]]]
```

Lemma.

```
In[38]:= implies[member[w, P[setpart[x]]],
           member[union[cart[dif[setpart[x], w], set[0]], cart[w, set[set[0]]]],
            map[setpart[x], succ[set[0]]]]] // AssertTest

Out[38]= or[member[
            union[cart[w, set[set[0]]], cart[intersection[complement[w], setpart[x]], set[0]]],
            map[setpart[x], succ[set[0]]]],
           not[member[w, V]], not[subclass[w, setpart[x]]]] == True

In[39]:= (% /. {w → w_, x → x_}) /. Equal → SetDelayed

Theorem.

In[40]:= Map[equal[V, #] &, SubstTest[class, w, implies[member[w, u], member[second[w], v]],
             {u -> CHAR[setpart[x]], v -> map[setpart[x], succ[set[0]]]}]] // Reverse

Out[40]= subclass[range[CHAR[setpart[x]]], map[setpart[x], succ[set[0]]]] == True

In[41]:= (% /. x → x_) /. Equal → SetDelayed
```

Eliminate the **setpart** wrapper.

```
In[42]:= SubstTest[implies, equal[x, setpart[y]],
           subclass[range[CHAR[x]], map[x, succ[set[0]]]], y → x]

Out[42]= or[not[member[x, V]], subclass[range[CHAR[x]], map[x, succ[set[0]]]]] == True

In[43]:= (% /. x → x_) /. Equal → SetDelayed
```

The same conclusion holds when **x** is not a set.

```
In[44]:= SubstTest[implies, equal[0, z], subclass[range[z], y], z → CHAR[x]]

Out[44]= or[member[x, V], subclass[range[CHAR[x]], y]] == True
```

```
In[45]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Combining these cases, one finds:

```
In[46]:= SubstTest[and, implies[p, q], or[p, q],
           {p → member[x, V], q -> subclass[range[CHAR[x]], map[x, succ[set[0]]]]}] // Reverse

Out[46]= subclass[range[CHAR[x]], map[x, succ[set[0]]]] == True

In[47]:= (% /. x → x_) /. Equal → SetDelayed
```

In the next section it is shown that this mapping is onto.

```
In[48]:= member[CHAR[x], map[P[x], map[x, succ[set[0]]]]] // AssertTest

Out[48]= member[CHAR[x], map[P[x], map[x, succ[set[0]]]]] == member[x, V]

In[49]:= member[CHAR[x_], map[P[x_], map[x_, succ[set[0]]]]] := member[x, V]
```

CHAR[x] is onto

Lemma.

```
In[50]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
           {u → range[x], w → inverse[x], v → union[y, z]}] /. {y → set[0], z → set[set[0]]}

Out[50]= or[not[subclass[range[x], succ[set[0]]]], subclass[domain[x],
            union[image[inverse[x], set[0]], image[inverse[x], set[set[0]]]]]] == True

In[51]:= (% /. x → x_) /. Equal → SetDelayed
```

Lemma.

```
In[52]:= SubstTest[implies, subclass[u, v],
          subclass[image[w, u], image[w, v]], {u → range[x], v → y, w → inverse[x]}]

Out[52]= or[not[subclass[range[x], y]], subclass[domain[x], image[inverse[x], y]]] == True

In[53]:= or[not[subclass[range[x_], y_]], subclass[domain[x_], image[inverse[x_], y_]]] := True
```

Lemma.

```
In[54]:= or[equal[funpart[v], union[cart[image[inverse[funpart[v]], set[set[0]]], set[set[0]]],
             cart[intersection[complement[image[inverse[funpart[v]], set[set[0]]]],
               domain[funpart[v]]], set[0]]]],
            not[subclass[range[funpart[v]], succ[set[0]]]]] // AssertTest // MapNotNot

Out[54]= or[equal[funpart[v], union[cart[image[inverse[funpart[v]], set[set[0]]], set[set[0]]],
             cart[intersection[complement[image[inverse[funpart[v]], set[set[0]]]],
               domain[funpart[v]]], set[0]]]],
           not[subclass[range[funpart[v]], succ[set[0]]]]] == True
```

```
In[55]:= (% /. v → v_) /. Equal → SetDelayed
```

Remove the **funpart** wrapper.

```
In[56]:= SubstTest[implies, equal[v, funpart[w]],
          or[equal[v, union[cart[image[inverse[v], set[set[0]]], set[set[0]]], cart[
              intersection[complement[image[inverse[v], set[set[0]]]], domain[v]], set[0]]]],
            not[subclass[range[v], succ[set[0]]]]], w → v]

Out[56]= or[equal[v, union[cart[image[inverse[v], set[set[0]]], set[set[0]]], cart[
              intersection[complement[image[inverse[v], set[set[0]]]], domain[v]], set[0]]]],
            not[FUNCTION[v]], not[subclass[range[v], succ[set[0]]]]] == True

In[57]:= (% /. v → v_) /. Equal → SetDelayed
```

Theorem.

```
In[58]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3],
            implies[p1, p4], implies[and[p2, p4], p5], implies[and[p3, p5], p6],
            not[implies[p1, p6]], {p1 → member[v, map[x, succ[set[0]]]], p2 → FUNCTION[v],
             p3 → equal[x, domain[v]], p4 → subclass[range[v], succ[set[0]]], p5 -> equal[v,
               union[cart[image[inverse[v], set[set[0]]], set[set[0]]], cart[intersection[
                  complement[image[inverse[v], set[set[0]]]], domain[v]], set[0]]]],
             p6 -> equal[v, union[cart[image[inverse[v], set[set[0]]], set[set[0]]],
                cart[intersection[complement[image[inverse[v], set[set[0]]]], x], set[0]]]]}]]

Out[58]= or[equal[v, union[cart[image[inverse[v], set[set[0]]], set[set[0]]],
              cart[intersection[x, complement[image[inverse[v], set[set[0]]]]], set[0]]]],
            not[member[v, map[x, succ[set[0]]]]]] == True

In[59]:= (% /. {v → v_, x → x_}) /. Equal → SetDelayed
```

Lemma.

```
In[60]:= equal[intersection[map[x, succ[set[0]]],
            P[complement[cart[complement[x], set[set[0]]]]]], map[x, succ[set[0]]]]

Out[60]= True

In[61]:= intersection[map[x_, succ[set[0]]],
            P[complement[cart[complement[x_], set[set[0]]]]]] := map[x, succ[set[0]]]
```

Eliminating the variables other than **x** yields:

```
In[62]:= Map[equal[0, domain[composite[complement[#], id[cart[V, V]]]]] &,
            SubstTest[class, pair[pair[u, v], w], implies[and[equal[v, setpart[w]],
               equal[image[inverse[v], set[set[0]]], u], member[v, y], subclass[u, x]],
              member[pair[u, v], z]], {y → map[x, succ[set[0]]], z → CHAR[x]}]] // Reverse

Out[62]= subclass[map[x, succ[set[0]]],
            fix[composite[CHAR[x], IMAGE[FIRST], IMAGE[id[cart[V, set[set[0]]]]]]]] == True

In[63]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[64]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → map[x, succ[set[0]]],
           v -> fix[composite[CHAR[x], IMAGE[FIRST], IMAGE[id[cart[V, set[set[0]]]]]]],
           w → range[CHAR[x]]}]

Out[64]= subclass[map[x, succ[set[0]]], range[CHAR[x]]] == True

In[65]:= (% /. x → x_) /. Equal → SetDelayed
```

Combining this inclusion with the one derived in the preceding section yields an equation for the range.

```
In[66]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> map[x, succ[set[0]]], v -> range[CHAR[x]]}]

Out[66]= True == equal[map[x, succ[set[0]]], range[CHAR[x]]]

In[67]:= range[CHAR[x_]] := map[x, succ[set[0]]]
```

CHAR[x] is one-to-one

Theorem.

```
In[68]:= Map[equal[0, inverse[complement[#]]] &,
           SubstTest[class, pair[u, v], implies[member[pair[u, v], s],
             equal[image[inverse[v], t], u]], {s -> CHAR[x], t → set[set[0]]}]] // Reverse

Out[68]= subclass[composite[inverse[CHAR[x]], inverse[IMAGE[id[cart[V, set[set[0]]]]]],
           IMAGE[FIRST]] == True

In[69]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[70]:= SubstTest[implies, subclass[u, v], subclass[composite[w, u], composite[w, v]],
          {u → composite[IMAGE[id[cart[V, set[set[0]]]]], CHAR[x]],
           v → inverse[IMAGE[FIRST]], w → IMAGE[FIRST]}]

Out[70]= subclass[composite[IMAGE[FIRST], IMAGE[id[cart[V, set[set[0]]]]], CHAR[x]], Id] == True

In[71]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary.

```
In[72]:= SubstTest[implies, subclass[composite[u, v], Id],
          FUNCTION[composite[inverse[v], id[domain[u]]]],
          {u -> composite[IMAGE[FIRST], IMAGE[id[cart[V, set[set[0]]]]]], v -> CHAR[x]}]

Out[72]= FUNCTION[inverse[CHAR[x]]] == True
```

```
In[73]:= FUNCTION[inverse[CHAR[x_]]] := True
```

equipollence theorem

Theorem.

```
In[74]:= SubstTest[implies, member[w, BIJ],
          member[pair[domain[w], range[w]], Q], w → CHAR[setpart[x]]]

Out[74]= member[pair[P[setpart[x]], map[setpart[x], succ[set[0]]]], Q] == True

In[75]:= member[pair[P[setpart[x_]], map[setpart[x_], succ[set[0]]]], Q] := True
```

Corollary.

```
In[76]:= Map[implies[member[x, y], #] &, SubstTest[implies,
           equal[x, setpart[z]], member[pair[P[x], map[x, succ[set[0]]]], Q], z → x]]

Out[76]= or[member[pair[P[x], map[x, succ[set[0]]]], Q], not[member[x, y]]] == True

In[77]:= or[member[pair[P[x_], map[x_, succ[set[0]]]], Q], not[member[x_, y_]]] := True
```

APPLY formula

Theorem.

```
In[78]:= ImageComp[composite[CUP,
           cross[IMAGE[RIGHT[set[0]]], composite[IMAGE[RIGHT[0]], RC[x]]]], DUP, set[y]]

Out[78]= image[CHAR[x], set[y]] ==
          intersection[complement[image[V, intersection[y, complement[x]]]],
           set[union[cart[y, set[set[0]]], cart[intersection[x, complement[y]], set[0]]]]]

In[79]:= image[CHAR[x_], set[y_]] :=
          intersection[complement[image[V, intersection[y, complement[x]]]],
           set[union[cart[y, set[set[0]]], cart[intersection[x, complement[y]], set[0]]]]]
```

Corollary.

```
In[80]:= SubstTest[A, image[z, set[y]], z → CHAR[x]] // Reverse

Out[80]= APPLY[CHAR[x], y] == union[cart[y, set[set[0]]],
           cart[intersection[x, complement[y]], set[0]], complement[image[V, set[x]]],
           complement[image[V, set[y]]], image[V, intersection[y, complement[x]]]]

In[81]:= APPLY[CHAR[x_], y_] := union[cart[y, set[set[0]]],
           cart[intersection[x, complement[y]], set[0]], complement[image[V, set[x]]],
           complement[image[V, set[y]]], image[V, intersection[y, complement[x]]]]
```

reify formula

A **reify** formula for **CHAR** is derived as follows:

```
In[82]:= SubstTest[reify, x,
          composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
            composite[inverse[SECOND], IMAGE[RIGHT[0]], g[f[x]]]]], g → RC]

Out[82]= reify[x, CHAR[f[x]]] ==
          composite[cross[Id, CUP], id[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
           cross[Id, composite[inverse[SECOND], IMAGE[RIGHT[0]]]],
           id[DISJOINT], inverse[CUP], VERTSECT[reify[x, f[x]]]]

In[83]:= reify[x_, CHAR[y_]] :=
          composite[cross[Id, CUP], id[composite[inverse[FIRST], IMAGE[RIGHT[set[0]]]],
           cross[Id, composite[inverse[SECOND], IMAGE[RIGHT[0]]]],
           id[DISJOINT], inverse[CUP], VERTSECT[reify[x, y]]]
```