characteristic functions and Boolean algebra

Johan G. F. Belinfante
2006 August 23

```
In[1]:= SetDirectory["l:"]; << goedel84.22a; << tools.m

        :Package Title: goedel84.22a      2006 August 22 at 4:50 p.m.

        It is now:  2006 Aug 23 at 21:19

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 August 22

        weightlimit = 40
```

summary

The one-to-one correspondence **CHAR[x]** between the set **P[x]** of subsets of a set **x** and the set **map[x, succ[set[0]]]** of their characteristic functions preserves all Boolean operations. The strategy will be to begin with equations involving **APPLY** and additional variables, and then to remove these extra variables by using reify twice, once for each side of the equation.

the relative complement formula

In this section, **reify** is used to eliminate the variable **y** in the following equation for the characterstic function corresponding to a relative complement of a subset of a set **x**. The **setpart** wrappper on the variable **x** can not be omitted, but one does not need a setpart wrapper on **y**.

```
In[6]:= composite[RC[set[0]], APPLY[CHAR[setpart[x]], intersection[setpart[x], y]]] ==
         APPLY[CHAR[setpart[x]], intersection[setpart[x], complement[y]]]

Out[6]= True
```

First **reify** is used on the right side of this equation, and the result is made into a temporary rewrite rule.

```
In[7]:= Map[composite[VERTSECT[#], id[P[setpart[x]]]] &, SubstTest[reify, y,
          APPLY[v, intersection[w, complement[y]]], {v → CHAR[setpart[x]], w → setpart[x]}]]

Out[7]= composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[0]]],
           composite[inverse[SECOND], IMAGE[RIGHT[set[0]]], RC[setpart[x]]]]] ==
         composite[CHAR[setpart[x]], RC[setpart[x]]]

In[8]:= composite[CUP, intersection[composite[inverse[FIRST], IMAGE[RIGHT[0]]],
           composite[inverse[SECOND], IMAGE[RIGHT[set[0]]], RC[setpart[x_]]]]] :=
         composite[CHAR[setpart[x]], RC[setpart[x]]]
```

Applying **reify** to the left side then yields:

```
In[9]:= Map[composite[VERTSECT[#], id[P[setpart[x]]]] &,
         SubstTest[reify, y, composite[u, APPLY[v, intersection[w, y]]],
          {u → RC[set[0]], v → CHAR[setpart[x]], w → setpart[x]}]]

Out[9]= composite[CHAR[setpart[x]], RC[setpart[x]]] == composite[
          IMAGE[cross[Id, union[cart[set[0], set[set[0]]], cart[set[set[0]], set[0]]]]],
          CHAR[setpart[x]]]

In[10]:= composite[CHAR[setpart[x_]], RC[setpart[x_]]] := composite[
           IMAGE[cross[Id, union[cart[set[0], set[set[0]]], cart[set[set[0]], set[0]]]]],
           CHAR[setpart[x]]]
```

The **setpart** wrapper can be replaced with a sethood literal:

```
In[11]:= SubstTest[implies, equal[x, setpart[y]],
          equal[composite[CHAR[x], RC[x]], composite[IMAGE[cross[Id,
              union[cart[set[0], set[set[0]]], cart[set[set[0]], set[0]]]]], CHAR[x]]], y → x]

Out[11]= or[equal[composite[CHAR[x], RC[x]], composite[
             IMAGE[cross[Id, union[cart[set[0], set[set[0]]], cart[set[set[0]], set[0]]]]],
             CHAR[x]]], not[member[x, V]]] == True

In[12]:= (% /. x → x_) /. Equal → SetDelayed
```

The same conclusion holds when **x** is not a set because then the equation reduces to **0 = 0**.

```
In[13]:= SubstTest[implies, equal[0, z], equal[composite[z, u], composite[v, z]], z → CHAR[x]]

Out[13]= or[equal[composite[v, CHAR[x]], composite[CHAR[x], u]], member[x, V]] == True

In[14]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

The two cases can be combined:

```
In[15]:= SubstTest[and, implies[p, q], or[p, q],
          {p → member[x, V], q -> equal[composite[CHAR[x], RC[x]], composite[IMAGE[cross[Id,
                union[cart[set[0], set[set[0]]], cart[set[set[0]], set[0]]]]], CHAR[x]]]}]

Out[15]= True == equal[composite[CHAR[x], RC[x]], composite[IMAGE[
             cross[Id, union[cart[set[0], set[set[0]]], cart[set[set[0]], set[0]]]]], CHAR[x]]]
```

The orientation of this rewrite rule is tentative:

```
In[16]:= composite[CHAR[x_], RC[x_]] := composite[IMAGE[
            cross[Id, union[cart[set[0], set[set[0]]], cart[set[set[0]], set[0]]]]], CHAR[x]]
```

comments about the setpart wrapper on x

The following counterexample shows that the **setpart** wrapper on **x** cannot be omitted from the equations involving **APPLY** in the preceding section. This is not surprising because the power set **P[V]** is not empty, but **map[V, succ[set[0]]** is the empty set.

```
In[17]:= equal[composite[RC[set[0]], APPLY[CHAR[x], intersection[x, setpart[y]]]],
            APPLY[CHAR[x], intersection[x, complement[setpart[y]]]]] /. x → V

Out[17]= False
```

Nonetheless, the final variable-free equations derived in the preceding section do hold without **setpart** wrappers because **CHAR[x]** is empty when **x** is not a set.

two associative pointwise operations

For any binary operation on natural numbers, there is a corresponding pointwise operation for number-valued functions. In particular, if **f** and **g** are numerical functions, then the pointwise minimum **h = min(f, g)** is the function defined by **h(x) = min(f(x), g(x))**, and similarly for the pointwise maximum. For natural numbers, the minimum and maximum operations are restrictions of the binary functions **CAP** and **CUP**, respectively. The corresponding pointwise operations on functions are associative:

```
In[2]:= SubstTest[implies, and[associative[x], thin[x]],
          associative[composite[IMAGE[cross[inverse[DUP], x]], CROSS]], x → CAP]

Out[2]= associative[composite[IMAGE[cross[inverse[DUP], CAP]], CROSS]] == True

In[3]:= associative[composite[IMAGE[cross[inverse[DUP], CAP]], CROSS]] := True

In[4]:= SubstTest[implies, and[associative[x], thin[x]],
          associative[composite[IMAGE[cross[inverse[DUP], x]], CROSS]], x → CUP]

Out[4]= associative[composite[IMAGE[cross[inverse[DUP], CUP]], CROSS]] == True

In[5]:= associative[composite[IMAGE[cross[inverse[DUP], CUP]], CROSS]] := True
```

To obtain formulas involving these expressions for the pointwise operations on functions that correspond to **CAP** and **CUP** one needs the following key rewrite rule:

```
In[18]:= Assoc[inverse[DUP], composite[TWIST,
            cross[cross[x, x], cross[inverse[FIRST], inverse[SECOND]]]], TWIST] /. x → Id

Out[18]= intersection[composite[cross[Id, inverse[FIRST]], FIRST],
           composite[cross[Id, inverse[SECOND]], SECOND]] ==
         composite[cross[inverse[DUP], Id], TWIST]
```

```
In[19]:= intersection[composite[cross[Id, inverse[FIRST]], FIRST],
           composite[cross[Id, inverse[SECOND]], SECOND]] :=
         composite[cross[inverse[DUP], Id], TWIST]
```

Lemma.

```
In[24]:= symdif[composite[IMAGE[cross[Id, x]], IMAGE[cross[inverse[DUP], Id]]],
           IMAGE[cross[inverse[DUP], x]]] // VSNormality

Out[24]= union[intersection[complement[IMAGE[cross[inverse[DUP], x]]],
            composite[IMAGE[cross[Id, x]], IMAGE[cross[inverse[DUP], Id]]]],
           intersection[composite[complement[IMAGE[cross[Id, x]]],
             IMAGE[cross[inverse[DUP], Id]]], IMAGE[cross[inverse[DUP], x]]]] == 0
```

```
In[25]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[26]:= SubstTest[equal, 0, symdif[u, v],
          {u -> composite[IMAGE[cross[Id, x]], IMAGE[cross[inverse[DUP], Id]]],
           v -> IMAGE[cross[inverse[DUP], x]]}]

Out[26]= True == equal[composite[IMAGE[cross[Id, x]], IMAGE[cross[inverse[DUP], Id]]],
           IMAGE[cross[inverse[DUP], x]]]
```

```
In[28]:= composite[IMAGE[cross[Id, x_]], IMAGE[cross[inverse[DUP], Id]]] :=
          IMAGE[cross[inverse[DUP], x]]
```

---

the CUP rule

In this section, **reify** is used to remove the variables **y** and **z** from the following equation. The two variables **y** and **z** can be replaced by a single variable by considering them to be the first and second components of an ordered pair.

```
In[29]:= composite[CUP, cross[APPLY[CHAR[setpart[x]], intersection[setpart[x], y]],
            APPLY[CHAR[setpart[x]], intersection[setpart[x], z]]], DUP] ==
          APPLY[CHAR[setpart[x]], intersection[setpart[x], union[y, z]]]

Out[29]= True
```

First **reify** is applied on right side of this equation, and the result is made into a temporary rewrite rule.

```
In[30]:= Map[composite[VERTSECT[#], cross[id[P[setpart[x]]], id[P[setpart[x]]]]] &,
          SubstTest[reify, y, APPLY[v, intersection[w, union[first[y], second[y]]]],
           {v → CHAR[setpart[x]], w → setpart[x]}]]

Out[30]= composite[CUP,
          intersection[composite[inverse[SECOND], CUP, intersection[composite[inverse[SECOND],
              CAP, cross[composite[IMAGE[RIGHT[0]], RC[setpart[x]], IMAGE[id[setpart[x]]]],
               composite[IMAGE[RIGHT[0]], RC[setpart[x]], IMAGE[id[setpart[x]]]]]],
             composite[inverse[FIRST], IMAGE[RIGHT[set[0]]], IMAGE[id[setpart[x]]], SECOND]]],
           composite[inverse[FIRST], IMAGE[RIGHT[set[0]]], IMAGE[id[setpart[x]]], FIRST]],
          id[cart[P[setpart[x]], P[setpart[x]]]] ==
         composite[CHAR[setpart[x]], CUP]

In[31]:= composite[CUP,
          intersection[composite[inverse[SECOND], CUP, intersection[composite[inverse[SECOND],
              CAP, cross[composite[IMAGE[RIGHT[0]], RC[setpart[x_]], IMAGE[id[setpart[x_]]]],
               composite[IMAGE[RIGHT[0]], RC[setpart[x_]], IMAGE[id[setpart[x_]]]]]],
             composite[inverse[FIRST], IMAGE[RIGHT[set[0]]],
              IMAGE[id[setpart[x_]]], SECOND]]],
           composite[inverse[FIRST], IMAGE[RIGHT[set[0]]], IMAGE[id[setpart[x_]]], FIRST]],
          id[cart[P[setpart[x_]], P[setpart[x_]]]] :=
         composite[CHAR[setpart[x]], CUP]
```

Apply **reify** to the left side now yields:

```
In[32]:= Map[composite[VERTSECT[#], cross[id[P[setpart[x]]], id[P[setpart[x]]]]] &,
          SubstTest[reify, y, composite[u,
            cross[APPLY[v, intersection[w, first[y]]], APPLY[v, intersection[w, second[y]]]],
            DUP], {u → CUP, v → CHAR[setpart[x]], w → setpart[x]}]]

Out[32]= composite[CHAR[setpart[x]], CUP] == composite[
           IMAGE[cross[inverse[DUP], CUP]], CROSS, cross[CHAR[setpart[x]], CHAR[setpart[x]]]]

In[33]:= composite[CHAR[setpart[x_]], CUP] := composite[
           IMAGE[cross[inverse[DUP], CUP]], CROSS, cross[CHAR[setpart[x]], CHAR[setpart[x]]]]
```

The **setpart** wrapper is replaced with a sethood literal:

```
In[34]:= SubstTest[implies, equal[x, setpart[y]], equal[composite[CHAR[x], CUP],
           composite[IMAGE[cross[inverse[DUP], CUP]], CROSS, cross[CHAR[x], CHAR[x]]]], y → x]

Out[34]= or[equal[composite[CHAR[x], CUP], composite[IMAGE[cross[inverse[DUP], CUP]],
             CROSS, cross[CHAR[x], CHAR[x]]]], not[member[x, V]]] == True

In[35]:= (% /. x → x_) /. Equal -> SetDelayed
```

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

```
In[36]:= SubstTest[implies, equal[0, z],
          equal[composite[z, u], composite[v, cross[z, z]]], z → CHAR[x]]

Out[36]= or[equal[composite[v, cross[CHAR[x], CHAR[x]]], composite[CHAR[x], u]], member[x, V]] ==
          True
```

```
In[37]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

Combining these two cases yields the final result:

```
In[38]:= SubstTest[and, implies[p, q], or[p, q],
          {p → member[x, V], q :> equal[composite[CHAR[x], CUP],
             composite[IMAGE[cross[inverse[DUP], CUP]], CROSS, cross[CHAR[x], CHAR[x]]]]}]

Out[38]= True == equal[composite[CHAR[x], CUP],
           composite[IMAGE[cross[inverse[DUP], CUP]], CROSS, cross[CHAR[x], CHAR[x]]]]

In[39]:= composite[CHAR[x_], CUP] :=
          composite[IMAGE[cross[inverse[DUP], CUP]], CROSS, cross[CHAR[x], CHAR[x]]]
```

the CAP rule

In this section, the variables **y** and **z** are eliminated from the following equation. As before, to simplify the application of **reify**, these two variables are replaced by a single one by using **first** and **second**.

```
In[40]:= composite[CAP, cross[APPLY[CHAR[setpart[x]], intersection[setpart[x], y]],
            APPLY[CHAR[setpart[x]], intersection[setpart[x], z]]], DUP] ==
           APPLY[CHAR[setpart[x]], intersection[setpart[x], y, z]]

Out[40]= True
```

First **reify** is applied to the right side and the result is made into a temporary rewrite rule:

```
In[41]:= Map[composite[VERTSECT[#], cross[id[P[setpart[x]]], id[P[setpart[x]]]]] &,
           SubstTest[reify, y, APPLY[v, intersection[w, first[y], second[y]]],
            {v → CHAR[setpart[x]], w → setpart[x]}]]

Out[41]= composite[CUP,
           intersection[composite[inverse[FIRST], CAP, cross[IMAGE[RIGHT[set[0]]], composite[
               IMAGE[RIGHT[set[0]]], IMAGE[id[setpart[x]]]]]], composite[inverse[SECOND], CUP,
             cross[composite[IMAGE[RIGHT[0]], RC[setpart[x]], IMAGE[id[setpart[x]]]],
              composite[IMAGE[RIGHT[0]], RC[setpart[x]], IMAGE[id[setpart[x]]]]]]],
           id[cart[P[setpart[x]], P[setpart[x]]]] == composite[CHAR[setpart[x]],
           CAP, id[cart[P[setpart[x]], P[setpart[x]]]]]

In[42]:= composite[CUP,
           intersection[composite[inverse[FIRST], CAP, cross[IMAGE[RIGHT[set[0]]], composite[
               IMAGE[RIGHT[set[0]]], IMAGE[id[setpart[x_]]]]]], composite[inverse[SECOND], CUP,
             cross[composite[IMAGE[RIGHT[0]], RC[setpart[x_]], IMAGE[id[setpart[x_]]]],
              composite[IMAGE[RIGHT[0]], RC[setpart[x_]], IMAGE[id[setpart[x_]]]]]]],
           id[cart[P[setpart[x_]], P[setpart[x_]]]] := composite[CHAR[setpart[x]],
           CAP, id[cart[P[setpart[x]], P[setpart[x]]]]]
```

Apply **reify** to the left side now yields:

```
In[43]:= Map[composite[VERTSECT[#], cross[id[P[setpart[x]]], id[P[setpart[x]]]]] &,
          SubstTest[reify, y, composite[u,
            cross[APPLY[v, intersection[w, first[y]]], APPLY[v, intersection[w, second[y]]]],
            DUP] , {u → CAP, v → CHAR[setpart[x]], w → setpart[x]}]]

Out[43]= composite[CHAR[setpart[x]], CAP, id[cart[P[setpart[x]], P[setpart[x]]]] == composite[
           IMAGE[cross[inverse[DUP], CAP]], CROSS, cross[CHAR[setpart[x]], CHAR[setpart[x]]]]

In[44]:= composite[CHAR[setpart[x_]], CAP, id[cart[P[setpart[x_]], P[setpart[x_]]]] :=
          composite[IMAGE[cross[inverse[DUP], CAP]],
           CROSS, cross[CHAR[setpart[x]], CHAR[setpart[x]]]]
```

The **setpart** wrapper is replaced with a sethood literal.

```
In[45]:= SubstTest[implies, equal[x, setpart[y]],
          equal[composite[CHAR[x], CAP, id[cart[P[x], P[x]]]],
           composite[IMAGE[cross[inverse[DUP], CAP]], CROSS, cross[CHAR[x], CHAR[x]]], y → x]

Out[45]= or[equal[composite[CHAR[x], CAP, id[cart[P[x], P[x]]]],
            composite[IMAGE[cross[inverse[DUP], CAP]], CROSS, cross[CHAR[x], CHAR[x]]],
           not[member[x, V]]] == True

In[46]:= (% /. x → x_) /. Equal → SetDelayed
```

Again, the same result holds when **x** is not a set, so one can dispense with the sethood literal.

```
In[47]:= SubstTest[and, implies[p, q], or[p, q],
          {p → member[x, V], q -> equal[composite[CHAR[x], CAP, id[cart[P[x], P[x]]]],
             composite[IMAGE[cross[inverse[DUP], CAP]], CROSS, cross[CHAR[x], CHAR[x]]]]}]

Out[47]= True == equal[composite[CHAR[x], CAP, id[cart[P[x], P[x]]]],
           composite[IMAGE[cross[inverse[DUP], CAP]], CROSS, cross[CHAR[x], CHAR[x]]]]

In[48]:= composite[CHAR[x_], CAP, id[cart[P[x_], P[x_]]]] :=
          composite[IMAGE[cross[inverse[DUP], CAP]], CROSS, cross[CHAR[x], CHAR[x]]]
```