

x2896

Johan G. F. Belinfante
2004 August 19

```
In[1]:= SetDirectory["i:"]; << goedel60.18c; << tools.m;

:Package Title: goedel60.18c          2004 August 18 at 9:00 p.m.

It is now: 2004 Aug 19 at 11:24

Loading Simplification Rules

TOOLS.M          Revised 2004 August 11

weightlimit = 40
```

summary

The use of **funpart** wrappers to prove theorems about functions is a favorite technique, which seems to work well because it enables one to exploit conditional rewrite rules for functions. This notebook contains a (slightly awkward) derivation of a rewrite rule for test suite example **x2896** that makes heavy use of **funpart** wrappers.

Comment: The total execution time for this notebook, not counting the considerable time it takes to load *Mathematica* and the **GOEDEL** program, is about 15 seconds on an HP Pavilion a574n computer, but this could be reduced to 3.5 seconds by turning off the **simplify** and **cond** flags, which do not affect any of the results obtained.

x2896

Lemma.

```
In[2]:= SubstTest[implies, equal[w, funpart[z]],
              FUNCTION[composite[funpart[x], w, funpart[y]]], w → z]

Out[2]= or[FUNCTION[composite[funpart[x], z, funpart[y]]], not[FUNCTION[z]]] == True

In[3]:= or[FUNCTION[composite[funpart[x_], z_, funpart[y_]]],
           not[FUNCTION[z_]]] := True
```

Removing the variable z yields a somewhat awkward technical result:

```
In[4]:= Map[equal[V, #] &, SubstTest[class, z,
      implies[subclass[P[z], w], subclass[P[composite[u, z, v]], w]],
      {u → funpart[y], v → funpart[x], w → FUNS}]] // Reverse

Out[4]= subclass[image[
      inverse[LB[composite[cross[inverse[funpart[x]], funpart[y]], inverse[E]]]],
      FUNS], FUNS] == True

In[5]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

A technical lemma is needed to replace this awkward result with something more understandable:

```
In[6]:= subclass[IMAGE[cross[inverse[funpart[x]], funpart[y]], inverse[LB[composite[
      cross[inverse[funpart[x]], funpart[y]], inverse[E]]]]] // AssertTest

Out[6]= subclass[IMAGE[cross[inverse[funpart[x]], funpart[y]], inverse[
      LB[composite[cross[inverse[funpart[x]], funpart[y]], inverse[E]]]]] == True

In[7]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

A simpler result is obtained by using the lemma to remove the awkward expression obtained when the variable z was removed.

```
In[8]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
      {u → image[IMAGE[cross[inverse[funpart[x]], funpart[y]], FUNS],
      v → image[inverse[LB[composite[cross[inverse[funpart[x]], funpart[y]],
      inverse[E]]]], FUNS], w → FUNS}]

Out[8]= subclass[image[IMAGE[cross[inverse[funpart[x]], funpart[y]], FUNS], FUNS] ==
      True

In[9]:= subclass[
      image[IMAGE[cross[inverse[funpart[y_]], funpart[x_]], FUNS], FUNS] := True
```

The final result is obtained by using equality substitution to remove the two **funpart** wrappers.

```
In[10]:= SubstTest[implies, and[equal[u, funpart[inverse[x]]], equal[v, funpart[y]],
      subclass[image[IMAGE[cross[inverse[u], v]], FUNS], FUNS],
      {u → inverse[x], v → y}]

Out[10]= or[not[FUNCTION[y]], not[FUNCTION[inverse[x]]],
      subclass[image[IMAGE[cross[x, y]], FUNS], FUNS]] == True
```

```
In[11]:= or[not[FUNCTION[y_]], not[FUNCTION[inverse[x_]]],  
          subclass[image[IMAGE[cross[x_, y_]], FUNS], FUNS]] := True
```