**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 \rightarrow 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:

```math
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
```

```math
    inverse[LB[composite[cross[inverse[funpart[x]], funpart[y]], inverse[E]]], 
    FUNS], FUNS] = True
```

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

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

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

```math
    LB[composite[cross[inverse[funpart[x]], funpart[y]], inverse[E]]]]] = True
```

```math
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.

```math
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}]
```

```math
```

```math
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.

```math
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}]
```

```math
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