

# equality substitution

*Johan G. F. Belinfante*  
2004 June 7

```
In[1]:= SetDirectory["c:/goedel/goedel/2004/jun/07"]; << goedel58.06c; << tools.m
:Package Title: goedel58.06c      2004 June 6 at 8:15 p.m.
It is now: 2004 Jun 7 at 16:19
Loading Simplification Rules
TOOLS.M              Revised 2004 May 14
weightlimit = 40
```

---

## summary

The **GOEDEL** program currently has only weak equality substitution rules, which makes for rather tedious derivations. In this notebook two new rewrite rules are proposed that would make it easier to use equality substitution arguments.

---

## behavior of current program on some examples

Here are some statements whose truth or falsity can not be decided with the current version of the **GOEDEL** program

```
In[2]:= implies[equal[x, y], equal[U[U[x]], U[U[y]]]]
Out[2]= or[equal[U[U[x]], U[U[y]]], not[equal[x, y]]]

In[3]:= implies[equal[x, P[y]], equal[U[x], y]]
Out[3]= or[equal[y, U[x]], not[equal[x, P[y]]]]

In[4]:= implies[equal[P[x], y], equal[x, U[y]]]
Out[4]= or[equal[x, U[y]], not[equal[y, P[x]]]]

In[5]:= implies[equal[x, y],
               equal[composite[x, id[z], inverse[x]], composite[y, id[z], inverse[y]]]]
Out[5]= or[equal[composite[x, id[z], inverse[x]], composite[y, id[z], inverse[y]]],
         not[equal[x, y]]]
```

```

In[6]:= and[equal[x, y], not[equal[U[x], U[y]]]] // NotNotTest
Out[6]= and[equal[x, y], not[equal[U[x], U[y]]]] = False

In[7]:= and[equal[0, x], equal[U[x], singleton[x]]]
Out[7]= and[equal[0, x], equal[singleton[x], U[x]]]

In[8]:= and[equal[singleton[0], x], equal[A[x], P[x]]]
Out[8]= and[equal[x, singleton[0]], equal[A[x], P[x]]]

```

---

## two new equality substitution rules

These new rules are considered so self-evident that no attempt is made to justify them.

```

In[9]:= or[not[equal[x_, y_]], p_] := True /; (p /. x → y)
In[10]:= and[equal[x_, y_], p_] := False /; (not[p] /. x → y)

```

---

## the examples revisited

With the new rules, the **GOEDEL** program can decide all the examples in the first section.

```

In[11]:= implies[equal[x, y], equal[U[U[x]], U[U[y]]]]
Out[11]= True

In[12]:= implies[equal[x, P[y]], equal[U[x], y]]
Out[12]= True

In[13]:= implies[equal[P[x], y], equal[x, U[y]]]
Out[13]= True

In[14]:= implies[equal[x, y],
  equal[composite[x, id[z], inverse[x]], composite[y, id[z], inverse[y]]]
Out[14]= True

In[15]:= and[equal[x, y], not[equal[U[x], U[y]]]]
Out[15]= False

In[16]:= and[equal[0, x], equal[0, singleton[x]]]
Out[16]= False

```

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

```
Out[17]= False
```

```
In[18]:= and[equal[0, x], equal[U[x], singleton[x]]]
```

```
Out[18]= False
```

```
In[19]:= and[equal[singleton[0], x], equal[A[x], P[x]]]
```

```
Out[19]= False
```