a wrapper for well-founded relations

Johan G. F. Belinfante
2004 September 18

```
In[1]:= SetDirectory["i:"]; << goedel61.14a; << tools.m
```

```
:Package Title: goedel61.14a          2004 September 14 at 10:00 p.m.

It is now:  2004 Sep 18 at 17:6

Loading Simplification Rules

TOOLS.M                       Revised 2004 September 13

weightlimit = 40
```

summary

A wrapper **wf[x]** for well-founded relations is introduced, and some of its basic properties are deduced.

definition

The wrapper **wf[x]** is defined by the following membership rule:

```
In[2]:= member[w_, wf[x_]] :=
         and[member[w, x], member[first[w], V], not[member[second[w], U[subvar[x]]]]]
```

normalization

The basic normalization result:

```
In[3]:= wf[x] // Normality // Reverse
```

```
Out[3]= composite[id[complement[U[subvar[x]]]], x] == wf[x]
```

```
In[4]:= composite[id[complement[U[subvar[x_]]]], x_] := wf[x]
```

A related rewrite rule:

```
In[5]:= Assoc[id[x], id[complement[U[subvar[y]]]], y] // Reverse

Out[5]= composite[id[intersection[x, complement[U[subvar[y]]]]], y] ==
         composite[id[x], wf[y]]

In[6]:= composite[id[intersection[x_, complement[U[subvar[y_]]]]], y_] :=
         composite[id[x], wf[y]]
```

basic properties

Simplification rule.

```
In[7]:= Assoc[Id, id[complement[U[subvar[x]]]], x]

Out[7]= composite[Id, wf[x]] == wf[x]

In[8]:= composite[Id, wf[x_]] := wf[x]
```

The well-founded part of **composite[Id, x]** is the same as the well-founded part of **x**.

```
In[9]:= wf[composite[Id, x]] // Normality

Out[9]= wf[composite[Id, x]] == wf[x]

In[10]:= wf[composite[Id, x_]] := wf[x]
```

using wf to remove literals involving the WELLFOUNDED predicate

Lemma.

```
In[11]:= equal[intersection[x, P[complement[U[x]]]],
           intersection[x, singleton[0]]] // AssertTest

Out[11]= equal[intersection[x, P[complement[U[x]]]],
           intersection[x, singleton[0]]] == True

In[12]:= intersection[x_, P[complement[U[x_]]]] := intersection[x, singleton[0]]
```

The only set subvariant under **wf[x]** is the empty set.

```
In[13]:= subvar[wf[x]] // Renormality

Out[13]= subvar[wf[x]] == singleton[0]

In[14]:= subvar[wf[x_]] := singleton[0]
```

The relation **wf[x]** is well-founded. This rewrite rule can be used to remove literals involving the **WELLFOUNDED** predicate by introducing **wf** wrappers.

```
In[15]:= WELLFOUNDED[wf[x]] // AssertTest
Out[15]= WELLFOUNDED[wf[x]] == True
In[16]:= WELLFOUNDED[wf[x_]] := True
```

For example, the theorem that any well-founded relation is irreflexive can now be expressed using a single literal:

```
In[17]:= SubstTest[implies, WELLFOUNDED[w], equal[fix[w], 0], w → wf[x]]
Out[17]= equal[0, fix[wf[x]]] == True
In[18]:= fix[wf[x_]] := 0
```

Similarly:

```
In[19]:= SubstTest[implies, WELLFOUNDED[w], equal[fix[trv[w]], 0], w → wf[x]]
Out[19]= equal[0, fix[trv[wf[x]]]] == True
In[20]:= fix[trv[wf[x_]]] := 0
```

---

wrapper removal

This section is concerned with rules needed to remove the **wf** wrapper, restoring the **WELLFOUNDED** predicate. We begin with a simple lemma:

```
In[21]:= equal[intersection[range[x], U[subvar[x]]], U[subvar[x]]]
Out[21]= True
```

This justifies adding the following rewrite rule:

```
In[22]:= intersection[range[x_], U[subvar[x_]]] := U[subvar[x]]
```

The following rule can be used to remove **wf** wrappers. Note that this also implies that every well-founded relation can be written in the form **wf[x]**.

```
In[23]:= equal[x, wf[x]] // AssertTest
Out[23]= equal[x, wf[x]] == WELLFOUNDED[x]
```

```
In[24]:= equal[x_, wf[x_]] := WELLFOUNDED[x]
```

Corollary: The **wf** wrapper satisfies the following idempotence property:

```
In[25]:= equal[wf[wf[x]], wf[x]]

Out[25]= True

In[26]:= wf[wf[x_]] := wf[x]
```

---

subclass and intersection results

The well-founded part of **x** is contained in **x**.

```
In[27]:= subclass[wf[x], x] // AssertTest

Out[27]= subclass[wf[x], x] == True

In[28]:= subclass[wf[x_], x_] := True
```

The reverse inclusion holds if and only if **x** is well-founded.

```
In[29]:= subclass[x, wf[x]] // AssertTest

Out[29]= subclass[x, wf[x]] == WELLFOUNDED[x]

In[30]:= subclass[x_, wf[x_]] := WELLFOUNDED[x]
```

Every subclass of a well-founded relation is well-founded.

```
In[31]:= SubstTest[implies,
          and[subclass[x, v], WELLFOUNDED[v]], WELLFOUNDED[x], v → wf[y]]

Out[31]= or[not[subclass[x, wf[y]]], WELLFOUNDED[x]] == True

In[32]:= or[not[subclass[x_, wf[y_]]], WELLFOUNDED[x_]] := True
```

Two corollaries of this are:

```
In[33]:= SubstTest[implies, and[subclass[u, v], WELLFOUNDED[v]],
          WELLFOUNDED[u], {u → intersection[x, wf[y]], v → wf[y]}]

Out[33]= WELLFOUNDED[intersection[x, wf[y]]] == True

In[34]:= WELLFOUNDED[intersection[x_, wf[y_]]] := True
```

```
In[35]:= SubstTest[implies, and[subclass[u, v], WELLFOUNDED[v]],
          WELLFOUNDED[u], {u → composite[id[x], wf[y]], v → wf[y]}]
Out[35]= WELLFOUNDED[composite[id[x], wf[y]]] == True

In[36]:= WELLFOUNDED[composite[id[x_], wf[y_]]] := True
```

Wrapper versions of these results:

```
In[37]:= equal[wf[intersection[x, wf[y]]], intersection[x, wf[y]]]
Out[37]= True

In[38]:= wf[intersection[x_, wf[y_]]] := intersection[x, wf[y]]

In[39]:= equal[wf[composite[id[x], wf[y]]], composite[id[x], wf[y]]]
Out[39]= True

In[40]:= wf[composite[id[x_], wf[y_]]] := composite[id[x], wf[y]]
```

---

some examples

The following two examples imply that **wf** cannot be monotone.

```
In[41]:= wf[0] // Normality
Out[41]= wf[0] == 0

In[42]:= wf[0] := 0

In[43]:= wf[V] // Normality
Out[43]= wf[V] == 0

In[44]:= wf[V] := 0
```

The following two examples of well-founded relations have been extensively investigated.

```
In[45]:= SubstTest[composite, id[complement[U[subvar[x]]]], x, x → E] // Reverse
Out[45]= wf[E] == composite[id[REGULAR], E]

In[46]:= wf[E] := composite[id[REGULAR], E]

In[47]:= SubstTest[composite, id[complement[U[subvar[x]]]], x, x → PS] // Reverse
Out[47]= wf[PS] == composite[id[FINITE], PS]
```

```
In[48]:= wf[PS] := composite[id[FINITE], PS]
```

some other examples

```
In[49]:= wf[Id] // Normality
Out[49]= wf[Id] == 0

In[50]:= wf[Id] := 0

In[51]:= wf[id[x]] // Normality
Out[51]= wf[id[x]] == 0

In[52]:= wf[id[x_]] := 0

In[53]:= wf[S] // Normality
Out[53]= wf[S] == 0

In[54]:= wf[S] := 0

In[55]:= wf[inverse[S]] // Normality
Out[55]= wf[inverse[S]] == 0

In[56]:= wf[inverse[S]] := 0

In[57]:= wf[inverse[E]] // Normality
Out[57]= wf[inverse[E]] == 0

In[58]:= wf[inverse[E]] := 0
```

serendipity: wf[Di] = 0

Some unexpected discoveries were made in the course of obtaining a formula for **wf[Di]**. The class of sets subvariant under the diversity relation **Di** is the complement of the class of singletons.

```
In[59]:= subvar[Di]
Out[59]= complement[range[SINGLETON]]
```

If one attempts to obtain a formula for **wf[Di]** by using a normality test, one quickly discovers that one needs a formula for the class **U[complement[range[SINGLE-TON]]]**. Most of this section is concerned with obtaining such a formula, and related

results concerning the class of sets with precisely two elements, **image[PAIRSET, Di]**. This class contains neither singletons nor the empty set:

```
In[60]:= member[singleton[x], image[PAIRSET, Di]] // AssertTest
Out[60]= member[singleton[x], image[PAIRSET, Di]] == False
In[61]:= member[singleton[x_], image[PAIRSET, Di]] := False
```

Removing the variable **x** here yields:

```
In[62]:= SubstTest[class, x, member[singleton[x], y], y -> image[PAIRSET, Di]] // Reverse
Out[62]= image[inverse[SINGLETON], image[PAIRSET, Di]] == 0
In[63]:= image[inverse[SINGLETON], image[PAIRSET, Di]] := 0
```

From this one readily derives the unsurprising fact that this class is disjoint fron the class of singetons:

```
In[64]:= ImageComp[SINGLETON, inverse[SINGLETON], image[PAIRSET, Di]]
Out[64]= intersection[image[PAIRSET, Di], range[SINGLETON]] == 0
In[65]:= intersection[image[PAIRSET, Di], range[SINGLETON]] := 0
```

The needed formula about the union of the complement of the class of singletons can be derived as follows:

```
In[66]:= Map[assert, SubstTest[implies, subclass[x, y], subclass[U[x], U[y]],
           {x -> image[PAIRSET, Di], y -> complement[range[SINGLETON]]}]]
Out[66]= equal[V, U[complement[range[SINGLETON]]]] == True
In[67]:= U[complement[range[SINGLETON]]] := V
```

Now one can use **Normality** to derive a formula for **wf[Di]**.

```
In[68]:= wf[Di] // Normality
Out[68]= wf[Di] == 0
In[69]:= wf[Di] := 0
```

---

empty wf[x]

The following provides a simple criterion for **wf[x]** to be empty.

```
In[70]:= SubstTest[equal, 0, composite[id[w], x], w → complement[U[subvar[x]]]]
Out[70]= equal[0, wf[x]] == subclass[range[x], U[subvar[x]]]
In[71]:= equal[0, wf[x_]] := subclass[range[x], U[subvar[x]]]
```