Theorem DK-K: DEDEKIND is invariant under K

Johan G. F. Belinfante
2005 February 4

```
In[1]:= SetDirectory["i:"]; << goedel66.03a; << tools.m

        :Package Title: goedel66.03a          2005 February 3 at 11:45 p.m.

        It is now:  2005 Feb 4 at 18:18

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 January 7

        weightlimit = 40
```

---

summary

If a single new element is adjoined to a Dedekind finite set, the union is again Dedekind finite. In other words, the class **DEDEKIND** is invariant under the cover relation **K**. The 26 step automated proof of this fact obtained by the author 2002 June 2 using McCune's automated reasoning program **Otter** inspired the shorter proof given here, but the three Skolem functions in that proof are avoided by taking advantage of new rewrite rules currently available in the **GOEDEL** program that were not available when the **Otter** proof was carried out. As an immediate corollary of this theorem, it follows that the union of any finite set and a Dedekind finite set is Dedekind finite.

---

derivation

Lemma 1 makes use of the fact that deleting an element from a set, and then adjoining another yields an equipollent set.

```
In[2]:= SubstTest[implies, subclass[u, v],
          subclass[composite[x, u, y], composite[x, v, y]],
          {u → composite[K, inverse[K]], v → Q, x → Q, y → PS}]

Out[2]= subclass[composite[Q, K, inverse[K], PS], composite[Q, PS]] == True

In[3]:= % /. Equal → SetDelayed
```

Lemma 2 follows from lemma 1 and facts about composites of **K** with **inverse[K]** and with the proper subset relation **PS**.

```
In[4]:= Map[equal[composite[Q, PS], #] &, Assoc[composite[Q, inverse[K]], K, PS]]

Out[4]= equal[composite[Q, PS], composite[Q, inverse[K], PS, PS]] == True

In[5]:= composite[Q, inverse[K], PS, PS] := composite[Q, PS]
```

Theorem. The class **DEDEKIND** is invariant under **K**.

```
In[6]:= SubstTest[subclass, image[x, fix[y]],
         fix[composite[x, y, inverse[x]]], {x → inverse[K], y → composite[Q, PS]}]

Out[6]= subclass[image[K, DEDEKIND], DEDEKIND] == True

In[7]:= % /. Equal → SetDelayed
```

---

a corollary

Invariance under the cover relation implies invariance under the process of adjoining any finite set, and conversely.

```
In[8]:= SubstTest[invariant, composite[z, id[cart[y, V]], inverse[SECOND]],
          x, {y → FINITE, z → CUP}] // Reverse

Out[8]= subclass[image[CUP, cart[FINITE, x]], x] == subclass[image[K, x], x]

In[9]:= subclass[image[CUP, cart[FINITE, x_]], x_] := subclass[image[K, x], x]
```

Corollary. The union of a finite set and a Dedekind finite set is Dedekind finite.

```
In[10]:= SubstTest[implies, and[member[u, v], subclass[v, w]],
          member[u, w], {u → pair[x, y], v → cart[FINITE, DEDEKIND],
           w → image[inverse[CUP], DEDEKIND]}]

Out[10]= or[member[union[x, y], DEDEKIND],
          not[member[x, FINITE]], not[member[y, DEDEKIND]]] == True

In[11]:= or[member[union[x_, y_], DEDEKIND],
          not[member[x_, FINITE]], not[member[y_, DEDEKIND]]] := True
```

Lemma.

```
In[12]:= SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
          {u → set[0], v → FINITE, w -> composite[CUP, id[cart[V, x]], inverse[FIRST]]}]

Out[12]= subclass[x, image[CUP, cart[FINITE, x]]] == True

In[13]:= subclass[x_, image[CUP, cart[FINITE, x_]]] := True
```

This yields a variable-free version of the corollary.

```
In[14]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u → image[CUP, cart[FINITE, DEDEKIND]], v → DEDEKIND}]

Out[14]= True == equal[DEDEKIND, image[CUP, cart[FINITE, DEDEKIND]]]

In[15]:= image[CUP, cart[FINITE, DEDEKIND]] := DEDEKIND
```

rewrite rules

Another theorem proved using **Otter** says that **DEDEKIND** is invariant under **inverse[K]**. This seems at first blush to be less interesting because it is an immediate corollary of a more general theorem that any subset of a Dedekind finite set is Dedekind finite.

```
In[16]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
          {u → inverse[K], v → inverse[S], w → DEDEKIND}]

Out[16]= subclass[image[inverse[K], DEDEKIND], DEDEKIND] == True

In[17]:= % /. Equal → SetDelayed
```

This inclusion can be strengthened to an equation. An elementary lemma is needed to do this:

```
In[18]:= Map[subclass[DEDEKIND, #] &, ImageComp[inverse[K], K, DEDEKIND]] // Reverse

Out[18]= subclass[DEDEKIND, image[inverse[K], image[K, DEDEKIND]]] == True

In[19]:= % /. Equal → SetDelayed
```

This yields the reverse inclusion:

```
In[20]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
          {u → DEDEKIND, v -> image[inverse[K], image[K, DEDEKIND]],
           w -> image[inverse[K], DEDEKIND]}]

Out[20]= subclass[DEDEKIND, image[inverse[K], DEDEKIND]] == True
```

```
In[21]:= % /. Equal → SetDelayed
```

The two inclusions can be combined into an equation and made into a rewrite rule.

```
In[22]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> DEDEKIND, v -> image[inverse[K], DEDEKIND]}]

Out[22]= True == equal[DEDEKIND, image[inverse[K], DEDEKIND]]

In[23]:= image[inverse[K], DEDEKIND] := DEDEKIND
```

A similar rewrite rule for **image[K, DEDEKIND]** will now be derived. Again one needs a reverse inclusion:

```
In[24]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
          {u → Id, v -> union[id[set[0]], composite[K, inverse[K]]], w → DEDEKIND}]

Out[24]= subclass[DEDEKIND, union[image[K, DEDEKIND], set[0]]] == True

In[25]:= % /. Equal → SetDelayed
```

This lemma is also needed:

```
In[26]:= Map[not, SubstTest[subclass, image[w, x], range[w], w → K]]

Out[26]= member[0, image[K, x]] == False

In[27]:= member[0, image[K, x_]] := False
```

An equation is obtained that can serve as a rewrite rule:

```
In[28]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> dif[DEDEKIND, set[0]], v -> image[K, DEDEKIND]}]

Out[28]= True == equal[image[K, DEDEKIND], intersection[DEDEKIND, complement[set[0]]]]

In[29]:= image[K, DEDEKIND] := intersection[DEDEKIND, complement[set[0]]]
```