properties of Dedekind finite sets

Johan G. F. Belinfante
2005 January 27

```
In[1]:= SetDirectory["i:"]; << goedel65.27b; << tools.m

        :Package Title: goedel65.27b           2005 January 27 at 1:30 p.m.

        It is now:  2005 Jan 27 at 14:52

        Loading Simplification Rules

        TOOLS.M                      Revised 2005 January 7

        weightlimit = 40
```

summary

Some basic properties of Dedekind finiteness are derived in this notebook, culminating with the theorem that a subset of a Dedekind finite set is Dedekind finite.

properties of DEDEKIND

The only subset of a Dedekind finite set **x** equipollent to **x** is **x** itself.

```
In[2]:= SubstTest[implies, and[member[x, u], subclass[u, v]], member[x, v],
          {u -> intersection[image[inverse[w], set[y]], P[y]], v -> set[y]}] /. w → Q

Out[2]= or[equal[x, y], not[member[y, DEDEKIND]],
          not[member[pair[x, y], Q]], not[subclass[x, y]]] == True

In[3]:= or[equal[x_, y_], not[member[y_, DEDEKIND]],
          not[member[pair[x_, y_], Q]], not[subclass[x_, y_]]] := True
```

A set equipollent to a Dedekind finite set is Dedekind finite.

```
In[4]:= SubstTest[implies, and[member[pair[x, y], composite[Id, z]], member[y, w]],
          member[x, image[inverse[z], w]], {w → DEDEKIND, z → Q}] // MapNotNot

Out[4]= or[member[x, DEDEKIND],
          not[member[y, DEDEKIND]], not[member[pair[x, y], Q]]] == True
```

```
In[5]:= or[member[x_, DEDEKIND],
          not[member[y_, DEDEKIND]], not[member[pair[x_, y_], Q]]] := True
```

A set equipollent to a set that is not Dedekind finite is not Dedekind finite.

```
In[6]:= SubstTest[implies, and[member[pair[x, y], composite[Id, z]], member[x, w]],
         member[y, image[z, w]], {w → DEDEKIND, z → Q}]

Out[6]= or[member[y, DEDEKIND],
          not[member[x, DEDEKIND]], not[member[pair[x, y], Q]]] == True

In[7]:= or[member[y_, DEDEKIND],
          not[member[x_, DEDEKIND]], not[member[pair[x_, y_], Q]]] := True
```

A denumerable set is not Dedekind finite.

```
In[8]:= SubstTest[or, member[y, DEDEKIND],
         not[member[x, DEDEKIND]], not[member[pair[x, y], Q]], y → omega]

Out[8]= or[not[member[x, DEDEKIND]], not[member[pair[x, omega], Q]]] == True

In[9]:= or[not[member[x_, DEDEKIND]], not[member[pair[x_, omega], Q]]] := True
```

A Dedekind finite set can not be denumerable.

```
In[10]:= and[member[x, DEDEKIND], member[pair[x, omega], Q]] // NotNotTest

Out[10]= and[member[x, DEDEKIND], member[pair[x, omega], Q]] == False

In[11]:= and[member[x_, DEDEKIND], member[pair[x_, omega], Q]] := False
```

---

equipollence of disjoint unions

The **GOEDEL** program currently only has this variable-free statement about equipollence of disjoint unions.

```
In[12]:= subclass[intersection[cart[DISJOINT, DISJOINT], cross[Q, Q]],
          composite[inverse[CUP], Q, CUP]]

Out[12]= True
```

Lemma.

```
In[13]:= member[pair[w, pair[x, y]], composite[inverse[CUP], z]] // AssertTest

Out[13]= member[pair[w, pair[x, y]], composite[inverse[CUP], z]] ==
          and[member[w, V], member[x, V], member[y, V], member[pair[w, union[x, y]], z]]
```

```
In[14]:= member[pair[w_, pair[x_, y_]], composite[inverse[CUP], z_]] :=
          and[member[w, V], member[x, V], member[y, V], member[pair[w, union[x, y]], z]]
```

Restatement of the theorem on equipollence of disjoint unions.

```
In[15]:= SubstTest[implies, and[member[r, s], subclass[s, t]],
           member[r, t], {r → pair[pair[u, v], pair[x, y]],
            s -> intersection[cart[DISJOINT, DISJOINT], cross[Q, Q]],
            t -> composite[inverse[CUP], Q, CUP]}] // MapNotNot

Out[15]= or[member[pair[union[u, v], union[x, y]], Q],
           not[equal[0, intersection[u, v]]], not[equal[0, intersection[x, y]]],
           not[member[pair[u, x], Q]], not[member[pair[v, y], Q]]] == True

In[16]:= or[member[pair[union[u_, v_], union[x_, y_]], Q],
           not[equal[0, intersection[u_, v_]]], not[equal[0, intersection[x_, y_]]],
           not[member[pair[u_, x_], Q]], not[member[pair[v_, y_], Q]]] := True
```

Lemma.

```
In[17]:= member[pair[x, x], Q] // AssertTest

Out[17]= member[pair[x, x], Q] == member[x, V]

In[18]:= member[pair[x_, x_], Q] := member[x, V]
```

Corollary.

```
In[19]:= Map[implies[member[z, w], #] &, SubstTest[implies,
           and[member[pair[x, y], Q], member[pair[u, v], Q], disjoint[x, u],
            disjoint[y, v]], member[pair[union[x, u], union[y, v]], Q], {u → z, v → z}]]

Out[19]= or[member[pair[union[x, z], union[y, z]], Q],
           not[equal[0, intersection[x, z]]], not[equal[0, intersection[y, z]]],
           not[member[z, w]], not[member[pair[x, y], Q]]] == True

In[20]:= or[member[pair[union[x_, z_], union[y_, z_]], Q],
           not[equal[0, intersection[x_, z_]]], not[equal[0, intersection[y_, z_]]],
           not[member[z_, w_]], not[member[pair[x_, y_], Q]]] := True
```

subsets of Dedekind finite sets

The following facts are used in the next theorem.

```
In[21]:= {implies[and[p1, p5], p6], implies[p5, p7], implies[and[p4, p5], p8],
           implies[and[p2, p6, p7, p8], p9], implies[and[p3, p5], q1],
           implies[and[p5, p9, q1], q2]} /. {p1 → subclass[x, y],
           p2 → member[pair[x, y], Q], p3 → subclass[y, z], p4 → member[z, DEDEKIND],
           p5 → equal[w, dif[z, y]], p6 → disjoint[x, w], p7 → disjoint[y, w],
           p8 → member[w, V], p9 → member[pair[union[x, w], union[y, w]], Q],
           q1 → equal[z, union[w, y]], q2 → member[pair[union[x, dif[z, y]], z], Q]}

Out[21]= {True, True, True, True, True, True}
```

The use of bundling makes this go faster. The temporary variable **t** is eliminated at the end. The hypothesis of Dedekind finiteness has been relaxed.

```
In[22]:= Map[not, SubstTest[and, implies[and[p1, p5], p6],
            implies[p5, p7], implies[and[p1, p5], p8],
            implies[and[p1, p6, p7, p8], p9], implies[and[p1, p5], q1],
            implies[and[p5, p9, q1], q2], not[implies[and[p1, p5], q2]],
            {p1 → and[subclass[x, y], member[pair[x, y], Q],
               subclass[y, z], member[z, w]], p5 → equal[t, dif[z, y]],
             p6 → disjoint[x, t], p7 → disjoint[y, t], p8 → member[t, V],
             p9 → member[pair[union[x, t], union[y, t]], Q], q1 → equal[z, union[t, y]],
             q2 → member[pair[union[x, dif[z, y]], z], Q]}]] /. t → dif[z, y]

Out[22]= or[member[pair[union[x, intersection[z, complement[y]]], z], Q],
           not[member[z, w]], not[member[pair[x, y], Q]],
           not[subclass[x, y]], not[subclass[y, z]]] == True

In[23]:= or[member[pair[union[x_, intersection[z_, complement[y_]]], z_], Q],
           not[member[z_, w_]], not[member[pair[x_, y_], Q]],
           not[subclass[x_, y_]], not[subclass[y_, z_]]] := True

In[24]:= or[equal[x, y], not[equal[z, union[x, intersection[z, complement[y]]]]],
           not[subclass[x, y]], not[subclass[y, z]]] // AssertTest

Out[24]= or[equal[x, y], not[equal[z, union[x, intersection[z, complement[y]]]]],
           not[subclass[x, y]], not[subclass[y, z]]] == True

In[25]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The following argument uses a temporary variable **t** that is eliminated at the end. This makes it easier to follow the reasoning.

```
In[26]:= Map[not, SubstTest[and, implies[and[p1, p2, p3, p4, p5], p6],
           implies[and[p1, p3, p5], p7], implies[and[p4, p6, p7], p8],
           implies[and[p1, p3, p5, p8], p9], not[implies[and[p1, p2, p3, p4, p5], p9]],
           {p1 → subclass[x, y], p2 → member[pair[x, y], Q],
            p3 → subclass[y, z], p4 → member[z, DEDEKIND],
            p5 → equal[t, union[x, intersection[z, complement[y]]]],
            p6 -> member[pair[t, z], Q], p7 → subclass[t, z],
            p8 → equal[t, z], p9 → equal[x, y]}]] /.
         t -> union[x, intersection[z, complement[y]]]

Out[26]= or[equal[x, y], not[member[z, DEDEKIND]], not[member[pair[x, y], Q]],
           not[subclass[x, y]], not[subclass[y, z]]] == True

In[27]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

The variable for the innermost set is eliminated:

```
In[28]:= Map[equal[V, #] &, SubstTest[class, w,
            or[equal[w, x], not[member[y, u]], not[member[pair[w, x], v]],
             not[subclass[w, x]], not[subclass[x, y]]], {u → DEDEKIND, v → Q}]] // Reverse

Out[28]= or[member[x, DEDEKIND], not[member[x, V]],
           not[member[y, DEDEKIND]], not[subclass[x, y]]] == True

In[29]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

This can be cleaned up:

```
In[30]:= Map[not, SubstTest[and, implies[and[p1, p2], p3], implies[and[p1, p2, p3], p4],
           not[implies[and[p1, p2], p4]], {p1 → subclass[x, y],
            p2 → member[y, DEDEKIND], p3 → member[x, V], p4 → member[x, DEDEKIND]}]]

Out[30]= or[member[x, DEDEKIND], not[member[y, DEDEKIND]], not[subclass[x, y]]] == True

In[31]:= or[member[x_, DEDEKIND],
           not[member[y_, DEDEKIND]], not[subclass[x_, y_]]] := True
```

The remaining variables can be eliminated to produce a variable-free statement of the theorem.

```
In[33]:= Map[equal[0, composite[Id, complement[#]]] &,
           SubstTest[class, pair[x, y], or[member[x, z],
             not[member[y, z]], not[subclass[x, y]]], z → DEDEKIND]] // Reverse

Out[33]= subclass[image[inverse[S], DEDEKIND], DEDEKIND] == True

In[34]:= % /. Equal → SetDelayed
```

The inclusion can be sharpened to an equation.

```
In[35]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[inverse[S], DEDEKIND], v -> DEDEKIND}]

Out[35]= True == equal[DEDEKIND, image[inverse[S], DEDEKIND]]

In[37]:= image[inverse[S], DEDEKIND] := DEDEKIND
```