the class of connected relations

Johan G. F. Belinfante
2009 October 30

```
In[1]:= SetDirectory["l:"]; << goedel.09oct29c;<< tools.m

        :Package Title: goedel.09oct29c              2009 October 29 at 3:55 p.m.

        It is now:  2009 Oct 30 at 13:1

        Loading Simplification Rules

        TOOLS.M                                     Revised 2009 October 28

        weightlimit = 40
```

summary

The class **CONNEX** of connected relations is defined and its relation to the class of dichotomous relations is studied.

definition of CONNEX

A relation is **connected** if for every pair of distict elements belonging to the union of its domain and range, one of them is related to the other. For example, if **u** and **v** are numbers, then the law of trichotomy implies that **u** < **v** or **u** = **v** or **u** > **v**. In this case, these three possibilities are mutually exclusive. For a connected relation in general the three possibilities need not be mutually exclusive.

The class of connected relations is defined by the following membership rule.

```
In[2]:= member[x_, CONNEX] :=
         and[member[x, V], subclass[x, cart[V, V]], subclass[cart[union[domain[x], range[x]],
           union[domain[x], range[x]]], union[Id, x, inverse[x]]]]
```

a formula for CONNEX

Theorem. A formula for the class of connected relations.

```
In[3]:= Map[empty, symdif[CONNEX, fix[
            composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], CART, DUP, CUP, DORA]]] // Normality]

Out[3]= equal[CONNEX,
          fix[composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], CART, DUP, CUP, DORA]]] == True
```

```
In[4]:= fix[composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], CART, DUP, CUP, DORA]] := CONNEX
```

There is a similar formula for the class of irreflexive connected relations.

Lemma.

```
In[5]:= (symdif[intersection[CONNEX, P[Di]],
           fix[composite[inverse[HULL[SYM]], IMAGE[id[Di]], CART, DUP, CUP, DORA]] //
         Normality) /. Equal → SetDelayed
```

Theorem.

```
In[6]:= SubstTest[empty, symdif[u, v], {u -> intersection[CONNEX, P[Di]],
          v -> fix[composite[inverse[HULL[SYM]], IMAGE[id[Di]], CART, DUP, CUP, DORA]]}]

Out[6]= equal[fix[composite[inverse[HULL[SYM]], IMAGE[id[Di]], CART, DUP, CUP, DORA]],
          intersection[CONNEX, P[Di]]] == True
```

```
In[7]:= fix[composite[inverse[HULL[SYM]], IMAGE[id[Di]], CART, DUP, CUP, DORA]] :=
         intersection[CONNEX, P[Di]]
```

---

dichotomous relations

A relation is **dichotomous** if for every pair of elements belonging to the union of its domain and range, one of them is related to the other. In the literature, such a relation is also called a **total relation** or a **strictly connected relation**. A dichotomous relation must be reflexive. When **u** and **v** are numbers, either **u ≤ v** or **v ≤ u** (or both). The two possibilities are not exclusive, but if both **u ≤ v** or **v ≤ u** hold for numbers, then **u = v**. In general however the definition of a dichotomous relation does not require this antisymmetric property. In the **GOEDEL** program, the class of dichotomous relations is called **DICHOT**. The membership rule for this class uses the fixed point class instead of the union of the domain and range.

```
In[8]:= member[x, DICHOT]

Out[8]= and[equal[cart[fix[x], fix[x]], union[x, inverse[x]]], member[x, V]]
```

Using **fix[x]** instead of **udora[x] = union[domain[x], range[x]]** makes no difference. To avoid needless duplication, the **GOEDEL** program has this rewrite rule:

```
In[9]:= equal[cart[udora[x], udora[x]], union[x, inverse[x]]]

Out[9]= equal[cart[fix[x], fix[x]], union[x, inverse[x]]]
```

Lemma. A dichotomous relation is connected.

```
In[10]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
           not[implies[p1, p3]], {p1 → equal[cart[fix[x], fix[x]], union[x, inverse[x]]],
            p2 -> subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
              union[x, inverse[x]]], p3 -> subclass[cart[union[domain[x], range[x]],
               union[domain[x], range[x]]], union[Id, x, inverse[x]]]}]] // Reverse

Out[10]= or[not[equal[cart[fix[x], fix[x]], union[x, inverse[x]]]],
          subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
           union[Id, x, inverse[x]]]] == True

In[11]:= or[not[equal[cart[fix[x_], fix[x_]], union[inverse[x_], x_]]],
          subclass[cart[union[domain[x_], range[x_]], union[domain[x_], range[x_]]],
           union[Id, inverse[x_], x_]]] := True
```

Theorem.

```
In[12]:= Map[equal[V, #] &, SubstTest[class, x, not[member[x, t]], t → dif[DICHOT, CONNEX]]]

Out[12]= subclass[DICHOT, CONNEX] == True

In[13]:= subclass[DICHOT, CONNEX] := True
```

sum class

Lemma.

```
In[14]:= Map[empty, dif[CONNEX, P[cart[V, V]]] // Normality]

Out[14]= subclass[U[CONNEX], cart[V, V]] == True

In[15]:= % /. Equal→ SetDelayed
```

Lemma.

```
In[17]:= SubstTest[implies, subclass[u, v],
           subclass[U[u], U[v]], {u → DICHOT, v → CONNEX}] // Reverse

Out[17]= subclass[cart[V, V], U[CONNEX]] == True

In[18]:= % /. Equal→ SetDelayed
```

Theorem. Formula for the sum class of the class of connected relations.

```
In[20]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → U[CONNEX], v → cart[V, V]}]

Out[20]= equal[cart[V, V], U[CONNEX]] == True

In[22]:= U[CONNEX] := cart[V, V]
```

reflexive connected relations

In this section it is shown that a relation is dichotomous if and only if it is reflexive and connected.

Lemma.

```
In[23]:= SubstTest[cliques, intersection[u, v],
          {u → union[Id, x, inverse[x]], v → cartsq[fix[x]]}]

Out[23]= intersection[chains[union[Id, x]], P[fix[x]]] == chains[x]
```

```
In[24]:= intersection[chains[union[Id, x_]], P[fix[x_]]] := chains[x]
```

Corollary.

```
In[25]:= SubstTest[subclass, P[fix[x]], intersection[u, v],
          {u → chains[union[Id, x]], v → P[fix[x]]}]

Out[25]= subclass[cart[fix[x], fix[x]], union[Id, x, inverse[x]]] ==
          subclass[cart[fix[x], fix[x]], union[x, inverse[x]]]
```

```
In[26]:= subclass[cart[fix[x_], fix[x_]], union[Id, x_, inverse[x_]]] :=
          subclass[cart[fix[x], fix[x]], union[x, inverse[x]]]
```

Because of a special rewrite rule for **fix[rfx[x]]**, a separate corollary is needed when the **rfx** wrapper is used.

Corollary. A temporary rewrite rule.

```
In[27]:= SubstTest[subclass, cart[fix[t], fix[t]],
           union[Id, t, inverse[t]], t → rfx[x]] // Reverse

Out[27]= subclass[cart[fix[x], fix[x]], union[Id, inverse[rfx[x]], rfx[x]]] ==
          subclass[cart[fix[x], fix[x]], union[inverse[rfx[x]], rfx[x]]]
```

```
In[28]:= subclass[cart[fix[x_], fix[x_]], union[Id, inverse[rfx[x_]], rfx[x_]]] :=
          subclass[cart[fix[x], fix[x]], union[inverse[rfx[x]], rfx[x]]]
```

Eliminating the **rfx** wrapper yields the following lemma.

Lemma.

```
In[29]:= SubstTest[implies, and[equal[x, rfx[t]], subclass[cart[union[domain[x], range[x]],
              union[domain[x], range[x]]], union[Id, x, inverse[x]]]],
           subclass[cart[fix[x], fix[x]], union[x, inverse[x]]], t → x] // Reverse

Out[29]= or[not[REFLEXIVE[x]],
           not[subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
             union[Id, x, inverse[x]]]],
           subclass[cart[fix[x], fix[x]], union[x, inverse[x]]]] == True
```

```
In[30]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. A reflexive connected relation is dichotomous.

```
In[31]:= SubstTest[and, implies[p, subclass[u, v]], implies[p, subclass[v, u]],
          {p → and[REFLEXIVE[x], subclass[cart[union[domain[x], range[x]],
               union[domain[x], range[x]]], union[Id, x, inverse[x]]]],
           u -> cart[fix[x], fix[x]], v -> union[x, inverse[x]]}]

Out[31]= or[equal[cart[fix[x], fix[x]], union[x, inverse[x]]], not[REFLEXIVE[x]],
           not[subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
             union[Id, x, inverse[x]]]]] == True

In[32]:= or[equal[cart[fix[x_], fix[x_]], union[inverse[x_], x_]], not[REFLEXIVE[x_]],
           not[subclass[cart[union[domain[x_], range[x_]], union[domain[x_], range[x_]]],
             union[Id, inverse[x_], x_]]]] := True
```

Corollary. A concise variable-free restatement of the theorem (and its converse).

```
In[33]:= equal[intersection[RFX, CONNEX], DICHOT] // AssertTest

Out[33]= equal[DICHOT, intersection[CONNEX, RFX]] == True

In[34]:= intersection[CONNEX, RFX] := DICHOT
```

from dichotomous to connected and back

In this section it is shown how to obtain a connected relation from a dichotomous one and vice versa.

Lemma.

```
In[35]:= SubstTest[implies, equal[cartsq[udora[x]], y],
           subclass[cartsq[udora[intersection[Di, x]]], y], y → union[x, inverse[x]]] // Reverse

Out[35]= or[not[equal[cart[fix[x], fix[x]], union[x, inverse[x]]]],
           subclass[cart[union[fix[composite[Di, x]], fix[composite[x, Di]]],
             union[fix[composite[Di, x]], fix[composite[x, Di]]]], union[x, inverse[x]]] == True

In[36]:= (% /. x → x_) /. Equal→ SetDelayed
```

Theorem. If **x** is a dichotomous relation, then **intersection[Di, x]** is connected.

```
In[37]:= Map[not, SubstTest[and, implies[p1, p2], implies[p2, p3],
            not[implies[p1, p3]], {p1 -> equal[cart[fix[x], fix[x]], union[x, inverse[x]]],
             p2 -> subclass[cart[union[fix[composite[Di, x]], fix[composite[x, Di]]],
                union[fix[composite[Di, x]], fix[composite[x, Di]]]], union[x, inverse[x]]],
             p3 -> subclass[cart[union[fix[composite[Di, x]], fix[composite[x, Di]]],
                union[fix[composite[Di, x]], fix[composite[x, Di]]]],
               union[Id, x, inverse[x]]]}]] // Reverse

Out[37]= or[not[equal[cart[fix[x], fix[x]], union[x, inverse[x]]]],
           subclass[cart[union[fix[composite[Di, x]], fix[composite[x, Di]]], union[
              fix[composite[Di, x]], fix[composite[x, Di]]]], union[Id, x, inverse[x]]]] == True
```

```
In[39]:= or[not[equal[cart[fix[x_], fix[x_]], union[inverse[x_], x_]]],
           subclass[cart[union[fix[composite[Di, x_]], fix[composite[x_, Di]]],
             union[fix[composite[Di, x_]], fix[composite[x_, Di]]]],
            union[Id, inverse[x_], x_]]] := True
```

The following variable-free restatement is much easier to read.

Theorem. The strict part of a dichotomous relation is connected.

```
In[40]:= Map[equal[V, #] &, SubstTest[class, x, not[member[x, t]],
           t -> dif[DICHOT, image[inverse[IMAGE[id[Di]]], CONNEX]]]]
Out[40]= subclass[image[IMAGE[id[Di]], DICHOT], CONNEX] == True

In[41]:= subclass[image[IMAGE[id[Di]], DICHOT], CONNEX] := True
```

One can go from a connected relation to a dichotomous relation by forming the reflexive closure.

Lemma. Simplification rule.

```
In[42]:= equal[image[HULL[RFX], DICHOT], DICHOT]
Out[42]= True

In[43]:= image[HULL[RFX], DICHOT] := DICHOT
```

Theorem. An inclusion in one direction.

```
In[44]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
           {t → HULL[RFX], u → DICHOT, v → CONNEX}] // Reverse
Out[44]= subclass[DICHOT, image[HULL[RFX], CONNEX]] == True

In[45]:= % /. Equal → SetDelayed
```

To obtain an inclusion in the opposite direction, it is useful to temporarily introduce a variable.

Lemma.

```
In[46]:= SubstTest[implies, subclass[u, v],
           subclass[image[t, u], image[t, v]], {t → id[cartsq[udora[x]]],
            u -> cart[union[domain[x], range[x]], union[domain[x], range[x]]],
            v -> union[Id, x, inverse[x]]}] // Reverse
Out[46]= or[not[subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
             union[Id, x, inverse[x]]]],
           subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
            union[composite[Id, x], id[union[domain[x], range[x]], inverse[x]]]] == True

In[47]:= (% /. x → x_) /. Equal → SetDelayed
```

The expression **composite[Id, x]** in the above lemma can be viewed as a wrapper for a relation. Eliminating this wrapper yields the next lemma.

Lemma.

```
In[48]:= SubstTest[implies, equal[x, composite[Id, t]],
           implies[subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
             union[Id, x, inverse[x]]],
            subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
             union[x, id[union[domain[x], range[x]]], inverse[x]]]], t → x] // Reverse

Out[48]= or[not[subclass[x, cart[V, V]]],
           not[subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
             union[Id, x, inverse[x]]]],
           subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
            union[x, id[union[domain[x], range[x]]], inverse[x]]]] == True

In[49]:= (% /. x → x_) /. Equal → SetDelayed
```

The next lemma replaces an inclusion in the preceding lemma with an equation.

Lemma. The reflexive closure of a connected relation is dichotomous.

```
In[50]:= SubstTest[and, implies[p, subclass[u, v]],
          implies[p, subclass[v, u]], {p -> member[x, CONNEX],
           u -> cart[union[domain[x], range[x]], union[domain[x], range[x]]],
           v -> union[x, id[union[domain[x], range[x]]], inverse[x]]}]

Out[50]= or[equal[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
            union[x, id[union[domain[x], range[x]]], inverse[x]]],
           not[member[x, V]], not[subclass[x, cart[V, V]]],
           not[subclass[cart[union[domain[x], range[x]], union[domain[x], range[x]]],
             union[Id, x, inverse[x]]]]] == True

In[51]:= (% /. x → x_) /. Equal → SetDelayed
```

Restatement.

```
In[52]:= implies[member[x, CONNEX], member[union[x, id[udora[x]]], DICHOT]]

Out[52]= True
```

To eliminate the variable **x** it is convenient to introduce a **setpart** wrapper.

Lemma. Introducing a **setpart** wrapper.

```
In[53]:= SubstTest[implies, member[t, CONNEX],
           member[union[t, id[udora[t]]], DICHOT], t → setpart[x]] // Reverse

Out[53]= or[equal[cart[union[domain[setpart[x]], range[setpart[x]]],
             union[domain[setpart[x]], range[setpart[x]]]],
            union[id[union[domain[setpart[x]], range[setpart[x]]],
             inverse[setpart[x]], setpart[x]]],
           not[subclass[cart[union[domain[setpart[x]], range[setpart[x]]],
              union[domain[setpart[x]], range[setpart[x]]]],
            union[Id, inverse[setpart[x]], setpart[x]]]],
           not[subclass[setpart[x], cart[V, V]]]] == True

In[54]:= (% /. x → x_) /. Equal→ SetDelayed
```

Lemma. A simplification rule.

```
In[55]:= ImageComp[HULL[image[inverse[IMAGE[id[cart[V, V]]]], RFX]],
           id[P[cart[V, V]]], CONNEX] // Reverse

Out[55]= image[HULL[image[inverse[IMAGE[id[cart[V, V]]]], RFX]], CONNEX] ==
          image[HULL[RFX], CONNEX]

In[56]:= % /. Equal→ SetDelayed
```

Lemma. Elimination of the variable **x**.

```
In[57]:= Map[equal[V, #] &,
          SubstTest[class, x, implies[member[setpart[x], u], member[setpart[x], v]], {u → CONNEX,
            v → image[inverse[HULL[image[inverse[IMAGE[id[cart[V, V]]]], RFX]]], DICHOT]}]]

Out[57]= subclass[image[HULL[RFX], CONNEX], DICHOT] == True

In[58]:= % /. Equal→ SetDelayed
```

The final step is the sharpen the inclusion to an equation.

Theorem. Every dichotomous relation is the reflexive closure of a trichotomous relation.

```
In[59]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> image[HULL[RFX], CONNEX], v -> DICHOT}]

Out[59]= equal[DICHOT, image[HULL[RFX], CONNEX]] == True

In[60]:= image[HULL[RFX], CONNEX] := DICHOT
```

Corollary.

```
In[61]:= SubstTest[implies, subclass[u, v], subclass[image[u, w], image[v, w]],
           {u → id[CONNEX], v → composite[inverse[HULL[RFX]], HULL[RFX]], w → CONNEX}] // Reverse

Out[61]= subclass[CONNEX, image[inverse[HULL[RFX]], DICHOT]] == True

In[62]:= subclass[CONNEX, image[inverse[HULL[RFX]], DICHOT]] := True
```