CONNEX normalization rule

Johan G. F. Belinfante
2010 March 3

In[1]:=
```
SetDirectory["l:"]; << goedel.10mar03a;<< tools.m
```

```
:Package Title: goedel.10mar03a                2010 March 3 at 9:45 a.m.

It is now:  2010 Mar 3 at 12:58

Loading Simplification Rules

TOOLS.M                                        Revised 2010 February 26

weightlimit = 40
```

summary

A relation **r** is said to be **connected** on a class **s** if $\mathbf{r} \subset \mathbf{s} \times \mathbf{s} \subset \mathbf{r} \bigcup \mathbf{Id} \bigcup \mathbf{inverse[r]}$. (See, for example, page 11 of the following reference.)

In[2]:=
```
"Keith Devlin, The Joy of Sets: Fundamentals of Contemporary
  Set Theory, 2nd edition, Springer-Verlag, New York, 1993.";
```

In this notebook a variable-free rewrite rule is derived that says that a (small) relation is connected on some set if and only if it is connected on the union of its domain and range.

temporary abbreviation

It is useful to introduce the following abbreviation for the statement that **r** is connected on **s**.

In[3]:=
```
connected[r_, s_] :=
  and[subclass[r, cart[s, s]], subclass[cart[s, s], union[Id, r, inverse[r]]]]
```

The class of relations that are connected on some set is given by the following expression.

In[4]:=
```
class[r, exists[s, connected[r, s]]] // InvertFix
```

Out[4]=
```
fix[composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], id[image[CART, Id]], S]]
```

It will be shown that this class is equal to the class **CONNEX** of all relations that are connected on the union of their domain and range. The union of the domain and range of a relation is often called the field of the relation, but to avoid confusion with other meanings of the word 'field' the abbreviation **udora** is used in the **GOEDEL** program.

```
In[5]:= udora[r]

Out[5]= union[domain[r], range[r]]
```

normalization

Theorem. A rewrite rule that normalizes the class **CONNEX**.

```
In[6]:= CONNEX // Normality // Reverse

Out[6]= intersection[
          fix[composite[inverse[HULL[SYM]], intersection[composite[S, IMAGE[id[Di]], CART],
             composite[S, IMAGE[id[Di]], CART, SWAP]], DORA]],
          fix[composite[inverse[HULL[invar[SWAP]]], S, IMAGE[id[Di]], CART, DUP, IMAGE[FIRST]]],
          fix[composite[inverse[HULL[invar[SWAP]]], S,
            IMAGE[id[Di]], CART, DUP, IMAGE[SECOND]]]] == CONNEX

In[7]:= intersection[fix[composite[inverse[HULL[SYM]], intersection[
             composite[S, IMAGE[id[Di]], CART], composite[S, IMAGE[id[Di]], CART, SWAP]], DORA]],
          fix[composite[inverse[HULL[invar[SWAP]]], S, IMAGE[id[Di]], CART, DUP, IMAGE[FIRST]]],
          fix[composite[inverse[HULL[invar[SWAP]]], S,
            IMAGE[id[Di]], CART, DUP, IMAGE[SECOND]]]] := CONNEX
```

implication in one direction

Lemma.

```
In[8]:= SubstTest[implies, subclass[u, v], subclass[image[t, u], image[t, v]],
          {t -> intersection[composite[x, SECOND], composite[inverse[y], FIRST]],
           u → Id, v → S}] // Reverse

Out[8]= subclass[fix[composite[x, y]], fix[composite[x, S, y]]] == True

In[9]:= subclass[fix[composite[x_, y_]], fix[composite[x_, S, y_]]] := True
```

Lemma.

```
In[10]:= SubstTest[subclass, fix[composite[u, v]], fix[composite[u, S, v]],
           {u -> composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], CART, DUP],
            v → composite[CUP, DORA]}] // Reverse

Out[10]= subclass[CONNEX, fix[composite[inverse[HULL[SYM]], S,
             IMAGE[id[Di]], id[image[CART, Id]], S, IMAGE[id[cart[V, V]]]]]]] == True

In[11]:= % /. Equal → SetDelayed
```

Theorem,

```
In[13]:= SubstTest[subclass, t, intersection[u, v],
           {t → CONNEX, u → P[cart[V, V]], v -> fix[composite[inverse[HULL[SYM]], S,
              IMAGE[id[Di]], id[image[CART, Id]], S, IMAGE[id[cart[V, V]]]]]]}] // Reverse

Out[13]= subclass[CONNEX,
           fix[composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], id[image[CART, Id]], S]]] == True

In[14]:= % /. Equal → SetDelayed
```

the reverse direction

The inclusion **s × s ⊂ r ⋃ Id ⋃ inverse[r]** implies that either **s = domain[r] ⋃ range[r]** or **s** is a singleton. The latter possibility must be dealt with separately.

Lemma.

```
In[15]:= SubstTest[implies, subclass[x, set[t]],
           or[empty[x], equal[x, set[t]]], t → PAIR[y, y]] // Reverse

Out[15]= or[equal[0, x], equal[x, cart[set[y], set[y]]],
           not[subclass[x, cart[set[y], set[y]]]]] == True

In[16]:= or[equal[0, x_], equal[x_, cart[set[y_], set[y_]]],
           not[subclass[x_, cart[set[y_], set[y_]]]]] := True
```

Observation.

```
In[17]:= equal[s, set[U[s]]]

Out[17]= member[s, range[SINGLETON]]
```

Corollary.

```
In[18]:= SubstTest[implies, and[subclass[r, cart[s, s]], equal[s, set[t]]],
           or[empty[r], equal[r, id[set[t]]]], t → U[s]] // Reverse

Out[18]= or[equal[0, r], equal[r, cart[set[U[s]], set[U[s]]]],
           not[member[s, range[SINGLETON]]], not[subclass[r, cart[s, s]]]] == True

In[19]:= (% /. {r → r_, s → s_}) /. Equal → SetDelayed
```

Theorem. If **r** is connected on **s**, then **r** is connected on **udora[r]**.

```
In[20]:= Map[not, SubstTest[and, implies[p1, or[p2, p3]],
           implies[and[p1, p3], p7], implies[and[p1, p2], or[p5, p6]],
           implies[p5, p7], implies[p6, p7], not[implies[p1, p7]],
           {p1 → connected[r, s], p2 → member[s, range[SINGLETON]], p3 → equal[s, udora[r]],
            p5 → empty[r], p6 → equal[r, id[set[U[s]]]], p7 → connected[r, udora[r]]}]] // Reverse

Out[20]= or[not[subclass[r, cart[s, s]]], not[subclass[cart[s, s], union[Id, r, inverse[r]]]],
          subclass[cart[union[domain[r], range[r]], union[domain[r], range[r]]],
           union[Id, r, inverse[r]]]] == True
```

```
In[21]:= or[not[subclass[cart[s_, s_], union[Id, inverse[r_], r_]]],
          not[subclass[r_, cart[s_, s_]]], subclass[cart[union[domain[r_], range[r_]],
            union[domain[r_], range[r_]]], union[Id, inverse[r_], r_]]] := True
```

The following lemma is needed to help with the elimination of variables.

Lemma.

```
In[22]:= implies[and[member[r, V], member[s, V], connected[r, s]], member[r, CONNEX]] //
          NotNotTest

Out[22]= or[and[subclass[r, cart[V, V]],
           subclass[cart[union[domain[r], range[r]], union[domain[r], range[r]]],
            union[Id, r, inverse[r]]]], not[member[r, V]],
          not[member[s, V]], not[subclass[r, cart[s, s]]],
          not[subclass[cart[s, s], union[Id, r, inverse[r]]]]] == True
```

```
In[23]:= (% /. {r → r_, s → s_}) /. Equal → SetDelayed
```

Theorem. Removing the variables **r** and **s** yields an inclusion in the reverse direction.

```
In[24]:= Map[empty[composite[Id, complement[#]]] &,
          SubstTest[class, pair[r, s], implies[and[member[r, V], member[s, V], connected[r, s]],
            member[r, t]], t → CONNEX]] // InvertFix

Out[24]= subclass[fix[composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], id[image[CART, Id]], S]],
          CONNEX] == True
```

```
In[25]:= % /. Equal → SetDelayed
```

The final step is to combine the two inclusions into an equation and make it into a rewrite rule.

Main Theorem. A relation is connected on some set if and only if it is connected on the union of its domain and range.

```
In[26]:= SubstTest[and, subclass[u, v], subclass[v, u], {u → CONNEX,
          v -> fix[composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], id[image[CART, Id]], S]]}]

Out[26]= equal[CONNEX,
          fix[composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], id[image[CART, Id]], S]]] == True
```

```
In[27]:= fix[composite[inverse[HULL[SYM]], S, IMAGE[id[Di]], id[image[CART, Id]], S]] := CONNEX
```