transitive closure of a reflexive relation

Johan G. F. Belinfante and Ming Li
2006 October 7

```
In[1]:= SetDirectory["l:"]; << goedel86.07a; << tools.m

        :Package Title: goedel86.07a          2006 October 7 at 4:45 a.m.

        It is now:  2006 Oct 7 at 12:32

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 October 7

        weightlimit = 40
```

summary

The transitive closure of a reflexive relation is reflexive.

a general result

The transitivity of inclusion implies this general result:

```
In[2]:= SubstTest[implies, and[subclass[x, y], subclass[y, z]], subclass[x, z],
          {y → cart[s, t], z → cart[u, v]}] // MapNotNot

Out[2]= or[not[subclass[s, u]], not[subclass[t, v]],
          not[subclass[x, cart[s, t]]], subclass[x, cart[u, v]]] == True

In[3]:= or[not[subclass[s_, u_]], not[subclass[t_, v_]],
          not[subclass[x_, cart[s_, t_]]], subclass[x_, cart[u_, v_]]] := True
```

To show that a relation is reflexive, it suffices to show that its domain, range and fixed-point class are equal.

```
In[4]:= SubstTest[implies,
         and[equal[u, w], equal[v, w], subclass[x, cart[u, v]]], subclass[x, cart[w, w]],
         {u → domain[x], v -> range[x], w → fix[x]}]

Out[4]= or[not[equal[domain[x], fix[x]]], not[equal[fix[x], range[x]]],
          not[subclass[x, cart[V, V]]], REFLEXIVE[x]] == True

In[5]:= or[not[equal[domain[x_], fix[x_]]], not[equal[fix[x_], range[x_]]],
          not[subclass[x_, cart[V, V]]], REFLEXIVE[x_]] := True
```

transitive closures of reflexive relations

Lemma.

```
In[6]:= SubstTest[subclass, fix[y], fix[trv[y]], y → rfx[x]]

Out[6]= subclass[fix[x], fix[trv[rfx[x]]]] == True

In[7]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem.

```
In[8]:= SubstTest[and, subclass[domain[y], fix[y]],
          subclass[range[y], fix[y]], y → trv[rfx[x]]] // Reverse

Out[8]= REFLEXIVE[trv[rfx[x]]] == True

In[9]:= REFLEXIVE[trv[rfx[x_]]] := True

In[10]:= (% /. x → x_) /. Equal → SetDelayed
```

Corollary. The transitive closure of a reflexive relation is reflexive.

```
In[11]:= SubstTest[implies, equal[x, rfx[y]], REFLEXIVE[trv[x]], y → x]

Out[11]= or[not[REFLEXIVE[x]], REFLEXIVE[trv[x]]] == True

In[12]:= or[not[REFLEXIVE[x_]], REFLEXIVE[trv[x_]]] := True
```

variable-free formulation

Using **reify**, one obtains this variable-free statement:

```
In[13]:= Map[VERTSECT, SubstTest[reify, x, rfx[trv[f[x]]], f → rfx]] // Reverse

Out[13]= composite[CORE[RFX], HULL[TRV], CORE[RFX]] == composite[HULL[TRV], CORE[RFX]]

In[14]:= composite[CORE[RFX], HULL[TRV], CORE[RFX]] := composite[HULL[TRV], CORE[RFX]]
```

Corollary. The class **RFX** of (small) reflexive relations is invariant under the transitive closure operation **HULL[TRV]**.

```
In[15]:= Map[subclass[#, RFX] &, ImageComp[composite[CORE[RFX], HULL[TRV]], CORE[RFX], V]]

Out[15]= subclass[image[HULL[TRV], RFX], RFX] == True

In[16]:= subclass[image[HULL[TRV], RFX], RFX] := True
```

a fix formula

The inclusion for **fix[trv[rfx[x]]]** derived earlier can be sharpened to an equation:

```
In[17]:= SubstTest[domain, rfx[y], y → trv[rfx[x]]] // Reverse
Out[17]= fix[trv[rfx[x]]] == fix[x]

In[18]:= fix[trv[rfx[x_]]] := fix[x]
```

This result too can be restated in variable-free form using **reify**.

```
In[19]:= Map[VERTSECT, SubstTest[reify, x, fix[trv[f[x]]], f → rfx]] // Reverse
Out[19]= composite[IMAGE[inverse[DUP]], HULL[TRV], CORE[RFX]] == IMAGE[inverse[DUP]]

In[20]:= composite[IMAGE[inverse[DUP]], HULL[TRV], CORE[RFX]] := IMAGE[inverse[DUP]]
```