intersections with ACYCLIC

Johan G. F. Belinfante and Ming Li
2006 January 30

```
In[1]:= SetDirectory["l:"]; << goedel77.29a; << tools.m

        :Package Title: goedel77.29a             2006 January 29 at 11:20 p.m.

        It is now:  2006 Jan 30 at 16:46

        Loading Simplification Rules

        TOOLS.M                      Revised 2006 January 2

        weightlimit = 40
```

summary

Rewrite rules are derived for intersections of the class **ACYCLIC** of acyclic relations with other named classes. A relation is **acyclic** if its transitive closure has no fixed points.

```
In[2]:= class[x, and[subclass[x, cart[V, V]], equal[0, fix[trv[x]]]]]

Out[2]= ACYCLIC
```

wellfounded relations

A relation is **well-founded** if it does not permit infinite regress. The class of such relations is:

```
In[3]:= class[x, and[subclass[x, cart[V, V]],
          forall[y, implies[subclass[y, image[x, y]], equal[0, y]]]]]

Out[3]= WF
```

Any well founded relation must be acyclic. The **GOEDEL** program recognizes that the intersection of the classes **ACYCLIC** and **WF** is equal to **WF**, when asked, but it lacks a corresponding rewrite rule. One is therefore justified in adding a rewrite rule expressing this fact.

```
In[4]:= equal[intersection[WF, ACYCLIC], WF]

Out[4]= True

In[5]:= intersection[ACYCLIC, WF] := WF
```

using the associativity of intersection

A number of new rewrite rules can be derived using the **AssInt** tool available in the file **tools.m**. In particular, one can take advantage if the fact that the **GOEDEL** program already contains a rewrite rule for the intersection of **ACYCLIC** with the class **PO** of all partial order relations to deduce the following new rewrite rules:

```
In[6]:= AssInt[ACYCLIC, PO, CL]
Out[6]= intersection[ACYCLIC, CL] == 0

In[7]:= intersection[ACYCLIC, CL] := 0

In[8]:= AssInt[ACYCLIC, PO, FUNS]
Out[8]= intersection[ACYCLIC, P[Id]] == set[0]

In[9]:= intersection[ACYCLIC, P[Id]] := set[0]

In[10]:= AssInt[ACYCLIC, PO, TO]
Out[10]= intersection[ACYCLIC, TO] == set[0]

In[11]:= intersection[ACYCLIC, TO] := set[0]

In[12]:= AssInt[ACYCLIC, PO, WO]
Out[12]= intersection[ACYCLIC, WO] == set[0]

In[13]:= intersection[ACYCLIC, WO] := set[0]
```

Likewise, since a rewrite rule for the intersection of **P[Di]** and **ACYCLIC** is available, one can use this information to deduce new facts:

```
In[14]:= AssInt[RFX, P[Di], ACYCLIC]
Out[14]= intersection[ACYCLIC, RFX] == set[0]

In[15]:= intersection[ACYCLIC, RFX] := set[0]
```

intersection with EQV

Lemma.

```
In[16]:= AssInt[P[Di], RFX, EQV]
Out[16]= intersection[EQV, P[Di]] == set[0]

In[17]:= intersection[EQV, P[Di]] := set[0]
```

This yields a rule for the intersection of **ACYCLIC** with the class **EQV** of equivalence relations.

```
In[18]:= AssInt[EQV, P[Di], ACYCLIC]

Out[18]= intersection[ACYCLIC, EQV] == set[0]

In[19]:= intersection[ACYCLIC, EQV] := set[0]
```

intersection with ANTISYM

Lemma. Any acyclic relation has no cycles of length 2.

```
In[20]:= Map[implies[#, equal[0, fix[composite[x, x]]]] &, SubstTest[and,
           subclass[u, v], equal[0, v], {u → fix[composite[x, x]], v → fix[trv[x]]}]]

Out[20]= or[equal[0, fix[composite[x, x]]], not[equal[0, fix[trv[x]]]]] == True

In[21]:= or[equal[0, fix[composite[x_, x_]]], not[equal[0, fix[trv[x_]]]]] := True
```

The class of relations with no cycles of length two is the class of relations where **x** is disjoint fom its inverse:

```
In[22]:= class[x, and[subclass[x, cart[V, V]], equal[0, fix[composite[x, x]]]]]

Out[22]= fix[composite[DISJOINT, INVERSE]]
```

Corollary.

```
In[23]:= Map[equal[0, #] &, dif[ACYCLIC, fix[composite[DISJOINT, INVERSE]]] // Renormality]

Out[23]= subclass[ACYCLIC, fix[composite[DISJOINT, INVERSE]]] == True

In[24]:= subclass[ACYCLIC, fix[composite[DISJOINT, INVERSE]]] := True
```

The condition of antisymmetry is slightly weaker, so it follows that any acyclic relation is anstisymmetric:

```
In[25]:= SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
           {u → ACYCLIC, v -> fix[composite[DISJOINT, INVERSE]], w → ANTISYM}]

Out[25]= subclass[ACYCLIC, ANTISYM] == True

In[26]:= subclass[ACYCLIC, ANTISYM] := True
```

This implies an intersection rule:

```
In[27]:= equal[intersection[ACYCLIC, ANTISYM], ACYCLIC]

Out[27]= True

In[28]:= intersection[ACYCLIC, ANTISYM] := ACYCLIC
```

intersection with SYM

Theorem

```
In[29]:= AssInt[SYM, ANTISYM, ACYCLIC]

Out[29]= intersection[ACYCLIC, SYM] == set[0]

In[30]:= intersection[ACYCLIC, SYM] := set[0]
```