Monotonicity

Johan G. F. Belinfante
2001 November 5

```
<< goedel52.L08; << tests.m
```

```
:Package Title: GOEDEL52.L08      2001 October 30 at 4:50 p.m.

It is now:  2001 Nov 5 at 11:35

Loading Simplification Rules

TESTS.M                   Revised 2001 October 18

weightlimit = 30


Context switch to ‘Goedel‘Private is needed for ReplaceTest

Just ignore the error message about Unterminated use of BeginPackage

Get::bebal : Unterminated uses of BeginPackage or Begin in << tests.m.
```

■ Unary functors

```
UnaryFunctors
```

```
{A, Aclosure, ADJOIN, cliques, complement, CORE, domain, fix, flip,
 funpart, GLB, GREATEST, H, HULL, id, IMAGE, invar, inverse, LB, LEAST,
 LEFT, LUB, MAXIMAL, MINIMAL, P, range, rank, RC, RIGHT, rotate, singleton,
 subcommutant, subvar, succ, tc, twist, U, UB, Uclosure, VERTSECT}
```

Known antitone:

```
Map[implies[subclass[x, y], subclass[#[y], #[x]]] &, {A, complement, invar}] // TableForm
```

```
True
True
True
```

Known monotone:

```
Map[implies[subclass[x, y], subclass[#[x], #[y]]] &,
  {Aclosure, domain, fix, flip, id, P, range, U, Uclosure}] // TableForm
```

```
True
True
True
True
True
True
True
True
True
```

■ inverse

```
SubstTest[implies, subclass[x, y], subclass[image[w, x], image[w, y]], w -> SWAP]
```

```
or[not[subclass[x, y]], subclass[composite[Id, x], y]] == True
```

```
or[not[subclass[x_, y_]], subclass[composite[Id, x_], y_]] := True
```

■ rotate

Permanent rule:

```
SubstTest[implies, subclass[x, y], subclass[image[w, x], image[w, y]], w -> ROT]
```

```
or[not[subclass[x, y]], subclass[rotate[x], rotate[y]]] == True
```

```
or[not[subclass[x_, y_]], subclass[rotate[x_], rotate[y_]]] := True
```

■ twist

Permanent rule:

```
SubstTest[implies, subclass[x, y], subclass[image[w, x], image[w, y]], w -> TWIST]
```

```
or[not[subclass[x, y]], subclass[twist[x], twist[y]]] == True
```

```
or[not[subclass[x_, y_]], subclass[twist[x_], twist[y_]]] := True
```

■ What’s left?

Subsumed by new rules.

```
Map[implies[subclass[x, y], subclass[#[x], #[y]]] &, {LB, UB}]
 // TableForm
```

```
True
True
```

Monotonicity counterexamples:

```
((Map[implies[subclass[x, y], subclass[#[x], #[y]]] &,
    {ADJOIN, RC, singleton, subcommutant, VERTSECT}]) /. {x -> 0, y -> V})
 //
 TableForm
```

```
False
False
False
False
False
```

More counterexamples, obtained by adding an **assert**.

```
Map[assert[(implies[subclass[x, y], subclass[#[x], #[y]]] /. {x -> 0, y -> V})] &,
 {CORE, IMAGE, LEFT, MAXIMAL, MINIMAL, RIGHT}]
 // TableForm
```

```
False
False
False
False
False
False
```

A different counterexample:

```
Map[
 assert[(implies[subclass[x, y], subclass[#[x], #[y]]] /. {x -> Id, y -> V})] &, {funpart}]
 // TableForm
```

```
False
```

■ Attempt to get counterexample for HULL

```
subclass[HULL[FULL], HULL[V]]
```

```
subclass[TC, Id]
```

```
subclass[HULL[range[POWER]], HULL[V]]
```

```
subclass[composite[POWER, BIGCUP], Id]
```

■ Counterexample for succ.

```
subclass[succ[0], succ[singleton[singleton[0]]]]
```

```
False
```

■ LEAST

```
LEAST[intersection[x, y]]
```

```
intersection[LEAST[x], LEAST[y]]
```

```
GREATEST[intersection[x, y]]
```

```
intersection[GREATEST[x], GREATEST[y]]
```

```
intersection[inverse[E], LB[x]]
```

```
LEAST[x]
```

```
subclass[LEAST[x], LEAST[y]]
```

```
subclass[composite[x, id[fix[x]]], y]
```

```
SubstTest[implies, subclass[u, v], subclass[intersection[w, u], intersection[w, v]],
    {u -> LB[x], v -> LB[y], w -> inverse[E]}]
```

```
or[not[subclass[composite[Id, x], y]], subclass[composite[x, id[fix[x]]], y]] == True
```

```
or[not[subclass[composite[Id, x_], y_]], subclass[composite[x, id[fix[x_]]], y_]] := True
```

```
Map[not, SubstTest[and, implies[p, q], implies[q, r], not[implies[p, r]],
        {p -> subclass[x, y],
   q -> subclass[LB[x], LB[y]], r -> subclass[LEAST[x], LEAST[y]]}]]
```

```
or[not[subclass[x, y]], subclass[composite[x, id[fix[x]]], y]] == True
```

```
or[not[subclass[x_, y_]], subclass[composite[x_, id[fix[x_]]], y_]] := True
```

```
implies[subclass[x, y], subclass[LEAST[x], LEAST[y]]]
```

```
True
```

■ Three similar cases

```
U[image[CLIQUES, P[x]]]
```

```
cliques[x]
```

```
U[image[HC, P[x]]]
```

```
H[x]
```

```
U[image[TC, P[x]]]
```

```
tc[x]
```

```
SubstTest[implies, subclass[u, v], subclass[image[w, u], image[w, v]],
    {u -> P[x], v -> P[y]}]
```

```
or[not[subclass[x, y]], subclass[image[w, P[x]], image[w, P[y]]]] == True
```

```
or[not[subclass[x_, y_]], subclass[image[w_, P[x_]], image[w_, P[y_]]]] := True
```

```
Map[not, SubstTest[and, implies[p, q], implies[q, r], not[implies[p, r]],
        {p -> subclass[x, y], q -> subclass[image[w, P[x]], image[w, P[y]]],
            r -> subclass[U[image[w, P[x]]], U[image[w, P[y]]]]}]]
```

```
or[not[subclass[x, y]], subclass[U[image[w, P[x]]], U[image[w, P[y]]]] == True
```

```
or[not[subclass[x_, y_]], subclass[U[image[w_, P[x_]]], U[image[w_, P[y_]]]] := True
```

```
Map[SubstTest[implies, subclass[x, y], subclass[U[image[w, P[x]]], U[image[w, P[y]]]],
    w -> #] &, {CLIQUES, HC, TC}] // TableForm
```

```
or[not[subclass[x, y]], subclass[cliques[x], cliques[y]]] == True
or[not[subclass[x, y]], subclass[H[x], y]] == True
or[not[subclass[x, y]], subclass[tc[x], tc[y]]] == True
```

```
or[not[subclass[x_, y_]], subclass[cliques[x_], cliques[y_]]] := True
```

```
or[not[subclass[x_, y_]], subclass[H[x_], y_]] := True
```

```
or[not[subclass[x_, y_]], subclass[tc[x_], tc[y_]]] := True
```

```
equiv[subclass[x, tc[y]], subclass[tc[x], tc[y]]] // AssertTest
```

```
True
```

■ subvar

```
(Map[not, SubstTest[and, implies[p, q], implies[q, r], not[implies[p, r]],
    {p → subclass[x, y], q → subclass[composite[E, x], composite[E, y]],
     r → subclass[fix[composite[E, x]], fix[composite[E, y]]]}]] /.
 {x → UB[u], y → UB[v]}
```

```
or[not[subclass[composite[Id, u], v]],
  subclass[domain[GREATEST[u]], domain[GREATEST[v]]]] == True
```

```
or[not[subclass[composite[Id, u_], v_]],
  subclass[domain[GREATEST[u_]], domain[GREATEST[v_]]]] := True
```

```
(Map[not, SubstTest[and, implies[p, q], implies[q, r], not[implies[p, r]],
    {p → subclass[u, v], q -> subclass[composite[Id, u], v],
     r → subclass[domain[GREATEST[u]], domain[GREATEST[v]]]}]] /.
 {u → complement[y], v → complement[x]}
```

```
or[not[subclass[x, y]], subclass[subvar[x], subvar[y]]] == True
```

```
or[not[subclass[x_, y_]], subclass[subvar[x_], subvar[y_]]] := True
```

■ rank

We have proved using Otter that rank is monotonic.

■ Remaining cases: GLB and LUB

The cases of **GLB** and **LUB** need to be looked into further.

■ Binary functors

```
BinaryFunctors
```

```
{cart, composite, cross, image, intersection, map, PAIR, pairset, union}
```

```
implies[and[subclass[u, x], subclass[v, y]],
    subclass[cart[u, v], cart[x, y]]]
```

```
True
```

```
implies[and[subclass[u, x], subclass[v, y]],
    subclass[cross[u, v], cross[x, y]]]
```

```
or[not[subclass[u, x]], not[subclass[v, y]], subclass[cross[u, v], cross[x, y]]]
```

```
implies[and[subclass[u, x], subclass[v, y]],
    subclass[image[u, v], image[x, y]]]
```

```
or[not[subclass[u, x]], not[subclass[v, y]], subclass[image[u, v], image[x, y]]]
```

■ composite

Temporary rule:

```
SubstTest[implies, implies[p, q], implies[p, and[q, r]],
    {p -> and[subclass[u, x], subclass[v, y]],
        q -> subclass[composite[u, v], composite[x, v]],
        r -> subclass[composite[x, v], composite[x, y]]}]
```

```
or[and[subclass[composite[u, v], composite[x, v]],
   subclass[composite[x, v], composite[x, y]]],
  not[subclass[u, x]], not[subclass[v, y]]] == True
```

Temporary rule:

```
or[and[subclass[composite[u_, v_], composite[x_, v_]],
   subclass[composite[x_, v_], composite[x_, y_]]],
  not[subclass[u_, x_]], not[subclass[v_, y_]]] := True
```

```
implies[and[subclass[u, x], subclass[v, y]],
    and[subclass[composite[u, v], composite[x, v]],
        subclass[composite[x, v], composite[x, y]]]]

True
```

Permanent rule:

```
Map[not, SubstTest[and, implies[p, q], implies[q, r], not[implies[p, r]],
        {p -> and[subclass[u, x], subclass[v, y]],
            q -> and[subclass[composite[u, v], composite[x, v]],
        subclass[composite[x, v], composite[x, y]]],
            r -> subclass[composite[u, v], composite[x, y]]}]]

or[not[subclass[u, x]], not[subclass[v, y]],
  subclass[composite[u, v], composite[x, y]]] == True

or[not[subclass[u_, x_]], not[subclass[v_, y_]],
  subclass[composite[u_, v_], composite[x_, y_]]] := True
```

Success!

```
implies[and[subclass[u, x], subclass[v, y]],
    subclass[composite[u, v], composite[x, y]]]

True
```

■ cross

Temporary rule:

```
SubstTest[implies, subclass[s, t], subclass[twist[s], twist[t]],
    {s -> cart[u, v], t -> cart[x, y]}]

or[and[not[equal[0, u]], not[equal[0, v]], not[subclass[u, x]]],
  and[not[equal[0, u]], not[equal[0, v]], not[subclass[v, y]]],
  subclass[cross[u, v], cross[x, y]]] == True

or[and[not[equal[0, u_]], not[equal[0, v_]], not[subclass[u_, x_]]],
  and[not[equal[0, u_]], not[equal[0, v_]], not[subclass[v_, y_]]],
  subclass[cross[u_, v_], cross[x_, y_]]] := True
```

Permanent rule:

```
Map[not, SubstTest[and, implies[p, q], implies[q, r], not[implies[p, r]],
        {p -> and[subclass[u, x], subclass[v, y]], q -> subclass[cart[u, v], cart[x, y]],
            r -> subclass[cross[u, v], cross[x, y]]}]]

or[not[subclass[u, x]], not[subclass[v, y]], subclass[cross[u, v], cross[x, y]]] == True

or[not[subclass[u_, x_]], not[subclass[v_, y_]],
  subclass[cross[u_, v_], cross[x_, y_]]] := True
```

■ image

Temporary rule:

```
SubstTest[implies, and[subclass[u, x], subclass[s, t]],
 subclass[composite[u, s], composite[x, t]], {s -> id[v], t -> id[y]}]
```

```
or[and[subclass[composite[u, id[v]], x], subclass[intersection[v, domain[u]], y]],
  not[subclass[u, x]], not[subclass[v, y]]] == True
```

```
or[and[subclass[composite[u_, id[v_]], x_], subclass[intersection[v_, domain[u_]], y_]],
  not[subclass[u_, x_]], not[subclass[v_, y_]]] := True
```

Temporary rule:

```
SubstTest[implies, subclass[s, t], subclass[range[s], range[t]],
    {s -> composite[u, id[v]], t -> composite[x, id[y]]}]
```

```
or[not[subclass[composite[u, id[v]], x]], not[subclass[intersection[v, domain[u]], y]],
  subclass[image[u, v], image[x, y]]] == True
```

```
or[not[subclass[composite[u_, id[v_]], x_]],
  not[subclass[intersection[v_, domain[u_]], y_]],
  subclass[image[u_, v_], image[x_, y_]]] := True
```

Permanent:

```
Map[not, SubstTest[and, implies[p, q], implies[q, r], not[implies[p, r]],
        {p -> and[subclass[u, x], subclass[v, y]],
   q -> subclass[composite[u, id[v]], composite[x, id[y]]],
           r -> subclass[image[u, v], image[x, y]]}]]
```

```
or[not[subclass[u, x]], not[subclass[v, y]], subclass[image[u, v], image[x, y]]] == True
```

```
or[not[subclass[u_, x_]], not[subclass[v_, y_]],
  subclass[image[u_, v_], image[x_, y_]]] := True
```

■ intersection

Permanent rule:

```
SubstTest[implies, implies[p, q], implies[p, and[q, r]],
    {p -> and[subclass[u, x], subclass[v, y]], q -> subclass[intersection[u, v], x],
        r -> subclass[intersection[u, v], y]}]
```

```
or[and[subclass[intersection[u, v], x], subclass[intersection[u, v], y]],
  not[subclass[u, x]], not[subclass[v, y]]] == True
```

```
or[and[subclass[intersection[u_, v_], x_], subclass[intersection[u_, v_], y_]],
  not[subclass[u_, x_]], not[subclass[v_, y_]]] := True
```

```
implies[and[subclass[u, x], subclass[v, y]],
    subclass[intersection[u, v], intersection[x, y]]]
```

```
True
```

■ union

Permanent rule:

```
SubstTest[implies, and[subclass[u, v], subclass[v, w]], subclass[u, w],
    {u -> x, v -> y, w -> union[y, z]}]

or[not[subclass[x, y]], subclass[x, union[y, z]]] == True

or[not[subclass[x_, y_]], subclass[x_, union[y_, z_]]] := True
```

Permanent rule:

```
SubstTest[implies, implies[p, q], implies[p, and[q, r]],
    {p -> and[subclass[u, x], subclass[v, y]], q -> subclass[u, union[x, y]],
        r -> subclass[v, union[x, y]]}]

or[and[subclass[u, union[x, y]], subclass[v, union[x, y]]],
  not[subclass[u, x]], not[subclass[v, y]]] == True

or[and[subclass[u_, union[x_, y_]], subclass[v_, union[x_, y_]]],
  not[subclass[u_, x_]], not[subclass[v_, y_]]] := True

implies[and[subclass[u, x], subclass[v, y]],
    subclass[union[u, v], union[x, y]]]

True
```