monotone transforms of cliques and chains

Johan G. F. Belinfante
2011 January 20

```
In[1]:= SetDirectory["l:"]; << goedel.11jan19a
```

```
:Package Title: goedel.11jan19a           2011 January 19 at 2:45 p.m.

It is now:  2011 Jan 20 at 18:7

Loading Simplification Rules

TOOLS.M is now incorporated in the GOEDEL program as of 2010 September 3

weightlimit = 40
```

summary

If **x** is monotone from **y** to **z**, then **x** transforms chains in **y** to chains in **z**. In particular, the range of certain monotone iterate expressions are chains with respect to inclusion.

derivation

Lemma.

```
In[2]:= SubstTest[implies, subclass[u, v],
          subclass[image[x, u], image[x, v]], {u → domain[x], v → y}] // Reverse
```

```
Out[2]= or[not[subclass[domain[x], y]], subclass[range[x], image[x, y]]] == True
```

```
In[3]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem.

```
In[4]:= Map[not, SubstTest[and, implies[p1, p3],
           implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
           {p1 → subclass[domain[x], y], p2 → subclass[image[x, y], z],
            p3 -> subclass[range[x], image[x, y]], p4 → subclass[range[x], z]}]] // Reverse
```

```
Out[4]= or[not[subclass[domain[x], y]],
          not[subclass[image[x, y], z]], subclass[range[x], z]] == True
```

```
In[5]:= or[not[subclass[domain[x_], y_]],
          not[subclass[image[x_, y_], z_]], subclass[range[x_], z_]] := True
```

Theorem. If **x** is monotone from **y** to **z**, then cliques in **y** are transformed by **x** to cliques in **z**.

```
In[6]:= SubstTest[implies, and[subclass[domain[t], y], subclass[image[t, y], z]],
          subclass[range[t], z], t → cross[x, x]] // Reverse

Out[6]= or[not[subclass[cart[domain[x], domain[x]], y]],
          not[subclass[composite[x, y, inverse[x]], z]],
          subclass[cart[range[x], range[x]], z]] == True

In[7]:= or[not[subclass[cart[domain[x_], domain[x_]], y_]],
          not[subclass[composite[x_, y_, inverse[x_]], z_]],
          subclass[cart[range[x_], range[x_]], z_]] := True
```

Lemma.

```
In[8]:= SubstTest[implies,
          and[subclass[domain[t], u], subclass[image[t, u], v]], subclass[range[t], v],
          {t → cross[x, x], u → union[y, inverse[y]], v → union[z, inverse[z]]}] // Reverse

Out[8]= or[not[subclass[cart[domain[x], domain[x]], union[y, inverse[y]]]],
          not[subclass[composite[x, y, inverse[x]], union[z, inverse[z]]]],
          not[subclass[composite[x, inverse[y], inverse[x]], union[z, inverse[z]]]],
          subclass[cart[range[x], range[x]], union[z, inverse[z]]]] == True

In[9]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Lemma.

```
In[10]:= SubstTest[implies, subclass[composite[x, u, inverse[x]], v],
           subclass[composite[x, u, inverse[x]], union[v, inverse[v]]],
           {u → inverse[y], v → inverse[z]}] // Reverse

Out[10]= or[not[subclass[composite[x, y, inverse[x]], z]],
           subclass[composite[x, inverse[y], inverse[x]], union[z, inverse[z]]]] == True

In[11]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Theorem. If **x** is monotone from **y** to **z**, then chains in **y** are transformed to chains in **z**.

```
In[12]:= Map[not, SubstTest[and, implies[and[p1, p3, p4], p5], implies[p2, p3], implies[p2, p4],
            not[implies[and[p1, p2], p5]], {p1 → subclass[P[domain[x]], chains[y]],
             p2 → subclass[composite[x, y, inverse[x]], z],
             p3 -> subclass[composite[x, y, inverse[x]], union[z, inverse[z]]],
             p4 -> subclass[composite[x, inverse[y], inverse[x]], union[z, inverse[z]]],
             p5 -> subclass[P[range[x]], chains[z]]}]] // Reverse

Out[12]= or[not[subclass[cart[domain[x], domain[x]], union[y, inverse[y]]]],
           not[subclass[composite[x, y, inverse[x]], z]],
           subclass[cart[range[x], range[x]], union[z, inverse[z]]]] == True

In[13]:= or[not[subclass[cart[domain[x_], domain[x_]], union[inverse[y_], y_]]],
           not[subclass[composite[x_, y_, inverse[x_]], z_]],
           subclass[cart[range[x_], range[x_]], union[inverse[z_], z_]]] := True
```

eliminating a variable

In this section more transparent versions of the results of the preceding section are derived by eliminating one of the three variables.

Theorem. Monotone relations preserve cliques.

```
In[14]:= Map[equal[V, #] &, SubstTest[class, t, implies[member[t, u], member[t, v]],
           {u → intersection[monotone[x, y], image[inverse[IMAGE[FIRST]], cliques[x]]],
            v → image[inverse[IMAGE[SECOND]], cliques[y]]}]]

Out[14]= subclass[image[image[DORA, monotone[x, y]], cliques[x]], cliques[y]] == True

In[15]:= subclass[image[image[DORA, monotone[x_, y_]], cliques[x_]], cliques[y_]] := True
```

Theorem. Monotone relations preserve chains.

```
In[16]:= Map[equal[V, #] &, SubstTest[class, t, implies[member[t, u], member[t, v]],
           {u → intersection[monotone[x, y], image[inverse[IMAGE[FIRST]], chains[x]]],
            v → image[inverse[IMAGE[SECOND]], chains[y]]}]]

Out[16]= subclass[image[image[DORA, monotone[x, y]], chains[x]], chains[y]] == True

In[17]:= subclass[image[image[DORA, monotone[x_, y_]], chains[x_]], chains[y_]] := True
```

application to iterate

Lemma. The domain of **iterate[x, y]** is a chain with respect to inclusion.

```
In[18]:= SubstTest[implies, member[t, OMEGA],
           member[t, chains[S]], t → domain[iterate[x, y]]] // Reverse

Out[18]= subclass[cart[domain[iterate[x, y]], domain[iterate[x, y]]],
           union[S, inverse[S]]] == True

In[19]:= subclass[cart[domain[iterate[x_, y_]], domain[iterate[x_, y_]]],
           union[S, inverse[S]]] := True
```

Theorem. If **P[iterate[x, y]] ⊂ monotone[S, S]**, then **P[range[iterate[x,y]]] ⊂ chains[S]**.

```
In[20]:= SubstTest[implies, and[subclass[cart[domain[t], domain[t]], union[S, inverse[S]]],
            subclass[composite[t, S, inverse[t]], S]],
           subclass[cart[range[t], range[t]], union[S, inverse[S]]], t → iterate[x, y]] // Reverse

Out[20]= or[not[subclass[composite[iterate[x, y], S, inverse[iterate[x, y]]], S]], subclass[
            cart[range[iterate[x, y]], range[iterate[x, y]]], union[S, inverse[S]]]] == True

In[21]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Corollary.

```
In[22]:= SubstTest[implies, subclass[P[iterate[u, v]], monotone[S, S]],
           subclass[P[range[iterate[u, v]]], chains[S]], {u → funpart[x], v → set[y]}] // Reverse

Out[22]= or[not[subclass[composite[
              iterate[funpart[x], set[y]], S, inverse[iterate[funpart[x], set[y]]]], S]],
           subclass[cart[hull[invar[funpart[x]], set[y]], hull[invar[funpart[x]], set[y]]],
            union[S, inverse[S]]]] == True

In[23]:= or[not[subclass[composite[iterate[funpart[x_], set[y_]],
              S, inverse[iterate[funpart[x_], set[y_]]]], S]],
           subclass[cart[hull[invar[funpart[x_]], set[y_]], hull[invar[funpart[x_]], set[y_]]],
            union[S, inverse[S]]]] := True
```

Theorem. A special case.

```
In[24]:= SubstTest[subclass,
           P[hull[invar[composite[CUP, id[funpart[t]], inverse[FIRST]]], set[y]]],
           chains[S], t -> composite[IMAGE[x], CART, DUP]] // Reverse

Out[24]= subclass[cart[hull[
             invar[composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]]], set[y]],
            hull[invar[composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]]],
             set[y]]], union[S, inverse[S]]] == True

In[25]:= subclass[cart[hull[invar[
              composite[CUP, id[composite[IMAGE[x_], CART, DUP]], inverse[FIRST]]], set[y_]],
            hull[invar[composite[CUP, id[composite[IMAGE[x_], CART, DUP]], inverse[FIRST]]],
             set[y_]]], union[S, inverse[S]]] := True
```