two-sided restrictions to final segments

Johan G. F. Belinfante
2010 July 15

In[1]:=
```
SetDirectory["l:"]; << goedel.10jul13a; << tools.m
```

```
:Package Title: goedel.10jul13a              2010 July 13 at 12:20 p.m.

It is now:  2010 Jul 15 at 14:45

Loading Simplification Rules

TOOLS.M                                      Revised 2010 February 26

weightlimit = 40
```

summary

In the recently posted notebook **trv-fnsg.nb** a variable-free statement about transitivity for final segments was derived. In the present notebook another variable-free statement concerning the transitivity of final segments is derived in which the predicate **TRANSITIVE** occurs explicitly.

acknowledgment

Theorem 8.10 on page 32 in the following reference inspired this work.

In[2]:=
```
"Egbert Harzheim, Ordered Sets, Advances in Mathematics, volume 7, Springer
  Science+Business Media, Inc., 2005.  ISBN 0387-24219-8.  QA171.48 .H37";
```


temporary abbreviations

Two-sided restriction can be viewed as an action of sets on relations. If **s** is a set and **r** is a relation, then this action of **s** on **r** could be denoted by **s · r = restrict[r, s, s]**. Actually one can allow **r** to be any set. There is no need to require **r** to be a relation. The following temporary abbreviation for the binary function for this action will be used.

In[3]:=
```
TWOSIDED := composite[CAP, cross[composite[CART, DUP], Id]]
```

The **APPLY** rule for this action simplifies when one uses **setpart** wrappers:

In[4]:=
```
APPLY[TWOSIDED, PAIR[setpart[s], setpart[r]]]
```

Out[4]=
```
composite[id[setpart[s]], setpart[r], id[setpart[s]]]
```

Harzheim defines a set **s** to be a **final segment** of a relation **r** if **image[r, s] = s**. The following temporary abbreviation for the relation of being a final segment will be used here.

```
In[5]:= FINSEG := fix[composite[inverse[SECOND], IMG]]
```

The following membership rule for this relation helps explain its significance.

```
In[6]:= member[pair[setpart[r], setpart[s]], FINSEG]

Out[6]= equal[image[setpart[r], setpart[s]], setpart[s]]
```

This notebook concerns only the restriction of the two-sided restriction action to final segments:

```
In[7]:= FNRS := composite[TWOSIDED, id[inverse[FINSEG]]]
```

It will be shown below that the relation **FNRS** ∘ **inverse[SECOND]** is transitive. An ordered pair of relations **pair[x, y]** belongs to this transitive relation if **y** is the two-sided restriction of **x** to some final segment of **x**.

---

a quasi-associative law

The **TWOSIDED** action satisfies the following quasi-associative law: **u · (v · x) = (u ⋂ v) · x**. This quasi-associative law simplifies for the restricted action **FNRS** in that **u ⋂ v** can be replaced with **u**. Explicitly, if one writes **r ▪ s** for this restriction of the action **r · s**, the quasi-associative law effectively becomes the inclusion **u ▪ (v ▪ x) ⊂ u ▪ x**. (Note that one must replace equality by inclusion here because the restricted action is no longer total. The following lemma formally expresses this simplification of the quasi-associative law for the action **FNRS** in the form of an implication involving two additional variables: **y = v ▪ x & z = u ▪ y ⟹ z = u ▪ x**.

Lemma. A quasi-associative law for the action **FNRS** with five variables wrapped with **setpart**.

```
In[8]:= implies[and[member[pair[pair[setpart[u], setpart[x]], setpart[y]], FNRS],
        member[pair[pair[setpart[v], setpart[y]], setpart[z]], FNRS]],
          member[pair[pair[setpart[v], setpart[x]], setpart[z]], FNRS]] // NotNotTest

Out[8]= or[and[equal[composite[id[setpart[v]], setpart[x], id[setpart[v]]], setpart[z]],
           equal[image[setpart[x], setpart[v]], setpart[v]]],
          not[equal[composite[id[setpart[u]], setpart[x], id[setpart[u]]], setpart[y]]],
          not[equal[composite[id[setpart[v]], setpart[y], id[setpart[v]]], setpart[z]]],
          not[equal[image[setpart[x], setpart[u]], setpart[u]]],
          not[equal[image[setpart[y], setpart[v]], setpart[v]]]] == True

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

The main problem encountered in eliminating the variables is that the complexity of the formula for **FNRS** causes various rewrite rules to produce complicated expressions. To shield **FNRS** from unwanted rewriting in the process of eliminating variables, yet another variable **t** and an equality literal will be introduced. Accordingly, the above simplified statement of the quasi-associative law will first be rewritten as follows:

```
In[10]:= implies[and[equal[t, FNRS], member[pair[pair[setpart[v], setpart[x]], setpart[y]], t],
           member[pair[pair[setpart[u], setpart[y]], setpart[z]], t]],
          member[pair[pair[setpart[u], setpart[x]], setpart[z]], t]]

Out[10]= True
```

The goal is to eliminate all six variables to derive a variable-free statement of transitivity.

---

eliminating x, y and z

Lemma. (Elimination of three variables.)

```
In[11]:= Map[empty[composite[complement[#], id[cart[V, V]]]] &,
          SubstTest[class, pair[pair[x, z], y],
           implies[and[equal[t, s], member[pair[pair[setpart[v], setpart[x]], setpart[y]], t],
             member[pair[pair[setpart[u], setpart[y]], setpart[z]], t]],
            member[pair[pair[setpart[u], setpart[x]], setpart[z]], t]], s → FNRS]]

Out[11]= or[not[equal[0, fix[composite[fix[composite[inverse[SECOND], IMG]], fix[
               composite[inverse[t], complement[CAP], cross[composite[CART, DUP], Id]]]]]]],
           not[subclass[t, cart[inverse[fix[composite[inverse[SECOND], IMG]]], V]]],
           not[subclass[inverse[fix[composite[inverse[SECOND], IMG]]],
             fix[composite[inverse[t], CAP, cross[composite[CART, DUP], Id]]]]],
           subclass[composite[t, LEFT[setpart[u]], t, LEFT[setpart[v]]],
            composite[t, LEFT[setpart[u]]]]] == True

In[12]:= (% /. {t → t_, u → u_, v → v_}) /. Equal → SetDelayed
```

This rather messy statement can be cleaned up as follows:

Theorem. The composite of the functions **FNRS ∘ LEFT[u]** and **FNRS ∘ LEFT[v]** is contained in the left hand factor: **FNRS ∘ LEFT[u]**.

```
In[13]:= Map[not, SubstTest[and, implies[p1, p2], implies[p1, p3], implies[p1, p4],
            implies[and[p2, p3, p4], p5], not[implies[p1, p5]], {p1 → equal[t, FNRS],
             p2 -> equal[0, fix[composite[fix[composite[inverse[SECOND], IMG]], fix[
                  composite[inverse[t], complement[CAP], cross[composite[CART, DUP], Id]]]]]],
             p3 -> subclass[t, cart[inverse[fix[composite[inverse[SECOND], IMG]]], V]],
             p4 -> subclass[inverse[fix[composite[inverse[SECOND], IMG]]],
               fix[composite[inverse[t], CAP, cross[composite[CART, DUP], Id]]]],
             p5 -> subclass[composite[t, LEFT[setpart[u]], t, LEFT[setpart[v]]],
               composite[t, LEFT[setpart[u]]]]}]] // Reverse

Out[13]= or[not[equal[t, composite[CAP, cross[composite[CART, DUP], Id],
              id[inverse[fix[composite[inverse[SECOND], IMG]]]]]]],
           subclass[composite[t, LEFT[setpart[u]], t, LEFT[setpart[v]]],
            composite[t, LEFT[setpart[u]]]]] == True

In[14]:= (% /. {t → t_, u → u_, v → v_}) /. Equal → SetDelayed
```

The following observation implies a corollary.

```
In[15]:= subclass[LEFT[u], inverse[SECOND]]

Out[15]= True
```

Corollary. The composite of the functions **FNRS ∘ LEFT[u]** and **FNRS ∘ LEFT[v]** is contained in the relation **FNRS ∘ inverse[SECOND]**.

```
In[16]:= Map[not, SubstTest[and, implies[p1, p2],
            implies[p2, p3], not[implies[p1, p3]], {p1 → equal[t, FNRS],
             p2 → subclass[composite[t, LEFT[setpart[u]], t, LEFT[setpart[v]]],
               composite[t, LEFT[setpart[u]]]], p3 -> subclass[composite[t, LEFT[setpart[u]],
                t, LEFT[setpart[v]]], composite[t, inverse[SECOND]]]}]] // Reverse

Out[16]= or[not[equal[t, composite[CAP, cross[composite[CART, DUP], Id],
              id[inverse[fix[composite[inverse[SECOND], IMG]]]]]]]],
           subclass[composite[t, LEFT[setpart[u]], t, LEFT[setpart[v]]],
            composite[t, inverse[SECOND]]]] == True

In[17]:= (% /. {t → t_, u → u_, v → v_}) /. Equal → SetDelayed
```

After some experimentation it was discovered that it is much easier to eliminate the variables **u** and **v** if one works with right multiplications for the flipped action **flip[FNRS]** instead of left multiplications of **FNRS**. Some additional steps are needed to do this.

Corollary.

```
In[18]:= SubstTest[implies, equal[s, FNRS],
           subclass[composite[s, LEFT[setpart[u]], s, LEFT[setpart[v]]],
            composite[s, inverse[SECOND]]], s → flip[t]] // Reverse

Out[18]= or[not[equal[composite[t, SWAP], composite[CAP, cross[composite[CART, DUP], Id],
              id[inverse[fix[composite[inverse[SECOND], IMG]]]]]]]],
           subclass[composite[t, RIGHT[setpart[u]], t, RIGHT[setpart[v]]],
            composite[t, inverse[FIRST]]]] == True

In[19]:= (% /. {t → t_, u → u_, v → v_}) /. Equal → SetDelayed
```

Corollary.

```
In[20]:= (Map[not, SubstTest[and, implies[and[p1, p2], p3],
              implies[and[p2, p3], p4], not[implies[and[p1, p2], p4]],
              {p1 → equal[s, flip[FNRS]], p2 → equal[s, t], p3 -> equal[flip[t], FNRS],
               p4 -> subclass[composite[t, RIGHT[setpart[u]], t, RIGHT[setpart[v]]],
                 composite[t, inverse[FIRST]]]}]] // Reverse) /. s → flip[FNRS]

Out[20]= or[not[equal[t, composite[CAP,
              cross[Id, composite[CART, DUP]], id[fix[composite[inverse[SECOND], IMG]]]]]]],
           subclass[composite[t, RIGHT[setpart[u]], t, RIGHT[setpart[v]]],
            composite[t, inverse[FIRST]]]] == True
```

```
In[21]:= (% /. {t → t_, u → u_, v → v_}) /. Equal → SetDelayed
```

All the remaining variables can now be eliminated all at once to obtain a transitive law.

Theorem. Two-sided restriction to final segments is transitive.

```
In[22]:= Map[empty[composite[Id, complement[#]]] &, SubstTest[class, pair[u, v],
           implies[equal[t, s], subclass[composite[t, RIGHT[setpart[u]], t, RIGHT[setpart[v]]],
             composite[t, inverse[FIRST]]]], s → flip[FNRS]]] /. t → flip[FNRS]

Out[22]= TRANSITIVE[
          composite[CAP, id[composite[CART, DUP, fix[composite[inverse[SECOND], IMG]]]],
           inverse[FIRST]]] == True
```

```
In[23]:= TRANSITIVE[
          composite[CAP, id[composite[CART, DUP, fix[composite[inverse[SECOND], IMG]]]],
           inverse[FIRST]]] := True
```

Restatement.

```
In[24]:= TRANSITIVE[composite[FNRS, inverse[SECOND]]]

Out[24]= True
```

final comments

Comment. The relation **FNRS ∘ inverse[SECOND]** is also antisymmetric.

```
In[25]:= ANTISYMMETRIC[composite[FNRS, inverse[SECOND]]]

Out[25]= True
```

The union of the domain and range of **FNRS ∘ inverse[SECOND]** is the universal class **V**.

```
In[33]:= udora[composite[FNRS, inverse[SECOND]]]

Out[33]= V
```

The following statement also follows automatically now.

```
In[36]:= PARTIALORDER[union[Id, composite[FNRS, inverse[SECOND]]]]

Out[36]= True
```