a function that subcommutes with S

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 0:56

        Loading Simplification Rules

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

        weightlimit = 40
```

summary

The function **CUP ∘ id[x] ∘ inverse[FIRST]** subcommutes with **S** if **x** does.

an assertion

```
In[2]:= subclass[composite[id[x], inverse[FIRST], y],
          composite[inverse[SECOND], z]] // AssertTest

Out[2]= subclass[composite[id[x], inverse[FIRST], y], composite[inverse[SECOND], z]] ==
         subclass[composite[x, y], z]

In[3]:= subclass[composite[id[x_], inverse[FIRST], y_], composite[inverse[SECOND], z_]] :=
         subclass[composite[x, y], z]
```

a theorem about subcommute

Lemma.

```
In[4]:= Assoc[CUP, cross[S, S], composite[id[x], inverse[FIRST]]]

Out[4]= composite[CUP,
          intersection[composite[inverse[FIRST], S], composite[inverse[SECOND], S, x]]] ==
         composite[S, CUP, id[x], inverse[FIRST]]

In[5]:= composite[CUP,
          intersection[composite[inverse[FIRST], S], composite[inverse[SECOND], S, x_]]] :=
         composite[S, CUP, id[x], inverse[FIRST]]
```

Theorem. If **x** subcommutes with **S**, then so does the function **CUP ∘ id[x] ∘ inverse[FIRST]** .

```
In[6]:= SubstTest[implies, subclass[u, v], subclass[composite[t, u], composite[t, v]],
          {t → CUP, u -> composite[id[x], inverse[FIRST], S], v -> intersection[
             composite[inverse[FIRST], S], composite[inverse[SECOND], S, x]]}] // Reverse

Out[6]= or[not[subclass[composite[x, S], composite[S, x]]],
          subclass[composite[CUP, id[x], inverse[FIRST], S],
           composite[S, CUP, id[x], inverse[FIRST]]]] == True

In[7]:= or[not[subclass[composite[x_, S], composite[S, x_]]],
          subclass[composite[CUP, id[x_], inverse[FIRST], S],
           composite[S, CUP, id[x_], inverse[FIRST]]]] := True
```

Corollary. The function **CUP ∘ id[IMAGE[t] ∘ CART ∘ DUP] ∘ inverse[FIRST]** subcommutes with **S**.

```
In[8]:= SubstTest[implies, subcommute[t, S],
          subcommute[composite[CUP, id[t], inverse[FIRST]], S],
          t → composite[IMAGE[x], CART, DUP]] // Reverse

Out[8]= subclass[composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST], S],
          composite[S, CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]]] == True

In[9]:= subclass[composite[CUP, id[composite[IMAGE[x_], CART, DUP]], inverse[FIRST], S],
          composite[S, CUP, id[composite[IMAGE[x_], CART, DUP]], inverse[FIRST]]] := True
```

Corollary. The function **CUP ∘ id[IMAGE[t] ∘ CART ∘ DUP] ∘ inverse[FIRST]** is monotone.

```
In[10]:= SubstTest[subcommute, funpart[t], S,
          t -> composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST]]]

Out[10]= subclass[composite[CUP, id[composite[IMAGE[x], CART, DUP]], inverse[FIRST],
            S, FIRST, id[composite[IMAGE[x], CART, DUP]], inverse[CUP]], S] == True

In[11]:= subclass[composite[CUP, id[composite[IMAGE[x_], CART, DUP]], inverse[FIRST],
            S, FIRST, id[composite[IMAGE[x_], CART, DUP]], inverse[CUP]], S] := True
```