words for a given alphabet

Johan G. F. Belinfante
2008 October 22

```
In[1]:= SetDirectory["l:"]; << goedel.08oct22a;<< tools.m

        :Package Title: goedel.08oct22a          2008 October 22 at 5:55 a.m.

        It is now:  2008 Oct 22 at 5:58

        Loading Simplification Rules

        TOOLS.M                                    Revised 2008 October 21

        weightlimit = 40
```

summary

The terms **lists**, **strings** and **words** are used interchangeably in the literature for functions whose domain is a natural number. In this notebook the focus is on restricting the alphabet. The class of words with a given alphabet **x** is.

```
In[2]:= class[t, and[FUNCTION[t], member[domain[t], omega], subclass[range[t], x]]]

Out[2]= intersection[LISTS, P[cart[omega, x]]]
```

reference

The words for a given alphabet **x** form a semigroup under concatenation, known as the **free semigroup** on **x**.

```
In[4]:= "Saunders MacLane, Categories for the Working
         Mathematician, Springer-Verlag, New York, 1971. (cf. page 140)";
```

sethood theorem

Lemma. (A formula for the class of words on a given alphabet **x**.)

```
In[5]:= U[image[MAP, cart[omega, P[x]]]] // Normality

Out[5]= U[image[MAP, cart[omega, P[x]]]] == intersection[LISTS, P[cart[omega, x]]]

In[6]:= U[image[MAP, cart[omega, P[x_]]]] := intersection[LISTS, P[cart[omega, x]]]
```

Lemma.

```
In[7]:= SubstTest[U, U[image[MAP, cart[w, P[x]]]], w → omega] // Reverse

Out[7]= core[LISTS, cart[omega, x]] == cart[omega, x]

In[8]:= core[LISTS, cart[omega, x_]] := cart[omega, x]
```

Theorem. The class of words using an alphabet **x** is a set if and only if **x** itself is a set.

```
In[9]:= SubstTest[member, core[u, v], V, {u → LISTS, v → cart[omega, x]}]

Out[9]= member[intersection[LISTS, P[cart[omega, x]]], V] == member[x, V]

In[10]:= member[intersection[LISTS, P[cart[omega, x_]]], V] := member[x, V]
```

concatenating words for a given alphabet

Technical Lemma.

```
In[11]:= composite[id[LISTS], LB[UB[complement[
             cross[Id, union[cart[V, complement[omega]], cart[complement[x], V]]]]]], PLUS,
           IMAGE[FIRST], id[intersection[LISTS, P[cart[omega, x]]]] // ReifNormality

Out[11]= composite[id[LISTS], LB[UB[complement[
             cross[Id, union[cart[V, complement[omega]], cart[complement[x], V]]]]]],
           PLUS, IMAGE[FIRST], id[intersection[LISTS, P[cart[omega, x]]]] ==
          cart[intersection[LISTS, P[cart[omega, x]]], intersection[LISTS, P[cart[omega, x]]]]

In[12]:= composite[id[LISTS], LB[UB[complement[
             cross[Id, union[cart[V, complement[omega]], cart[complement[x_], V]]]]]],
           PLUS, IMAGE[FIRST], id[intersection[LISTS, P[cart[omega, x_]]]] :=
          cart[intersection[LISTS, P[cart[omega, x]]], intersection[LISTS, P[cart[omega, x]]]]
```

Theorem.

```
In[13]:= ImInComp[CUP,
          composite[intersection[composite[inverse[SECOND], SECOND], composite[inverse[FIRST],
             COMPOSE, cross[Id, composite[INVERSE, PLUS, IMAGE[FIRST]]]]],
           SWAP, id[cart[LISTS, LISTS]]], P[cart[omega, x]]]

Out[13]= image[inverse[JOIN], P[cart[omega, x]]] ==
          cart[intersection[LISTS, P[cart[omega, x]]], intersection[LISTS, P[cart[omega, x]]]]

In[14]:= image[inverse[JOIN], P[cart[omega, x_]]] :=
          cart[intersection[LISTS, P[cart[omega, x]]], intersection[LISTS, P[cart[omega, x]]]]
```

Theorem. (The class of words for a given alphabet is invariant under concatenation.)

```
In[15]:= ImageComp[JOIN, inverse[JOIN], P[cart[omega, x]]] // Reverse

Out[15]= image[JOIN, cart[P[cart[omega, x]], P[cart[omega, x]]]] ==
          intersection[LISTS, P[cart[omega, x]]]
```

```
In[16]:= image[JOIN, cart[P[cart[omega, x_]], P[cart[omega, x_]]]] :=
          intersection[LISTS, P[cart[omega, x]]]
```

the free semigroup on x

Theorem. (Concatenation restricted to a given alphabet is a binary operation.)

```
In[17]:= SubstTest[and, member[t, FUNS], equal[domain[t], cartsq[fix[domain[t]]]],
          subclass[range[t], fix[domain[t]]], t -> composite[id[P[cart[omega, x]]], JOIN]]
Out[17]= member[composite[id[P[cart[omega, x]]], JOIN], BINOPS] == member[x, V]

In[18]:= member[composite[id[P[cart[omega, x_]]], JOIN], BINOPS] := member[x, V]
```

Theorem. (Simplification rule.)

```
In[19]:= equal[composite[JOIN, id[cart[P[cart[omega, x]], P[cart[omega, x]]]]],
           composite[id[P[cart[omega, x]]], JOIN]] // AssertTest
Out[19]= equal[composite[JOIN, id[cart[P[cart[omega, x]], P[cart[omega, x]]]]],
           composite[id[P[cart[omega, x]]], JOIN]] == True

In[20]:= composite[JOIN, id[cart[P[cart[omega, x_]], P[cart[omega, x_]]]]] :=
          composite[id[P[cart[omega, x]]], JOIN]
```

Theorem. The associative law.

```
In[21]:= SubstTest[implies, and[associative[x], subclass[image[x, cartsq[t]], t]],
           associative[composite[x, id[cartsq[t]]]],
           {x → JOIN, t -> intersection[LISTS, P[cart[omega, x]]]}] // Reverse
Out[21]= associative[composite[id[P[cart[omega, x]]], JOIN]] == True

In[22]:= associative[composite[id[P[cart[omega, x_]]], JOIN]] := True
```

Theorem. (The free semigroup with alphabet **x**.)

```
In[23]:= member[composite[id[P[cart[omega, x]]], JOIN], SEMIGPS] // AssertTest
Out[23]= member[composite[id[P[cart[omega, x]]], JOIN], SEMIGPS] == member[x, V]

In[24]:= member[composite[id[P[cart[omega, x_]]], JOIN], SEMIGPS] := member[x, V]
```