the class ENUMS

Johan G. F. Belinfante
2010 December 3

In[1]:= SetDirectory["l:"; << goedel.10dec01a

:Package Title: goedel.10dec01a

2010 December 1 at 12:30 noon

It is now: 2010 Dec 3 at 8:19

Loading Simplification Rules

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

weightlimit = 40

summary

The definition of \texttt{enum}[x] as the union of partial enumerations of the ordinals in the class \( x \) does not provide a membership rule, and makes it difficult to quantify over the variable \( x \). It is nonetheless possible to derive variable-free statements about all enumerations by exploiting the fact that \texttt{enum}[x] is the unique function contained in \( \Omega \times \Omega \) that subcommutes with the membership relation and whose range is the class of ordinals in \( x \). In this notebook the class \texttt{ENUMS} of all (small) enumerations is defined, and some of its properties are derived.

introduction

In this section a rewrite rule is derived that allows one to regard \texttt{enum}[x] as a wrapper for enumerations.

Lemma.

In[4]:= SubstTest[subcommute, funpart[t], E, t -> enum[x]] // Reverse

Out[4]= subclass[composite[enum[x], E], composite[inverse[IMAGE[inverse[enum[x]]]], E]] := True

In[5]:= subclass[composite[enum[x_], E],
                  composite[inverse[IMAGE[inverse[enum[x_]]]], E]] := True

The following rewrite rule characterizing enumerations resembles a wrapper-removal rule.

Theorem.

In[6]:= equiv[equal[x, enum[range[x]]], and[FUNCTION[x],
               subclass[x, cart[OMEGA, OMEGA]], subclass[composite[x, E], composite[E, x]]]]

The class `ENUMS` of all (small) enumerations is defined by the following equation.

```plaintext
In[8]:= intersection[FUNS,
    image[INVERSE, subcommutant[inverse[E]]], P[cart[OMEGA, OMEGA]]] := ENUMS
```

It is easier to reason about the class `ENUMS` if membership statements do not automatically expand out. Accordingly, only theorems about membership will be derived.

Theorem.

```plaintext
In[9]:= Map[implies[#, FUNCTION[x]] &, SubstTest[member, x, intersection[u, v],
    {u -> intersection[FUNS, image[INVERSE, subcommutant[inverse[E]]]],
    v -> P[cartsq[OMEGA]]}]] // Reverse
Out[9]= or[FUNCTION[x], not[member[x, ENUMS]]] = True
```

Theorem.

```plaintext
In[10]:= or[FUNCTION[x_], not[member[x_, ENUMS]]] := True
```

Theorem.

```plaintext
In[11]:= Map[implies[#, subcommute[x, E]] &, SubstTest[member, x, intersection[u, v],
    {u -> intersection[FUNS, image[INVERSE, subcommutant[inverse[E]]]],
    v -> P[cartsq[OMEGA]]}]] // Reverse
Out[11]= or[not[member[x, ENUMS]], subclass[composite[x, E], composite[E, x]]] = True
```

Theorem.

```plaintext
In[12]:= or[not[member[x_, ENUMS]], subclass[composite[x_, E], composite[E, x_]]] := True
```

Theorem.

```plaintext
In[13]:= Map[implies[#, subclass[x, cartsq[OMEGA]]] &, SubstTest[member, x, intersection[u, v],
    {u -> intersection[FUNS, image[INVERSE, subcommutant[inverse[E]]]],
    v -> P[cartsq[OMEGA]]}]] // Reverse
Out[13]= or[not[member[x, ENUMS]], subclass[x, cart[OMEGA, OMEGA]]] = True
```

In the reverse direction, one has:

Theorem.
The following may nonetheless be convenient.

Theorem.

Lemma.

Theorem.
In[31]:= and[equal[x, enum[range[x]]], member[x, V]]
Out[31]= member[x, ENUMS]

ranges of enumerations

Lemma.

In[32]:= SubstTest[implies, member[pair[x, y], composite[Id, t]], member[x, domain[t]],
{t -> composite[id[ENUMS], inverse[IMAGE[SECOND]], y -> enum[x]]} // Reverse
Out[32]= or[member[x, image[IMAGE[SECOND], ENUMS]],
not[member[x, V]], notsubclass[x, OMEGA]] = True

In[33]:= (% /. x -> x_) /. Equal -> SetDelayed

Theorem. (Eliminate the variable x.)

In[34]:= Map[equal[V, #] & , SubstTest[class, x, implies[member[x, u], member[x, v]],
{u -> P[OMEGA], v -> image[IMAGE[SECOND], ENUMS]]]
Out[34]= subclass[P[OMEGA], image[IMAGE[SECOND], ENUMS]] = True

In[35]:= subclass[P[OMEGA], image[IMAGE[SECOND], ENUMS]] := True

Lemma. (Eliminating the variable x.)

In[40]:= SubstTest[implies, and[member[x, v], equal[x, enum[range[x]]]],
subclass[range[x], OMEGA], v -> V] // Reverse
Out[40]= or[not[member[x, ENUMS]], subclass[range[x], OMEGA]] = True

In[41]:= or[not[member[x_, ENUMS]], subclass[range[x_], OMEGA]] := True

Lemma. (Eliminating the variable x.)

In[42]:= Map[equal[V, #] & , SubstTest[class, x, implies[member[x, u], member[x, v]],
{u -> ENUMS, v -> image[inverse[IMAGE[SECOND]], P[OMEGA]]}]]

In[43]:= % /. Equal -> SetDelayed

Theorem.

In[44]:= SubstTest[and, subclass[u, v], subclass[v, u],
{u -> P[OMEGA], v -> image[IMAGE[SECOND], ENUMS]]]
Out[44]= equal[image[IMAGE[SECOND], ENUMS], P[OMEGA]] = True

In[45]:= image[IMAGE[SECOND], ENUMS] := P[OMEGA]
domains of enumerations

Theorem.

\[
\text{SubstTest[implies, and[member[x, V], equal[x, enum[t]]], member[domain[x], OMEGA], t \to range[x]]} \quad \text{// Reverse}
\]

\[
\text{or[member[domain[x], OMEGA], not[member[x, ENUMS]]]} = \text{True}
\]

Lemma. (Eliminate the variable \( x \).)

\[
\text{Map[equal[V, #] \&, SubstTest[class, x, implies[member[x, u], member[x, v]], \{u \to \text{ENUMS, } v \to \text{image[\text{inverse[IMAGE[\text{FIRST}], OMEGA}]\}}\]}}
\]

\[
\text{subclass[\text{image[IMAGE[\text{FIRST}], ENUMS], OMEGA]} = \text{True}}
\]

An inclusion in the opposite direction can be derived by considering enumerations of ordinals.

Lemma. The identity function on any ordinal is an enumeration.

\[
\text{SubstTest[member, enum[t], ENUMS, t \to \text{ord[x]}]} \quad \text{// Reverse}
\]

\[
\text{member[id[\text{ord[x]}], ENUMS]} = \text{True}
\]

Corollary. A simple special case.

\[
\text{SubstTest[member, id[\text{ord[x]}], ENUMS, x \to 0]} \quad \text{// Reverse}
\]

\[
\text{member[0, ENUMS]} = \text{True}
\]

Theorem.

\[
\text{Map[empty, SubstTest[reify, x, dif[set[\text{ord[x]}], t], t \to \text{image[\text{inverse[IMAGE[DUP]], ENUMS}]}}]}
\]

\[
\text{subclass[\text{image[IMAGE[DUP], OMEGA], ENUMS}} = \text{True}
\]

\[
\text{subclass[\text{image[IMAGE[DUP], ENUMS]} = \text{True}}
\]

Corollary. An inclusion in the opposite direction form an earlier one.
Theorem. The class of domains of (small) enumerations is $\Omega$.

Lemma. Enumerations with the same range are equal.

Theorem. The function $\text{IMAGE}[\text{SECOND}]$ restricted to $\text{ENUMS}$ is one-to-one.
This provides a one-to-one correspondence between sets of ordinals and enumerations. The composite of this function with \texttt{IMAGE[FIRST]} yields a one-to-one correspondence between sets of ordinals and ordinals.

**Corollary.**  

\texttt{In[87]}:= \texttt{SubstTest[FUNCTION, composite[funpart[u], funpart[v]],  
\{u \rightarrow \texttt{IMAGE[FIRST]}, v \rightarrow \texttt{composite[id[ENUMS], inverse[IMAGE[SECOND]]]}})]} // \texttt{Reverse}

\texttt{Out[87]}= \texttt{FUNCTION[inverse[image[DORA, ENUMS]]]} = True

\texttt{In[88]}:= \texttt{FUNCTION[inverse[image[DORA, ENUMS]]]} := True

**Comment.** The function \texttt{inverse[image[DORA, ENUMS]]} assigns an ordinal to each set of ordinals, usually called its \textit{order type}.

---

**closure under unions of chains**

In this section it is shown that the class of enumerations is closed under unions of chains. This is true because each of the three classes whose intersection is \texttt{ENUMS} also has that property. In fact, two of these three classes is closed under arbitrary unions. The class \texttt{FUNS} is the only one of the three that is not closed under arbitrary unions.

**Lemma.**  (Closure under arbitrary unions implies closure under unions of chains.)

\texttt{In[91]}:= \texttt{SubstTest[Uchains, Uclosure[t], t \rightarrow \texttt{image[INVERSE, subcommutant[x]]}})]} // \texttt{Reverse}

\texttt{Out[91]}= \texttt{Uchains[image[INVERSE, subcommutant[x]]]} = \texttt{image[INVERSE, subcommutant[x]]}

\texttt{In[92]}:= \texttt{Uchains[image[INVERSE, subcommutant[x_]]]} := \texttt{image[INVERSE, subcommutant[x]]}

**Theorem.** The class \texttt{ENUMS} is closed under unions of chains.

\texttt{In[93]}:= \texttt{SubstTest[implies, and[equal[x, Uchains[x]], equal[y, Uchains[y]]],  
equal[intersection[x, y], Uchains[intersection[x, y]]],  
{x \rightarrow \texttt{FUNS}, y \rightarrow \texttt{intersection[  
image[INVERSE, subcommutant[\texttt{inverse[E]}]], P[cart[\texttt{OMEGA}, \texttt{OMEGA}]]}})]} // \texttt{Reverse}

\texttt{Out[93]}= \texttt{equal[ENUMS, Uchains[ENUMS]]} = True

\texttt{In[94]}:= \texttt{Uchains[ENUMS]} := \texttt{ENUMS}

---

**enumerations are one-to-one**

**Lemma.**

\texttt{In[96]}:= \texttt{SubstTest[implies, and[member[x, V], equal[x, enum[t]]],  
\text{member[x, BIJ], t \rightarrow range[x]}  \text{]}  \text{ // MapNotNot} // Reverse}

\texttt{Out[96]}= \texttt{or[FUNCTION[inverse[x]], not[member[x, ENUMS]]]} = True
Lemma.

\textbf{In}[103]:= member[x, \text{dif[ENUMS, BIJ]}] // NotNotTest

\textbf{Out}[103]= \text{or[and[member[x, ENUMS], not [FUNCTION[x]]],}
\text{and[member[x, ENUMS], not [FUNCTION[inverse[x]]]]] = False}

\textbf{In}[104]= \text{nothing} /. \text{Equal} \rightarrow \text{SetDelayed}

Theorem. Every enumeration is a bijection.

\textbf{In}[105]= Map[empty, \text{dif[ENUMS, BIJ]} // Normality]

\textbf{Out}[105]= \text{subclass[ENUMS, BIJ] = True}

\textbf{In}[106]= \text{subclass[ENUMS, BIJ] := True}

Corollary. Any set of ordinals is equipollent to its order type.

\textbf{In}[108]= \text{SubstTest[implies, subclass[u, v],}
\text{subclass[image[t, u], image[t, v]], \{t \rightarrow \text{DORA}, u \rightarrow \text{ENUMS}, v \rightarrow \text{BIJ}\] // Reverse}

\textbf{Out}[108]= \text{subclass[image[\text{DORA, ENUMS}], Q] = True}

\textbf{In}[109]= \text{subclass[image[\text{DORA, ENUMS}], Q] := True}