CURRY

Johan G. F. Belinfante
2006 November 6

```
In[1]:= SetDirectory["l:"]; << goedel87.06a; << tools.m

        :Package Title: goedel87.06a          2006 November 6 at 5:15 p.m.

        It is now:  2006 Nov 6 at 17:14

        Loading Simplification Rules

        TOOLS.M                       Revised 2006 November 1

        weightlimit = 40
```

summary

A function **CURRY** is introduced which can be used to transform functions with two arguments to functions with one argument. A typical example is the conversion of **NATADD** to **PLUS**. Comments. This function is named in honor of Haskell B. Curry. The definition used was inspired by the process called currying in the lambda calculus, but unlike the lambda calculus, the definition used here is formulated in a set-theoretic framework. The definition of **CURRY** is formulated in a **class**-wrapped membership rewrite rule:

```
In[2]:= Begin["Goedel`Private`"];

In[3]:= FirstMatch[class[y_, member[z_, HoldPattern[CURRY]]]]

Out[3]= class[t_, member[z_, CURRY]] := ReleaseHold[Module[{w = Unique[], x = Unique[], y =
          Unique[]}, class[t, exists[w, x, y, and[equal[z, pair[x, y]], equal[w, image[
          ASSOC], x]], member[pair[w, y], VS]], subclass[x, cart[cart[V, V], V]]]]]]]
```

one-to-one restrictions of IMAGE[ASSOC]

Lemma. (A temporary rewrite rule.)

```
In[4]:= SubstTest[implies, subclass[composite[x, y], Id],
          FUNCTION[composite[inverse[y], id[domain[x]]]], {x → IMAGE[inverse[ASSOC]],
           y → composite[IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]]}] // Reverse

Out[4]= FUNCTION[composite[id[P[cart[cart[V, V], V]]], inverse[IMAGE[ASSOC]]]] == True

In[5]:= % /. Equal → SetDelayed
```

Corollary of the lemma.

```
In[6]:= SubstTest[composite, funpart[x], inverse[funpart[x]],
          x -> composite[id[P[cart[cart[V, V], V]]], inverse[IMAGE[ASSOC]]]] // Reverse

Out[6]= composite[id[P[cart[cart[V, V], V]]], inverse[IMAGE[ASSOC]],
          IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]] == id[P[cart[cart[V, V], V]]]

In[7]:= % /. Equal → SetDelayed
```

Lemma. (Special case of the main theorem.)

```
In[8]:= Assoc[composite[id[P[cart[cart[V, V], V]]], inverse[IMAGE[ASSOC]]],
         composite[IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]],
         composite[IMAGE[inverse[ASSOC]], id[P[cart[V, cart[V, V]]]]]]

Out[8]= composite[id[P[cart[cart[V, V], V]]], inverse[IMAGE[ASSOC]]] ==
         composite[IMAGE[inverse[ASSOC]], id[P[cart[V, cart[V, V]]]]]

In[9]:= % /. Equal → SetDelayed
```

Theorem.

```
In[10]:= Assoc[id[P[cart[cart[x, y], z]]],
           id[P[cart[cart[V, V], V]]], inverse[IMAGE[ASSOC]]] // Reverse

Out[10]= composite[id[P[cart[cart[x, y], z]]], inverse[IMAGE[ASSOC]]] ==
          composite[IMAGE[inverse[ASSOC]], id[P[cart[x, cart[y, z]]]]]

In[11]:= composite[id[P[cart[cart[x_, y_], z_]]], inverse[IMAGE[ASSOC]]] :=
          composite[IMAGE[inverse[ASSOC]], id[P[cart[x, cart[y, z]]]]]
```

Corollary.

```
In[12]:= composite[IMAGE[ASSOC], id[P[cart[cart[x, y], z]]]] // DoubleInverse // Reverse

Out[12]= composite[id[P[cart[x, cart[y, z]]]], inverse[IMAGE[inverse[ASSOC]]]] ==
          composite[IMAGE[ASSOC], id[P[cart[cart[x, y], z]]]]

In[13]:= composite[id[P[cart[x_, cart[y_, z_]]]], inverse[IMAGE[inverse[ASSOC]]]] :=
          composite[IMAGE[ASSOC], id[P[cart[cart[x, y], z]]]]
```

---

a formula for CURRY

```
In[14]:= CURRY // Normality // Reverse

Out[14]= composite[VS, IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]] == CURRY

In[15]:= composite[VS, IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]] := CURRY
```

CURRY is one-to-one

```
In[16]:= SubstTest[FUNCTION,
           composite[funpart[v], IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]], v → VS] // Reverse

Out[16]= FUNCTION[CURRY] == True

In[17]:= FUNCTION[CURRY] := True

In[18]:= composite[IMAGE[inverse[ASSOC]],
           id[P[cart[V, cart[V, V]]]], inverse[VS]] // DoubleInverse

Out[18]= composite[IMAGE[inverse[ASSOC]], id[P[cart[V, cart[V, V]]]], inverse[VS]] ==
          inverse[CURRY]

In[19]:= composite[IMAGE[inverse[ASSOC]],
           id[P[cart[V, cart[V, V]]]], inverse[VS]] := inverse[CURRY]

In[20]:= SubstTest[FUNCTION, composite[IMAGE[x], id[P[cart[V, cart[V, V]]]], inverse[VS]],
           x -> inverse[ASSOC]] // Reverse

Out[20]= FUNCTION[inverse[CURRY]] == True

In[21]:= FUNCTION[inverse[CURRY]] := True
```

domain and range

```
In[22]:= IminComp[composite[VS, IMAGE[ASSOC]], id[P[cart[cart[V, V], V]]], V]

Out[22]= domain[CURRY] == P[cart[cart[V, V], V]]

In[23]:= domain[CURRY] := P[cart[cart[V, V], V]]

In[24]:= ImageComp[composite[VS, IMAGE[ASSOC]], id[P[cart[cart[V, V], V]]], V]

Out[24]= range[CURRY] ==
          intersection[FUNS, P[cart[V, intersection[complement[set[0]], P[cart[V, V]]]]]]

In[25]:= range[CURRY] :=
          intersection[FUNS, P[cart[V, intersection[complement[set[0]], P[cart[V, V]]]]]]
```

vertical sections and APPLY

```
In[26]:= SubstTest[image, funpart[w], set[x], w → CURRY] // Reverse

Out[26]= image[CURRY, set[x]] == set[APPLY[CURRY, x]]

In[27]:= image[CURRY, set[x_]] := set[APPLY[CURRY, x]]
```

```
In[28]:= SubstTest[image, funpart[w], set[x], w → inverse[CURRY]] // Reverse

Out[28]= image[inverse[CURRY], set[x]] == set[APPLY[inverse[CURRY], x]]

In[29]:= image[inverse[CURRY], set[x_]] := set[APPLY[inverse[CURRY], x]]

In[30]:= Map[A, ImageComp[inverse[CURRY], CURRY, set[x]]] // Reverse

Out[30]= APPLY[inverse[CURRY], APPLY[CURRY, x]] == union[x, complement[image[V, set[x]]],
           image[V, intersection[x, complement[cart[cart[V, V], V]]]]]

In[31]:= APPLY[inverse[CURRY], APPLY[CURRY, x_]] := union[x, complement[image[V, set[x]]],
           image[V, intersection[x, complement[cart[cart[V, V], V]]]]]
```

---

inverse[CURRY]

Lemma.

```
In[32]:= Map[VERTSECT, Assoc[id[cart[V, V]], ASSOC, inverse[E]]]

Out[32]= composite[IMAGE[id[cart[V, V]]], IMAGE[ASSOC]] == IMAGE[ASSOC]

In[33]:= composite[IMAGE[id[cart[V, V]]], IMAGE[ASSOC]] := IMAGE[ASSOC]
```

Theorem.

```
In[34]:= Assoc[IMAGE[cross[Id, inverse[E]]], VS,
          composite[IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]]]

Out[34]= composite[IMAGE[cross[Id, inverse[E]]], CURRY] ==
          composite[IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]]

In[35]:= composite[IMAGE[cross[Id, inverse[E]]], CURRY] :=
          composite[IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]]
```

An alternate formula for **inverse[CURRY]** is this:

```
In[36]:= Assoc[composite[IMAGE[inverse[ASSOC]], IMAGE[cross[Id, inverse[E]]]],
          CURRY, inverse[CURRY]]

Out[36]= composite[IMAGE[inverse[ASSOC]], IMAGE[cross[Id, inverse[E]]], id[intersection[FUNS,
             P[cart[V, intersection[complement[set[0]], P[cart[V, V]]]]]]]] == inverse[CURRY]

In[37]:= composite[IMAGE[inverse[ASSOC]], IMAGE[cross[Id, inverse[E]]], id[intersection[FUNS,
             P[cart[V, intersection[complement[set[0]], P[cart[V, V]]]]]]]] := inverse[CURRY]
```

an example: NATADD and PLUS

```
In[38]:= Map[A, ImageComp[VS, composite[IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]], set[NATADD]]]
Out[38]= APPLY[CURRY, NATADD] == PLUS

In[39]:= APPLY[CURRY, NATADD] := PLUS

In[40]:= SubstTest[APPLY, inverse[CURRY], APPLY[CURRY, x], x → NATADD] // Reverse
Out[40]= APPLY[inverse[CURRY], PLUS] == NATADD

In[41]:= APPLY[inverse[CURRY], PLUS] := NATADD
```

an example: NATMUL and TIMES

```
In[42]:= Map[A, ImageComp[VS, composite[IMAGE[ASSOC], id[P[cart[cart[V, V], V]]]], set[NATMUL]]]
Out[42]= APPLY[CURRY, NATMUL] == TIMES

In[43]:= APPLY[CURRY, NATMUL] := TIMES

In[44]:= SubstTest[APPLY, inverse[CURRY], APPLY[CURRY, x], x → NATMUL] // Reverse
Out[44]= APPLY[inverse[CURRY], TIMES] == NATMUL

In[45]:= APPLY[inverse[CURRY], TIMES] := NATMUL
```