band wrapper

Johan G. F. Belinfante
2009 February 17

```
In[1]:= << l:goedel.09feb16a;<< l:tools.m

        :Package Title: goedel.09feb16a         2009 February 16 at 5:20 a.m.

        It is now:  2009 Feb 17 at 5:56

        Loading Simplification Rules

        TOOLS.M                                      Revised 2009 February 15

        weightlimit = 40
```

summary

A band wrapper **band[x]** is defined to enable the properties of bands to be formulated as rewrite rules.

definition

The following rewrite rule serves to define the **band[x]** wrapper:

```
In[2]:= image[V, intersection[band[x_], set[y_]]] :=
          intersection[image[V, intersection[BANDS, set[x]]], image[V, intersection[x, set[y]]]]

        General::spell1 : Possible spelling error: new symbol name "band" is similar to existing symbol "and". More…
```

normalization

Theorem. Normalization rewrite rule for **band[x]**.

```
In[3]:= Map[fix, SubstTest[reify, y, image[V, intersection[t, set[y]]], t → band[x]]] // Reverse

Out[3]= intersection[x, image[V, intersection[BANDS, set[x]]]] == band[x]

In[4]:= intersection[x_, image[V, intersection[BANDS, set[x_]]]] := band[x]
```

wrapper introduction and removal rules

Lemma. Simplification rule.

```
In[5]:= equiv[or[equal[0, x], member[x, BANDS]], member[x, BANDS]]
Out[5]= True

In[6]:= or[equal[0, x_], member[x_, BANDS]] := member[x, BANDS]
```

Theorem. (Wrapper removal rule.)

```
In[7]:= SubstTest[equal, x,
          intersection[x, image[V, intersection[t, set[x]]]], t → BANDS] // Reverse
Out[7]= equal[x, band[x]] == member[x, BANDS]

In[8]:= equal[x_, band[x_]] := member[x, BANDS]
```

Theorem. (Automatic removal.)

```
In[9]:= implies[member[x, BANDS], equal[band[x], x]]
Out[9]= True

In[10]:= band[x_] := x /; member[x, BANDS]
```

Theorem. (Wrapper introduction rule.)

```
In[11]:= SubstTest[member,
           intersection[x, image[V, intersection[t, set[x]]]], t, t → BANDS] // Reverse
Out[11]= member[band[x], BANDS] == True

In[12]:= member[band[x_], BANDS] := True
```

Corollary. Bands are sets.

```
In[14]:= SubstTest[implies, member[u, v], member[u, V], {u → band[x], v → BANDS}] // Reverse
Out[14]= member[band[x], V] == True

In[15]:= member[band[x_], V] := True
```

FUNCTION rules

Theorem. Bands are binary operations.

```
In[16]:= SubstTest[implies, member[t, BANDS], member[t, BINOPS], t → band[x]] // Reverse
Out[16]= member[band[x], BINOPS] == True

In[17]:= member[band[x_], BINOPS] := True
```

Theorem. Bands are functions.

```
In[18]:= SubstTest[FUNCTION, binop[t], t → band[x]] // Reverse
Out[18]= FUNCTION[band[x]] == True

In[19]:= FUNCTION[band[x_]] := True
```

domain and range

Theorem. The domain of **band[x]** is the cartesian square of its range.

```
In[20]:= SubstTest[implies, member[t, BANDS],
           equal[domain[t], cartsq[range[t]]], t → band[x]] // Reverse
Out[20]= equal[cart[range[band[x]], range[band[x]]], domain[band[x]]] == True

In[21]:= cart[range[band[x_]], range[band[x_]]] := domain[band[x]]
```

Corollary. The domain of the domain is the range.

```
In[22]:= SubstTest[domain, cartsq[t], t → range[band[x]]] // Reverse
Out[22]= domain[domain[band[x]]] == range[band[x]]

In[23]:= domain[domain[band[x_]]] := range[band[x]]
```

Corollary. The fixed-point set of the domain is the range.

```
In[24]:= SubstTest[fix, cartsq[t], t → range[band[x]]] // Reverse
Out[24]= fix[domain[band[x]]] == range[band[x]]

In[25]:= fix[domain[band[x_]]] := range[band[x]]
```

Corollary. The domain is symmetric.

```
In[26]:= SubstTest[inverse, cartsq[t], t → range[band[x]]] // Reverse
Out[26]= inverse[domain[band[x]]] == domain[band[x]]

In[27]:= inverse[domain[band[x_]]] := domain[band[x]]
```

Corollary.

```
In[28]:= SubstTest[inverse, inverse[t], t → domain[band[x]]]

Out[28]= composite[Id, domain[band[x]]] == domain[band[x]]

In[29]:= composite[Id, domain[band[x_]]] := domain[band[x]]
```

Corollary. The range of the domain is the range.

```
In[30]:= SubstTest[range, cartsq[t], t → range[band[x]]] // Reverse

Out[30]= range[domain[band[x]]] == range[band[x]]

In[31]:= range[domain[band[x_]]] := range[band[x]]
```

Theorem. Membership in the domain.

```
In[32]:= SubstTest[member, pair[u, v], cartsq[t], t → range[band[x]]] // Reverse

Out[32]= member[pair[u, v], domain[band[x]]] ==
          and[member[u, range[band[x]]], member[v, range[band[x]]]]

In[33]:= member[pair[u_, v_], domain[band[x_]]] :=
          and[member[u, range[band[x]]], member[v, range[band[x]]]]
```

cartesian product upper bounds

Theorem.

```
In[35]:= SubstTest[subclass, composite[Id, t], cart[y, z], t → band[x]] // Reverse

Out[35]= subclass[band[x], cart[y, z]] ==
          and[subclass[domain[band[x]], y], subclass[range[band[x]], z]]

In[36]:= subclass[band[x_], cart[y_, z_]] :=
          and[subclass[domain[band[x]], y], subclass[range[band[x]], z]]
```

Theorem.

```
In[37]:= SubstTest[subclass, composite[Id, t], cart[y, z], t → domain[band[x]]] // Reverse

Out[37]= subclass[domain[band[x]], cart[y, z]] ==
          and[subclass[range[band[x]], y], subclass[range[band[x]], z]]

In[38]:= subclass[domain[band[x_]], cart[y_, z_]] :=
          and[subclass[range[band[x]], y], subclass[range[band[x]], z]]
```

APPLY rules

Theorem. An **APPLY** rule.

```
In[39]:= SubstTest[image, funpart[t], set[PAIR[u, v]], t → band[x]] // Reverse
Out[39]= image[band[x], cart[set[u], set[v]]] == set[APPLY[band[x], PAIR[u, v]]]

In[40]:= image[band[x_], cart[set[u_], set[v_]]] := set[APPLY[band[x], PAIR[u, v]]]
```

Lemma.

```
In[41]:= SubstTest[composite, binop[t], id[cartsq[V]], t → band[x]] // Reverse
Out[41]= composite[band[x], id[cart[V, V]]] == band[x]

In[42]:= composite[band[x_], id[cart[V, V]]] := band[x]
```

Theorem. Automatic replacement of **pair** with **PAIR**.

```
In[43]:= ApComp[band[x], id[cart[V, V]], pair[u, v]] // Reverse
Out[43]= APPLY[band[x], pair[u, v]] == APPLY[band[x], PAIR[u, v]]

In[44]:= APPLY[band[x_], pair[u_, v_]] := APPLY[band[x], PAIR[u, v]]
```

Theorem.

```
In[45]:= SubstTest[member, APPLY[funpart[t], w],
           range[funpart[t]], {t → band[x], w → pair[u, v]}] // Reverse
Out[45]= member[APPLY[band[x], PAIR[u, v]], range[band[x]]] ==
          and[member[u, range[band[x]]], member[v, range[band[x]]]]

In[46]:= member[APPLY[band[x_], PAIR[u_, v_]], range[band[x_]]] :=
          and[member[u, range[band[x]]], member[v, range[band[x]]]]
```

---

associative law

Theorem. Bands are semigroups.

```
In[47]:= SubstTest[implies, member[t, BANDS], member[t, SEMIGPS], t → band[x]] // Reverse
Out[47]= member[band[x], SEMIGPS] == True

In[48]:= member[band[x_], SEMIGPS] := True
```

Corollary. Bands are associative.

```
In[49]:= SubstTest[associative, semigp[t], t → band[x]] // Reverse
Out[49]= associative[band[x]] == True

In[50]:= associative[band[x_]] := True
```

Theorem. The associative law as a rewrite rule.

```
In[51]:= (SubstTest[APPLY, semigp[t], PAIR[u, APPLY[semigp[t], PAIR[v, w]]], t → band[x]]) //
          Reverse

Out[51]= APPLY[band[x], PAIR[u, APPLY[band[x], PAIR[v, w]]]] ==
          APPLY[band[x], PAIR[APPLY[band[x], PAIR[u, v]], w]]

In[52]:= APPLY[band[x_], PAIR[u_, APPLY[band[x_], PAIR[v_, w_]]]] :=
          APPLY[band[x], PAIR[APPLY[band[x], PAIR[u, v]], w]]
```

idempotence

Theorem. Every element **u ∈ range[band[x]]** is idempotent.

```
In[53]:= SubstTest[implies, and[member[u, range[t]], member[t, BANDS]],
            equal[APPLY[t, PAIR[u, u]], u], t → band[x]] // Reverse

Out[53]= or[equal[u, APPLY[band[x], PAIR[u, u]]], not[member[u, range[band[x]]]]] == True

In[54]:= or[equal[u_, APPLY[band[x_], PAIR[u_, u_]]], not[member[u_, range[band[x_]]]]] := True
```

Lemma. The range of a band is contained in the class of its idempotent elements.

```
In[55]:= (implies[member[t, BANDS], subclass[fix[domain[t]], fix[composite[t, DUP]]]] //
            AssertTest) /. t → band[x]

Out[55]= subclass[range[band[x]], fix[composite[band[x], DUP]]] == True

In[56]:= (% /. x → x_) /. Equal → SetDelayed
```

Theorem. The class of idempotents of a band is its range.

```
In[57]:= SubstTest[and, subclass[u, v], subclass[v, u],
           {u -> range[band[x]], v -> fix[composite[band[x], DUP]]}]

Out[57]= equal[fix[composite[band[x], DUP]], range[band[x]]] == True

In[58]:= fix[composite[band[x_], DUP]] := range[band[x]]
```

Corollary. (Restatement without the **band** wrapper.)

```
In[59]:= SubstTest[implies, equal[x, band[t]],
            equal[fix[composite[x, DUP]], range[x]], t → x] // Reverse

Out[59]= or[equal[fix[composite[x, DUP]], range[x]], not[member[x, BANDS]]] == True

In[60]:= or[equal[fix[composite[x_, DUP]], range[x_]], not[member[x_, BANDS]]] := True
```

Theorem. A formula for the composite of the fixed point class of **DUP** and **band[x]** in the reverse order.

```
In[61]:= SubstTest[id, fix[composite[t, DUP]], t → band[x]]

Out[61]= fix[composite[DUP, band[x]]] == id[range[band[x]]]
```

```
In[62]:= fix[composite[DUP, band[x_]]] := id[range[band[x]]]
```

duality

Theorem. A corollary of the definition of **BANDS**.

```
In[63]:= implies[and[member[x, SEMIGPS], equal[fix[domain[x]], fix[composite[x, DUP]]]],
           member[x, BANDS]] // AssertTest

Out[63]= or[member[x, BANDS],
           not[equal[fix[composite[x, DUP]], fix[domain[x]]]], not[member[x, SEMIGPS]]] == True

In[64]:= or[member[x_, BANDS], not[equal[fix[composite[x_, DUP]], fix[domain[x_]]]],
           not[member[x_, SEMIGPS]]] := True
```

Theorem. The dual of a band is a band.

```
In[65]:= SubstTest[implies, and[equal[fix[composite[t, DUP]], fix[domain[t]]],
             member[t, SEMIGPS]], member[t, BANDS], t → flip[band[x]]] // Reverse

Out[65]= member[composite[band[x], SWAP], BANDS] == True

In[66]:= member[composite[band[x_], SWAP], BANDS] := True
```

Corollary. (Restatement without the **band** wrapper.) If **x** is a band, then **flip[x]** is a band.

```
In[67]:= SubstTest[implies, equal[x, band[t]], member[flip[x], BANDS], t → x] // Reverse

Out[67]= or[member[composite[x, SWAP], BANDS], not[member[x, BANDS]]] == True

In[68]:= or[member[composite[x_, SWAP], BANDS], not[member[x_, BANDS]]] := True
```