computing rational hulls

Johan G. F. Belinfante
2012 August 7

```
In[1]:= SetDirectory["l:"]; << goedel.12aug05a

:Package Title: goedel.12aug05a                2012 August 5 at 9:00 a.m.

Loading takes about sixteen minutes, half that time due to builtin pauses.

It is now:  2012 Aug 7 at 12:9

Loading Simplification Rules

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

weightlimit = 40

Loading completed.

It is now:  2012 Aug 7 at 12:25
```

summary

In the **GOEDEL** program, rational numbers are maximal straight lines through the origin in the integer plane. The theorem that one rational number cannot be a subset of another is generalized. The **rational hull** of a set is the intersection of all rational numbers that contain that set. The final results derived in the first section of this notebook are useful for deriving explicit formulas for rational hulls. One special case is provided as an example in the second section.

derivation

A **sub-rational** function is defined as any subset of a rational number.

Lemma.

```
In[2]:= Map[not, SubstTest[and, implies[and[p2, p3], p5],
           implies[and[p1, p4, p5], p6], not[implies[and[p1, p2, p3, p4], p7]],
           {p1 → member[x, RATS], p2 → subclass[x, y], p3 → subclass[y, z], p4 → member[z, RATS],
            p5 → subclass[x, z], p6 -> equal[x, z], p7 → equal[x, y]}]] // Reverse

Out[2]= or[equal[x, y], not[member[x, RATS]],
          not[member[z, RATS]], not[subclass[x, y]], not[subclass[y, z]]] == True

In[3]:= (% /. {x → x_, y → y_, z → z_}) /. Equal → SetDelayed
```

Eliminating the variable **z** yields the following theorem, which generalizes the theorem that one rational number cannot be a subset of another.

Theorem. If a rational number is a subset of a sub-rational function, then they are equal.

```
In[4]:= Map[equal[V, domain[#]] &,
         SubstTest[reify, z, case[or[equal[x, y], not[member[x, t]], not[member[z, t]],
            not[subclass[x, y]], not[subclass[y, z]]]], t → RATS]]

Out[4]= or[equal[x, y], not[member[x, RATS]],
          not[member[y, image[inverse[S], RATS]]], not[subclass[x, y]]] == True

In[5]:= or[equal[x_, y_], not[member[x_, RATS]],
          not[member[y_, image[inverse[S], RATS]]], not[subclass[x_, y_]]] := True
```

Corollary. (The result of eliminating the variable **y**.)

```
In[6]:= Map[equal[V, domain[#]] &, SubstTest[reify, y,
          case[or[equal[x, y], not[member[x, u]], not[member[y, v]], not[subclass[x, y]]]],
          {u → RATS, v → image[inverse[S], RATS]}]]

Out[6]= or[not[member[x, RATS]], not[member[x, image[inverse[PS], RATS]]]] == True

In[7]:= or[not[member[x_, RATS]], not[member[x_, image[inverse[PS], RATS]]]] := True
```

Theorem. A variable-free rewrite rule.

```
In[8]:= equal[intersection[RATS, image[inverse[PS], RATS]], 0]

Out[8]= True

In[9]:= intersection[RATS, image[inverse[PS], RATS]] := 0
```

Another variable-free rewrite rule will be derived. The starting point is a lemma about a set sandwiched between two rationals.

Lemma. If $\mathbf{u} \subset \mathbf{x} \subset \mathbf{v}$ and if $\mathbf{u}, \mathbf{v}$ are rationals, then so is $\mathbf{x}$. (Indeed, $\mathbf{u} = \mathbf{x} = \mathbf{v}$.)

```
In[10]:= Map[not,
           SubstTest[and, implies[and[p1, p2, p3, p4], p5], not[implies[and[p1, p2, p3, p4], p6]],
            {p1 → member[u, RATS], p2 → subclass[u, x], p3 → subclass[x, v],
             p4 → member[v, RATS], p5 → equal[u, x], p6 → member[x, RATS]}]] // Reverse

Out[10]= or[member[x, RATS], not[member[u, RATS]],
           not[member[v, RATS]], not[subclass[u, x]], not[subclass[x, v]]] == True

In[11]:= (% /. {u → u_, v → v_, x → x_}) /. Equal → SetDelayed
```

Lemma. (Elimination of two variables.)

```
In[12]:= Map[equal[V, domain[#]] &, SubstTest[reify, t,
           case[or[member[x, z], not[member[first[t], z]], not[member[second[t], z]],
             not[subclass[first[t], x]], not[subclass[x, second[t]]]]]], z → RATS]]

Out[12]= or[equal[0, intersection[RATS, P[x]]],
           member[x, RATS], not[member[x, image[inverse[S], RATS]]]] == True

In[13]:= or[equal[0, intersection[RATS, P[x_]]],
           member[x_, RATS], not[member[x_, image[inverse[S], RATS]]]] := True
```

Lemma. Elimination of the remaining variable.

```
In[14]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, x, case[or[equal[0, intersection[t, P[x]]], member[x, t],
             not[member[x, image[inverse[S], t]]]]]], t → RATS]]

Out[14]= subclass[intersection[image[S, RATS], image[inverse[S], RATS]], RATS] == True

In[15]:= % /. Equal → SetDelayed
```

Theorem. A better variable-free rewrite rule.

```
In[16]:= SubstTest[and, subclass[u, v], subclass[v, u],
          {u -> intersection[image[S, RATS], image[inverse[S], RATS]], v -> RATS}]

Out[16]= equal[RATS, intersection[image[S, RATS], image[inverse[S], RATS]]] == True

In[17]:= intersection[image[S, RATS], image[inverse[S], RATS]] := RATS
```

Corollary.

```
In[18]:= Assoc[id[image[inverse[S], RATS]], id[image[S, RATS]], composite[S, id[RATS]]]

Out[18]= composite[id[image[inverse[S], RATS]], S, id[RATS]] == id[RATS]

In[19]:= composite[id[image[inverse[S], RATS]], S, id[RATS]] := id[RATS]
```

The intersection of all rationals is the singleton of the origin of the integer plane **Z** × **Z**.

```
In[20]:= A[RATS] == set[PAIR[id[omega], id[omega]]]

Out[20]= True
```

The rational hull of a sub-rational function is a rational number unless the sub-rational is a subset of **A[RATS]** .

Lemma.

```
In[21]:= Map[or[#, equal[first[u], id[omega]]] &,
           SubstTest[implies, and[member[u, x], subclass[x, y]],
            member[u, y], y -> cart[set[id[omega]], set[id[omega]]]]] // Reverse

Out[21]= or[equal[first[u], id[omega]], not[member[u, x]],
           not[subclass[x, cart[set[id[omega]], set[id[omega]]]]]] == True
```

```
In[22]:= (% /. {u → u_, x → x_}) /. Equal → SetDelayed
```

Theorem. If **u** is a member of a sub-rational function **x** and **first[u]** is not **id[ω]**, then the rational hull of **x** is a rational number.

```
In[23]:= Map[not, SubstTest[and, implies[p2, or[p5, p6]], implies[p3, not[p4]],
            implies[and[p1, p3], not[p5]], not[implies[and[p1, p2, p3], p6]],
            {p1 -> member[u, x], p2 -> member[x, image[inverse[S], RATS]],
             p3 -> not[equal[first[u], id[omega]]], p4 → equal[u, pair[id[omega], id[omega]]],
             p5 → subclass[x, A[RATS]], p6 -> member[hull[RATS, x], RATS]}]] // Reverse

Out[23]= or[equal[first[u], id[omega]], member[hull[RATS, x], RATS],
           not[member[u, x]], not[member[x, image[inverse[S], RATS]]]] == True

In[24]:= or[equal[first[u_], id[omega]], member[hull[RATS, x_], RATS],
           not[member[u_, x_]], not[member[x_, image[inverse[S], RATS]]]] := True
```

Corollary. Variant with **pair** instead of **first**.

```
In[25]:= Map[or[equal[u, id[omega]], #] &, SubstTest[or,
            equal[first[w], id[omega]], member[hull[RATS, x], RATS], not[member[w, x]],
            not[member[x, image[inverse[S], RATS]]], w → pair[u, v]]] // Reverse

Out[25]= or[equal[u, id[omega]], member[hull[RATS, x], RATS],
           not[member[x, image[inverse[S], RATS]]], not[member[pair[u, v], x]]] == True

In[26]:= or[equal[u_, id[omega]], member[hull[RATS, x_], RATS],
           not[member[x_, image[inverse[S], RATS]]], not[member[pair[u_, v_], x_]]] := True
```

The following theorem is useful for computing rational hulls.

Theorem. If **u ∈ x ⊂ y ∈ RATS** and **first[u]** is not **id[ω]**, then **y = hull[RATS, x]**.

```
In[27]:= Map[not, SubstTest[and, (*implies[and[p2,p3],p5],*) implies[and[p1, p4, p5], p6],
            implies[and[p2, p3], p7], (*implies[and[p3,p6,p7],p8],*)
            not[implies[and[p1, p2, p3, p4], p8]], {p1 -> member[u, x], p2 -> subclass[x, y],
             p3 -> member[y, RATS], p4 -> not[equal[first[u], id[omega]]],
             p5 → member[x, image[inverse[S], RATS]], p6 → member[hull[RATS, x], RATS],
             p7 → subclass[hull[RATS, x], y], p8 → equal[hull[RATS, x], y]}]] // Reverse

Out[27]= or[equal[y, hull[RATS, x]], equal[first[u], id[omega]],
           not[member[u, x]], not[member[y, RATS]], not[subclass[x, y]]] == True

In[28]:= or[equal[y_, hull[RATS, x_]], equal[first[u_], id[omega]],
           not[member[u_, x_]], not[member[y_, RATS]], not[subclass[x_, y_]]] := True
```

Corollary. Variant with **pair** instead of **first**.

```
In[29]:= Map[or[#, equal[u, id[omega]]] &,
           SubstTest[or, equal[y, hull[RATS, x]], equal[first[w], id[omega]], not[member[w, x]],
            not[member[y, RATS]], not[subclass[x, y]], w → pair[u, v]]] // Reverse

Out[29]= or[equal[u, id[omega]], equal[y, hull[RATS, x]],
           not[member[y, RATS]], not[member[pair[u, v], x]], not[subclass[x, y]]] == True

In[30]:= or[equal[u_, id[omega]], equal[y_, hull[RATS, x_]], not[member[y_, RATS]],
           not[member[pair[u_, v_], x_]], not[subclass[x_, y_]]] := True
```

It follows from this result that to compute the rational hull of a (non-trivial) sub-rational function **x** one needs to do two things. First, one needs to find a rational number **y** such that **x ⊂ y**. Second, one needs to find a point **pair[u, v] ∈ x** with **u** not equal to the zero integer **id[ω]**. Then **y** is the rational hull of **x**.

an example

Lemma. If **x** and **y** are integers, then **pair[x, y] ∈ inttimes[y] ∘ inverse[inttimes[x]]**.

```
In[31]:= SubstTest[implies, and[member[pair[x, t], composite[Id, v]],
            member[pair[t, y], composite[Id, u]]], member[pair[x, y], composite[u, v]],
           {t → plus[set[0]], u -> inttimes[y], v -> inverse[inttimes[x]]}] // Reverse

Out[31]= or[member[pair[x, y], composite[inttimes[y], inverse[inttimes[x]]]],
           not[member[x, Z]], not[member[y, Z]]] == True

In[32]:= or[member[pair[x_, y_], composite[inttimes[y_], inverse[inttimes[x_]]]],
           not[member[x_, Z]], not[member[y_, Z]]] := True
```

Lemma. (Application of the final theorem of the preceding section.)

```
In[33]:= SubstTest[or, equal[s, id[omega]], equal[v, hull[RATS, u]],
           not[member[v, RATS]], not[member[pair[s, t], u]], not[subclass[u, v]],
           {s → x, t → y, u → composite[inttimes[y], inverse[inttimes[x]]],
            v → composite[inverse[inttimes[x]], inttimes[y]]}] // Reverse

Out[33]= or[equal[x, id[omega]], equal[composite[inverse[inttimes[x]], inttimes[y]],
            hull[RATS, composite[inttimes[y], inverse[inttimes[x]]]]],
           not[member[x, Z]], not[member[y, Z]],
           not[member[pair[x, y], composite[inttimes[y], inverse[inttimes[x]]]]]] == True

In[34]:= (% /. {x → x_, y → y_}) /. Equal → SetDelayed
```

Theorem. (An example.) If **pair[x, y] ∈ domain[RATIO]**, then the rational hull of the composite of **inttimes[y]** and **inverse[inttimes[y]]** is their composite in the reverse order.

```
In[35]:= Map[not, SubstTest[and, implies[p1, p2], implies[and[p1, p2], p3],
            not[implies[p1, p3]], {p1 → member[pair[x, y], domain[RATIO]],
             p2 -> member[pair[x, y], composite[inttimes[y], inverse[inttimes[x]]]],
             p3 -> equal[composite[inverse[inttimes[x]], inttimes[y]],
               hull[RATS, composite[inttimes[y], inverse[inttimes[x]]]]]}]] // Reverse

Out[35]= or[equal[x, id[omega]], equal[composite[inverse[inttimes[x]], inttimes[y]],
            hull[RATS, composite[inttimes[y], inverse[inttimes[x]]]]],
          not[member[x, Z]], not[member[y, Z]]] == True

In[36]:= or[equal[composite[inverse[inttimes[x_]], inttimes[y_]],
            hull[RATS, composite[inttimes[y_], inverse[inttimes[x_]]]]],
          equal[id[omega], x_], not[member[x_, Z]], not[member[y_, Z]]] := True
```

A variable-free reformulation can be derived.

Lemma. A special vertical section rule for the function **RATIO**.

```
In[37]:= SubstTest[image, funpart[t], set[PAIR[x, y]], t → RATIO] // Reverse

Out[37]= image[RATIO, cart[set[x], set[y]]] ==
          intersection[image[V, intersection[omega, complement[fix[x]]]],
           image[V, intersection[Z, set[x]]], image[V, intersection[Z, set[y]]],
           set[composite[inverse[inttimes[x]], inttimes[y]]]]

In[38]:= image[RATIO, cart[set[x_], set[y_]]] :=
          intersection[image[V, intersection[omega, complement[fix[x]]]],
           image[V, intersection[Z, set[x]]], image[V, intersection[Z, set[y]]],
           set[composite[inverse[inttimes[x]], inttimes[y]]]]
```

Lemma. (Eliminating two variables. This takes a while.)

```
In[39]:= Map[equal[V, domain[#]] &,
          SubstTest[reify, t, case[implies[member[t, domain[funpart[v]]],
             equal[image[funpart[v], set[PAIR[first[t], second[t]]]],
              image[funpart[w], set[PAIR[first[t], second[t]]]]]]],
           {v → RATIO, w -> composite[HULL[RATS], COMPOSE, SWAP,
              cross[composite[INVERSE, INTTIMES], INTTIMES]]}]]

Out[39]= subclass[cart[intersection[Z, complement[set[id[omega]]]], Z],
          fix[composite[inverse[RATIO], HULL[RATS], COMPOSE, SWAP,
            cross[composite[INVERSE, INTTIMES], INTTIMES]]]] == True

In[40]:= % /. Equal → SetDelayed
```

Theorem.

```
In[41]:= SubstTest[subclass, domain[funpart[u]],
          fix[composite[inverse[funpart[u]], v]], {u → RATIO, v -> composite[
             HULL[RATS], COMPOSE, SWAP, cross[composite[INVERSE, INTTIMES], INTTIMES]]}]

Out[41]= subclass[RATIO, composite[HULL[RATS], COMPOSE,
            SWAP, cross[composite[INVERSE, INTTIMES], INTTIMES]]] == True
```

```
In[42]:= % /. Equal → SetDelayed
```

Theorem. A better rewrite rule.

```
In[43]:= SubstTest[equal, u, composite[funpart[v], id[domain[u]]],
           {u → RATIO, v -> composite[HULL[RATS], COMPOSE, SWAP,
              cross[composite[INVERSE, INTTIMES], INTTIMES]]}] // Reverse

Out[43]= equal[RATIO, composite[HULL[RATS], COMPOSE, SWAP, cross[
             composite[INVERSE, INTTIMES, id[complement[set[id[omega]]]]], INTTIMES]]] == True

In[44]:= composite[HULL[RATS], COMPOSE, SWAP, cross[
            composite[INVERSE, INTTIMES, id[complement[set[id[omega]]]]], INTTIMES]] := RATIO
```

Corollary. (Moving **SWAP** out.)

```
In[45]:= Assoc[composite[HULL[RATS], COMPOSE, SWAP],
          cross[composite[INVERSE, INTTIMES, id[complement[set[id[omega]]]]], INTTIMES], SWAP]

Out[45]= composite[HULL[RATS], COMPOSE,
           cross[INTTIMES, composite[INVERSE, INTTIMES, id[complement[set[id[omega]]]]]]] ==
          composite[RATIO, SWAP]

In[46]:= composite[HULL[RATS], COMPOSE,
           cross[INTTIMES, composite[INVERSE, INTTIMES, id[complement[set[id[omega]]]]]]] :=
          composite[RATIO, SWAP]
```