```
-- Copyright 1997 by Mike Stillman and Harry Tsai

---------------------------------
-- Inhomogeneous SAGBI bases ----
---------------------------------
sagbiEngine = (Gens, maxnloops, printlevel) -> (
     (R, G, S, RS, RStoS, Gmap, inGmap, J) := 8:null;
     (d, maxdeg, nloops, Pending) := 4:null;
     numnewsagbi := null;
     R = ring Gens;
     maxdeg = maxnloops;
     Pending = new MutableList from toList(maxdeg+1:{});
     RtoRS := null;
     RStoR := null;
     insertPending := (m) -> (
      -- append the entries of the one row matrix 'm' to Pending.
      i := 0;
      while i < numgens source m do (
           f := m_(0,i);
           e := (degree f)_0;
           Pending#e = append(Pending#e, f);
           i = i+1;
           ));
     lowestDegree := () -> (
      -- returns maxdeg+1 if Pending list is empty, otherwise
      -- returns the smallest non-empty strictly positive degree.
      i := 0;
      while i <= maxdeg and Pending#i === {} do i=i+1;
      i);
     appendToBasis := (m) -> (
      R := ring m;
      M := monoid R;
      G = G | m;
      nR := numgens R;
      nG := numgens source G;
      newOrder := appendElimination(M.Options.MonomialOrder, nR, nG);
      k := coefficientRing R;
      N := monoid [
           Variables => nR + nG,
           Degrees=>join(degrees source vars R, degrees source G),
           MonomialOrder => newOrder];
      RS = k N;
      RtoRS = map(RS,R,(vars RS)_{0..nR-1});
      RStoS = map(RS,RS, matrix {toList(nR:0_RS)} |
           (vars RS)_{nR .. nR+nG-1});
      J = ideal((vars RS)_{nR..nR+nG-1}-RtoRS(leadTerm G));
      Gmap = map(RS,RS,(vars RS)_{0..nR-1} | RtoRS(G));
      RStoR = map(R,RS,(vars R) | matrix {toList(nG:0_R)});
      );
     grabLowestDegree := () -> (
      -- assumes: lowest degree pending list is already autosubducted.
      -- this row reduces this list, placing all of the
      -- entries back into Pending, but then appends the lowest
      -- degree part into the basis.
```

Inputs: Gens = |x+y  xy  xy^2| matrix
max n loops = limit
print level (default 0)
remaining are null

Set R to ring of gens

max deg = limit

Mutable list of length at most max deg + 1
displays as limit+1 {}'s.
These are bins, one for each degree

Not local

m is a matrix

only take entries in first row. Assume m has at most numgens columns.

f is only get degree of f

Not local

Not local

add f to the correct list

Searches through Pending until you get a list which isn't empty or the for loop stops

Not local

Not local

R is the ring containing m

monoid generated by gens of R

m is a matrix of a single element

I can multiply generators, but not add them.

Coefficients

number of Gens of R

Number of columns of G

Adding an elimination order. Not sure what this does, eliminate the first n G elts for nG+nR? What does append do?

Coeff ring.

Put all the vars together eliminate one set.

Polynomial ring with coefficients k.

Gens of R maps to $p_0, \dots, p_{nR-1}$

Generators map to 0.

Syzygies (relations between higher order elements from Groebner basis)

Gens, then substitute gens into map all but generators to 0
```
```

```
e := lowestDegree();
if e <= maxdeg then (
     trr := timing rowReduce(matrix{Pending#e}, e);
     timerr := trr#0;
     if printlevel > 0 then
       << "      rowred  done in " << timerr << " seconds" << endl;
     m := trr#1;
     Pending#e = {};
     insertPending m;
     e = lowestDegree();
     numnewsagbi = #Pending#e;
     timeapp := (timing appendToBasis matrix{Pending#e})#0;
     if printlevel > 0 then
       << "      append  done in " << timeapp << " seconds" << endl;
     Pending#e = {};
     );
e);
G = matrix(R, {{}});
Gensmaxdeg := (max degrees source Gens)_0;
Gens = compress submatrixBelowDegree(Gens, maxdeg+1);
insertPending Gens;
Pending#0 = {};
d = grabLowestDegree();  -- initializes G
d = d+1;
nloops = d;
isdone := false;
while nloops <= maxnloops and not isdone do (
  ttotal := timing(
 nloops = nloops+1;
 if printlevel > 0 then
   << "--- degree " << d << " ---" << endl;
     tgbJ := timing gb(J, DegreeLimit=>d);
 gbJ := tgbJ#1;
 if printlevel > 0 then
   << "      gb comp done in " << tgbJ#0 << " seconds" << endl;
 -- spairs = time mingens ideal selectInSubring(1, gens gbJ);
 spairs := submatrixByDegrees(selectInSubring(1, gens gbJ), d);
 if printlevel > 1 then
   << "spairs = " << transpose spairs << endl;
 tGmap := timing Gmap(spairs);
 spairs = tGmap#1;
 if printlevel > 0 then
   << "      Gmap    done in " << tGmap#0 << " seconds" << endl;
 if Pending#d != {} then (
      newgens := RtoRS(matrix{Pending#d});
      spairs = spairs | newgens;
      Pending#d = {};);
 tsub := timing map(RS,rawSubduction(rawMonoidNumberOfBlocks raw monoid R, raw
  spairs, raw Gmap, raw gbJ));
 if printlevel > 0 then
   << "      subduct done in " << tsub#0 << " seconds" << endl;
     tRS := timing compress RStoR(tsub#1);
 newguys := tRS#1;
 if printlevel > 0 then
```

```
  << "    RStoR   done in " << tRS#0 << " seconds" << endl;
if numgens source newguys > 0
then (
     if printlevel > 0 then
          << "    GENERATORS ADDED!" << endl;
     insertPending newguys;
     d = grabLowestDegree();
     if printlevel > 0 then
       << "    " << numnewsagbi << " NEW GENERATORS!" << endl;
     )
else (
     numnewsagbi = 0;
     ngens := sum apply(toList Pending,i -> #i);
     if ngens === 0 and gbDone gbJ and d>Gensmaxdeg then (
         isdone = true;
     if printlevel > 0 then
       << "    SAGBI basis is FINITE!" << endl;
     );
    );
  );
if printlevel > 0 then (
  << "    deg " << d << "  done in " << ttotal#0 << " seconds" << endl;
  );
d=d+1;
);
G)
```

If there are new ones.

Insert & get lowest degree.

No new things to add.

number of pending

nothing left

d = Integer (larger than largest generator)

next degree