Multinomial Division Preliminaries

Dylan Zwick

Tuesday, June 24th, 2008

Yesterday we wrote a procedure that divides one polynomial in one variable by another polynomial. Today, we're going to extend this to a procedure that divides a polynomial in many variables by a set of polynomials.

The pseudocode for this algorithm is on page 64 of the textbook. In order to write this program we've got to have a procedure that picks out the leading term from a multivariable polynomial based upon some monomial order.

I've written a procedure that does this (I think and hope). The procedure requires some earlier procedures, so in order to use this procedure (called LT) you should enter in each of the procedures at the end of this handout in order. Then, assuming that you entered everything in correctly, you should have a procedure that calculates the leading term of a polynomial based upon a monomial ordering.

In order to run the leading term procedure you need to enter three things: a polynomial, a list of variables, and a list of weights. The polynomial and the list of variables is straightforward, but the weights require some explanation.

Basically, a list of weights is a list of lists. For example, here would be a list of weights:

$$L1 := [[1,1,1],[1,0,0],[0,1,0],[0,0,1]];$$

This would be a list of weights that would implement $grlex$ ordering in three variables. If you combined this with the variable list $[x,y,z]$ it would implement $grlex$ ordering with a variable ordering $x > y > z$. The way this works is that it looks at the first entry in the list for the weight to give

to the power of each monomial. The second entry in the list is the weight to give to each monomial in case of a tie. So, in this case first each power would be given equal weight, and in the caes of a tie then the x variable would be the only one investigated, then in the case of a tie there the y variable would be the only one investigated, and in the cae of a tie there the z variable would be the only one investigated. In this manner you can implement any ordering described in the book. If this is confusing please ask me about it.

So, for example, if you entered:

$$LT(4 * x^2 * y^3 * z + y^7 - 2 * x^3 * z, [x, y, z], [[1, 1, 1], [1, 0, 0], [0, 1, 0], [0, 0, 1]]);$$

the output would be the leading term according to $grlex$ with $x > y > z$, which in this case would be y^7. If, on the other hand, you entered:

$$LT(4 * x^2 * y^3 * z + y^7 - 2 * x^3 * z, [x, y, z], [[1, 0, 0], [0, 1, 0], [0, 0, 1]]);$$

the output would be the leading term according to lex with $x > y > z$, which in this case would be $-2 * x^3 * z$.

You should be able to use this LT procedure, along with the pseudocode in the textbook and the example of the program you wrote yesterday, to write a procedure for doing multinomial division. Pleaes write this procedure today.

1 Code for Creating the LT Procedure

```
MONDEGREE := proc(f,variables,weights) local deg, i;
if not type(f,polynom) then
ERROR('First argument must be a monomial.');
elif type(f,'+') then
ERROR('First argument must be a monomial.');
fi;
if nops(variables) <> nops(weights) then
ERROR('Must have the same number of weights as variables'); fi;
i := 0; deg := 0;
while i < nops(variables)
do
```

```
deg := deg + degree(f,variables[i+1])*weights[i+1];
i := i+1;
od;
deg;
end;



DEGREECOMP := proc(m1,L1,m2,L2) local i, best, notbestfound;
if nops(L1) <> nops(L2) then
ERROR(‘Your weight score vectors must be of the same length‘) fi;
i := 1; best := m1; notbestfound := true;
while notbestfound
do
if L1[i] > L2[i] then best := m1; notbestfound := false;
elif L1[i] < L2[i] then best := m2; notbestfound := false;
fi;
i := i+1;
if i > nops(L1) then notbestfound := false fi;
od;
best;
end;



CALCWEIGHTS := proc(f,variables,weights) local scores, i;
i := 1; scores := array(1..nops(weights));
while i < nops(weights) + 1
do
scores[i] := MONDEGREE(f,variables,weights[i]);
i := i+1;
od;
[f,eval(scores)];
end;



LT := proc(f,variables,weights) local i, leadterm;
```

```
if not type(f,polynom) then
ERROR(‘First argument must be a polynomial.‘);
elif not type(f,‘+‘) then leadterm := f;
else
i := 2;
leadterm := op(1,f);
while i < nops(f) + 1
do
leadterm
:=DEGREECOMP(leadterm,CALCWEIGHTS(leadterm,variables,weights)[2],
op(i,f),CALCWEIGHTS(op(i,f),variables,weights)[2]);
i := i+1;
od;
fi;
leadterm;
end;
```