

## A draft Manual for the package Ell2

Main new procedures:

$E\_OeqS(Q, vxa, va)$  and  $E\_OeqW(Q, vxa, va)$  (there should be no bug for these two procedures)

$E\_OgeS(Q, vxa, va)$  and  $E\_OgeW(Q, vxa, va)$  (you are welcome to report bug for these two procedures)

To use the Ell2-package, first type

read "D://GarsiaXin/Ell2.mpl":

Then the following command will compute the constant term.

$$G := E\_OeqW(F, [vxa], [va]);$$

Here  $F$  is the crude generating function,  $vxa$  will be a sequence like  $x_1, \dots, x_n, a_1, \dots, a_k$ , and  $va$  will be  $a_1, a_2, \dots, a_k$ , a sequence of variable to be eliminated. Then  $G$  will be the result from  $F$  after eliminating all of the variables  $a_i$ .

The working field is  $\mathbb{Q}\langle\langle a_k, \dots, a_1, x_n, \dots, x_1 \rangle\rangle$ , in other words, the variable comes earlier in our Maple command is much smaller than the variable comes later. Unlike the old version, we include the  $a$ 's in  $vxa$  when defining the total ordering, and then we make a choice of  $a_i$  to eliminate. The old program uses  $E\_Oge(F, [x_1, \dots, x_n], [a_1, \dots, a_k])$  to compute the constant term of

$F(x_1, \dots, x_n, a_1, \dots, a_k)$  in the field  $\mathbb{Q}\langle\langle a_1, \dots, a_k, x_1, \dots, x_n \rangle\rangle$ , eliminating  $a_1, \dots, a_k$  in a fixed order.

There are two ideas implemented in the new package to reduce the computation time. In order to explain the idea, we need to fix the inputs: the crude generating function  $F$ , the sequence  $vxa$  defining the working field  $\mathbb{Q}\langle\langle a_k, \dots, a_1, x_n, \dots, x_1 \rangle\rangle$ , and the sequence  $va$  of the variables to be eliminated. The result will be

$$G(x_1, \dots, x_n) = \underset{a_1, \dots, a_k}{\text{CT}} F(x_1, \dots, x_n, a_1, \dots, a_k).$$

1. When taking the constant term in a specified variable, we decide whether to compute those fractional parts with contribution or those fractional parts without contribution.
2. We decide which variable to eliminate first.

The second is from a well-known fact: the order of the variables to eliminate can make a difference for the computational time. To take advantage from it, we use the fact that the number of rational functions produced by eliminating  $\lambda_i$  is equal to the number  $cf(\lambda_i)$  of factors in the denominator of  $F$  that have contributions with respect to  $\lambda_i$ . If  $cf(\lambda_{i_0})$  is the smallest among all the  $cf(\lambda_i)$ , then we shall eliminate the  $\lambda_{i_0}$  first. Note that This way does not guarantee the best result.

The first is better explained by an example.

Suppose we are eliminating  $\lambda$  and the partial fraction of  $F$  is

$$F(\lambda) = p(\lambda) + \sum_{i=1}^8 \frac{A_i}{1 - \lambda x_i} + \frac{A_9}{\lambda - x_9},$$

where  $p(\lambda)$  is a polynomial, and  $A_i$  are constants with respect to  $\lambda$ . Then the constant term of  $F(\lambda)$  in  $\lambda$  is equal to  $p(0) + A_1 + \dots + A_8$ , since

$$\text{CT}_{\lambda} \frac{A_9}{\lambda - x_9} = \text{CT}_{\lambda} \frac{A_9}{\lambda(1 - x_9/\lambda)} = \text{CT}_{\lambda} A_9 \sum_{n \geq 0} x_9^n / \lambda^{n+1} = 0.$$

If we want to use the above formula, we need to compute  $p(0), A_1, \dots, A_8$ . However, we have  $F(0) = p(0) + A_1 + \dots + A_8 - A_9/x_9$  by simply substituting  $\lambda$  by 0. Therefore

$$\text{CT}_{\lambda} F(\lambda) = F(0) + A_9/x_9.$$

Using this alternative formula, we need only to compute  $A_9$ , while  $F(0)$  is easy to evaluate.

When implementing this idea, we first search if the denominator of  $F$  divides  $\lambda$ . If not, and the number of contribution terms is more than the number of non-contribution terms then we apply the alternative formula.

Our Ell2 package searches for the smallest number, with the corresponding variable  $\lambda_0$ , among all of the choices of the variables and the numbers of contribution terms and non-contribution terms (with the described condition). Then we take the constant term. The result will be a sum. Then we shift the variables to make  $a_k$  disappear. This procedure is done by the command  $E\_OeqS(F, [vx], [va])$ , where  $S$  stands for “single”. The command  $E\_OeqW(F, [vx], [va])$  is an iteration of the  $E\_OeqS$  command. The  $W$  stands for “whole”.