A QUICK DIP INTO MATHEMATICA

ULRICH KRÄHMER

Abstract. These are notes written during the process of learning how to use mathematica. It ended abruptly when I got the (maybe wrong) impression that it can not (yet) do what I really need for my research, but sitll it is a helpful tool for many other things.

Contents

1. The afternoon

1.1. **Introduction.** OK, so here I am, not knowing anything at all about computer algebra systems. In three weeks' time I have to teach students how to use mathematica, and on that occasion I want to learn also how to use it for simplfying expressions in a noncommutative algebra defined in terms of generators and relations.

1.2. **Compute 2+2.** Most of you will work with the graphical user interface. But you can aslo work in shell, my mathematica is for example only installed on a university server and I use it in a shell by sshing there and typing

```
/maths/mathematica7/Executables/math
```

which produces

```
Mathematica 7.0 for Linux x86 (64-bit)
Copyright 1988-2008 Wolfram Research, Inc.
```

```
In[1]:=
```

Now I can type in a command. When I'm done I hit ENTER and the command will be executed. If you use the graphical front end you need SHIFT plus ENTER instead. For example, if I type 2+2 and hit ENTER the whole screen looks like this:

```
Mathematica 7.0 for Linux x86 (64-bit)
Copyright 1988-2008 Wolfram Research, Inc.

In[1]:= 2+2

Out[1]= 4
```

and I can now type the second command.

1.3. **Counting the primes less than 50.** Now I googled a bit for examples and tutorials. I spare you banalities, let us go right away into something more substantial but then study it step by step. Type (or better copy and paste) in one go the following after the new input prompt In[2]:= and hit ENTER.

```
sieve[n_]:=
    Module[{count, result, i},
      result = Range[n];
      result[[1]] = 0; i = 2;
      While[i^2 <= n,
        If[result[[i]] != 0,
          For[j = 2*i, j <= n, j += i, result[[j]] = 0]];
        i++];
      count = 0;
      For[k = 1, k <= n, k++,
        If[result[[k]] > 0, count++]];
      count]
```

There should be no ouput, just In[3]:=. Now type

```
sieve[50]
```

and evaluate (meaning hit ENTER). You should get

```
Out[3]= 15
```

which is the number of primes less or equal than 50 - the whole code has implemented the sieve of Eratosthenes.

Let us analyse this step by step.

```
sieve[n_]:=
```

defines a function called “sieve”. It has one variable “n”, the lower dash is absolutely crucial and tells mathematica this is a variable I think. “Module” defines local variables “count”, “result”, “i”. If later in this mathematica session you introduce a constant named “count” with some value, mathematica will be clever enough not to be confused when you call “sieve” afterwards. Note that the square bracket of “Module” is closed only at the very end of the whole algorithm.

“{count, result, i}” should be clear. Then after a comma comes the actual algrotihm starting with “result = Range[n];” which says that “result” should be the list of natural numbers from 1 to n. The semicolon says that here a command ends and should be carried out.

Next comes “ result[[1]] = 0;” - well, “result” is by its definition given in the previous step an array, i.e. a list of things, and this here addresses its first entry and sets it to zero. To check I understood this correctly I do some experiments. The glorious VPN of my university has kicked me out in the meantime so I have to restart everything, hence it starts with In[1] again.

```
In[1]:= testlist=Range[5]

Out[1]= {1, 2, 3, 4, 5}

In[2]:= testlist[[1]]

Out[2]= 1

In[3]:= testlist[[4]]

Out[3]= 4

In[4]:= testlist[[2]]=3
```

```
Out[4]= 3

In[5]:= testlist

Out[5]= {1, 3, 3, 4, 5}
```

I hope this array concept is clear now, so let us get back to our "sieve" function. "i=2" should be clear.

Now comes a huge "While"-loop. While something is true, here $i^2 \leq n$, she (mathematica) is doing something, namely

```
If[result[[i]] != 0,
          For[j = 2*i, j <= n, j += i,
 result[[j]] = 0]];
```

which we discuss in a second. After mathematica has carried this out "i++" adds 1 to the variable "i" and she checks again whether $i^2 \leq n$ is still true and so on.

In the nested "If" and "For"-thing, "!=" means $\neq$ and "j += i" means "replace j by j+i". Things should be clear now as long as you know the mathematics behind the sieve of Eratosthenes, for this see

http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

So far the algorithm has produced a list of the integers from 1 to n and then replaced all integers that are not prime by zeroes. Now mathematica counts with

```
     count = 0;
      For[k = 1, k <= n, k++,
        If[result[[k]] > 0, count++]];
```

these zeroes, and the last line

```
   count]
```

makes her to return the value of "count" as the value of the whole function "sieve". Recall that the square bracket just closes the one from "Module[".

I guess we are done with this example which I have taken from the very good and one (long) page long tutorial

```
http://www.outbacksoftware.com/mathematica/
mathematica-intro.html
```

which contains also some other excellent and instructive examples. Furthermore, I have used

```
http://reference.wolfram.com/
search.html?query=!%3D&collection=reference&lang=en
```

to find out what a single command does if it was not clear.

1.4. **NCAlgebra.**

Now I feel comfortable with the basics so I searched the web for the things I want to do. The first I found was the NCAlgebra suite, see

```
http://www.math.ucsd.edu/~ncalg/
```

You have to download this package from the above page. Details on how to do this, how to install it, and how to use it can be found in a 344 pages documentation to be downloaded there as well.

In my case I downloaded "NC2010.tgz", copied it into my home directory on the uni server, untared it which produced a folder "NC/", and then I could restart mathematica and things worked fine.

So I assume now this installation is complete and you have started mathematica. Right now my funtasting VPN has again stopped working so I anyway have to restart. Then I load the package by saying

```
In[1]:= << NC`
```

which produces (hit ENTER now) the output

```
You are using the version of NCAlgebra which is found in:
  /home/staff1/ukraehmer/NC.
You can now use "<< NCAlgebra`" to load
  NCAlgebra or
"<< NCGB`" to load NCGB.
```

I am interested in the NCGB package which does noncommutative Gröbner bases, so I say

```
In[2]:= << NCGB‘
```

which yields a longish output. As you guess, “<<” reads a file, I will talk about this later again.

Now follow the following steps which is a simplified example from the NCGB documentation. I always list input and the output produced when evaluating the input.

```
In[3]:=  SetNonCommutative[x,y]

Out[3]= {False, False}
```

Mathematica now knows that “x” and “y” are some noncommuting variables.

```
In[4]:=  SetMonomialOrder[{x,y}]
```

If you know something about Gröbner bases, you have to give some ordering to your variables, and this is done here. I will not explain the theory behind this, but later when you deal with complicated relations it might matter which ordering you use, the whole algorithm might not work with some orderings.

Now comes the first real computation:

```
In[5]:=  NCSimplifyAll[{x**y**x-3*y**x**x},{x**y-4**y**x},4]
```

There is some longish output, and then, at the end

```
         x ** x ** y
Out[6]= {-----------}
             16
```

Impressive, isn’t it? She understands that $xy = 4yx$ and then simplifies $xyx - 3yx^2$. The 4 at the very end of the input tells her how hard to try to simplify things, it seems she would otherwise get lost in more difficult ones, its some depth of search. Feel free to check the manual of NCAlgebra if you want to know it, I personally don’t but I guess I have to keep the number large enough in bigger computations.

So far, I have now lost my VPN connection for the third time and should have left anyway ten minutes ago to fetch my son from his piano lessons, so I call this a day and will continue tomorrow.

2. The evening

There was an hour left before bed time, so I thought I'd finish the example from the NCGB documentation. First of all I should say that I noticed that above I made a typo, I meant to use "**" everywhere but once just wrote "*" and it seems mathematica can cope with this. "*" is the usual commutative multiplication that you will use in standard mathematica without the NCAlgebra package when dealing with numbers, polynomials etc., "**" is the new one defined by the noncommutative algebra package.

Anyway, what I really want to say is: I played a bit and read this and that and now I think NCAlgebra is not really good for what I want to do which is for example dealing with relations that involve a parameter q so that at the end the objects one handles are quotients of free algebras in finitely many generators but with coefficients in $\mathbb{Q}(q)$, the field of rationl functions in q with values in the rational numbers. Hence I guess I forget about matheatica very soon again unless I have to use it for teaching.