



Re: generate all possible math expr of one term

{ $1/(x^2*y^2)$ ,  $1/(x*y^2)$ ,  $x/y^2$ ,  $1/(x*y)$ ,  $x/y$ ,  $x^2/y$ ,  $x*y$ ,  $x^2*y$ ,  $x^2*y^2$ ,  $\text{Cos}[x]/y^2$ ,  $\text{Cos}[x]/y$ ,  $\text{Cos}[x]/(x*y)$ ,  $(x*\text{Cos}[x])/y$ ,  $y*\text{Cos}[x]$ ,  $(y*\text{Cos}[x])/x$ ,  $x*y*\text{Cos}[x]$ ,  $y^2*\text{Cos}[x]$ ,  $\text{Cos}[x]*\text{Cos}[y]$ ,  $\text{Cot}[x]/y^2$ ,  $\text{Cot}[x]/(x*y^2)$ }

For a gallery of selection plots of these equations, see  
[http://xahlee.org/MathGraphicsGallery\\_dir/dense/dense1.html](http://xahlee.org/MathGraphicsGallery_dir/dense/dense1.html)

The above i wrote in 2002. If there are requests, i'll translate the above code into emacs lisp. The result lisp expression should match Mathematica's, token for token. (the code make a lot use of nested lambda and or apply and or map) If you are interested, you could translate the above into lisp too, it's not difficult (though the number of lines will increase maybe 10 fold. And if Common Lisp doesn't have combinatorics library providing KSubsets, and also since CL doesn't have Outer, so the above in CL might be few hundred lines). (see here for a example of how to:[http://xahlee.org/UnixResource\\_dir/writ/notations.html](http://xahlee.org/UnixResource_dir/writ/notations.html))

PS as a after-thought, i decided to post this to perl, python, and java too. This will take about the same number of lines in perl as in Common Lisp. Probably llightly more in Python due to syntax. In Java, it will be one million lines.

Gratuitous poem of the day:

in the climb to geekdom,  
you have few rungs to catch,  
before you see my ass.

Xah Lee, 2005

Xah  
x...@xxxxxxxxxxx  
<http://xahlee.org/>

The thing is stuff like that is relatively in Lisp. Let's see your Emacs Lisp version!

In Common Lisp you get the benefit of nicer code.

To give a hint:

```
(defun example ()  
  (pprint  
   (cross-product '(* +)  
   (cons 'x (cross-product '(sin cos tan /) '(x)))))
```

Re: generate all possible math expr of one term

Re: generate all possible math expr of one term

(cons 'y (cross-product '(sin cos tan /)'(y))))))

CL-USER 33 > (time (example))

Timing the evaluation of (EXAMPLE)

((+ X Y)  
(+ X (/ Y))  
(+ X (TAN Y))  
(+ X (COS Y))  
(+ X (SIN Y))  
(+ (/ X) Y)  
(+ (/ X) (/ Y))  
(+ (/ X) (TAN Y))  
(+ (/ X) (COS Y))  
(+ (/ X) (SIN Y))  
(+ (TAN X) Y)  
(+ (TAN X) (/ Y))  
(+ (TAN X) (TAN Y))  
(+ (TAN X) (COS Y))  
(+ (TAN X) (SIN Y))  
(+ (COS X) Y)  
(+ (COS X) (/ Y))  
(+ (COS X) (TAN Y))  
(+ (COS X) (COS Y))  
(+ (COS X) (SIN Y))  
(+ (SIN X) Y)  
(+ (SIN X) (/ Y))  
(+ (SIN X) (TAN Y))  
(+ (SIN X) (COS Y))  
(+ (SIN X) (SIN Y))  
(\* X Y)  
(\* X (/ Y))  
(\* X (TAN Y))  
(\* X (COS Y))  
(\* X (SIN Y))  
(\* (/ X) Y)  
(\* (/ X) (/ Y))  
(\* (/ X) (TAN Y))  
(\* (/ X) (COS Y))  
(\* (/ X) (SIN Y))  
(\* (TAN X) Y)  
(\* (TAN X) (/ Y))  
(\* (TAN X) (TAN Y))  
(\* (TAN X) (COS Y))  
(\* (TAN X) (SIN Y))  
(\* (COS X) Y)  
(\* (COS X) (/ Y))  
(\* (COS X) (TAN Y))  
(\* (COS X) (COS Y))  
(\* (COS X) (SIN Y))  
(\* (SIN X) Y)

Re: generate all possible math expr of one term

Re: generate all possible math expr of one term

```
(* (SIN X) (/ Y))  
(* (SIN X) (TAN Y))  
(* (SIN X) (COS Y))  
(* (SIN X) (SIN Y))
```

User time = 0.002

That should be easy to extend to get a solution for your problem.

Then you map a simplifier over the result. (mapcar 'simplify result)  
and you get a nice list of simplified forms.

With some added recursion you can change the depth of the term  
generation.

.