Computer Algebra Independent Integration Tests

Nasser M. Abbasi

December 16, 2018 Compiled on December 16, 2018 at 10:28am [public]

These reports and the web pages themselves were written in LaTeX using TeXLive distribution and compiled to HTML using TeX4ht.

FULL CAS integration tests

These reports show the result of using full CAS test suite on different CAS's at same time. This compares how one CAS performes against another.

The test suite used is the full Rubi CAS intergation tests.

1. Completed August 2, 2018 Started June 6, 2018. This build is now complete.

   CAS systems used are : Rubi 4.15.2, Mathematica 11.3, Maple 2018, Maxima 5.41, Fricas 1.3.3, Sympy 1.1.1 and Giac/XCAS 1.4.9.

   Number of integrals [69,522].

2. In progress. Updated December 16, 2018 Number of tests completed [28,425]

   This report which includes the same CAS systems as above, but with the addition of Rubi in Sympy implemetation. This is a port of Rubi to Sympy.

   Sympy 1.3 was used in this test for both Sympy itself and for Rubi in Sympy.

LITE CAS integration tests

These reports show the result of using same test suite on different versions of same CAS. Only Maple, Mathematica and Rubi are used. This is for regression testing to find how different version of the same CAS improved over the years. The test suite used is the lite version of the full CAS intergation tests. It contains smaller number of integrals compared to the full test suite used above.

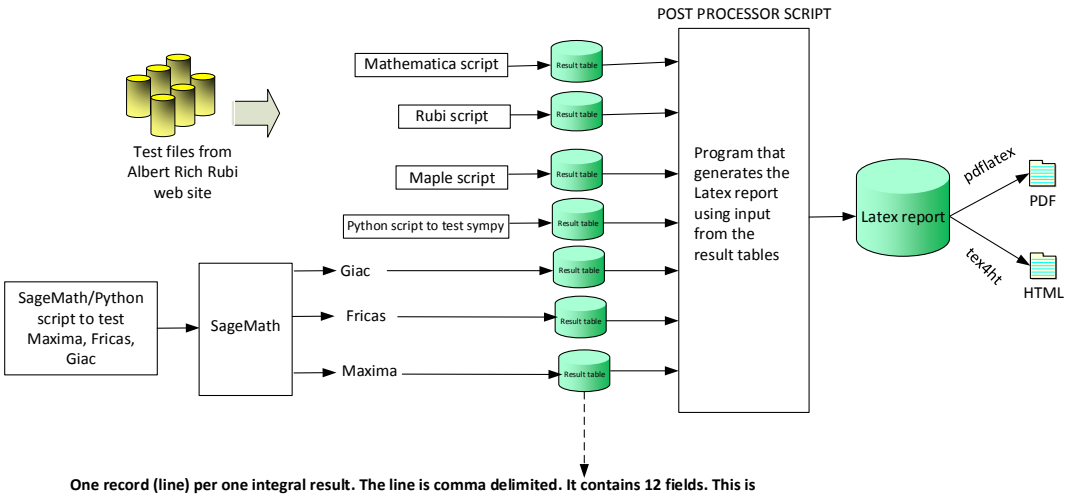
1. Mathematica This zip file contains the raw input tests used which was obtained from Rubi web site.

2. Rubi Uses same zip file as above for input.

3. Maple This zip file contains the raw input tests used which was obtained from Rubi web site.

Note on build system

The following diagram gives a high level view of the current test build system.



One record (line) per one integral result. The line is comma delimited. It contains 12 fields. This is description of each record (line)

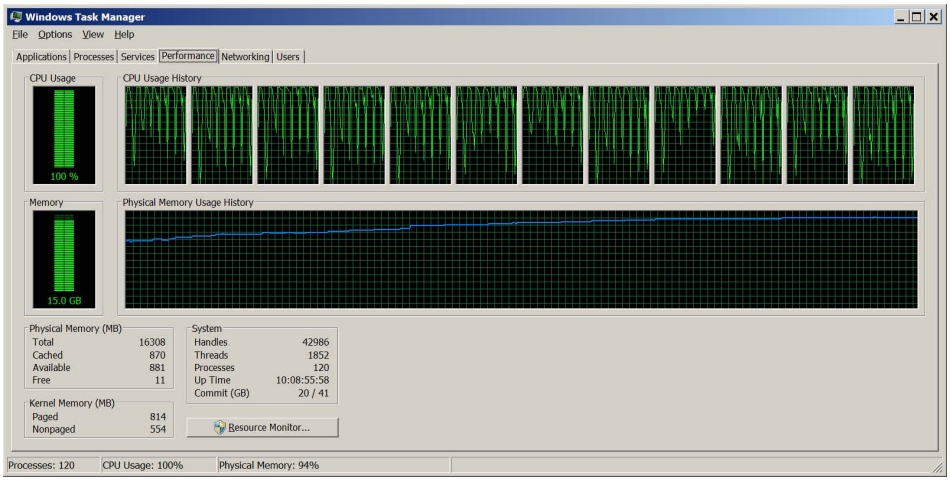
1. integer, the problem number.
2. integer. 0 or 1 for failed or passed. (this is not the grade field)
3. integer. Leaf size of result.
4. integer. Leaf size of the optimal antiderivative.
5. number. CPU time used to solve this integral. 0 if failed.
6. string. The integral in Latex format
7. string. The input used in CAS own syntx.
8. string. The result (antiderivative) produced by CAS in Latex format
9. string. The optimal antiderivative in Latex format.
10. integer. 0 or 1. Indicates if problem has known antiderivative or not
11. String. The result (antiderivative) in CAS own syntax.
12. String. The grade of the antiderivative. Can be “A”, “B”, “C”, or “F”

High level overview of the CAS independent integration test build system

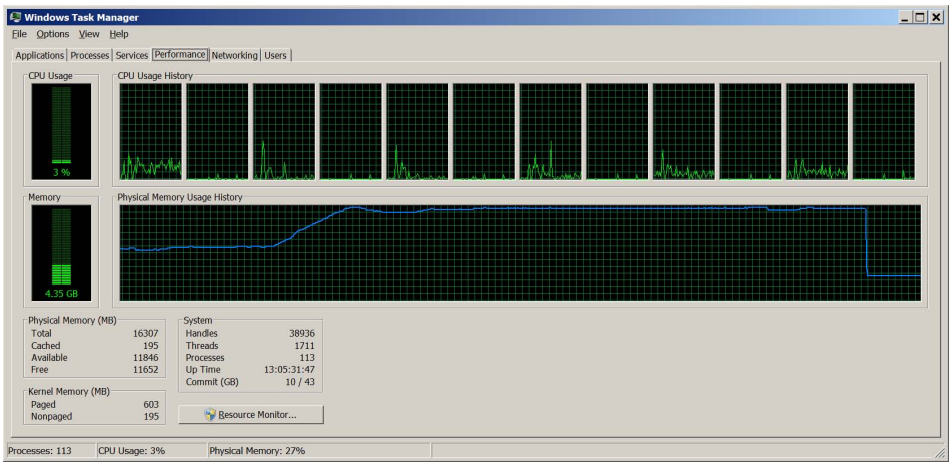
Nasser M. Abbasi
June 22, 2018

My PC during running the tests

I really need a faster PC with much more RAM !



This below shows example of CAS suddenly consuming all RAM in PC, and I had to terminate the process, since it did not time out as instructed, and just hanged.



my cheat sheet notes

Fricas

To use Fricas online, go to `http://axiom-wiki.newsynthesis.org/FriCASIntegration`
Go to bottom of page, type

```
\begin{axiom}
setSimplifyDenomsFlag(true)
integrate(sin(x),x)
\end{axiom}
```

Then click the preview button

to send email to supprt group use fricas-devel@googlegroups.com

To download go to `https://sourceforge.net/projects/fricas`

To install, see `http://fricas.sourceforge.net/doc/INSTALL.txt`

To build

```
#install sbcl
#sudo apt-get install sbcl #do not use this. bug
sudo apt-get install clisp #This lisp works
sudo apt-get install dvipng
sudo apt-get install auctex
bzip2 -dk fricas-1.3.2-full.tar.bz2
tar -xf fricas-1.3.2-full.tar
```

Now build it

```
make clean
fricas-1.3.2>./configure --with-lisp=/usr/bin/clisp
checking build system type... x86_64-linux-gnu
checking host system type... x86_64-linux-gnu
checking target system type... x86_64-linux-gnu
checking for make... make
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for a BSD-compatible install... /usr/bin/install -c
checking for touch... touch
checking for mktemp... mktemp
checking for gawk... gawk
checking for gtar... no
checking for tar... tar
checking for ranlib... ranlib
checking for ar... ar
checking for latex... /usr/local/texlive/2017/bin/x86_64-linux/latex
checking for makeindex... makeindex
checking PREGENERATED... "/home/me/data/fricas/fricas-1.3.2/pre-generated"
checking for sbcl... /usr/bin/sbcl
checking Lisp implementation... This is SBCL 1.3.1.debian, an implementatio
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no warranty.
It is mostly in the public domain; some portions are provided under
BSD-style licenses.  See the CREDITS and COPYING files in the
distribution for more information.
* sbcl
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
```

```
checking for stdint.h... yes
checking for unistd.h... yes
fasl
checking dirent.h usability... yes
checking dirent.h presence... yes
checking for dirent.h... yes
checking whether closedir is declared... yes
checking whether opendir is declared... yes
checking whether readdir is declared... yes
checking whether dirfd is declared... yes
checking whether fchdir is declared... yes
checking signal.h usability... yes
checking signal.h presence... yes
checking for signal.h... yes
checking whether sigaction is declared... yes
checking for sys/stat.h... (cached) yes
checking for unistd.h... (cached) yes
checking whether getuid is declared... yes
checking whether geteuid is declared... yes
checking whether getgid is declared... yes
checking whether getegid is declared... yes
checking whether kill is declared... yes
checking sys/socket.h usability... yes
checking sys/socket.h presence... yes
checking for sys/socket.h... yes
checking util.h usability... no
checking util.h presence... no
checking for util.h... no
checking pty.h usability... yes
checking pty.h presence... yes
checking for pty.h... yes
checking whether openpty is declared... yes
checking for openpty in -lc... no
checking for openpty in -lutil... yes
checking sys/wait.h usability... yes
checking sys/wait.h presence... yes
checking for sys/wait.h... yes
checking whether wait is declared... yes
checking whether fork is declared... yes
checking for X... no
configure: The Graphics and HyperDoc components are disabled.
configure: WARNING: Aldor interface will not be built.
configure: creating ./config.status
config.status: creating src/clef/Makefile
config.status: creating src/sman/Makefile
config.status: creating src/hyper/Makefile
config.status: creating src/doc/Makefile
config.status: creating Makefile
```

```
config.status: creating src/Makefile
config.status: creating src/lib/Makefile
config.status: creating src/lisp/Makefile
config.status: creating src/boot/Makefile
config.status: creating src/interp/Makefile
config.status: creating src/algebra/Makefile
config.status: creating src/input/Makefile
config.status: creating src/etc/Makefile
config.status: creating src/aldor/Makefile
config.status: creating src/aldor/Makefile2
config.status: creating src/aldor/Makefile3
config.status: creating contrib/emacs/Makefile
config.status: creating config/fricas_c_macros.h
extracting list of SPAD type definitions
Type 'make' (without quotes) to build FriCAS
```

And

```
make
```

Add these to .bashrc

```
export PATH=/home/me/data/fricas/fricas-1.3.2/target/x86_64-linux-gnu/bin:$
export AXIOM=/home/me/data/fricas/fricas-1.3.2/target/x86_64-linux-gnu
export DAASE=/home/me/data/fricas/fricas-1.3.2/target/x86_64-linux-gnu
```

Example of integral which fails in Fricas

```
ii:=integrate(log(1+x)/x/(1+(1+x)^(1/2))^(1/2),x)


   >> Error detected within library code:
   integrate: implementation incomplete (constant residues)
```

For Fricas, use these commands to get 1D plain text output

```
setSimplifyDenomsFlag(true)
)set output algebra on
ii:=integrate(1/(x*(3*x^2 - 6*x + 4)^(1/3)),x);
unparse(ii::InputForm)
```

Otherwise the output will go to console in 2D. To get Latex output do

```
setSimplifyDenomsFlag(true)
)set message prompt plain
)set output algebra off
)set output tex on
s:=asin(sqrt(1-x^2))/sqrt(1-x^2);
ii:=integrate(s,x)
```

And this will give

$$-\frac{\arcsin\left(\sqrt{-x^2+1}\right)^2}{2} \tag{19}$$

To record console session to file, use (from `https://github.com/daly/axiom/blob/master/faq`)

```
)spool filename
    starts sending output to the file called filename
)spool )off
stops sending output to the file
```

Like this

```
)spool /home/me/data/output.txt
)set message prompt plain
)set output algebra off
)set output tex on
ii:=integrate(sin(x), x)
)set output tex off
)set output algebra on
unparse(ii::InputForm)
)spool )off
```

To send Latex output to file do

```
)set output tex  on
)set output tex  filename
```

To turn off, just do `)set output tex off`. Make sure to tex on first. To send back to console, do `)set output tex console`

To read commands from file do

To read an input file into Axiom, use the)read system command. read For example, you can read a file in a particular directory by issuing

)read /spad/src/input/matrix.input

To make record

```
)clear prop r
r: Record(a: String,b:Integer)
r:=["hello",10]
```

Some hints below thanks to Waldek Hebisch http://mathforum.org/kb/message.jspa?messageID=9791385

copied here so easy for me to get to:

```
1. How to measure CPU time used from the int call?
(3) -> )set messages time on
(3) -> integrate(x^(3/2)/sqrt(1 + x^5), x)

Type: Union(Expression(Integer),...)
Time: 0.06 (EV) + 0.00 (OT) = 0.06 sec

That is command ')set messages time on' tell FriCAS to print
time needed to execute following command. The (EV) part means
actual computation (OT) printing and at the end there is total

2. How to measure leaf count/size of result? (Maple and Mathematica have
build in function to do this)

That is a bit tricky. FriCAS (like Axiom) has quite different
representation of expressions tham Maple or Mathematica.
In FriCAS expression is a quotient of two polnomials, with
variables beeing kernels:

(13) -> f := exp(x^3 + 1/x) + 1
(14) -> numer(f)

4
x + 1
------
x

(15) -> denom(f)


(16) -> variables(numer(f))

4
x + 1
------
x
(16) [%e ]
```

```
Type: List(Kernel(Expression(Integer)))
Time: 0.00 (OT) = 0.00 sec

From FriCAS point of view natural measure of size is number
of monomials in numerator and denominator:

(17) -> numberOfMonomials(numer(f))

(17) 2
(18) -> numberOfMonomials(denom(f))
(18) 1


but this may underestimate size because kernels may be big.
Also internally given kernel is stored only once, but in
printed output it may appear several times.

It is possible to traverse expression is somewhat tree like
manner, so it is possible to give better approximation
to say Maple result, but that would require investigating
what Maple is doing. I personally never used Maple node
count so I do not know it this is simple or if there are
some traps.

3. How to check if int passed or failed? Aborted? nil result? etc..

integrate may produce an error (in particular it will do so
if it can not decide if integral is elementary). Or may
produce unevaluated integral. Or normal result. At programistic
level it is possible to catch errors, but a bit tricky,
If you look at printed output, than errors are reasonably
easy to match via a regex:

(22) -> integrate(1/sqrt(exp(x) - x + 1), x)
integrate: implementation incomplete (constant residues)

the '>> Error' part indicates error. Unevaluated integrals
always have integral sign at top level. FriCAS never returns
partial results, so text match is relatively easy. Or you
may use code like:


test_int(f) ==
res : Union(Expression Integer, LE)
res := integrate(f, 'x)
res0 : List Expression Integer :=
res case Expression Integer =>
[res::(Expression Integer)]@(List Expression Integer)
```

```
res case List Expression Integer =>
res::(List Expression Integer)
error "test_int: impossible 1"
for ri in res0 repeat
#(kernels ri) > 0 and is?(operator first kernels ri, 'integral) =>
print("Unevaluated integral"::OutputForm)
```

Note1: this is part extracted from bigger test script, I did not test it alone.

Note2: FriCAS may either return list of results or a single result. Middle part converts this to have always a list (typically of length 1). The last part is a loop so that all solutions can be examined. It you only want to know if result is evaluated, than loop is not needed: more than one result means that integral is evaluated.

Note3: In FriCAS testsuite I use a bit different test for evaluated integrals. Namely, FriCAS can do useful computations on unevaluated integrals and in particular unevaluated integrals may appear in the argument to integrate. Such unevaluated integrals of course may propagate form input to the output. The test above may misclasify such integral as unevaluated, while better test checks that unevaluated integral came from the input.

4. And most importantly, how to export the result (if it passed) to a plain text file in _Latex_ format?

```
(24) -> )set output tex on
(24) -> integrate(exp(x)*exp(1/(exp(x)+1)-x), x)
```

Here one have to decide what to do with errors. Without error trapping error will abort currently executing function and propagate to top level (up to command line).

After doing:

```
)set break resume
```

after error FriCAS will continue executing file from next command (but still abort current command). If you need real loop then I can provide error catcher, but it is a bit more complicated than snippets above.
Waldek Hebisch

To get type of value in fricas do `typeOf(r)`

To clear everything do

)clear all

)clear completely

sympy

http://docs.sympy.org/0.7.1/modules/integrals.html

http://docs.sympy.org/dev/tutorial/index.html

To install python package do conda install package-name to update do conda update package for example conda update spyder

To integrate the command is

from sympy import *

or

import sympy #but now have to add sympy. to each call

```
import os
os.getcwd()
os.chdir('X:\\data\\public_html\\my_notes\\CAS_integration_tests\\reports\\rubi_4_11\\code')
os.getcwd()
import math
init_printing()
init_printing(use_unicode=False, wrap_line=False, no_global=True)
x = symbols('x', real=True)
r=integrate(x,x)
r0=latex(r)
text_file = open("python.txt", "w")
text_file.write("\"%s\"" % r0)
text_file.close()
```

To check if integral fails?

```
r=integrate(f*g, (x, L/2, L))
type(r) is integrals.Integral

Out[41]: True
```

or

```
isinstance(r, integrals.Integral)
Out[48]: True
```

```
r=integrate(x,x)
type(r) is integrals.Integral
Out[43]: False

if isinstance(r, integrals.Integral):
    result = 0
else:
    result=1
```

To get cwd

os.getcwd()

To get list of folder in cwd do

os.listdir(os.getcwd())

or

ls

```
Out[85]:
['1_Algebraic_functions',
 '2_Exponentials',
 '3_Logarithms',
 '4_Trig_functions',
 '5_Inverse_trig_functions',
 '6_Hyperbolic_functions',
 '7_Inverse_hyperbolic_functions',
 '8_Special_functions',
 'Independent_test_suites']

f = open('Hebisch_Problems.txt','r')
f.readline()
f.close()
```

how to do timeout? `simple timeouts using signal.alarm module` check `http://stackoverflow.com/questions/492519/timeout-on-a-function-call` for some code. There is also talk about it here `https://groups.google.com/forum/#!topic/sympy/qsPImy6WqcI` code from above is

```
def _timeout(self, function, timeout):
        def callback(x, y):
            signal.alarm(0)
            raise Skipped("Timeout")
        signal.signal(signal.SIGALRM, callback)
        signal.alarm(timeout)  # Set an alarm with a given timeout
```

```
function()
signal.alarm(0)  # Disable the alarm
```

elementry functions `http://docs.sympy.org/latest/modules/functions/elementary.html` and `http://docs.sympy.org/dev/modules/functions/special.html`

and `http://docs.sympy.org/latest/modules/functions/index.html`

Need to change

```
AppellF1 -->            #hypergeometric function of two variables
Ellipticpi-->
arctanh -->atanh
arctan --> atan
arccosh->acosh
Pi --> pi.
arcsin -> asin
arccos -> acos
hypergeom -> hyper,
arccoth -> acoth
GAMMA -> uppergamma()
arccsc -> acsc
arcsec -> asec
arccot -> acot,
EllipticF -> elliptic_f
EllipticE -> elliptic_e
Li -> Li
Si -> Si
Ci -> Ci
Ei -> Ei
FresnelS -> fresnels
FresnelC -> fresnelc
```

I also removed 4th field that contains `integrate(...)` in it as it hangs the reading of the input file.

To run python tests do

```
cd /media/data/public_html/my_notes/CAS_integration_tests/reports/rubi_4_11
python sympy_main.py
```

To add new test, or run more tests, edit `sympy_main.py` and change the line

```
for n in range(0,1):  #change the last number to the number of tests to do
```

To update conda and python do

```
conda update conda
conda update python
```

To update sympy do

```
pip list | grep sympy
pip install --upgrade sympy
```

To find version, from inside python, type

```
import sympy
sympy.___version___
```

from github `https://github.com/sympy/sympy/wiki/SymPy-vs.-Sage`

To obtain a Rational in SymPy, one of these methods must be used:

```
>>> from sympy import Rational
>>> Rational(2, 7)
2/7
```

In python/sympy, to loop using index 1, can use this:

```
>>> for num,val in enumerate(sol,start=1):
...   print("root number {} is {}".format(num,val))
```

For timeout in recent python, on windows, this seems to work

```
from subprocess import STDOUT, check_output
output = check_output('dir', shell=True, stderr=STDOUT, timeout=5)
print(output)
```

To download development sympy

```
sudo git clone git://github.com/sympy/sympy.git
sudo git clone https://github.com/HPAC/matchpy.git
```

To install using conda `sudo conda install multiset`

packages are here `/home/me/anaconda3/lib/python3.6/site-packages`

To install `pymatch`

```
>sudo conda install -c conda-forge matchpy
[sudo] password for me:
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda
```

```
  added / updated specs:
    - matchpy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    hopcroftkarp-1.2.4         |           py_0          20 KB  conda-forge
    certifi-2018.4.16          |        py36_0         142 KB  conda-forge
    conda-4.5.5                |        py36_0         624 KB  conda-forge
    matchpy-0.4.4              |           py_0          49 KB  conda-forge
    multiset-2.1.1             |           py_0          11 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         845 KB

The following NEW packages will be INSTALLED:

    hopcroftkarp: 1.2.4-py_0       conda-forge
    matchpy:      0.4.4-py_0       conda-forge
    multiset:     2.1.1-py_0       conda-forge

The following packages will be UPDATED:

    certifi:      2018.4.16-py36_0           --> 2018.4.16-py36_0 conda-forge
    conda:        4.5.4-py36_0               --> 4.5.5-py36_0     conda-forge

Proceed ([y]/n)? y


Downloading and Extracting Packages
hopcroftkarp-1.2.4   |   20 KB | ##################################################
certifi-2018.4.16    |  142 KB | ##################################################
conda-4.5.5          |  624 KB | ##################################################
matchpy-0.4.4        |   49 KB | ##################################################
multiset-2.1.1       |   11 KB | ##################################################
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
>
```

To update sympy in conda do `sudo conda update sympy` I do not know now if I should always use sudo for this, but it works. To check conda version do

```
>conda list anaconda$
# packages in environment at /opt/anaconda:
#
# Name                    Version                   Build  Channel
anaconda                  5.2.0                   py36_3

>sudo conda update conda
[sudo] password for me:
```

```
Solving environment: done

# All requested packages already installed.


>conda update anaconda
Solving environment: done

# All requested packages already installed.
```

Can also do `conda update --all` to update other conda stuff.

To get all info about conda do

```
>conda info

     active environment : None
       user config file : /home/me/.condarc
 populated config files :
          conda version : 4.5.5
    conda-build version : 3.10.5
         python version : 3.6.5.final.0
       base environment : /opt/anaconda  (read only)
           channel URLs : https://repo.anaconda.com/pkgs/main/linux-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/free/linux-64
                          https://repo.anaconda.com/pkgs/free/noarch
                          https://repo.anaconda.com/pkgs/r/linux-64
                          https://repo.anaconda.com/pkgs/r/noarch
                          https://repo.anaconda.com/pkgs/pro/linux-64
                          https://repo.anaconda.com/pkgs/pro/noarch
          package cache : /opt/anaconda/pkgs
                          /home/me/.conda/pkgs
       envs directories : /home/me/.conda/envs
                          /opt/anaconda/envs
               platform : linux-64
             user-agent : conda/4.5.5 requests/2.18.4 CPython/3.6.5 Linux/4.14.53-1-MANJARO manjaro/17.1.11 glibc/2.2
                UID:GID : 1000:1000
             netrc file : None
           offline mode : False
```

see also `https://conda.io/docs/_downloads/conda-cheatsheet.pdf`

This will list all versions of specific package `conda search sympy`

sagemath hints

see `http://doc.sagemath.org/html/en/tutorial/programming.html`

To check if integral evaluated or not, do

```
var('x')
sage: anti=integrate(sqrt(1+x^3),x)

sage: anti
integrate(sqrt(x^3 + 1), x)

sage: isinstance(anti.operator(), sage.symbolic.integration.integral.IndefiniteIntegral)

True
```

To simplify, use `expr.full_simplify()`

to find what methods are there for some expressions to appy to, type `expr.<TAB>` i.e. hit TAB key after the dot.

see help on `http://doc.sagemath.org/html/en/genindex.html`

To find versions of installed package do `ver = installed_packages()`

0.1 Trying rubi in sympy

First install 1.2 sympy then install this

```
>sudo conda install -c conda-forge matchpy
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda

  added / updated specs:
    - matchpy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2018.8.24  |       ha4d7672_0         136 KB  conda-forge
    certifi-2018.8.24          |           py36_1         139 KB  conda-forge
    openssl-1.0.2o             |       h470a237_1         3.5 MB  conda-forge
    conda-4.5.11               |           py36_0         625 KB  conda-forge
    hopcroftkarp-1.2.4         |             py_0          20 KB  conda-forge
    multiset-2.1.1             |             py_0          11 KB  conda-forge
    matchpy-0.4.4              |             py_0          49 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         4.5 MB

The following NEW packages will be INSTALLED:

    hopcroftkarp:     1.2.4-py_0         conda-forge
```

```
    matchpy:          0.4.4-py_0         conda-forge
    multiset:         2.1.1-py_0         conda-forge

The following packages will be UPDATED:

    ca-certificates: 2018.03.07-0                  --> 2018.8.24-ha4d7672_0 conda-forge
    certifi:         2018.8.13-py36_0              --> 2018.8.24-py36_1     conda-forge
    conda:           4.5.10-py36_0                 --> 4.5.11-py36_0        conda-forge

The following packages will be DOWNGRADED:

    openssl:         1.0.2p-h14c3975_0             --> 1.0.2o-h470a237_1    conda-forge

Proceed ([y]/n)? y


Downloading and Extracting Packages
ca-certificates-2018 | 136 KB    | ####################################################
certifi-2018.8.24    | 139 KB    | ####################################################
openssl-1.0.2o       | 3.5 MB    | ####################################################
conda-4.5.11         | 625 KB    | ####################################################
hopcroftkarp-1.2.4   | 20 KB     | ####################################################
multiset-2.1.1       | 11 KB     | ####################################################
matchpy-0.4.4        | 49 KB     | ####################################################
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Now try it for few commands

```
>python
Python 3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from sympy import *
>>> from sympy.integrals.rubi.rubi import rubi_integrate
>>> x = symbols('x')
>>> rubi_integrate(exp(1-exp(x**2)*x+2*x**2)*(2*x**3+x)/(1-exp(x**2)*x)**2, x)
Integral((2*x**3 + x)*exp(2*x**2 - x*exp(x**2) + 1)/(-x*exp(x**2) + 1)**2, x)

>>> x,a,b = symbols('x a b')
>>> rubi_integrate(1/(a**2-b**2*x), x)
-log(a**2 - b**2*x)/b**2

>>> rubi_integrate(1/(a**2-b**2*x**2), x)
x*hyper((1, 1/2), (3/2,), b**2*x**2/a**2)/a**2

%the above seems to be wrong

>>> rubi_integrate(sec(2*a*x), x)
Integral(sec(2*a*x), x)
```

```
>>> rubi_integrate(1/(1+cos(x)), x)
Integral(1/(cos(x) + 1), x)

%why it can't do the above?

>>> integrate(1/(1+cos(x)), x)
tan(x/2)

%it looks like not all rules are there
```

build logs

1. anaconda.txt This is build log for anaconda 5.2 on Linux on June 2018
2. sagemath.txt This is build log for sage math 8.2 on Linux on June 2018
3. fricas.txt This is build log for Fricas 1.3.3-1 on Linux on June 2018
4. giac.txt This is build log for giac 1.4.9.59-4 on Linux on June 2018