Computer Algebra Independent Integration Tests

Nasser M. Abbasi

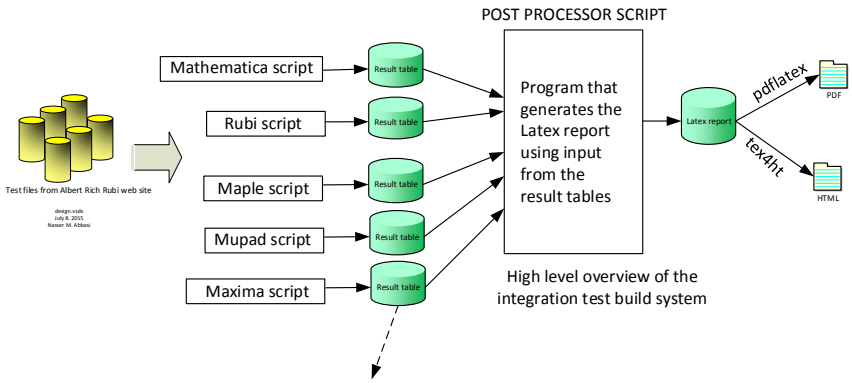
June 18, 2017 compiled on — Sunday June 18, 2017 at 06:55 PM [public]

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

1. Rubi 4.11, Mathematica 11.1, Maple 2017, Mupad 7.0, Sympy 1.0 [63,648 integrals (not completed yet)]
2. Rubi 4.9.8, Mathematica 11.0, Maple 2016 and Mupad 7.0 (Matlab 2016a) [58,469 integrals]
3. Rubi 4.9, Mathematica 10.4, Maple 2016 and Mupad 7.0 (Matlab 2016a) [58,469 integrals]

1 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”

The following are the steps to run one test suite

1. open Maple and run `convert_maple_to_sympy.mw` This translates all Maple input files to sympy syntax. This needs to be done once only. This coverts all Maple (177) test files to sympy syntax.
2. load Rubi package

3. open main.nb and definitions.nb. Evaluate definitions.nb. Then in main.nb, increment counter to run the test number (under RUN THE TESTS). This will run both Mathematica and Rubi test and grade and create the result tables in the test folder.

4. In main.nb, increment counter for Rubi rules and run. This will create Rubi rules used for each integral.

5. Now run Maple. open `master_file_maple.mw` and adjust the counter to same test number and evaluate. This will create Maple result table.

6. open `grade_maple.mw` in Maple and adjust the counter and run. This will grade each integral in the result table. This uses `GradeAntiderivative.mpl`

7. Now open Matlab and load `mupad.mn` and increment the counter to same test number and run it. This will create mupad result table.

8. Open Maple again and load `grade_mupad.mw` and adjust the counter to the test number and run it. This will read mupad result table and grade it.

9. Edit `sympy_main.py` which is the Python driver to run the sympy tests and adjust the counter, then type (on Linux only) `python sympy_main.py` This will create sympy result table. Wait few days for this to complete.

10. Now go back to main.nb and run command `makeReportLatexDriver[n]` where n is the test number. This will create the test report Latex document. Then run `makeIndexLatexReportDriver[m]` command, where m is the total completed tests so far. This will build the main index latex document.

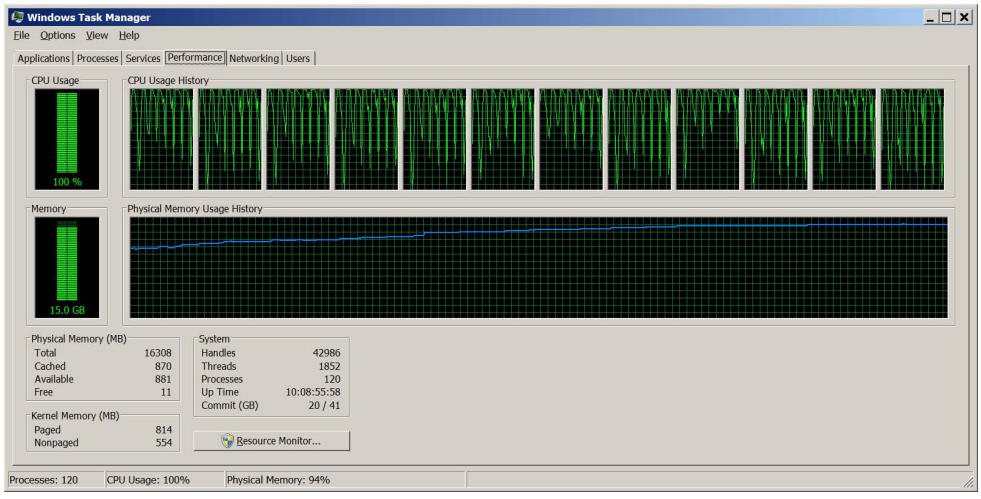
11. Now compile everything on Linux using the command `CAS_integration_tests>make -I$HOME all`

12. after many hrs it should be done.

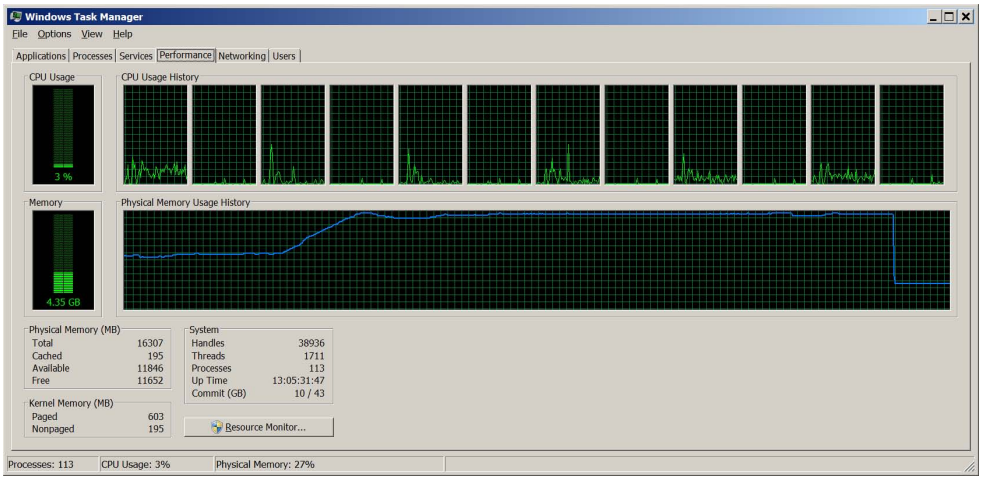
13. When build is done, upload to site.

2 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.



3 cheat sheet notes

3.1 Fricas

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 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
```

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 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
```

3.2 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
```