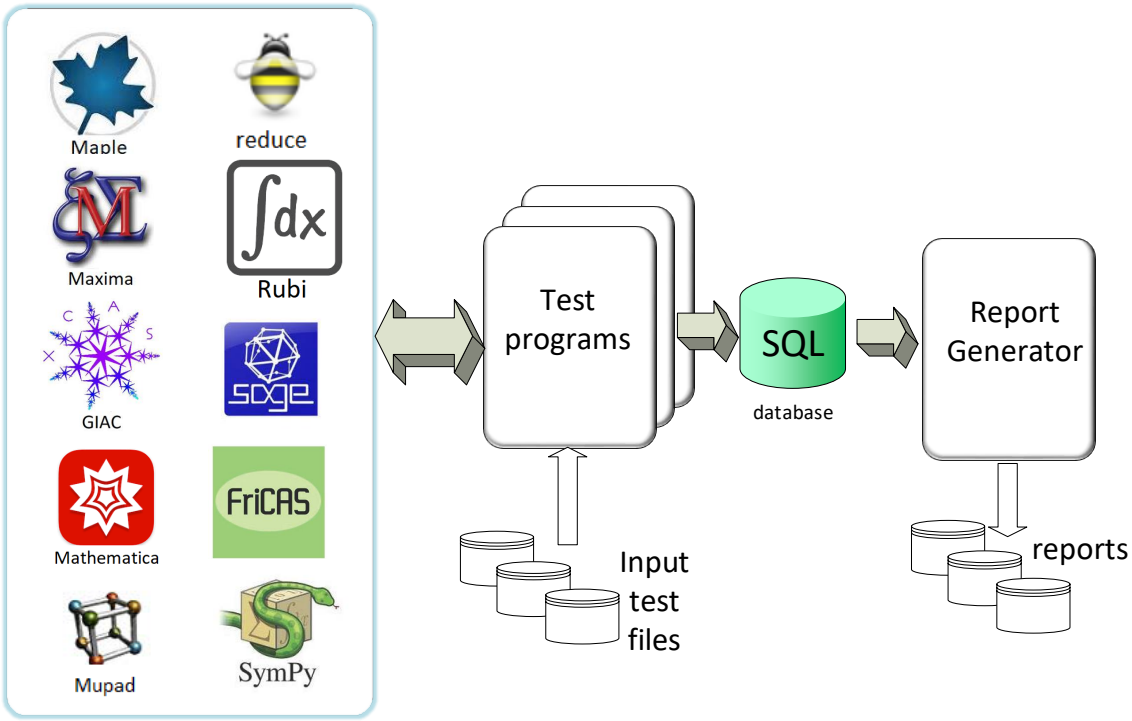
# Computer Algebra Independent Integration Tests




Nasser M. Abbasi

May 17, 2024

# Contents

# 1 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 performed against another.

## 1.1 Summer 2024

Started January 6 2024, Completed April 19, 2024.

Updated May 20, 2024 to use new GIAC 1.9-0.99, update from 1.9-0.97.

Main page is here

The following are the CAS systems tested:

1. Mathematica 14 (January 9, 2024) on windows 10
2. Rubi 4.17.3 (Sept 25, 2023) on Mathematica 14 on windows 10
3. Maple 2024 on windows 10
4. Maxima 5.47 (June 1, 2023) using Lisp SBCL 2.4.0 on Linux Manjaro 23.1.2 KDE via sagemath 10.3
5. FriCAS 1.3.10 built with sbcl 2.3.11 (January 10, 2024) on Linux Manjaro 23.1.2 KDE via sagemath 10.3
6. Giac/Xcas 1.9.0-99 on Linux via sagemath 10.3
7. Sympy 1.12 using Python 3.11.6 (Nov 14 2023, 09:36:21) [GCC 13.2.1 20230801] on Linux Manjaro 23.1.2 KDE
8. Mupad using Matlab 2021a with Symbolic Math Toolbox Version 8.7 on windows 10
9. Reduce CSL rev 6687 (January 9, 2024) on Linux Manjaro 23.1.2 KDE

Verification of anti-derivative implemented only for Mathematica, Rubi and Maple.

Maxima and Fricas and Giac are called using Sagemath. This was done using Sagemath `integrate` command by changing the name of the algorithm to use the different CAS systems.

Sympy was run directly in Python not via sagemath. Reduce was run directly.

The following report shows how to interface to the data generated using SQLite3 Summer 2024 database

## 1.2 Summer 2023 build with Rubi V 4.17.3

Main page is here

This build is same as the one below except for using Rubi 4.17.3 instead of 4.16.1. It uses the same 213 tests files as summer 2023 edition.

Verification of anti-derivative implemented only for Mathematica, Rubi and Maple.

The following are the CAS systems tested:

1. Mathematica 13.3.1 (August 16, 2023) on windows 10.
2. Rubi 4.17.3 (Sept 25, 2023) on Mathematica 13.3.1 on windows 10
3. Maple 2023.1 (July, 12, 2023) on windows 10.
4. Maxima 5.47 (June 1, 2023) using Lisp SBCL 2.3.0 on Linux via sagemath 10.1 (Aug 20, 2023).
5. FriCAS 1.3.9 (July 8, 2023) based on sbcl 2.3.0 on Linux via sagemath 10.1 (Aug 20, 2023).
6. Giac/Xcas 1.9.0-57 (June 26, 2023) on Linux via sagemath 10.1 (Aug 20, 2023).
7. Sympy 1.12 (May 10, 2023) Using Python 3.11.3 on Linux.
8. Mupad using Matlab 2021a with Symbolic Math Toolbox Version 8.7 on windows 10.

Maxima and Fricas and Giac are called using Sagemath. This was done using Sagemath `integrate` command by changing the name of the algorithm to use the different CAS systems.

Sympy was run directly in Python not via sagemath.

The following Rubi regression report shows integrals failed in 4.17.3 but passed in 4.16.1.

The following Rubi reverse regression report shows integrals failed in 4.16.1 but passed in 4.17.3.

The following report shows how to interface to the data generated using SQLite3 Summer 2023 database

## 1.3 Summer 2023 build with Rubi V 4.16.1

Main page is here

**Completed** September 6, 2023.

Number of integrals **85,963** with grading for all CAS systems except for Mupad.

Verification of anti-derivative implemented only for Mathematica, Rubi and Maple.

The following are the CAS systems tested:

1. Mathematica 13.3.1 (August 16, 2023) on windows 10.
2. Rubi 4.16.1 (Dec 19, 2018) on Mathematica 13.3 on windows 10
3. Maple 2023.1 (July, 12, 2023) on windows 10.
4. Maxima 5.47 (June 1, 2023) using Lisp SBCL 2.3.0 on Linux via sagemath 10.1 (Aug 20, 2023).
5. FriCAS 1.3.9 (July 8, 2023) based on sbcl 2.3.0 on Linux via sagemath 10.1 (Aug 20, 2023).
6. Giac/Xcas 1.9.0-57 (June 26, 2023) on Linux via sagemath 10.1 (Aug 20, 2023).
7. Sympy 1.12 (May 10, 2023) Using Python 3.11.3 on Linux.
8. Mupad using Matlab 2021a with Symbolic Math Toolbox Version 8.7 on windows 10.

Maxima and Fricas and Giac are called using Sagemath. This was done using Sagemath `integrate` command by changing the name of the algorithm to use the different CAS systems.

Sympy was run directly in Python not via sagemath.

The following report shows how to interface to the data generated using SQLite3 Summer 2023 database

## 1.4 Summer 2022 build

Main page is here

**Completed** August 21, 2022.
**Frozen effective March 27,2023**.

No changes will be made to this version. See above for current version.

Number of integrals **85,483** with grading for all CAS systems except for Mupad.

Verification of anti-derivative implemented only for Mathematica, Rubi.

The test integrals used are made of the following 210 files.

1. Rubi test suite. Files 1 to 208. Thanks to Alert Rich.
2. IntegrateAlgebraic test file. File 209. Thanks to Sam Blake.
3. Fricas test file. File 210. Thanks to Waldek Hebisch.

These 8 CAS systems are used:

1. Mathematica 13.1 (June 29, 2022) on windows 10.
2. Rubi 4.16.1 (Dec 19, 2018) on Mathematica 13.1 on windows 10.
3. Maple 2022.1 (June 1, 2022) on windows 10.
4. Maxima 5.46 (April 13, 2022) using Lisp SBCL 2.1.11.debian on Linux via sagemath 9.6.
5. Fricas 1.3.8 (June 21, 2022) based on sbcl 2.1.11.debian on Linux via sagemath 9.6.
6. Giac/Xcas 1.9.0-13 (July 3, 2022) on Linux via sagemath 9.6.
7. Sympy 1.10.1 (March 20, 2022) Using Python 3.10.4 on Linux.
8. Mupad using Matlab 2021a with Symbolic Math Toolbox Version 8.7 on windows 10.

The following Summer 2022 regression reports shows the integrals which failed in current summer 2022 build but passed in summer 2021 build. This was done for each CAS which changed version since last year.

The following Summer 2022 progress reports shows the integrals which were solved in current summer 2022 build but failed in summer 2021 build. This was done for each CAS which has a changed version since last year.

The following report shows how to interface to the data generated using SQLite3 Summer 2022 database

## 1.5 Summer 2021 build

Main page is here

**Completed**. Started May 15, 2021, finished July 28, 2021.

Number of integrals **71,994**, with grading for all CAS systems except for Mupad.

Eight CAS systems were tested:

1. Rubi 4.16.1.
2. Mathematica 12.3 on windows 10.
3. Maple 2021.1 on windows 10.
4. Maxima 5.44/sagemath 9.3 on Linux/Manjaro.
5. Fricas 1.3.7/sagemath 9.3 on Linux/Manjaro.
6. Giac 1.7.0/sagemath 9.3 on Linux/Manjaro.
7. Sympy 1.8 Using Python 3.7.9 on Linux/Ubuntu 18.
8. Mupad/Matlab 2021a on windows 10.

## 1.6 April 2020 build

Main page is here

Started March 9, 2020, Completed April 6, 2020.

Uses Rubi 4.16.1 input data. Number of integrals **71,994**. With grading for all CAS systems.

CAS systems used are : Rubi 4.16.1, Mathematica 12.1, Maple 2020, Maxima 5.43, Fricas 1.3.6, Sympy 1.5, Giac/XCAS 1.5 and Sagemath.

## 1.7 December 2018 build with Rubi in sympy

Main page is here

Number of tests completed **28,425**. Completed December 16, 2018.

CAS systems used are : Rubi 4.15.2, Mathematica 11.3, Maple 2018, Maxima 5.41, Fricas 1.3.3, Sympy 1.1.1, Giac/XCAS 1.4.9, SageMath version 8.3 and Rubi in Sympy implemetation (port of Rubi to Sympy)

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

# 2 Specialized integration tests

## 2.1 January 2024 special build with Reduce as guest CAS

Main page is here

Number of integrals **3,809** with grading for all CAS systems except for Mupad and Reduce.

Verification of anti-derivative implemented only for Mathematica, Rubi and Maple.

The following are the CAS systems tested:

1. Mathematica 13.3.1 (August 16, 2023) on windows 10.
2. Rubi 4.17.3 (Sept 25, 2023) on Mathematica 13.3.1 on windows 10
3. Maple 2023.1 (July, 12, 2023) on windows 10.
4. Maxima 5.47 (June 1, 2023) using Lisp SBCL 2.3.0 on Linux via sagemath 10.1 (Aug 20, 2023).
5. FriCAS 1.3.9 (July 8, 2023) based on sbcl 2.3.0 on Linux via sagemath 10.1 (Aug 20, 2023).
6. Giac/Xcas 1.9.0-61 (Sept. 26, 2023) on Linux via sagemath 10.1 (Aug 20, 2023).
7. Sympy 1.12 (May 10, 2023) Using Python 3.11.3 on Linux.
8. Mupad using Matlab 2021a with Symbolic Math Toolbox Version 8.7 on windows 10.
9. Reduce CSL rev. 6657. December 10, 2023. On Linux.

Maxima and Fricas and Giac are called using Sagemath. This was done using Sagemath `integrate` command by changing the name of the algorithm to use the different CAS systems.

Sympy was run directly in Python not via sagemath.

Reduce was run directly on Linux using its own build script. Postprocessing of Reduce output was done by Maple.

Link to final database will be made here when all tests completed.

The following report shows how to interface to the data generated using SQLite3 january 2024 with reduce database

## 2.2 First 13 files only using Mathics 4.0 via sagemath 9.6 as Guest CAS

This build shows only the result of testing the first 13 input files and includes a guest CAS Mathics 4.0 via sagemath 9.6 as a proof of concept.

Summer 2022 special build

Number of integrals tested [3809] with grading for all CAS systems except for Mupad.

The test integrals used are made of the following 210 files.

1. Rubi test suite. Files 1 to 208. Thanks to Alert Rich.
2. IntegrateAlgebraic test file. File 209. Thanks to Sam Blake.
3. Fricas test file. File 210. Thanks to Waldek Hebisch.

These 9 CAS systems are used:

1. Mathematica 13.0.1 (February 17, 2022) on windows 10.
2. Rubi 4.16.1 (Dec 19, 2018) on Mathematica 13.0.1 on windows 10.
3. Maple 2022.1 (June 1, 2022) on windows 10.
4. Maxima 5.46 (April 13, 2022) using Lisp SBCL 2.0.1.debian on Ubuntu 20.04 Linux under window 10 WSL 2.0 subsystem via sagemath 9.6.
5. Fricas 1.3.7 (June 30, 2021) based on based on ecl 21.2.1 on Ubuntu 20.04 Linux under window 10 WSL 2.0 subsystem via sagemath 9.6.

6. Giac/Xcas 1.9.0-7 (April 2022) on on Ubuntu 20.04 Linux under window 10 WSL 2.0 subsystem. Now uses direct calls using giac C++ API and not via sagemath.

7. Sympy 1.10.1 (March 20, 2022) using Python 3.10.4 Ubuntu 20.04 Linux under window 10 WSL 2.0 subsystem via sagemath 9.6.

8. Mupad using Matlab 2021a with Symbolic Math Toolbox Version 8.7 on windows 10.

9. Mathics 4.0 via sagemath 9.6.

## 2.3 Waldek Hebisch's Yet another integration test

Completed on Sept 15, 2021

This report uses the integrals provided thanks to Waldek Hebisch taken from this sci.math.symbolic post

The Number of integrals is [10,335]. With grading for all CAS systems except for Mupad.

These CAS systems were tested : Rubi 4.16.1, Mathematica 12.3.1, Maple 2021.1, Maxima 5.45, Fricas 1.3.7, Sympy 1.8 under Python 3.7.9, Giac/XCAS 1.7 and Mupad/Matlab 2021a and Sagemath 9.4.

Maxima, Fricas and Giac were used from within Sagemath 9.4 interface.

## 2.4 Elementary Algebraic integrals

September 2021 version

Number of integrals [22,622]. With grading for all CAS systems except for Mupad.

Nine CAS systems were tested : Rubi 4.16.1, Mathematica 12.3.1, Maple 2021.1, Maxima 5.44, Fricas 1.3.7, Sympy 1.8 under Python 3.7.9, Giac/XCAS 1.7 and Mupad/Matlab 2021a and Sagemath 9.3 and `IntegrateAlgebraic` under Mathematica 12.3.1 on windows 10.

Maxima, Fricas and Giac were used from within Sagemath 9.3 interface.

## 2.5 Sam Blake's IntegrateAlgebraic function Test

IntegrateAlgebraic test problems

Finished August 26, 2021.

This test Uses IntegrateAlgebraic test suite provided thanks to Sam Blake.

Number of integrals [3,154]. With grading for all CAS systems except for Mupad.

Nine CAS systems were tested : Rubi 4.16.1, Mathematica 12.3.1, Maple 2021.1, Maxima 5.44, Fricas 1.3.7, Sympy 1.8 under Python 3.7.9, Giac/XCAS 1.7 and Mupad/Matlab 2021a and Sagemath 9.3 and `IntegrateAlgebraic` under Mathematica 12.3.1 on windows 10.

Maxima, Fricas and Giac were used from within Sagemath 9.3 interface.

# 3 LITE CAS integration tests (same CAS over different versions)

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.

## 3.1 Mathematica

Main report is here

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

## 3.2 Rubi

Main report is here

Uses same zip file as above for input.

## 3.3 Maple

Main report is here

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

# 4 Current tree layout of integration tests

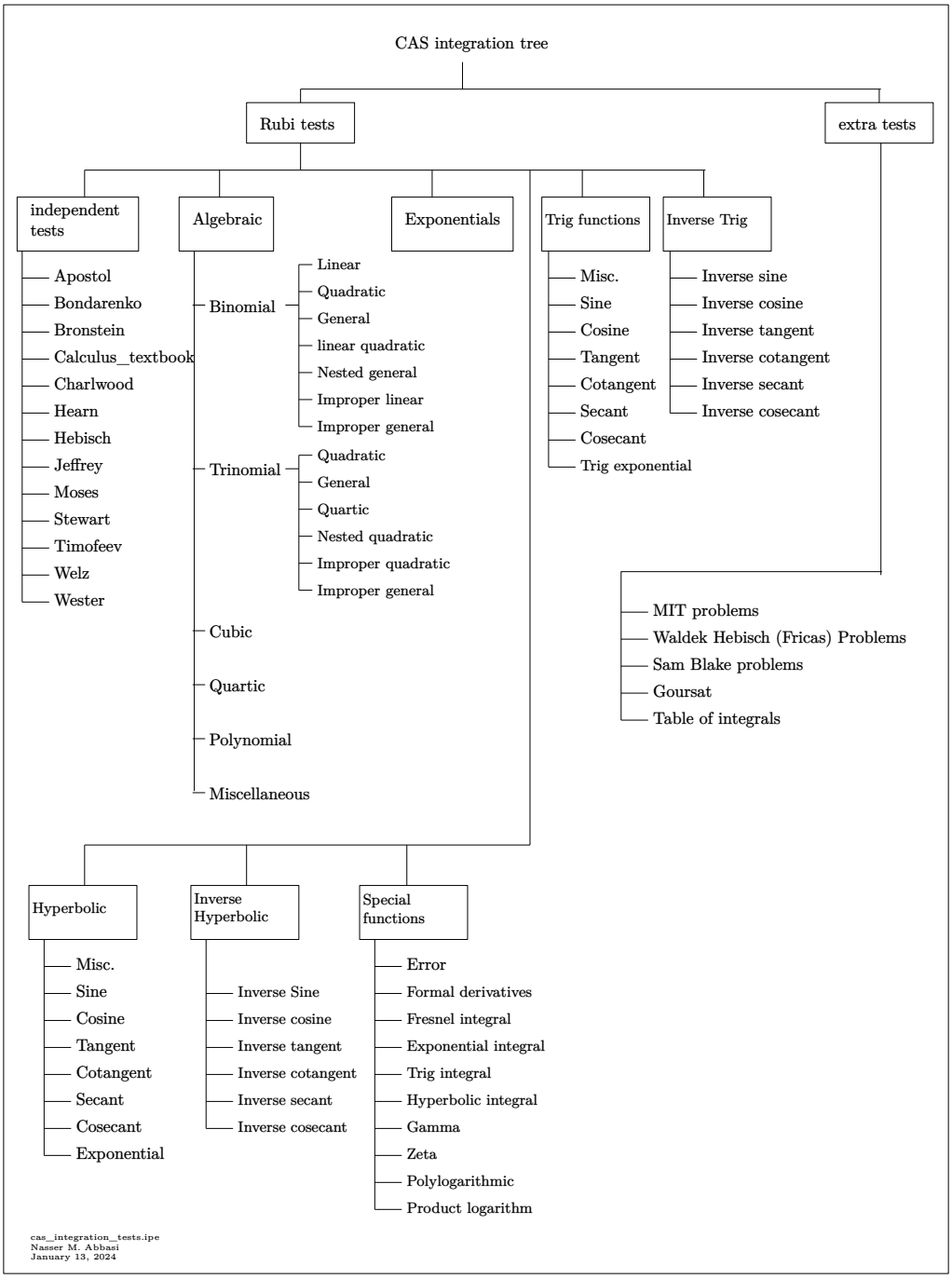
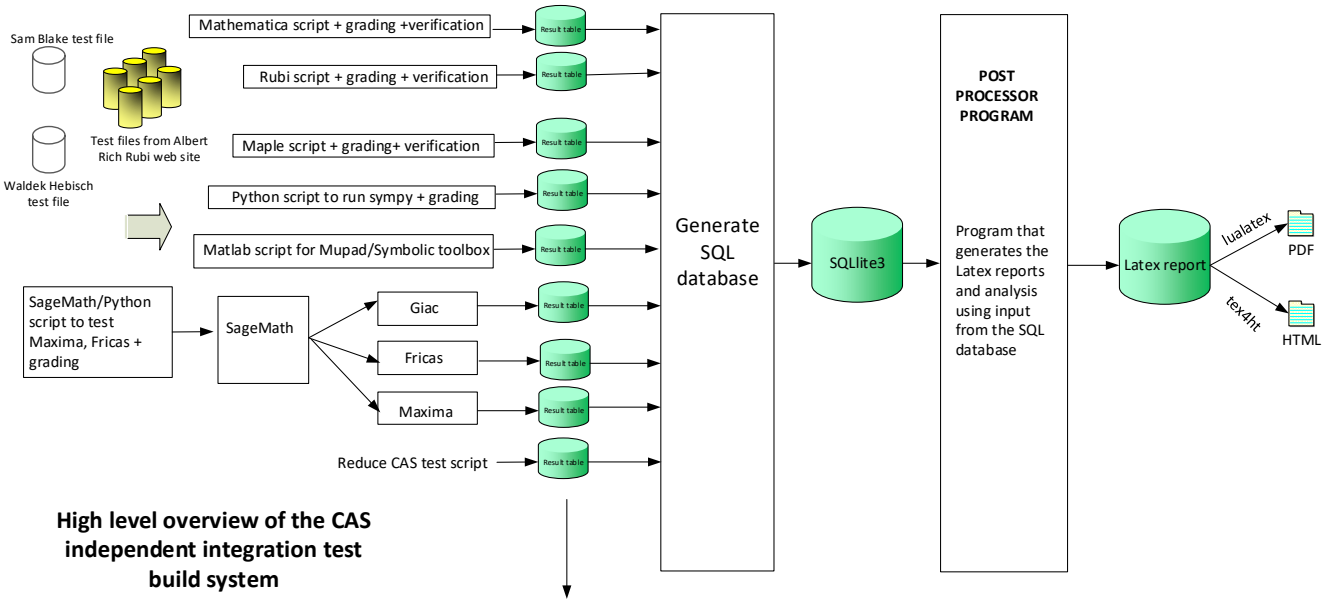



Figure 1: CAS integration tests tree

# 5 Note on build system

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




**High level overview of the CAS independent integration test build system**

**One record (line) per one integral result. The line is CSV comma separated. This is description of each record**

1. integer, the problem number.
2. integer. 0 for failed, 1 for passed, -1 for timeout, -2 for CAS specific exception. (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 syntax.
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"
13. String. Small string description of why the grade was given.
14. integer. 1 if result was verified or 0 if not verified. (For mma, rubi and maple only)

*The following fields are present only in **Rubi** Table file*

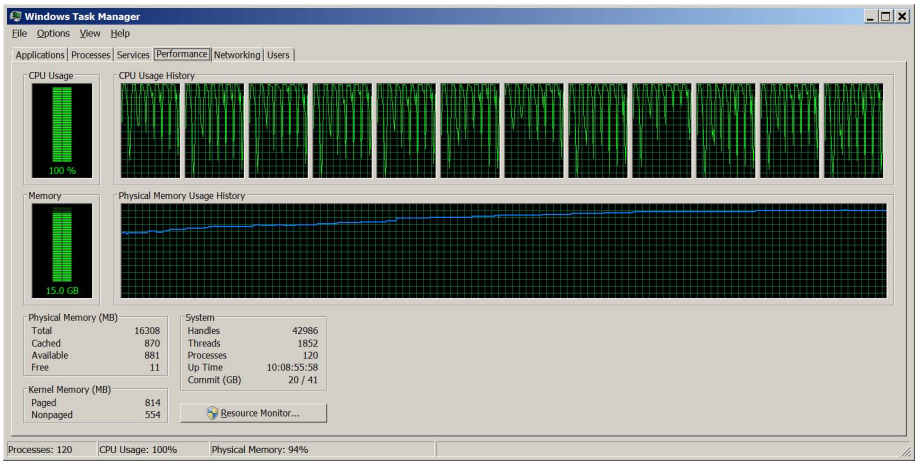
15. integer. Number of steps used.
16. integer. Number of rules used.
17. integer. Integrand leaf size.
18. real number. Ratio. Field 16 over field 17
19. String of form "{n,n,..}" which is list of the rules used by Rubi
20. String. The optimal antiderivative in Mathematica syntx

Nasser M. Abbasi
January 13, 2024
Design.vsdx

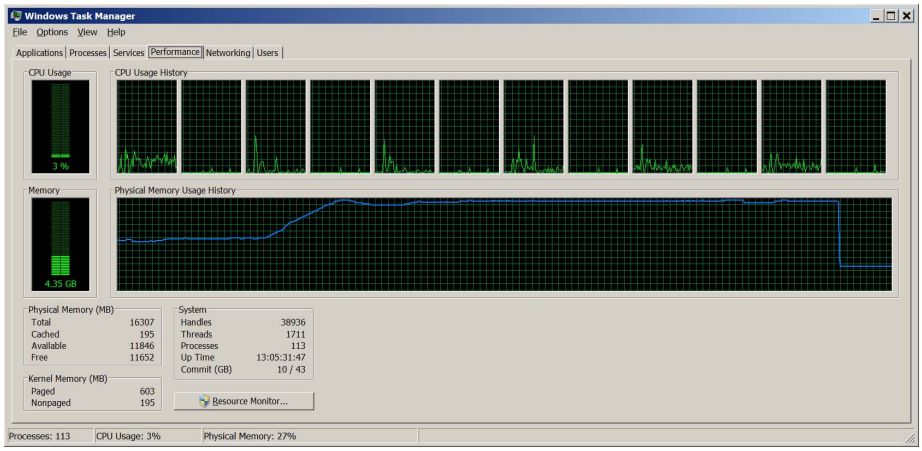
Figure 2: CAS integration test system

# 6 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.




# 7 my cheat sheet notes

## 7.1 Reduce

Downloaded source tar file from

`https://sourceforge.net/projects/reduce-algebra/files/snapshot_2023-12-18/Reduce-svn6658-src.tar.gz/download`

The extracted it to my `$HOME/TMP/` folder and cd to the folder created and did `./configure --with-csl` and then `make`

When done, make sure to add `$HOME/TMP/Reduce-svn6658-src/bin/` to `PATH` by editing my .bashrc

To start do `redcsl -w` this starts in console mode. Or `redcsl` to start in GUI window.

To run script do `redcsl < script.red`

1. `https://github.com/reduce-algebra`
2. `https://reduce-algebra.sourceforge.io/index.php`

To do integration do , after typing `redcsl` the GUI will come up then type

```
load_package "algint";
off fancy;
on latex;
int(sin(x),x);
```

The reduce document at `https://reduce-algebra.sourceforge.io/reduce38-docs/reduce.pdf`

To run reduce script, do this. First make file r.txt and write in it reduce commands. Let r.txt be this file

```
load_package "algint";
load_package "rlfi";
off fancy;
on latex;
r:=int(sin(x),x);
```

Now do

```
redcsl --nogui < r.txt
Reduce (CSL, rev 6339), 16-Jun-2022 ...

1:
2:
3:
***** FANCY is not current output handler
```

```
4: \documentstyle{article}
\begin{document}

5: \begin{displaymath}
r=-\cos \,x
\end{displaymath}

6:




*** End-of-file read
>
```

See `https://reduce-algebra.sourceforge.io/tutorials/EggerScripts.en.php` for examples of reduce scripts.

## 7.2 Julia

1. see `https://symbolicnumericintegration.sciml.ai/dev/`

   It says:

   "Function integrate returns the integral of a univariate expression with constant real or complex coefficients. integrate returns a tuple with three values. The first one is the solved integral, the second one is the sum of the unsolved terms, and the third value is the residual error. If integrate is successful, the unsolved portion is reported as 0."

   see also the paper

   `https://arxiv.org/abs/2201.12468`

```
import Pkg; Pkg.add("Symbolics")` to install the Symbolics package.

julia> using SymbolicNumericIntegration
  Package SymbolicNumericIntegration not found, but a package named SymbolicNumericInte
  Install package?
    (@v1.7) pkg> add SymbolicNumericIntegration
  (y/n) [y]:

Precompiling project...
```

```
  64 dependencies successfully precompiled in 539 seconds (110 already precompiled)
```

To do integration, the commands are (where it is assumed the integrand say 3*x is in file called `input.txt` in same folder

```
using Symbolics
using SymbolicNumericIntegration
using Dates
using CSV
using DataFrames
using Elliptic
using Elliptic.Jacobi
using SpecialFunctions

@syms x z y t a

integrand = include("input.txt")
integrate(integrand,x)

julia> @time integrate(3*sin(x)+x*cos(x),x)
     0.138087 seconds (19.18 k allocations: 9.491 MiB)
    (x*sin(x) - 2cos(x), 0, 0)
```

To measure time used

```
julia> time1 = now();

julia> anti,n_unsolved,residual_error = integrate(3*sin(x)+x*cos(x),x)
(x*sin(x) - 2cos(x), 0, 0)

julia> time2 = now();

julia> (Dates.DateTime(time2)-Dates.DateTime(time1))/ Millisecond(1) * (1 / 1000)
0.197

julia> anti
x*sin(x) - 2cos(x)
```

To read the integrals, create text file like this (say input.txt)

```
[[3*x,x,0,3/2*x^2],
[4*x,x,0,2*x^2]]
```

Then do

```
data=include("input.txt")

julia> length(data)
2

julia> data[1][1]
3x

julia> data[2][1]
4x
```

Hence to loop over all integrands in file, do

```
data=include("input.txt")

julia> for n in 1:length(data)
       println(integrate(data[n][1],data[n][2]))
       end
((3//2)*(x^2), 0, 0)
(2(x^2), 0, 0)
```

To read all lines as strings do

```
data=readlines("input.txt")

julia> data[2]
"[x*sqrt(1 + 3*x), x, 2],"

julia> data[3]
"[x^2*sqrt(1 + x), x, 2],"
```

This gives error

```
expr=1/((1 + x)^2*sqrt(2)*sqrt(-im + x^2)) + 1/((1 + x)^2*sqrt(2)*sqrt(im + x^2))
ERROR: TypeError: non-boolean (Num) used in boolean context
Stacktrace:
```

```
 [1] sqrt(z::Complex{Num})
   @ Base ./complex.jl:501
 [2] top-level scope
   @ REPL[115]:1
```

A better way to read the integrals from file is this. The file has to be in this format

```
[x^2,x]
[x^3,x]
.
.
.
[sin(x),x]
```

Then do

```
using Symbolics
using SymbolicNumericIntegration
using Dates
using SpecialFunctions
using SymbolicUtils

@syms x z y t a
@syms _C0 _C1 _C2 _C3 _C4 _C5 _C6 _C7 _C8 _C9

#import Base: exp

#exp(x::Real) = Symbolics.Term(exp,[x])
#exp(x::Int64) = Symbolics.Term(exp,[x])



data=readlines("julia_input.txt");

number_failed = 0
for counter in 1:10 #length(data)
   #println("processing line ",counter)
   try
      item = eval(Meta.parse(data[counter]))
      print("\n\n****calling integrate(",item[2],",",item[3],")")
      anti,n_unsolved,residual_error = integrate(item[2],item[3])
```

```
      println("anti=",anti)
      println("n_unsolved=",n_unsolved)
      println("residual_error=",residual_error)

   catch
      number_failed = number_failed + 1
      #println(">>>>>>>>>error on integral ",counter)
      println(">>>>>>>>>error on integral ",Meta.parse(data[counter]))
   end
end

println("Failed to read ", number_failed, " integrals")
```

This way I know which line has an error when reading the interals.

Find why this error

```
julia> (-x)^(1//2)
ERROR: DomainError with -1.0:
Exponentiation yielding a complex result requires a complex argument.
Replace x^y with (x+0im)^y, Complex(x)^y, or similar.
Stacktrace:
 [1] throw_exp_domainerror(x::Float64)
   @ Base.Math ./math.jl:37
 [2] ^(x::Float64, y::Float64)
   @ Base.Math ./math.jl:1003
 [3] ^
   @ ./promotion.jl:422 [inlined]
 [4] ^
   @ ./rational.jl:481 [inlined]
 [5] unstable_pow
   @ ~/.julia/packages/SymbolicUtils/qulQp/src/types.jl:801 [inlined]
 [6] ^(a::SymbolicUtils.Mul{Number, Int64, Dict{Any, Number}, Nothing}, b::Rational{Int
   @ SymbolicUtils ~/.julia/packages/SymbolicUtils/qulQp/src/types.jl:1047
 [7] top-level scope
   @ REPL[32]:1
```

But the integrate command currently only supports univariate expression with constant real or complex coefficients. So it can integrate $2*x$ but can not integrate $a*x$ so not possible to use it in CAS integration tests.

2. to read symbolic expression from file see this post `https://stackoverflow.co`

m/questions/59543961/how-to-read-a-polynomial-from-a-text-file-in-julia

## 7.3 Fricas

### 7.3.1 Running Fricas script

create file, say F1.input like this

```
f(n)==n^2
f(4)
```

And run it using the command

```
fricas -nosman < F1.input > o.txt
```

The output will go to file o.txt

### 7.3.2 Finding type of variable

Use `typeOf`

```
r:=sqrt(-5)::Complex(Float)
   (3)  2.2360679774_997896964 %i

(5) -> typeOf(r)
   (5)  Complex(Float)
```

### 7.3.3 Home page

Home page `https://fricas.github.io/index.html`

To install fricas on Linux Manjaro `sudo pacman -S sagemath`.

in case of error while installing the package, try `sudo pacman -Rs sagemath` and then repeat the previous command

If copy paste does not work between windows and Linux VBox, try `VBoxClient --clipboard`

To use Fricas online, go to

~~`http://axiom-wiki.newsynthesis.org/FriCASIntegration`~~

~~http://fricas-wiki.math.uni.wroc.pl/SandBox~~

https://wiki.fricas.org/SandBox

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 installl on Linux Manjaro, type `yay -S fricas` . Do not use the Manjaro GUI package manager, it always gives build error for some reason. Make sure to install yay first using GUI manager.

Make sure when installing Linux on VBox, to add second CPU to configuration of VBox. i.e. do not use ONE CPU (which is default). There is a bug in fricas installation which makes it fail due to use of multi-threading if only one CPU exist. This should be fixed in future, but just in case.

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

To build directly (but I use yay now and not this any more)

```
#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 implementation of ANSI Common
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:$PATH
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)
```

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 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
```

There 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
```

## 7.4 sympy

To install type `pip install -U sympy`

To install using anaconda:

First install anaconda using Linux Manjaro package manager.

To install latest sympy in anaconda try `conda install -c anaconda sympy` see `https://anaconda.org/anaconda/sympy`

To update to latest conda do

```
sudo –i
#conda update conda
```

To update to latest sympy do

```
sudo –i
#conda update sympy
... now check version
#python
>>> import sympy
>>> sympy.___version___
'1.8'
```

Next must install matchpy, without it Rubi will not worK:

```
>sudo conda install -c conda-forge matchpy
```

```
[sudo] password for me:
Collecting package metadata: done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda

  added / updated specs:
    - matchpy


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2019.3.9   |       hecc5488_0         146 KB  conda-forge
    certifi-2019.3.9           |           py37_0         149 KB  conda-forge
    conda-4.6.14               |           py37_0         2.1 MB  conda-forge
    hopcroftkarp-1.2.4         |             py_0          20 KB  conda-forge
    matchpy-0.5.1              |             py_0          52 KB  conda-forge
    multiset-2.1.1             |             py_0          11 KB  conda-forge
    openssl-1.1.1b             |       h14c3975_1         4.0 MB  conda-forge
    ------------------------------------------------------------
                                           Total:         6.4 MB

The following NEW packages will be INSTALLED:

  hopcroftkarp       conda-forge/noarch::hopcroftkarp-1.2.4-py_0
  matchpy            conda-forge/noarch::matchpy-0.5.1-py_0
  multiset           conda-forge/noarch::multiset-2.1.1-py_0

The following packages will be UPDATED:

  ca-certificates    pkgs/main::ca-certificates-2019.1.23-0 --> conda-forge::ca-certificates

The following packages will be SUPERSEDED by a higher-priority channel:

  certifi                                       pkgs/main --> conda-forge
  conda                                         pkgs/main --> conda-forge
  openssl              pkgs/main::openssl-1.1.1b-h7b6447c_1 --> conda-forge::openssl-1.1.1b-
```

```
Proceed ([y]/n)? y


Downloading and Extracting Packages
multiset-2.1.1       | 11 KB     | ############################################################
openssl-1.1.1b       | 4.0 MB    | ############################################################
matchpy-0.5.1        | 52 KB     | ############################################################
conda-4.6.14         | 2.1 MB    | ############################################################
hopcroftkarp-1.2.4   | 20 KB     | ############################################################
ca-certificates-2019 | 146 KB    | ############################################################
certifi-2019.3.9     | 149 KB    | ############################################################
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
>
```

In general, To install python package do `conda install package-name` to update do `conda update package` for example `conda update spyder`

To use Rubi in sympy, do (note this no longer works in recent version of sympy. 2023)

```
>python
Python 3.7.3 (default, Mar 27 2019, 22:11:17)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from sympy import *
>>> from sympy.integrals.rubi.rubimain import rubi_integrate
```

For testing do

```
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 failed

```
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
                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`

## 7.5 sagemath hints

To download latest stable release use `https://mirrors.mit.edu/sage/src/index.html`.

To download development releases use `https://www.sagemath.org/download-latest.html`

To build do

```
unset SAGE_ROOT
unset SAGE_LOCAL

#Do this to force sage to use system's before building sage

export GIAC=/usr/local/bin/giac
export MAXIMA=/usr/bin/giac

#Now do (to build with newer python), make sure to tell it to
```

```
#use system giac, which I installed before, since it is newer
#version. see my giac notes on how to install giac

./configure --with-system-python3=no --with-system-giac=force
make
make install
```

To obtain version numbers of installed CAS systems do (see also https://ask.sagemath.org/question/42617/how-to-find-version-number-of-cas-used-by-sage/

```
>sage

    SageMath version 9.8, Release Date: 2023-02-11
    Using Python 3.11.1. Type "help()" for help.

#ignore this, since I install maxima manually outside sage
age: ver = installed_packages()
sage: ver['maxima']  #WRONG. version. I have deleted maxima used by sage
'5.45.0.p0'



#use this below, since I installed maxima outside sage.
#but make sure to remove the maxima's binaries installed by sage from its local/bin
#after installing external maxima and do the following
#         >cd $SAGE_ROOT/local/bin
#         >rm rmaxima
#         >rm maxima
#         >rm xmaxima

#         >ln -s /usr/bin/rmaxima
#         >ln -s /usr/bin/maxima
#         >ln -s /usr/bin/xmaxima

#now do this to get the correct version of the external maxima used

sage: print(maxima.version())
5.46.0

#to install fricas, use the command
#     > sage -i fricas
```

```
#AFTER installing sagemath

sage: print(fricas.eval(")lisp |$build_version|"))
Value = "FriCAS 1.3.8"

#this will now print the version by external giac since we asked
#sage to use system giac before

sage: print(giac.version())
"giac 1.9.0, (c) B. Parisse and R. De Graeve, Institut Fourier, Universite de Grenoble I"

age: gap.version()
'4.11.1'

sage: pari.version()
(2, 15, 2)

sage: gp.version()
((2, 15, 2), 'GP/PARI CALCULATOR Version 2.15.2 (released)')
```

To enter an ode do

```
age: x = var('x')
sage: y = function('y')
sage: ode=diff(y(x),x)+y(x)==sin(x)

sage: ode
y(x) + diff(y(x), x) == sin(x)
```

To obtain lhs and rhs of equation

```
age: ode.lhs()
y(x) + diff(y(x), x)
sage: ode.rhs()
sin(x)
```

To use sagemath to solve ODE

```
x = var('x')
y = function('y')
```

```
sage: desolve(diff(y(x),x,2) + y(x) ==0, y(x),algorithm='fricas')
_C0*cos(x) + _C1*sin(x)

sage: desolve(diff(y(x),x,2) + y(x) ==0, y(x),algorithm='maxima')
_K2*cos(x) + _K1*sin(x)
```

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`

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

## 7.6 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.rubimain 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
```

## 7.7 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