ADAM 2013

Prover9 and other provers with Sage/Python/LaTeX input and output

Peter Jipsen

Chapman University, Orange, California

ADAM 2013, June 7, U. New Mexico

Outline

Residuated idempotent semirings

Generalized Hoops

SMT-solver for Basic Logic

UACalc

Aims for ATP interfaces

We start with showing how Prover9/Mace4 can be called from Sage (uses a simple Python interface that could also be used independently).

To use Sage in this way, one has to install the Universal Algebra package from http://math.chapman.edu/~jipsen/sagepkg/

There are several theories built in. To see a list of abbreviations for all such theories use the command

FOclasses

or to see full names for these classes, enter

for x in FirstOrderClasses: print x

```
for x in FirstOrderClasses: print x
```

evaluate

```
IdSring
```

```
FOclass("Idempotent semirings", syntax="Prover9", axioms=[
"(x*y)*z = x*(y*z)",
"x*1 = x",
"1*x = x",
"(x+y)+z = x+(y+z)",
"x+y = y+x",
"x+0 = x",
"x*(y+z) = (x*y)+(x*z)",
"(x+y)*z = (x*z)+(y*z)",
"x*0 = 0",
"0*x = 0",
"x+x = x"])
```

```
ISrng = FOclass("ISrng", "Idempotent semirings",
syntax="Prover9", axioms=[
"(x*y)*z = x*(y*z)",
"(x+y)+z = x+(y+z)",
"x+y = y+x",
"x*(y+z) = (x*y)+(x*z)",
"(x+y)*z = (x*z)+(y*z)",
"x+x = x"])
```

```
ISrng.find_models(2)
```

```
Number of nonisomorphic models of cardinality 2  is  6
[
Model(cardinality = 2, index = 0, operations = {
```

```
"+":[
[0,0],
[0,1]],
"*":[
[0,0],
[0,0]]}),
Model(cardinality = 2, index = 1, operations = {
"+":[
[0,1],
[1,1]],
"*":[
[0,0],
[0,0]]}),
Model(cardinality = 2, index = 2, operations = {
"+":[
[0,0],
[0,1]],
"*":[
[0,0],
[0,1]]}),
Model(cardinality = 2, index = 3, operations = {
"+":[
[0,1],
[1,1]],
"*":[
[0,0],
[0,1]]}),
Model(cardinality = 2, index = 4, operations = {
"+":[
[0,0],
[0,1]],
"*":[
[0,0],
[1,1]]}),
Model(cardinality = 2, index = 5, operations = {
"+":[
[0,0],
[0,1]],
"*":[
[0,1],
[0,1]]})]
```

tmp.in tmp.out tmpe

```
[len(ISrng.find_models(n)) for n in [2..4]]
```

```
Number of nonisomorphic models of cardinality 2  is  6
```

```
Number of nonisomorphic models of cardinality 3  is  61
isofilter: 1000 interps read, 423 kept
isofilter: 2000 interps read, 686 kept
isofilter: 3000 interps read, 792 kept
isofilter: 4000 interps read, 845 kept
isofilter: 5000 interps read, 855 kept
isofilter: 6000 interps read, 861 kept
isofilter: 7000 interps read, 866 kept
isofilter: 8000 interps read, 866 kept
isofilter: 9000 interps read, 866 kept
isofilter: 10000 interps read, 866 kept
isofilter: 11000 interps read, 866 kept
isofilter: 12000 interps read, 866 kept
isofilter: 13000 interps read, 866 kept
Number of nonisomorphic models of cardinality 4  is  866
[6, 61, 866]
```

tmp.in   tmp.out   tmpe

```
[len(IdSring.find_models(n)) for n in [2..5]]
```

```
Number of nonisomorphic models of cardinality 2  is  1
Number of nonisomorphic models of cardinality 3  is  3
Number of nonisomorphic models of cardinality 4  is  20
Number of nonisomorphic models of cardinality 5  is  149
[1, 3, 20, 149]
```

tmp.in   tmp.out   tmpe

```
ISrng1=["(x*y)*z = x*(y*z)",
"x*1 = x",
"1*x = x",
"(x+y)+z = x+(y+z)",
"x+y = y+x",
"x*(y+z) = (x*y)+(x*z)",
"(x+y)*z = (x*z)+(y*z)",
"x+x = x"]
[len(prover9(ISrng1,[],200,0,n)) for n in [2..5]]
```

```
Number of nonisomorphic models of cardinality 2  is  2
Number of nonisomorphic models of cardinality 3  is  11
Number of nonisomorphic models of cardinality 4  is  73
isofilter: 1000 interps read, 371 kept
isofilter: 2000 interps read, 520 kept
isofilter: 3000 interps read, 587 kept
isofilter: 4000 interps read, 654 kept
isofilter: 5000 interps read, 654 kept
isofilter: 6000 interps read, 703 kept
```

```
isofilter: 7000 interps read, 703 kept
isofilter: 8000 interps read, 703 kept
isofilter: 9000 interps read, 703 kept
Number of nonisomorphic models of cardinality 5  is  703
[2, 11, 73, 703]
```

[tmp.in](tmp.in) [tmp.out](tmp.out) tmpe

```
ISrng0=["(x*y)*z = x*(y*z)",
"(x+y)+z = x+(y+z)",
"x+y = y+x",
"x+0 = x",
"x*(y+z) = (x*y)+(x*z)",
"(x+y)*z = (x*z)+(y*z)",
"x*0 = 0",
"0*x = 0",
"x+x = x"]
[len(prover9(ISrng0,[],200,0,n)) for n in [2..5]]
```

```
Number of nonisomorphic models of cardinality 2  is  2
Number of nonisomorphic models of cardinality 3  is  12
Number of nonisomorphic models of cardinality 4  is  129
isofilter: 1000 interps read, 381 kept
isofilter: 2000 interps read, 477 kept
isofilter: 3000 interps read, 477 kept
isofilter: 4000 interps read, 615 kept
isofilter: 5000 interps read, 755 kept
isofilter: 6000 interps read, 864 kept
isofilter: 7000 interps read, 905 kept
isofilter: 8000 interps read, 978 kept
isofilter: 9000 interps read, 978 kept
isofilter: 10000 interps read, 986 kept
isofilter: 11000 interps read, 1348 kept
isofilter: 12000 interps read, 1506 kept
isofilter: 13000 interps read, 1672 kept
isofilter: 14000 interps read, 1777 kept
isofilter: 15000 interps read, 1819 kept
isofilter: 16000 interps read, 1835 kept
isofilter: 17000 interps read, 1841 kept
isofilter: 18000 interps read, 1847 kept
isofilter: 19000 interps read, 1847 kept
isofilter: 20000 interps read, 1847 kept
isofilter: 21000 interps read, 1852 kept
isofilter: 22000 interps read, 1852 kept
isofilter: 23000 interps read, 1852 kept
isofilter: 24000 interps read, 1852 kept
isofilter: 25000 interps read, 1852 kept
```

```
isofilter: 26000 interps read, 1852 kept
isofilter: 27000 interps read, 1852 kept
isofilter: 28000 interps read, 1852 kept
isofilter: 29000 interps read, 1852 kept
isofilter: 30000 interps read, 1852 kept
isofilter: 31000 interps read, 1852 kept
Number of nonisomorphic models of cardinality 5  is  1852
[2, 12, 129, 1852]
tmp.in   tmp.out   tmpe
```

Generalized hoops were first studied by Bosbach [1969, 70] and the name hoop was introduced by Büchi and Owen [1975].

A **generalized hoop** $(A, \cdot, 1, \backslash, /)$ is a **residuated partially ordered monoid** in which $x \le y \iff \exists u(x = uy) \iff \exists v(x = yv)$.

I.e. the monoid is **naturally ordered**, hence **integral**: $x \le 1$

Residuated means: $xy \le z \iff y \le x\backslash z \iff x \le z/y$

```
GHax = ["x<=x", "x<=y & y<=x -> x=y", "x<=y & y<=z -> x<=z",

"(x*y)*z=x*(y*z)", "x*1=x", "1*x=x",

"x<=y -> x*z<=y*z", "x<=y -> z*x<=z*y",

"x*y<=z <-> y<=x\z", "x*y<=z <-> x<=z/y",

"x<=y <-> exists u(x=u*y)", "x<=y <-> exists v(x=y*v)"]

GHax
```

```
['x<=x', 'x<=y & y<=x -> x=y', 'x<=y & y<=z
-> x<=z', '(x*y)*z=x*(y*z)', 'x*1=x', '1*x=x', 'x<=y ->
x*z<=y*z', 'x<=y -> z*x<=z*y', 'x*y<=z <->
y<=x\\z', 'x*y<=z <-> x<=z/y', 'x<=y <-> exists
u(x=u*y)', 'x<=y <-> exists v(x=y*v)']
```

```
prover9(GHax,["x<=1"],0,20)
```

```
[
Proof(syntax="Prover9", formula_list=[
[1, 'x <= y <-> (exists z x = z * y) # label(non_clause)', []]
[2, 'x <= 1 # label(non_clause) # label(goal)', []],
[3, 'x * 1 = x', []],
```

```
[4, 'x <= y | z * y != x', [1]],
[5, '-(c1 <= 1)', [2]],
[6, 'x <= 1', [4, 3]],
[7, '$F', [6, 5]]])]
```

tmp.in tmp.out tmpe

```
prover9(GHax,["x/x=1"],0,20)
```

```
[
Proof(syntax="Prover9", formula_list=[
[1, 'x <= y & y <= x -> x = y # label(non_clause)', []],
[2, 'x * y <= z <-> x <= z / y # label(non_clause)', []],
[3, 'x <= y <-> (exists z x = z * y) # label(non_clause)', []],
[4, 'x / x = 1 # label(non_clause) # label(goal)', []],
[5, '-(x <= y) | -(y <= x) | y = x', [1]],
[6, 'x * 1 = x', []],
[7, '1 * x = x', []],
[8, '-(x * y <= z) | x <= z / y', [2]],
[9, 'x <= y | z * y != x', [3]],
[10, 'c1 / c1 != 1', [4]],
[11, 'x <= 1', [9, 6]],
[12, '1 * (x * y) <= y', [9, 7]],
[13, 'x * y <= y', [7, 12]],
[14, '-(1 <= x) | x = 1', [5, 11]],
[15, '-(1 <= c1 / c1)', [14, 10]],
[16, 'x <= y / y', [8, 13]],
[17, '$F', [16, 15]]])]
```

tmp.in tmp.out tmpe

```
prover9(GHax,["(x/y)*y=(y/x)*x"],0,20)
```

```
[]
```

tmp.in tmp.out tmpe

```
GHaxL = ["x<=x", "x<=y & y<=x -> x=y", "x<=y & y<=z -> x<=z",
"x*y<=z <-> y<=x\z", "x*y<=z <-> x<=z/y",
"x<=y <-> exists u(x=u*y)"]
prover9(GHaxL,["x<=y -> x=(x/y)*y"],0,20)
```

```
[
Proof(syntax="Prover9", formula_list=[
[1, 'x <= y & y <= x -> x = y # label(non_clause)', []],
[2, 'x <= y & y <= z -> x <= z # label(non_clause)',
[]],
```

```
[3, 'x * y <= z <-> x <= z / y # label(non_clause)', []],
[4, 'x <= y <-> (exists z x = z * y) # label(non_clause)', []],
[5, 'x <= y -> x = (x / y) * y # label(non_clause) # label(goal)',
[]],
[6, 'x <= x', []],
[7, '-(x <= y) | -(y <= x) | y = x', [1]],
[8, '-(x <= y) | -(y <= z) | x <= z', [2]],
[9, '-(x * y <= z) | x <= z / y', [3]],
[10, 'x * y <= z | -(x <= z / y)', [3]],
[11, '-(x <= y) | f1(x,y) * y = x', [4]],
[12, 'x <= y | z * y != x', [4]],
[13, 'c1 <= c2', [5]],
[14, '(c1 / c2) * c2 != c1', [5]],
[15, 'x <= (x * y) / y', [9, 6]],
[16, '(x / y) * y <= x', [10, 6]],
[17, 'x * y <= y', [12]],
[18, 'f1(c1,c2) * c2 = c1', [11, 13]],
[19, '-(x <= y) | z * x <= y', [8, 17]],
[20, 'x * y <= (y * z) / z', [19, 15]],
[21, '-(x <= (x / y) * y) | (x / y) * y = x', [7, 16]],
[22, '-(c1 <= (c1 / c2) * c2)', [21, 14]],
[23, 'f1(c1,c2) <= c1 / c2', [18, 15]],
[24, 'f1(f1(c1,c2),c1 / c2) * (c1 / c2) = f1(c1,c2)', [11, 23]],
[25, '(x * y) * z <= y * z', [10, 20]],
[26, '-(x <= (y * z) * u) | x <= z * u', [8, 25]],
[27, '-(c1 <= (x * (c1 / c2)) * c2)', [26, 22]],
[28, '-(c1 <= f1(c1,c2) * c2)', [24, 27]],
[29, '-(c1 <= c1)', [18, 28]],
[30, '$F', [29, 6]]])]
```

tmp.in tmp.out tmpe

```
prover9(GHaxL,["x<=y <- x=(x/y)*y"],0,20)
```

```
GHa = ["x<=x", "x<=y & y<=x -> x=y", "x<=y & y<=z -> x<=z",

#"(x*y)*z=x*(y*z)", "x*1=x", "1*x=x",

"x*y<=z <-> y<=x\z", "x*y<=z <-> x<=z/y",

"x<=y <-> x=(x/y)*y", "x<=y <-> x=y*(y\ x)"]

prover9(GHa,["x<=y -> x*z<=y*z"],0,20)
```

```
[
Proof(syntax="Prover9", formula_list=[
[1, 'x <= y & y <= z -> x <= z # label(non_clause)',
[]],
[2, 'x * y <= z <-> x <= z / y # label(non_clause)', []],
[3, 'x <= y -> x * z <= y * z # label(non_clause) #
label(goal)', []],
[4, 'x <= x', []],
[5, '-(x <= y) | -(y <= z) | x <= z', [1]],
[6, '-(x * y <= z) | x <= z / y', [2]],
[7, 'x * y <= z | -(x <= z / y)', [2]],
[8, 'c1 <= c2', [3]],
[9, '-(c1 * c3 <= c2 * c3)', [3]],
[10, 'x <= (x * y) / y', [6, 4]],
[11, '-(c1 <= (c2 * c3) / c3)', [7, 9]],
[12, '-(c2 <= x) | c1 <= x', [5, 8]],
[13, '-(c2 <= (c2 * c3) / c3)', [12, 11]],
[14, '$F', [13, 10]]])]
tmp.in   tmp.out   tmpe
```

```
prover9(GHa,["(x/y)*y=(y/x)*x"],0,20)
```

```
[]
tmp.in   tmp.out   tmpe
```

```
GHaR = ["x<=x", "x<=y & y<=x -> x=y", "x<=y & y<=z -> x<=z",

"(x*y)*z=x*(y*z)", "x*1=x", "1*x=x",

"x*y<=z <-> x<=z/y",

"x<=y <-> x=(x/y)*y"]

prover9(GHaR,["(x/y)*y=(y/x)*x"],0,20)
```

```
[
Proof(syntax="Prover9", formula_list=[
[1, 'x <= y & y <= x -> x = y # label(non_clause)', []],
[2, 'x <= y & y <= z -> x <= z # label(non_clause)',
[]],
[3, 'x * y <= z <-> x <= z / y # label(non_clause)', []],
[4, 'x <= y <-> x = (x / y) * y # label(non_clause)', []],
[5, '(x / y) * y = (y / x) * x # label(non_clause) # label(goa
[6, 'x <= x', []],
[7, '-(x <= y) | -(y <= x) | y = x', [1]],
[8, '-(x <= y) | -(y <= z) | x <= z', [2]],
```

```
[9, 'x * 1 = x', []],
[10, '1 * x = x', []],
[11, '-(x * y <= z) | x <= z / y', [3]],
[12, 'x * y <= z | -(x <= z / y)', [3]],
[13, '-(x <= y) | (x / y) * y = x', [4]],
[14, 'x <= y | (x / y) * y != x', [4]],
[15, '(c2 / c1) * c1 != (c1 / c2) * c2', [5]],
[16, 'x <= (x * y) / y', [11, 6]],
[17, '(x / y) * y <= x', [12, 6]],
[18, 'x <= 1 | x / 1 != x', [9, 14]],
[19, 'x <= x / 1', [9, 16]],
[20, '1 <= x / x', [10, 16]],
[21, '(((x / y) * y) / x) * x = (x / y) * y', [13, 17]],
[22, 'x / 1 <= x', [9, 17]],
[23, '-(x / 1 <= x) | x / 1 = x', [7, 19]],
[24, 'x / 1 = x', [23, 22]],
[25, 'x <= 1 | x != x', [24, 18]],
[26, 'x <= 1', [25]],
[27, '-(x / x <= 1) | x / x = 1', [7, 20]],
[28, 'x / x = 1', [27, 26]],
[29, 'x * y <= y | -(x <= 1)', [28, 12]],
[30, 'x * y <= y', [29, 26]],
[31, '-(x <= y) | (x / z) * z <= y', [8, 17]],
[32, '((x * y) / z) * z <= y', [31, 30]],
[33, '(x * y) / z <= y / z', [11, 32]],
[34, '-(x / y <= z) | (u * x) / y <= z', [8, 33]],
[35, '(x * y) / z <= ((y / z) * u) / u', [34, 16]],
[36, '((x * y) / z) * u <= (y / z) * u', [12, 35]],
[37, '(x / y) * y <= (y / x) * x', [21, 36]],
[38, '-((x / y) * y <= (y / x) * x) | (x / y) * y = (y / x) * 
37]],
[39, '(x / y) * y = (y / x) * x', [38, 37]],
[40, '$F', [39, 15]]]])]
```

[tmp.in]() [tmp.out]() [tmpe]()

```
prover9(GHaR,["(x/y)/z=x/(y*z)"],2,20)
```

```
[
Model(cardinality = 4, index = 0, operations = {"c3":2, "c2":0
"*":[
[0,0,2,3],
[0,1,2,3],
[0,2,2,3],
[3,3,3,3]],
"/":[
```

```
[1,0,3,1],
[1,1,1,1],
[3,2,1,1],
[3,3,3,1]]}, relations = {
"<=":[
[1,1,0,0],
[0,1,0,0],
[0,1,1,0],
[1,1,1,1]]})]
```

tmp.in tmp.out tmpe

```
prover9(GHaR,["(x/y)/z=x/(z*y)"],2,20)
```

```
[
Proof(syntax="Prover9", formula_list=[
[1, 'x <= y & y <= x -> x = y # label(non_clause)', []],
[2, 'x * y <= z <-> x <= z / y # label(non_clause)', []],
[3, '(x / y) / z = x / (z * y) # label(non_clause) # label(goa
[4, 'x <= x', []],
[5, '-(x <= y) | -(y <= x) | y = x', [1]],
[6, '(x * y) * z = x * (y * z)', []],
[7, '-(x * y <= z) | x <= z / y', [2]],
[8, 'x * y <= z | -(x <= z / y)', [2]],
[9, '(c1 / c2) / c3 != c1 / (c3 * c2)', [3]],
[10, '-(x * (y * z) <= u) | x * y <= u / z', [6, 7]],
[11, '(x / y) * y <= x', [8, 4]],
[12, '(((x / y) / z) * z) * y <= x', [8, 11]],
[13, '((x / y) / z) * (z * y) <= x', [6, 12]],
[14, '(x / (y * z)) * y <= x / z', [10, 11]],
[15, '(x / y) / z <= x / (z * y)', [7, 13]],
[16, 'x / (y * z) <= (x / z) / y', [7, 14]],
[17, '-(x / (y * z) <= (x / z) / y) | (x / z) / y = x / (y * z
15]],
[18, '-(c1 / (c3 * c2) <= (c1 / c2) / c3)', [17, 9]],
[19, '$F', [18, 16]]])]
```

tmp.in tmp.out tmpe

```
GH=[
"x*1 = x",
"x/x = 1",
"x\ x = 1",

"(x/y)*y = (y/x)*x",
"x*(x\y) = y*(y\ x)",
"x*(x\y) = (y/x)*x",

"x/(y*z) = (x/z)/y",
"(x*y)\z = y\(x\z)"
]
GH
```

```
['x*1 = x', 'x/x = 1', 'x\\ x = 1', '(x/y)*y = (y/x)*x', 'x*(x\
y*(y\\ x)', 'x*(x\\y) = (y/x)*x', 'x/(y*z) = (x/z)/y', '(x*y)\\
y\\(x\\z)']
```

```
prover9(GH,["1*x=x"],0,20)
```

```
[
Proof(syntax="Prover9", formula_list=[
[1, '1 * x = x # label(non_clause) # label(goal)', []],
[2, 'x * 1 = x', []],
[3, 'x / x = 1', []],
[4, 'x \\ x = 1', []],
[5, 'x * (x \\ y) = (y / x) * x', []],
[6, '1 * c1 != c1', [1]],
[7, 'x * (x \\ x) = 1 * x', [3, 5]],
[8, 'x * 1 = 1 * x', [4, 7]],
[9, 'x = 1 * x', [2, 8]],
[10, '1 * x = x', [9]],
[11, '$F', [10, 6]]])]
```

tmp.in tmp.out tmpe

```
prover9(GH,["(x*y)*z=x*(y*z)"],0,20)
```

```
[
Proof(syntax="Prover9", formula_list=[
[1, '(x * y) * z = x * (y * z) # label(non_clause) # label(goa
[2, 'x * 1 = x', []],
[3, 'x / x = 1', []],
[4, 'x \\ x = 1', []],
[5, '(x / y) * y = (y / x) * x', []],
[6, 'x * (x \\ y) = (y / x) * x', []],
```

```
[7, 'x / (y * z) = (x / z) / y', []],
[8, '(x / y) / z = x / (z * y)', [7]],
[9, '(c1 * c2) * c3 != c1 * (c2 * c3)', [1]],
[10, 'x * (x \\ x) = 1 * x', [3, 6]],
[11, 'x * 1 = 1 * x', [4, 10]],
[12, 'x = 1 * x', [2, 11]],
[13, '1 * x = x', [12]],
[14, '(x / (y * z)) / u = (x / z) / (u * y)', [8, 8]],
[15, 'x / (u * (y * z)) = (x / z) / (u * y)', [8, 14]],
[16, 'x / (u * (y * z)) = x / ((u * y) * z)', [8, 15]],
[17, 'x / ((y * z) * u) = x / (y * (z * u))', [16]],
[18, '((x * y) * z) / (x * (y * z)) = 1', [17, 3]],
[19, '1 * (x * (y * z)) = ((x * (y * z)) / ((x * y) * z)) * ((
z)', [18, 5]],
[20, 'x * (y * z) = ((x * (y * z)) / ((x * y) * z)) * ((x * y)
[13, 19]],
[21, 'x * (y * z) = ((x * (y * z)) / (x * (y * z))) * ((x * y)
[17, 20]],
[22, 'x * (y * z) = 1 * ((x * y) * z)', [3, 21]],
[23, 'x * (y * z) = (x * y) * z', [13, 22]],
[24, '(x * y) * z = x * (y * z)', [23]],
[25, '$F', [24, 9]]]])]
```

tmp.in   tmp.out   tmpe

Problem: Is the equational theory of GH decidable?

Need to try Waldmeister as a KB-completion procedure ...

SMT-solver for Hajek's Basic Logic

The term $(x/y) * y$ is a meet operation

Prover9 proved it is commutative, and associativity and idempotence $(x/x) * x = x$ also hold

Assume * is commutative, and add a join operation and a constant 0.

If we also assume **prelinearity**: $x/y \vee y/x = 1$ then we an equational axiomatization of **BL-algebras**.

They are the algebraic semantics of **Hajek's Basic Logic**.

BL-algebras include Boolean algebras, MV-algebras and linear Heyting algebras.

Yet they are very special residuated lattices since they have distributive lattice reducts.

In fact the subdirectly irreducible BL-algebras are linearly ordered.

The variety of BL-algebras is generated by ordinal sums of the unit interval (considered as an MV-algebra).

More precisely following result was proved by **Agliano and Montagna** [2003]:

An n-variable BL-identity fails in some BL-algebra if and only if it fails in A_n

where A_n is the interval $[0, n+1]$ and we define

$$x \cdot y = \begin{cases} \max(x+y-1-\lfloor y \rfloor, \lfloor x \rfloor) & \text{if } \lfloor x \rfloor = \lfloor y \rfloor \\ \min(x,y) & \text{otherwise} \end{cases}$$

$$x \to y = \begin{cases} n+1 & \text{if } x \le y \\ y & \text{if } \lfloor y \rfloor < \lfloor x \rfloor \\ \min(1+y-x+\lfloor x \rfloor, 1+\lfloor y \rfloor) & \text{otherwise} \end{cases}$$

```
Demo of SMT implementation (separate window)
```

UACalc: A program for finite universal algebras

Written in Java by R. Freese, E. Kiss, M. Valeriote

Can download it from uacalc.org, runs on PC, Mac, Linux

But works only on one algebra at a time.

Can read the algebra from an XML file or type in the operation tables

Tedious if Mace4 has computed a list of 61 algebras and you want to test them for some property

E.g. find the **simple** algebras in the list

```
ISrng = FOclass("ISrng", "Idempotent semirings",
syntax="Prover9", axioms=[
"(x*y)*z = x*(y*z)",
"(x+y)+z = x+(y+z)",
"x+y = y+x",
"x*(y+z) = (x*y)+(x*z)",
"(x+y)*z = (x*z)+(y*z)",
"x+x = x"])
```

```
a=ISrng.find_models(3)
```

```
Number of nonisomorphic models of cardinality 3  is  61
tmp.in   tmp.out   tmpe
```

```
s=[x for x in a if x.is_simple()]
s
```

```
[
Model(cardinality = 3, index = 43, operations = {
"+":[
[0,0,0],
[0,1,0],
[0,0,2]],
"*":[
[0,0,0],
[0,1,2],
[0,2,1]]},
con = ['|0|1|2|', '|0,1,2|']),
Model(cardinality = 3, index = 46, operations = {
"+":[
[0,0,0],
[0,1,1],
[0,1,2]],
"*":[
[0,0,0],
[0,1,2],
[2,2,2]]},
con = ['|0|1|2|', '|0,1,2|']),
Model(cardinality = 3, index = 49, operations = {
"+":[
[0,0,0],
[0,1,1],
[0,1,2]],
"*":[
```

```
[0,0,2],
[0,1,2],
[0,2,2]]},
con = ['|0|1|2|', '|0,1,2|'])]
tmpalgCon.ua   tmpoutCon.txt
```

```
Con(a[49])
```

```
['|0|1|2|', '|0,1,2|']
```

```
Sub(a[0])
```

```
s[0].Free(3)
```

```
93
tmpalgA.ua   tmpout.txt
```

```
s[1].Free(3)
```

```
180
tmpalgA.ua   tmpout.txt
```

```
s[2].Free(3)
```

```
180
tmpalgA.ua   tmpout.txt
```

```
len([x for x in a if x.is_SI()])
```

```
29
```

Aims:

bring ATP closer to mathematicians

use a standard LaTeX based language for I/O

little theories, also with reasoning over models (semantics)

integrate with computer algebra systems, use provers from a web browser develop more exploration tools, e.g. include term rewriting, completion, lemma extraction

present proofs in human readable form (standard typeset infix notation with superscripts, subscripts, greek letters, ...)

find short proofs, leave out trivial steps, remove symmetries

use ATP and ITP in undergraduate math classes

Thanks!

http://sagemath.org

http://www.cs.unm.edu/~mccune/

http://math.chapman.edu/~jipsen/sagepkg/