Brief Z3Py Tutorial

January 24, 2018

1 Introduction to Z3

Z3 is an SMT solver which means it is able to determine whether a logical statement is true or false (within certain limitations).
Here are some examples of using Z3 from within Python:
Proving DeMorgan's Law:

```
from z3 import *
s = Solver()      # Z3 solver instance
a = Bool('a')     # Z3 boolean variable a
b = Bool('b')     # Z3 boolean variable b
a_or_b = Or(a,b)  # a_or_b means a or b is true
nna_and_nb = Not(And(Not(a), Not(b)))
                  # nna_and_nb means not(not(a) and not(b))
                  # is true
s.add(a_or_b != nna_and_nb) # Here we are asserting to Z3
                            # (a or b) is not equivalent to
                            # ~(~a and ~b)
s.check()         # Here we ask Z3 if our statement is
                  # satisfiable or unsatisfiable
```

When running this in python interactive mode, Z3 reports `unsat` which means that our assertion that
$(a \vee b) \neq \neg(\neg a \wedge \neg b)$
is false. This means that it's negation (DeMorgan's Law) is true.

Finding Pythagorean triples, *e.g.*, integers x, y, z such that $x^2 + y^2 = z^2$:

```
from z3 import *
s = Solver() # Z3 solver instance
x = Int('x') # Z3 Integer variable x
y = Int('y') # Z3 Integer variable y
z = Int('z') # Z3 Integer variable z
s.add(And(x > 0, y > 0, z > 0))  # Assert x, y and z are all
                                 # positive
s.add(x*x + y*y == z*z) # Assert x^2 + y^2 = z^2
s.check() # Query Z3 to see if it can find an answer
s.model() # Get the satisfying assignment for x, y and z
s.add(x != 12) # Disallow the solution where x = 12
```

```
12 s.check() # Check if Z3 can still find an answer
13 s.model() # Get the satisfying assignment
```

In this example, `Int()` creates an integer variable in Z3.[1] On line 6 a constraint was added to the solver, *e.g.*, we constrained the values of x, y and z to be positive. On line 8 an additional constraint is added which states that we are interested only in integers that solve the equation $x^2 + y^2 = z^2$. On line 9 we ask Z3 if the added constraints can be solved, which in this case is true. On line 10 we ask Z3 for the assignment and get: x = 12, y = 9 and z = 15 (your answer may vary). On line 11 we assert that x cannot be equal to 12 (this makes Z3's previous answer illegal). On lines 12 and 13 we check for satisfiability and retrieve the solution which in this case is: x = 8, y = 6 and z = 10.

2 Introducing Python

A good tutorial to learn Python (this is the free version, you can't watch videos and might get pop ups):
http://learnpythonthehardway.org/book/

Read this to learn how to do list comprehensions in Python:
http://www.secnetix.de/olli/Python/list_comprehensions.hawk

More examples of using Python with Z3:
http://cpl0.net/ argp/papers/z3py-guide.pdf

3 Useful Python and Z3 Commands

`Solver()` - Creates a solver for Z3.
`s.add(constraint)` - Adds a constraint to a Solver s.
`s.check()` - Returns `sat` if the current constraints are satisfiable, and `unsat` if the constraints are unsatisfiable.
`s.model()` - Gets the satisfying instance from the solver. Note: this function should only be called if s.check() returned sat.
`Int("name")` - Declares an integer variable in Z3.

The following functions can take either multiple arguments, *e.g.*, `Or(a, b, c)` or a list `Or(my_list)`.
`And()` - Asserts that all arguments given are true.

[1] There is an important distinction between Int variables in Z3 and int variables in other programming languages: In Z3, this declaration is akin to saying that, x for instance, x is an element of the integers in a mathematical sense. In other programming languages, an int simply allocates some amount of space which represents a value in a finite range of integers.

`Or()` - Asserts at least one argument given is true.
`Distinct()` - Asserts that all given variables are distinct.
`Sum()` - Creates a variable which is equal to the sum of the arguments.
`Product()` - Creates a variable equal to the product of the arguments.

Z3 also interacts with Python's comparison and arithmetic operators:

```
from z3 import *
s = Solver()
x = Int('x')
y = Int('y')
z = Int('z')
a = Int('a')
s.add(x != y) # Z3 variables x and y are asserted to be different
s.add(x == 3) # Z3 variable x is now equal to 3. This is different
              # from assignment
s.add(z >= y) # Z3 variable z is asserted to be greater than or
              # equal to y
s.add(y > 7)  # y is asserted to be greater than 7
s.add(a == x + y) # a is now equal to x + y
s.add(x*y > 7)# The product of x and y is greater than 7
```