Z3 Tutorial

Perlen der Logik

Timo Lobitz

22.Juli 2024

Introduction

1. Smartphone company OnePluZ3 is about to launch their new flagship phone
2. You are facing several issues that need to be solved ASAP

Problem 1 - What to produce?

- You can produce 3 different items
  - Phone cases, chargers, and smartphones
- Each take different amounts of resources to produce and generate a different amount of profit
- You have limited labor hours, machine hours and material available

Problem 1 - What to produce?

Resources available:

- 500 labor hours
- 800 machine hours
- 600 units of material

Name	Profit	Labor Hours	Machine Time	Raw Materials
Phone Case	10	3	3	4
Phone Charger	30	5	3	2
Smartphone	50	4	5	6

Problem 1 - Formalization

This can be expressed as a linear programming problem.

$$\max f(x) = 10 * A + 30 * B + 50 * C \tag{1}$$

$$\text{with contraints} \tag{2}$$

$$3 * A + 5 * B + 4 * C <= 500 \tag{3}$$

$$3 * A + 3 * B + 5 * C <= 800 \tag{4}$$

$$4 * A + 2 * B + 6 * C <= 600 \tag{5}$$

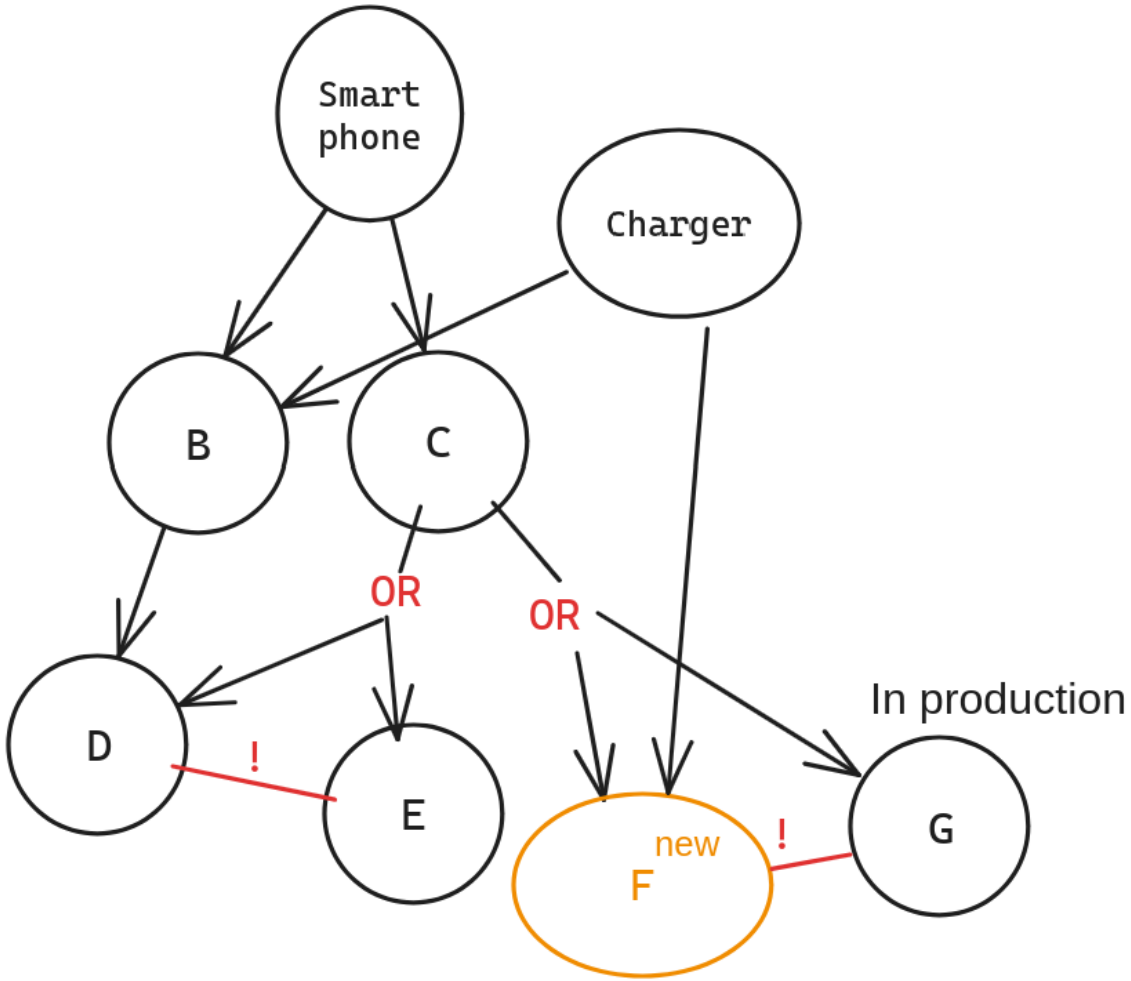
$$A >= 0 \tag{6}$$

$$B >= 0 \tag{7}$$

$$C >= 0 \tag{8}$$

Problem 2 - Dependency Chaos

As part of the production line, you need to manage different parts and chips that are used in different devices.



Problem 2 - Formalization

- Each part is represented by a boolean variable
  - True if in production
  - False if not in production
- A depends on B : $A \implies B$
- A conflicts with B : $\neg A \vee \neg B$

Problem 3 - Code Verification

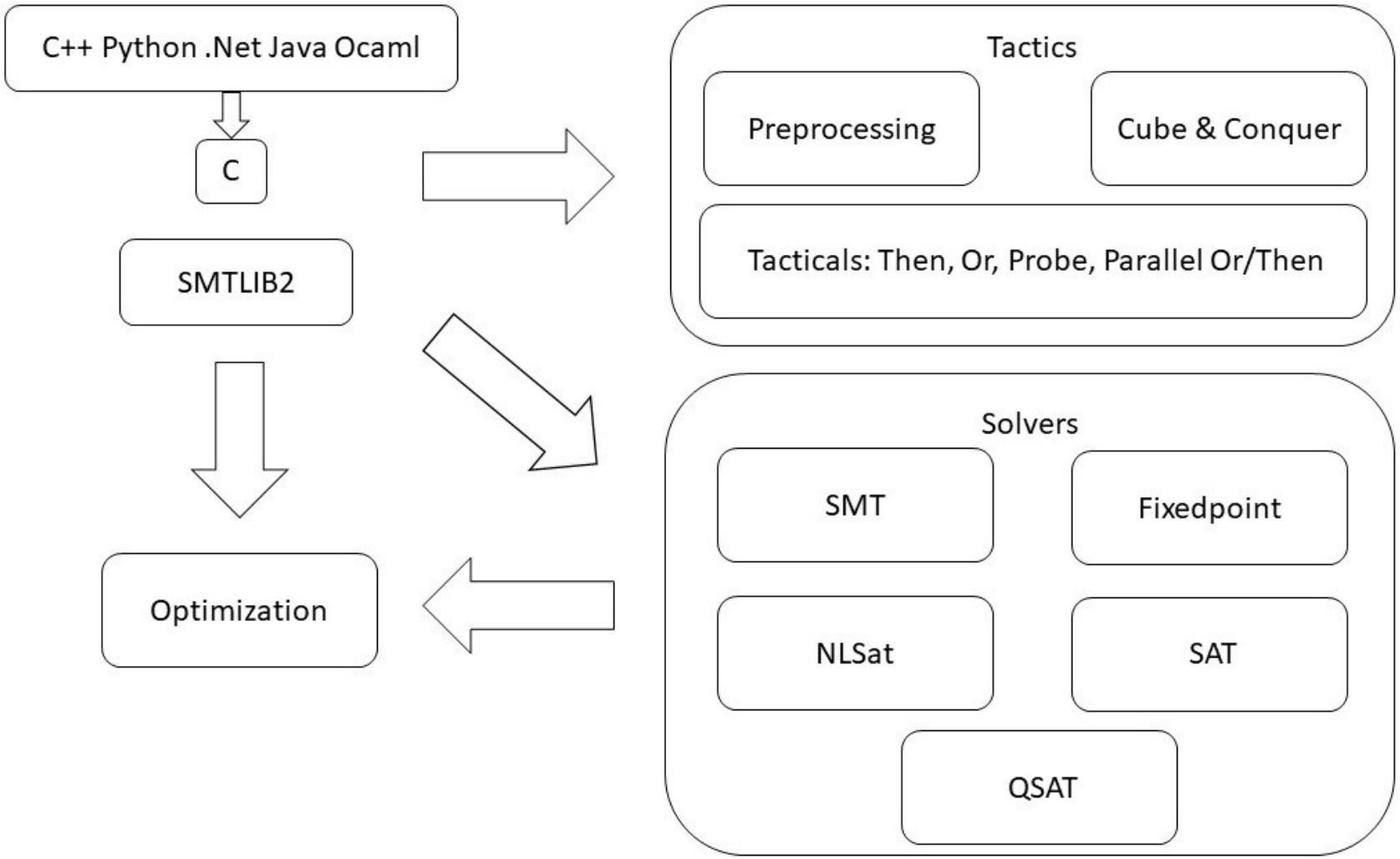
The day 1 patch is currently in code review. You notice a strange function written by a coworker.

```
// Magic function
uint32_t f(int32_t v) {
    int32_t const mask = v >> 31;
    uint32_t r = (v + mask) ^ mask;
    return r;
}
```

Z3

- SMT solver
- open-source
- from Microsoft Research

Z3


C++ Python .Net Java Ocaml
C
SMTLIB2
Optimization
Tactics
Preprocessing
Cube & Conquer
Tacticals: Then, Or, Probe, Parallel Or/Then
Solvers
SMT
Fixedpoint
NLSat
SAT
QSAT

Z3 - Usage

4 Step pattern:

1. Create Solver
2. Define variables
3. Add constraints
4. Check

Z3 - Usage

4 Step pattern:

1. Create Solver
2. Define variables
3. Add constraints
4. Check

```
s = Solver()
x = Real("x")

s.add(3*x + 6 == 0)

if s.check() == sat:
        m = s.model()
        x = m.evaluate(x)

        print(f"{x=}") # x=-2
else:
        print(s.check())
```

Variables

```
# Bool
x = Bool('a')

a = BoolVal(True)
b = BoolVal(False)

# Integers
x = Int('x')
y = Int('y')

a = IntVal(5)
```

```
# Real numbers
x = Real('x')
y,z = Reals("y-z")

a = RealVal(3.141)
b = Q(1,3) # 1/3

# Bit-vector
x = BitVec('x', 16) # 16 Bits
y = BitVec('y', 32) # 32 Bits

a = BitVecVal(16, 32) # 32 Bits
```

Arrays

Arrays in Z3 map from one datatype to another. They support Store and Select operations. Those create new Arrays and don't change the input array.

```
# A is an array mapping from integer to integer
A = Array("A", IntSort(), IntSort())
x = Int('x')

Store(A, x, 10)
Select(A, x) # = A[x]
```

Custom Datatypes - Enum

```
Color, (red, green, blue) = EnumSort('Color',
        ('red', 'green', 'blue'))


print (simplify(green == blue)) # False

c = Const('c', Color)
solve(c != green, c != blue) # [c = red]
```

Custom Datatypes - List

```
def DeclareList(sort):
  List = Datatype('List_of_%s' % sort.name())
  List.declare('cons', ('head', sort), ('tail', List))
  List.declare('nil')
  return List.create()

IntList     = DeclareList(IntSort())
RealList    = DeclareList(RealSort())
IntListList = DeclareList(IntList)

l1 = IntList.cons(10, IntList.nil)
l2 = RealList.cons("1/3", RealList.nil)
```

Conclusion

- Z3 is versatile and can be applied to a lot of problems, such as
  - Software Verification / Static analysis
  - Optimization problems
  - Mathematical proofs
  - ...
- Z3 is an efficient general solver for SAT and SMT problems
- Z3 can be used to determine satisfiability, optimize or proof formulas

References

- Programming Guide by Microsoft https://z3prover.github.io/papers/programmingz3.html
- Python tutorial by Microsoft https://microsoft.github.io/z3guide/programming/Z3%20Python%20-%20Readonly/Introduction
- Z3 - Github https://github.com/Z3Prover/z3
- SAT-SMT by example https://smt.st/SAT_SMT_by_example.pdf
- Z3 API https://z3prover.github.io/api/html/namespacez3py.html

Github

Notebooks used : https://github.com/TimoLob/Z3-Tutorial