[Computational Logic Lab](https://drive.google.com/drive/folders/1QsI7iK8T_OM3f5vP5aOlJOI3YKv56qZ_?usp=sharing)

13/10/2021

Use the SMT-solver z3 to solve the following problems.

# (1)

(1) Check that the following formula is satisfiable and supply a satisfying

assignment:

(p → q) ∧ ¬(p ∧ q) .

Are there more than one assignment satisfying it?

; this is a comment test

;declare constant p q r

(declare-const p Bool)

(declare-const q Bool)

(declare-const r Bool)

; assert (p → q) ∧ ¬(p ∧ q) .

**(assert (and (=> p q) (not(and p q))))**

; we check that is consistent

(check-sat)

; we get the satisfying assignment

(get-model)

sat

(

(define-fun q () Bool

false)

(define-fun p () Bool

false)

(define-fun r () Bool

false)

)

it is satisfiable:

q needs to be false

p need to be false

r need to b false

------------

# (2)

(2) Check that the following formula is a tautology:

(((p → q) → p) → p) .

; this is a comment test

;declare constant p q r

(declare-const p Bool)

(declare-const q Bool)

(declare-const r Bool)

; assert (((p → q) → p) → p) .

**(assert(=> (=> (=> p q) p ) p) )**

; we check that is consistent

(check-sat)

; we get the satisfying assignment

(get-model)

ANSWER:

sat

(

(define-fun p () Bool

false)

(define-fun q () Bool

false)

(define-fun r () Bool

false)

)

it is satisfiable:

q needs to be false

p need to be false

r need to b false

-----------

tautology:

p ^ ¬ p

**(assert(or p not p )**

# (3)

(3) Show that 3 colors are sufficient to color the part of the European map

that includes the following set of countries: Italy, Austria, Switzerland,

France, Germany.

[Hint: see the course notes.]

(declare-const italy1 Bool)

(declare-const italy2 Bool)

(declare-const italy3 Bool)

(declare-const austria1 Bool)

(declare-const austria2 Bool)

(declare-const austria3 Bool)

(declare-const switzerland1 Bool)

(declare-const switzerland2 Bool)

(declare-const switzerland3 Bool)

(declare-const france1 Bool)

(declare-const france2 Bool)

(declare-const france3 Bool)

(declare-const germany1 Bool)

(declare-const germany2 Bool)

(declare-const germany3 Bool)

;declaration of the ternary function xor3: it holds iff just one of its arguments is true

(define-fun xor3 ((x Bool) (y Bool) (z Bool)) Bool

(and

(or x y z)

(not (and x y))

(not (and x z))

(not (and y z))

)

)

(define−fun nand ((x Bool) (y Bool) ) Bool

(or

(and x (not y))

(and (not x) y)

(and (not x) (not y))

)

)

(assert (xor3 italy1 italy2 italy3))

(assert (xor3 austria1 austria2 austria3))

(assert (xor3 switzerland1 switzerland2 switzerland3))

(assert (xor3 france1 france2 france3))

(assert (xor3 germany1 germany2 germany3))

(assert (nand italy1 france1))

(assert (nand italy2 france2))

(assert (nand italy3 france3))

(assert (nand italy1 switzerland1))

(assert (nand italy2 switzerland2))

(assert (nand italy3 switzerland3))

(assert (nand italy1 austria1))

(assert (nand italy2 austria2))

(assert (nand italy3 austria3))

(assert (nand austria1 switzerland1))

(assert (nand austria2 switzerland2))

(assert (nand austria3 switzerland3))

(assert (nand austria1 germany1))

(assert (nand austria2 germany2))

(assert (nand austria3 germany3))

(assert (nand switzerland1 germany1))

(assert (nand switzerland2 germany2))

(assert (nand switzerland3 germany3))

(assert (nand switzerland1 france1))

(assert (nand switzerland2 france2))

(assert (nand switzerland3 france3))

(assert (nand france1 germany1))

(assert (nand france2 germany2))

(assert (nand france3 germany3))

; we check that is consistent

(check-sat)

; we get the satisfying assignment

(get-model)

---------

(define−fun nand ((x Bool) (y Bool) ) Bool

(or

(and x (not y))

(and (not x) y)

(and (not x) (not y))

)

)

**italia - francia**

**italia - svizzera**

**italia - austria**

**austria - svizzera**

austria - italia

**austria - germania**

svizzera - italia

**svizzera - francia**

**svizzera - germania**

svizzera - austria

francia - italia

francia - svizzera

**francia - germania**

germania - francia

germania - svizzera

germania - austria

# 

# 

# 

# (4)

(4) We must schedule the lectures of the following courses:

C=calculus, L=logic, P=programming, H=physics, A=algebra

We have many available classrooms, however only three time slots can be

used:

9am, 10am, 11am

So some classes will be held in parallel. We want to accomplish (if possible)

the following desiderata:

- student X wants to attend all C, H, A;

- student Y wants to attend both L and P;

- student Z wants to attend C and H but cannot come before 10am.

Check whether this is possible or not and - in the affirmative case - show how

to schedule the above five courses.

[Hint: use propositional letters C9, C10, C11 to say that Calculus is

at 9am, 10am, 11am. Then write down formulae saying that exactly one

among C9, C10, C11 hold. Do the same for Logic, Programming, Physics

and Algebra. Finally, formalize the students desiderata: to this aim, notice

that a student cannot attend two different courses if they are held in parallel.

Finally, run z3 to see whether all this is consistent or not. If it is, ask z3 to

give you a model.]

;xor 3 function to assert only one among 3

(define-fun xor3 ((x Bool) (y Bool) (z Bool)) Bool

(and

(or x y z)

(not (and x y))

(not (and x z))

(not (and y z))

)

)

; declare propositional for Calculus

(declare-const C9 Bool)

(declare-const C10 Bool)

(declare-const C11 Bool)

; declare propositional for Logic

(declare-const L9 Bool)

(declare-const L10 Bool)

(declare-const L11 Bool)

; declare propositional for Programming

(declare-const P9 Bool)

(declare-const P10 Bool)

(declare-const P11 Bool)

; declare propositional for Physics

(declare-const H9 Bool)

(declare-const H10 Bool)

(declare-const H11 Bool)

; declare propositional for Algebra

(declare-const A9 Bool)

(declare-const A10 Bool)

(declare-const A11 Bool)

;or 9 or 10 or 11

(assert (xor3 C9 C10 C11))

(assert (xor3 L9 L10 L11))

(assert (xor3 P9 P10 P11))

(assert (xor3 H9 H10 H11))

(assert (xor3 A9 A10 A11))

;student X wants to attend all C, H, A;

(assert (xor3 C9 H9 A9))

(assert (xor3 C10 H10 A10))

(assert (xor3 C11 H11 A11))

;student Y wants to attend both L and P;

(assert (xor L9 P9))

(assert (xor L10 P10))

(assert (xor L11 P11))

;student Z wants to attend C and H but cannot come before 10am.

(assert (xor C10 H10))

(assert (xor C11 H11))

;check if is consistent

(check-sat)

;get satisfying assignment

(get-model)

MANUEL:

(declare-const C9 Bool)

(declare-const C10 Bool)

(declare-const C11 Bool)

(declare-const L9 Bool)

(declare-const L10 Bool)

(declare-const L11 Bool)

(declare-const P9 Bool)

(declare-const P10 Bool)

(declare-const P11 Bool)

(declare-const H9 Bool)

(declare-const H10 Bool)

(declare-const H11 Bool)

(declare-const A9 Bool)

(declare-const A10 Bool)

(declare-const A11 Bool)

(define-fun xor3 ((x Bool) (y Bool) (z Bool)) Bool

(and

(or x y z)

(not (and x y))

(not (and x z))

(not (and y z))

)

)

(define-fun nand ((x Bool) (y Bool)) Bool

(and

(not (and x y))

)

)

(assert (xor3 C9 C10 C11))

(assert (xor3 L9 L10 L11))

(assert (xor3 P9 P10 P11))

(assert (xor3 H9 H10 H11))

(assert (xor3 A9 A10 A11))

(assert (nand C9 H9))

(assert (nand C9 A9))

(assert (nand H9 A9))

(assert (nand C10 H10))

(assert (nand C10 A10))

(assert (nand H10 A10))

(assert (nand C11 H11))

(assert (nand C11 A11))

(assert (nand H11 A11))

(assert (nand L9 P9))

(assert (nand L10 P10))

(assert (nand L11 P11))

(assert (not C9))

(assert (not H9))

(assert (nand C10 H10))

(assert (nand C11 H11))

(check-sat)

(get-model)