Hints of the exercises are on the notes

Paragraph 4.5

Quantifiers, all universal and all in front of the formula

Herbrand theorem

Let T be a set of universal sentences T does not have a model iff there is a finite set of ground instances of T, such that whose propositional abstraction is not satisfiable

Instances: if ∀x1,…,∀xn  A € T a ground instance is a formula of the kind A(t1/x1…tn/xn) where t1,…tn are ground terms

A SMT-solver take the formulas, replace the variable with ground terms, and ask sat fouler it it is satisfiable or not

Propositional satisfiability test can be done by resolution, DPLL…

Depends on how many ground terms there are. If the language contains at least one ground symbol there are infinitely many ground terms. Does not give you an aghortim that terminates

Start producing instances, ask to the SAT-solver if the instances are satisfiable, if no it keep add instances

If the formula are not sat, producing enough instances will be enough to detect unsatisfiability

To sum up: if T is not satisfiable, it is possible to discover this by producing enough ground instances and asking a prepositional solver whether this set of ground instances is proportionality satisfiable

**Important special case**:

If the Herbrand universe (namely the set of ground terms (terms without variables)) is finite, one can produce all finite instances and so Herbrand theorem gives a terminated algorithm

LAB: 3 problems, it has a Herbrand universe that is finite

CHAPTER 6

How to write down in z3 problem with first order

Domain should be declare and give it a name

(declare-sort D)

Declare a constants (a,b,c, pidocchio) and specify the domain D

(declare-const c D)

Declare a binary function f

(declare-fun f (D D) D )

Declare function g unary

(declare-fun g (D) D)

Declare predicate R binary but is a predicate

(declare-fun R (D D) Bool)

P unary predicate

(declare-fun P (D) Bool)

(declare-const p Bool)

P unary predicate and c constant

P(c) is wrong Z3 use (P c)

R binary predicate c,d constant

R (c,d) wrong z3 use (R c d)

How to write down formulas

Use assert

^ and

V or

→ =>

¬ not

P(c) ^ Rc,d)

(and (P c) (R c d))

Quantifiers

∃ x<formula>

(exist ((xD)) <formula>)

∃ x ∃ y <formula>

(exist ((x D) (y D)) <formula>)

Universal quantifiers

∀x <formula>

(forall ((x D)) <formula>)

∀ x ∀ y ∀ z <formula>

(forall ((x D) (y D)(z D)) <formula>)

* List of declaration:
* Sort (domain)
* Constants
* Functions
* Predicate
* List of assertion
* (check-sat)
* (get-model)

(get-value ( <list of terms of formulas> )) ! It is more useful than (get-model)!

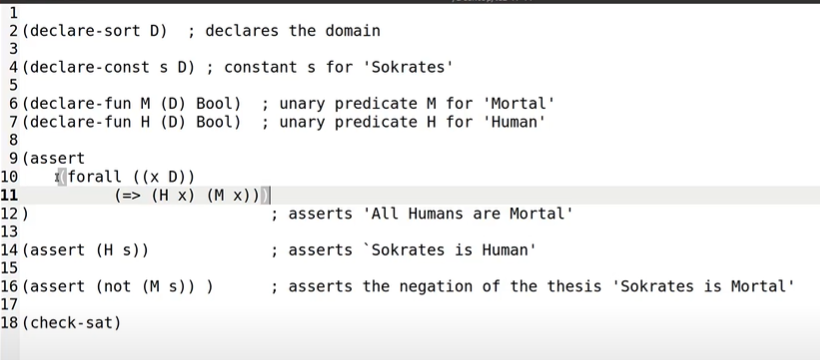
(get-value ((K1 Italy) (K2 Italy) (K3 Italy)))

The answer will be:

(K1 Italy) = true

(K2 Italy) = false

(K3 Italy) = false



!! indentation helps to have a good understanding of the program!!

Negate what you want to prove

Work by absurdity

**Proof of the Herbrand theorem**

T is a set of universal clauses

T is not satisfiable iff there is a finite set of ground instances of clauses of T whose propositional abstraction is not satisfiable

The main idea to prove this

It is possible to show that if T has a model (there is a model that satisfies) iff it has a model bases on the Herbrand universe

A = (A, I)

A is the set of ground terms

For f n-ary function I(f) maps elements t1,…tn into f(t1,…tn)

As a consequence for every ground term t ta = t typical of itself

The evolution of a ground term is the term itself

The interpretation of a predicate R can vary I(R) can vary

In the herbrand universe the interpretation of a function symbol is fixed, but the interpretation of a predicate can vary

Typical if some SAT-solver found an assignment V for all propositional abstraction of ground instances of formulas in T, than

R € Pn r is an n-ary predicate

I(R) = {(t1…tn) | V(PR(t1…tn)) = 1 }

the valuation of the propositional abstraction of R(t1…tn)is 1

If there is identity in the language, you need a literal modification to Herbrand theory

∀x (x = x) (\*)

∀x ∀y ∀z (x = y ^ y = z → x = z) (\*)

∀x ∀y (x= y → y = x) (\*)

∀x1…∀xn ∀y1…∀yn (x1= y1 ^… ^ xn = yn → f(x1..xn) = f(y1…yn)) (\*)

∀x1…∀xn ∀y1…∀yn (x1 = y1 ^...^ xn = yn ^ R(x1...xn) → R(y1...yn)) (\*)

**These are universal**

The herbrand theorem works with languages with identity if you add these sentences in addition

If there is identify in the language just add all the formulas (\*) to T and apply the Herbrand theorem to T ∪ {(\*)}

Not very efficient

But works

19/11/2021

A ↔ B (A → B) ^ (B→ A)

In Z3 use ( = A B) this is useful to know

C(a) → F

¬C(a) → ¬ F

Same as C(a) ↔ F in z3 is ( = (C a) F) only one line instead of 2

**Z3 pop and push**

(Push)

(Assert F1)

…

(Assert Fn)

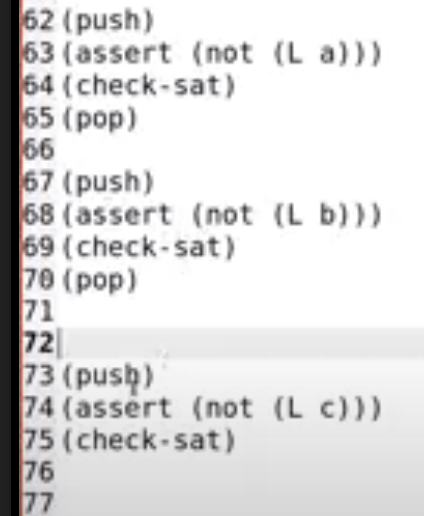
(Check-sat)

(Pop)

Everything between (push) and (pop) is removed, the insite

(Pop) Remove all the assertions before push

retract all assertions below push



Get three answer, try all the cases, ask 3 checks

It is very helpful if you have many solutions

Do a dialogue with the solver, check this, check the other one…

**Syllogism**

Invented by Aristotle

Quite easy for the solver

4 possibile proposition, **aristotle's square**

*All humans are mortal* ∀ x(H(x) → M(x) A

*Some humans are mortal*  ∃ x(H(x) ^ M(x)) I

*No humans is mortal* ∀ ¬ x(H(x) ^ M(x) E

*Some humans are not mortal ∃* x(H(x) ^ ¬ M(x)) O

Don’t consider if the sentences are true or false

A negation of O (and vice versa)

I negation of E (and vice versa)

A and E are universal

I and O are existential

Check:

*all humans are animals*

*All animals are mortal*

*—————*

*All humans are mortal* (as consequences)

syllogism bArbArA (3 affirmative propositions)

Check if the syllogism is ok:

1. Translate formulas in mathematical logic

∀ x(H(x) → A(x)) DON’T PUT AN AND (^) INSTEAD OF → COMPLETELY WRONG STATEMENT. It will be like saying *All x are humans and also animals*

∀ x(A(x) → M(x)

————

∀ x(H(x) → M(x))

I have to negate the thesis

2 kinds of exercise:

1. Prove something (pinocchio, knights and knives). You have to negate the thesis, assert the negation of the thesis and get an unsat of the answer
2. Find something (color countries), don’t need to negate anything

steps

1. take the english sentence
2. translate to mathematical logic
3. use skolna constant to remove existential quantifiers
4. get an universal problem with a finite herbrand universe
5. make all instantiations
6. apply a prepositional method (resolution, DPLL) to see if the set is consistent or not
7. If I get empty clause: syllogism is valid, if I don’t get empty clause then the syllogism is not valid

IMPORTANT

¬ ∀ xA = ∃ x ¬ A

¬ ∃ xA = ∀ x¬ A

∀ x(H(x) → M(x)) negate

∃ x ¬ (H(x) → M(x)) transform in Conjunctive Normal Form

∃ x (H(x) ^ ¬ M(x)) [A → B = ¬ A v B; ¬ (A → B) = A ^ ¬ B]

For satisfiability (or un satisfiability) problem I need to **replace the variable with a new constant**

The constant should be new (use a constant for every existential)

H(c) ¬ M(c) skolna constant

Replacing a variable with a constant preserves satisfiability

H(c) ^ ¬ M(c) now this is + completare

∀ x(H(x) → A(x))

∀ x(A(x) → M(x))

H(c) ^ ¬ M(c)

C is the ground term

1. H(c) → A(c) ¬ H(c) v A(c)
2. A(c) → M(c) ¬ A(c) v M(c)
3. H(c) ^ ¬ M(c) H(c) ¬ M(c)

If I want to use resolution

¬ A(c) v M(c) ¬ M(c)

———————————

¬ A(c)

H(c) ¬ H(c) v A(c)

———————-

A(c)

¬ A(c) A(c)

————————

⬜ Unsat

**Since it is unsat than the syllogism is valid**

I can use also advance DPLL instead of resolution

(ø | ¬ A(c) v A(c) , ¬ A(c) v M(c),H(c), ¬ M(c) | \* ) search state

If you enter a conflict state with no decision (all propagation) than you fail

+finire esercizio

Conflict state with no decision → empty clause

**Another syllogism**

*All humans are mortal*

*Some mortal are animals*

*—————————*

*Some humans are animals*  this syllogism is correct

∀ x (H(x) → M(x))

∃ x(M(x) ^ A(x)) skolar constant M(c) ^ A(c)

—————--

∃ x(H(x) ^ A(x)) negation ¬ ∃ (H(x) ^ (A(x)) = ∀ x ¬ (H(x) ^ A(x))

I have to write instantiations

H(c) → M(c) ¬H(c) v M(c)

M(c) ^ A(c) is ground M(c) A(c)

¬ (H(c) ^ A(c)) ¬ H(c) v ¬ A(c)

4 clauses

I can use resolution or CDCL

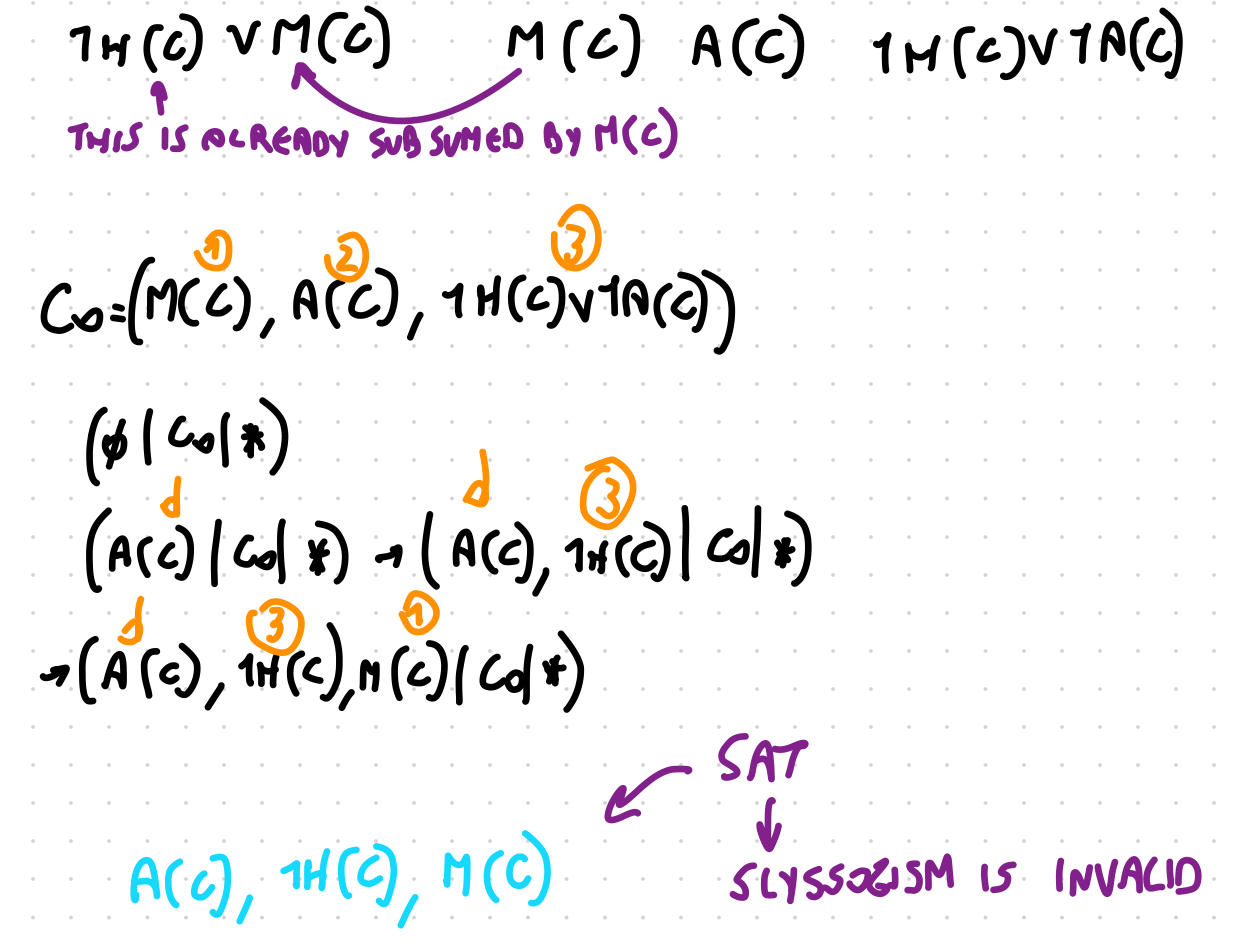
The first one is remove by subsumption because M(c) is true from the second sentence, so ¬H(c) v M(c) must be true

A(c) ¬ H(c) v ¬ A(c)

———————

¬ H(c) I can remove the last clauses by subsumption

I get M(c) A(c) ¬H(c) is SAT, syllogism is not valid



The counterexample of the syllogism A(c) ¬ H(c), M(c)

Is somebody that is animal, not human, but mortal

**The identity**

The propositional clauses that you get as results are all horn clauses

With horns clauses can only use propagation

Consider an equivalence relation on a set

Is a binary relation

∀ xR(x,x) is reflexive

∀ x ∀y( R(x,y) → R(y,x)) is symmetric

∀ x ∀y ∀ z (R(x,y) ^ R(y,z) → R(x,z) is transitive

whenever I have a binary relation, a reflexive relation, a symmetric relation and a transitive relation it is called **equivalent relation**

To have the same age, have the same jackets, same height…

The equivalence relation induces a partition, the set is divided in pieces



I put in the same piece all elements x,y s.t. R(x,y)

If the equivalence relation is to have the same height

In the same group I will put the elements that have the same height



Every element is only in one block and these pieces are disjoin, can’t be in 2 together

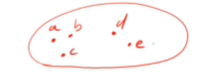
Those blocks are called **equivalent classes**

The set of equivalence classes is called the **quotient set**

If I have a problem with identity

a= b b=c a=c c ≠ d d= e f ≠ a f ≠ c

Herbrand universe: a,b,c,d,e (all ground terms) 5 atoms



↑ this herbrand universe is not a model, because a is not equal to b

To make that true

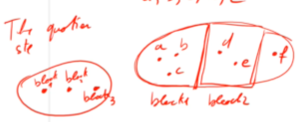
t = u iff t= u holds in the set

Division in the blocks



If idea is to consider the quotient set as a model

The quotient set is a set of three block, one with a,b,c one with d,e and one with f



a= **block 1** = b = c

d = e = **block 2**

f = **block 3**

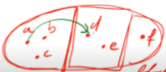
Whenever I have a problem with equivality I consider my problem and in consider I had the axiom (see before)

The axiom are universal, so there is no problem

However it might be a problem with function symbol

If I have a function symbol F

If F(a) = d it is a good idea to map block 1 to block 2



Map block 1 to block 2 Since a = block 1



F(c) = *an element of block 2*, like e it is ok, but if F(c) is equal to an element of block 3 (like f) that I have an inconsistency

F(a) = d

F(c) = f

a = c

however F(a) ≠ F(c) inconsistent (or unsat)

If they are the same element

If F is *father of*

d is block 2

f is in block 3

Need to add the axiom of before and also

**congruence axiom**

∀ x ∀ y. x= y → F(x) =f(y)

∀ x1…∀xn ∀ y1…∀yn (x1= y1 ^ xn = yn → G(x1…xn) = G(y1…yn) with G n-ary

also for predicates

If x = y → (P(x) ↔ P(y))

∀ x1…∀xn ∀ y1…∀yn (x1 = y1 xn = yn → (R(x1…xn) ↔ R(y1…yn))) with R n-ary

**Herbrand theorem with equality**

In conclusion:

A universal set of sentence T (in a language with equalities ) is inconsistent if there is a finite subset of ground instances of T ∪ {reflexivity, transitivity, symmetry, congruence} whose propositional abstraction is inconsistent

Not very efficient, in a first approximation it works but there are better methods

take the universal set of sentences I add reflexivity, transitivity, symmetry, congruence axiom. One congruence axiom for every symbol and one congruence axiom for every predicate symbol