Satisfiability problem

Universal formulas

∀x1… ∀xn F... Check sat?

F is universal (no quantifiers)

Consider all ground terms from herbrand universe and replace x1…xn in F by ground terms in all possible ways, check the resulting ground formulas for propositional satisfiability (resolution, basic DPLL, advance DPLL)

If the herbrand universe is finite, we have finite formulas, we can check all of them

If the herbrand universe is infinite, we get infinitely many formulas, feed the SAT-solver a finite number until we discover the inconsistency. There is the risk of no termination

If we have the identity in the language we can do the same, add some closes reflexivity, transitivity, symmetry and congruence

What about non-universal formulas?

There is a procedure called **skolemization** that transforms such formulas into universal sentences **keeping satisfiability**. The problem will be different

If it was unsat, with skolemization it will still be unsat

It is an algorithmic transformation, we can reduce everything. By skolemization we reduce to universal case by herbrand theorem we reduce to satisfiability

This is a general method that works in any case, but there are better methods

Skolemization have two steps

First part:

**Prenex normal form**

Series of quantifiers

Qx1…Qxn M

M: quantifiers free, called matrix

Qx1…Qxn sequences of quantifiers, called prefix (the quantifiers can be existential or universal or mixed).

!!! If you change the order of quantifiers you change the meaning !!!

∃ x, ∀ y ∀ z ∃ u (Px → R(y,z) ^ S(u))

Prefix

Matrix no quantifiers

**Theorem**

Every sentence (formulas without free variables) is logically equivalent to a sentence in prenex normal form

Algorithm that moves quantifiers outside

Tra formation algorithm is based on rewriting rules that move quantifiers outwords

Can apply the rule in the order you want, depleting on the way the order you get get different result (but only in apparente) the final result will be equivalent result, one formula can have many prenex normals formal equivalent to it

a ^ ∀xb ∀ x (a^b)

correct only if x is not free in a, otherwise the transformation is not correct

You can always rename

P(x) ^ ∀ xR(x,y) ∀ x1(P(x) ^ R(x1,y))

Change the name of all bound variable in such a way there is no free and bound variable

In any step you can rename

∀ x B ^ A ∀ x (B ^ A)

A ^ ∃ x B ∃ x (A^B)

∃ x B^A ∃ x (B^A)

A v ∀ x B ∀ x (A v B)

∀ x B v A ∀ x (B v A)

A v ∃ x B ∃ x (A v B)

∃ x B v A ∃ x (B v A)

¬ ∃ x A ∀ x ¬ A

¬ ∀ x A ∃ x ¬ A

A → ∃ x B ∃ x (A→ B)

A → ∀ x B ∀ x (A → B)

∃ x B → A ∀ x (B → A)

∀ x B → A ∃ x ( B → A)

It is useful to have

∀ x B ^ ∀ x A ∀ x (B ^ A)

∀ x (B ^ ∀ x A)

∀ x (B1 ∀ x1 A(x1/x))

∀ x ∀ x1 /B ^ A (x1/x)) Is logical equivalent to ∀ x (B^A) but the two formulas are not the same

∃ x B v ∃ x A ∃ x ( B v A)

**X must not be free in A**

(Qx1)…(QxN) M can out the matrix M in CNF using propositional transformations

Remove existential quantifiers carefully, each time we remove a ∃ we add to the language a **new** function or constant symbols - maintains satisfiability, not logical equivalence

∀ x1… ∀ xk ∃ y F

Take a new functions f symbols of arity k

Then ∀ x1… ∀ xk f(f(x1…xk)/y)

can have many possible y

same as ∀ ε ∃ δ δ (ε)

If k = 0 then f is a new constant

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

We name x as c

H(c) ^ M(c) some means at least one

Keep satisfiability

∀ x1 ∀ xm M Put matrix in CNF, so M = C1 ^ …Cn

(∀ x1 ∀ xN C1)^… ^ (∀ x1 ∀ xn Cn) universal quantified clauses, apply herbrand

Maybe cause the herbrand theorem to become infinity

D = {a,b,c} finite domain

∀ x P(x)

P(a) ^ P(b) ^ P(c)

∃ x P(x)

P(a) v P(b) v P(c)

26 nov 2021

*All men* (human = U) *are mortal*

∀ x (U(x) → M(x)) is an implication not an and ^

**First exercise**

*All men are mortal* ∀ x(U(x) → M(x))

*No mortal is mortal* ¬ ∃x(M(x) ^ P(x))

————————-

*No man is perfect* ¬ ∃ x(U(x) ^ P(x)). The thesis need to be

negated

¬¬ ∃ x(U(x) ^ P(x)) ∃ x(U(x) ^ P(x))

**Second exercise**

CHECK *CAI members* on Google Doc

[CAI members](https://docs.google.com/document/d/1DekFytrBnjAztckinritwufYDTQ3fwQKtvidAvLcUAg/edit?usp=sharing)

**Page 81**

(Declare-datatypes()(S a b c))

S = {a,b,c}

And

A ≠ b a ≠ c b ≠ c

With just one line

(Decare-sort S)

(Declare-const a S)

(Declare-const b S)

(Delcare-const c S)

(Assert (distinct a b c)) is like saying

S contains a,b,c (and all different) (but can contains others too)

And

A ≠ b a ≠ c b ≠ c

If you want to make s to contains precisely (just) a,b,c

Use the first data

Check Albert einstein riddle

Example 24

∀x ∀y (R(x,y ) → R(y,x) symmetric

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

∀x ∃y R(x,y) reflexive

————-

∀xR(x,x) need to prove this

Negate the thesis

¬∀xR(x,x) = ∃ ¬R(x,x)

* transform the formulas in prenex normal form
* Need to skolemize it (remove ∃ )
* Put matrix in CNF
* Apply herbrand: instantiate ∀ with ground terms.
* Use SAT-solver

1. ∀x ∀y( ¬ R(x,y) v R(y,x)

2. ∀x ∀y ∀z ( ¬ R (x,y) v ¬ R(y,z) v R(x,z))

3. ∀x R(x,f(x)) why f(x)

—————

4. ∃ x ¬ R(x,x) that becomes ¬ R(c,c)

Ready for instantons

Now the herbrand universe is infinite, it contains c,f(c), f(f(c))…

From clause 3. x→ c R(c,f(c))

From clause 1. x → c y → f(x) ¬ R(c,f(c)) v R(f(c),c)

From clause 2. x→ c y→ f(c) z→ c ¬ R( c, f(c)) v ¬ R(f(c),c) v R(c,c)

R(f(c),c) ¬ R(f(c),c) v R(c,c))

R(c,c)

empty clause

UNSAT

**So the thesis is right**

Two ways:

Improve instatianton (use resolution non ground). Saturation-based theorem proved

Or

Combine instantiation and decision procedure (CDCL(T)). Satisfiability modulo theories (SMT)

Persons, cigarettes, animas, houses, drinks

2 people can't have the same house, animal…

Better to have multiples domain

The herbrand universe is finite

*The German lives just near the Italian*

h(g) = h(i)+1 v h(i) = h(g) +1 h = house

Z3 supporters arithmetics

We have specific decision procedures for particular domains (integers, reals, equality, bitvectors (0,1), arrays, lists)

In z3 for each of this domain there are specific procedures, combined with each other

Put together with logic (SAT and herbrand)

Libera arithmetic +,-,0,1,<, =

Real or integers

**Fourier-motzkin algorithm**, is not the best one, but it easy to explain and powerful (can handle quantifiers)

For integers is the cooperer algorithm but is difficult, not in this course

DPLL solver, takes propositional abstractions of the item and assign them to truth value

x > 0 x< -1

P Q

Can’t be true together

Once I have a propositional assignment I must check whether it is good or not **in my specific domain**

Need a way to recognise when an assignment is good or not

In general: given a finite set of literal interpreted in my specific domain, I want a procedure to check satisfiability **in the domain**

**Example**: Linear equalities and inequalities between the reals

Solved by simplex algorithm but also by Fourier-motzkin

∃x ∃y (3y + x < 0 ^ x+6-2y>0 ^ y>0 ^ x<1)

Check satisfiability means to check when it is satisfiability

Remove quantifiers (up to to equivalence) until I do not have any quantifies more

Remove x

x<-3y ^ x> -6+2y ^ x<1 ^ y>0

x is less than -3y and 1 and bigger than -6+2y

If x exist then

-6+2y<1 ^ -6+2y < -3y ^ y>0 Take one element below x and one above x

Do the same for y

2y < 1+6 ^ 5y<6 ^ y>0

y<7/2 ^ y<6/5 ^ y>0

0 < 7/2 ^ 0<6/5 is true Hence SAT

Remove variables, util i have a relation with just number, simple but slow

**Quantifiers elimination**

Can be apply to any type of formulas

∀ x same ¬ ∃ x ¬

I can take any formulas with quantifiers and remove all of them