*Mike*

*Besh*

*Tito N.*

*Lele*

*British*

*Utto*

FOLLOW US ON INSTAGRAM: [@massoneria.ai](https://www.instagram.com/massoneria.ai/)

***All rights reserved to @massoneria.ai***

# **COMPUTATIONAL LOGIC DEFINITIONS**

# NNF (Negation Normal Form)

No implications and all the negations are in front of atoms

((¬P ∧ (Q ∨ R)) ∨ S) is NNF but not CNF

(¬P ∧ Q) ∨ (¬P)

# CNF (Conjunction Normal Form)

It's a conjunction of clauses (clause: disjunction of literals)

(P ∨ Q) **∧** (P ∨ ¬Q)

# HORN CLAUSES

It's a clause with **at most one positive literal.**

¬P ∨ ¬Q is a Horn Clause

¬P ∨ Q ∨ ¬R is a Horn Clause

P ∨ Q ∨ R is NOT a Horn Clause

# LOGIC EQUIVALENCE

Two formulas A, B are said to be logically equivalent iff A ↔ B is a tautology. Iff we have V(A) = V(B) for every assignment V. **Same truth table!**

# EQUI-SATISFIABLE

Two formulas A, B are said to be **equisatisfiable iff A is satisfiable iff B is satisfiable**

**(A → B)**

# SEARCH STATE

When you are in “search state” you have 3 possible cases:

* **look for a conflict**: *see conflict state*
* **propagation**: when you can learn a new literal using the set of partial assignments applied to the set of clauses.
* **decision**: when you made all the possible propagations, if you haven’t evaluated all the literals yet, you can decide arbitrarily a value for one of the remaining ones (i.e. V(p)=1) and go on with a new search state (so conflict, propagation…)

# CONFLICT STATE

You enter in “*conflict state*” when you have some literals (propagated or decided) that are in contradiction with a clause of the set. In case of conflict with no decided literals, we can say that the set of clauses is UNSAT; otherwise, a conflict with at least one decided literal oblige you to go on with the backjumping rule.

# BACKJUMPING

After the conflict state, you learn a new clause that must contain the negation of the literal that you have decided before (otherwise if it is a negation of a propagated literal or clause you will have unsat).

When you apply backjumping you don’t consider anymore what you have propagated after the conflict decision and substitutes the decided literal with the learned literal or clause. After that, you can continue with the procedure.

(i.e <V(p)[d] | C ∧ ~V(p)[*n*] | \*> → <V(p)[*n*] | C | \*>)

# EXPLAIN RULE

The explain rule is applicable right after the conflict rule, furthermore, it is always applicable as long as the literal of the conflict clause (whose negation was assigned last) was assigned subsequently to the last literal decided.

# T-PROPAGATE

If you are in the state ( ∨ | F | \* ) [search state] we can pass to the state ( V, l | F, D ∨ l \* ) if ¬D ∧ is not T-satisfiable, ∨ ⊨ ¬D and l is not defined in ∨ (but occurs in F)

# 

# HERBRAND

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. The risk of no termination is real!

Let T be a set of universal sentences; then T is unsatisfiable (i.e. it has no model) if and only if the set gr(T ) of the ground instances of the sentences of T is unsatisfiable.

Skolemization transform non-universal formulas in universal sentences, keeping satisfiability

# SUBSUMPTION AND UNIT RESOLUTION, DIFFERENCES

| **Subsumption** | **Unit resolution** |
| --- | --- |
| p ∨ q ∨ ¬r if V(p)=1 I can remove the clause | p ∨ q ∨ ¬r if V(p)=0 I can remove the literal, that becomes q ∨ ¬r |
| If I remove all clause, empty set of clause | If I remove all the literals, empty clause ⬜ |
| SAT | Failed branch |

# SYLLOGISM’S STEPS

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

# 

# PNF (Prenex Normal Form)

Qx1, Qx2, Qx3 M

Q: quantifiers (or ∃ or ∀ )

M quantifiers free, called matrix

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

Transformations:

| a ∧ ∀xb | ∀x (a ∧ b) |
| --- | --- |
| ∀xB ∧ A | ∀x(B ∧ A) |
| ∃xB ∧ A | ∃x(B ∧ A) |
| A ∨ ∀xB | ∀x(A ∨ B) |
| ∀xB ∨ A | ∀x(B ∨ A) |
| A ∨ ∃xB | ∃x(A ∨ B) |
| ∃xB ∨ A | ∃x(B ∨ A) |
| ¬∃x A | ∀x ¬A |
| ¬∀x A | ∃x ¬A |
| **A → ∃xB** | **∃x(A→B)** |
| **A → ∀xB** | **∀x(A→B)** |
| **∃xB → A** | **∀x(B→A)** |
| **∀xB → A** | **∃x(B→A)** |
| ∀xB ∧ ∀xA | ∀x(B ∧ A) |

The transformations can be applied in any order, the result may not be the same in different execution of the procedure, but in any case, you get a formula that is logically equivalent to the input one.

**You can always rename!**

(i.e. P(x) ∧ ∀xR(x,y) → ∀ x1 (P(x) ∧ R(x1,y)) )

**Change all bound variables’ names in such a way there is no free and bound variable!**

# SKOLEMIZATION

With this procedure, we substitute the existential sentences with universal sentences logically equivalent, introducing the Skolem function (that can also have arity 0, so constant). The arity of the function is strictly related to the number of universal quantifiers before the existential that we want to substitute.

# 

# DEFINITIONS WITH A SET OF SENTENCES

Taking T as a set of sentences:

* a Model of T is a structure A such that A (entails) B (where B is a sentence) for all B (in) T;

MODEL : STRUCTURE ENTAILS SENTENCE;

* B (sentence) is a Logical consequence of T iff B is true in all models of T;

LOGICAL CONSEQUENCE: SET OF SENTENCES ENTAILS A SENTENCE;

* B is a Logical Truth iff B is true in all structures;

LOGICAL TRUTH: NOTHING ENTAILS SENTENCE.

| **Grammarly correct** | Check the arity of predicates and of the function |  |
| --- | --- | --- |
| **Sentence** | formula in which no variable is free | If it is not a sentence then there is at least one variable that is not free |
| **Open** | At least one free variable | If it is not open then there is no free variable |
| **Ground** | without using variables | If it is not ground than there is at least one variable |
| **Universal** | Only universal quantifiers | If it is not universal that there are no quantifiers or there is at least one existential quantifiers ( ∃ ) |

| **Free** | A variable without ∃ or ∀ | A variable is not free (bounded) if it has ∃ or ∀ |
| --- | --- | --- |

# IDL

[*structure L* = (*A*; *J*)]

**predecessor**: P in *F(n)*, J(P)(n) = n - 1 (dal nome: *n* che hai - 1)

**successor**: S in *F(n)*, J(S)(n) = n + 1 (dal nome: *n* che hai + 1)

# Z3 TRICKS

*A iff B is* A ↔ B is (A → B) ∧ (B → A) this is equal to “e viceversa”

*if then else*  is (ite <condition> value1 value2)