8/10/2021

Want to check satisfiability of A

* transform in NNF
* Transform in CNF
  + Structural
  + Non structural (suggested one for the exercise)
* Rewrite clauses as sequences. It is done for a conventional way
* Saturate with the resolution rule. Add to your set of clauses all closes that are contained in the resolution law
* When there is nothing to do you get (you should do ALL possible applications)
  + SAT if you not get ⟹ (empty close)
  + UNSAT if you get ⟹ (empty close)

Γ, p ⟹ Δ Γ ‘ ⟹ p, Δ ‘

(Remove p)

———-

Γ Γ ‘ ⟹ Δ Δ ‘

If same appears it can be removed

Applausi same simplification [saturation up to redundancy]

Γ , p = > p, Δ is useless

if you have Γ ⟹ Δ and Γ Γ’ ⟹ Δ Δ’ the second is redundant and can be removed( *subsumed* is the tecnica therm)

SPASS logic (German tool than can be used)

EXeRCISE ON GOODNOTES

**Theorem**

If a sat of saturated clauses does not contain the empty clause ( ⟹ ) then it is satisfiable

Γ, p ⟹ Δ Γ ‘ ⟹ p, Δ ‘

(Remove p)

———-

Γ Γ ‘ ⟹ Δ Δ ‘

This clauses is a logical consequence of the premises in the sense that if an assignment V satisfies the premises then it also satisfies the conclusion

Suppose that this is not true: then the conclusion is false, then everything on the left of the arrow is true, everything on the right of the arrow is false

A sequence is false only when the atom before the arrow are true and the atom after the arrow are false

If V does not satisfied the conclusion then all Γ , Γ ‘ are true and all Δ, Δ ‘are false, but

V(p) = 1 in this case Γ , p ⟹ Δ is not satisfy

V(p) = 0 then Γ ‘ ⟹ Δ ‘p is false

Whenever I add a resolver I add a logical consequence

PROVE THAT A IS A TAUTOLOGY

You need to apply the above procedure (NNF, CNF, SATURATION) to **¬ A**

* if you get ⟹ (empty clause) than A is tautology, otherwise is not

Exercise 2