Revision Notes

14 April 2019 12:43 PM

Well formed formulae:

i) $\bot$ & $\top$ are well formed formulas (wff)
ii) Every Propositional variable is a wff
} atoms / atomic formulae

iii) If ϕ & ψ are wff then so are:

$$\neg\psi,\ \phi \wedge \psi,\ \phi \vee \psi,\ \phi \rightarrow \psi,\ \phi \leftrightarrow \psi$$

iv) Nothing else is a wff

An interpretation, $\mathcal{I}$, is a function or Mapping which assigns an atom a truth value

$$\mathcal{I}(P) \in \{T, F\}$$

- If $\mathcal{I}(P) = T$, P is true under the interpretation, $\mathcal{I}$
- If $\mathcal{I}(P) = F$ ....

A Model is defined as follows:

Let P be an atom, let $\mathcal{I}$ be an interpretation for P

$\mathcal{I}$ is a Model for P iff $\mathcal{I}(P) = T$

in such a case we right:

$\mathcal{I} \models P$ ("models")

We can define our semantics of connectives as:

Let $\mathcal{I}$ be an interpretation for a set of atoms $\mathcal{F}$,
let $\phi, \psi \in wff_{\mathcal{F}}(\mathcal{F})$

- $\mathcal{I}(\bot) = F$, $\mathcal{I}(\top) = T$
- $\mathcal{I} \models \neg\phi$ iff $\mathcal{I} \not\models \phi$
- $\mathcal{I} \models \phi \wedge \psi$ iff $\mathcal{I} \models \phi$ and $\mathcal{I} \models \psi$
- $\mathcal{I} \models \phi \vee \psi$ iff $\mathcal{I} \models \phi$ or $\mathcal{I} \models \psi$
- $\mathcal{I} \models \phi \rightarrow \psi$ iff $\mathcal{I} \not\models \phi$ or $\mathcal{I} \models \psi$
- $\mathcal{I} \models \phi \leftrightarrow \psi$ iff $(\mathcal{I} \not\models \phi$ and $\mathcal{I} \not\models \psi)$ or $(\mathcal{I} \models \phi$ and $\mathcal{I} \models \psi)$

Important Notions

Important Notions

Satisfiability - A sentence, s, is satisfiable iff $\exists \mathcal{I} \models s$

Falsifiability - A sentence, s, is falsifiable iff $\exists \mathcal{I} \models \neg s$

Unsatisfiability - A sentence, s, is unsatisfiable if it has no models, $\neg \exists \mathcal{I} \models s$

Validity - A sentence, s, is valid iff every interpretation is a model

$$\forall \mathcal{I}. \mathcal{I} \models s$$

Tautology - if a sentence is valid, it is a tautology

Contradiction - if it is unsatisfiable

Contingent - if it is both satisfiable & falsifiable

Semantic Consequence

- Let Δ be a finite set of sentences. Let ϕ be a sentence
-
- We say Δ semantically entails ϕ if for every interpretation, $\mathcal{I}$, where $\mathcal{I} \models \Delta$ we also have $\mathcal{I} \models \phi$

We write this as

$$\Delta \models \phi$$

if $\mathcal{I} \models \Delta$, and $\Delta = \{\phi_1, \phi_2, \ldots, \phi_n\}$
$\mathcal{I} \models \phi_1, \mathcal{I} \models \phi_2, \ldots, \mathcal{I} \models \phi_n$

Equivalence

We say two formulas ϕ & ψ are equivalent if

$$\{\phi\} \models \psi \text{ and } \{\psi\} \models \phi$$

(semantic entailment)

We write $\phi \cong \psi$

Syntactic Consequence

Let Δ be a finite set of sentences. Let ϕ be a sentence
Let S be an inference System

(e.g. natural deduction or PropRes)

We say Δ syntactically entails ϕ if there is a valid sequence of rules from S that allows us to derive ϕ from Δ

We will write

$$\Delta \vdash_S \phi$$

$$\Delta \models_S \phi$$

Normal forms

- There are several functionally complete sets of connectives
- Minimal sets, however, often produce unaccommodating structures with lots of negation
- If we relax our minimality, we can create normal forms of simple, uniform structure.

- A Literal is either an atom or the negation of an atom.

Disjunctive normal form – A formula, ϕ, is in DNF iff it is of the form $\psi_1 \vee \ldots \vee \psi_n$ $(n \geq 1)$ where each ψ_i is a Conjunction of literals

$$\bigvee_{i=1}^{n} \bigwedge_{j=1}^{n} l_{ij}$$

Conjunctive Normal form – We say a formula, ϕ, is in CNF iff it is of the form $\psi_1 \wedge \ldots \wedge \psi_n$ where each ψ is a disjunction of literals

(Conjunctive)

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} l_{ij}$$

Clausal Normal form – A formula is in Clausal Normal form if it is in Conjunctive Normal form & represented as a set of clauses

We will use this form for the rest of the module.

$$(P \vee \neg Q \vee R) \wedge (\neg P \vee Q)$$

$$\Downarrow$$

$$\{P, \neg Q, R\}, \{\neg P, Q\}$$

Computing Normal forms

1) Eliminate $\leftrightarrow$ and $\rightarrow$ connectives

2) Use laws of double negation, De Morgan's 1 & 2 repeatedly until all negations are immediately in front of atoms

3) Use laws of Distributivity for Conj & Disj until

3) Use laws of Distributivity for Conj & Disj. until desired clausal form is reached.

A Theorem in a logical Calculus is a formula or sentence that has a proof in that Calculus.

Using this we can now express Soundness & Completeness as a relation between the Syntactic & Semantic notion

Soundness — Let S be an inference system

S is Sound if every sentence in S is also a Valid tautology.

$$\vdash_S \phi \longrightarrow \models \phi$$

Essentially, every formula that has a proof in S is valid under all interpretations.

Completeness — Let S be an inference system, S is complete if every tautology of S is also a theorem of S

Shown as:

$$\models \phi \longrightarrow \vdash_S \phi$$

One Wants an inference system to be Sound & Complete although we can ignore Completeness in order for a system to be more Computable

Resolution

- Efficient logical Calculus
- uses the principle of Modus Ponens

$$\frac{P \quad P \rightarrow Q}{Q}$$

Using Clausal normal form

$$\frac{P \quad \neg P \vee Q}{Q}$$

The Resolution Rule

- Expresses the fact that for any interpretation, I to satisfy both P and $\neg P \vee Q$ we have to have $I(P) = T$ ∴ the satisfiability of the second formula depends on if $I \models Q$

We can express this as

$$\frac{\{l\} \cup C_1 \qquad \{\neg l\} \cup C_2}{C_1 \cup C_2}$$

resolvent

Where C_1, C_2 are clauses and l is a literal

- The conclusion of resolution is satisfiable. iff the premises are satisfiable.

The Resolution rule is Sound

Denote $\{\}$ as $\square$

Resolution Procedure

- Exhaustive application of the Resolution rule.
  - Either we have derived $\square$ or no new clauses can be derived.

1. let Φ_0 be the initial set of clauses
2. for Φ_i, $i \geq 0$, Choose 2 clauses $C_1, C_2 \in \Phi_i$ that have not yet been resolved & Contain One Complementary literal, let C be the resolvent clause
3. Set $\Phi_{i+1} = \Phi_i \cup \{C\}$
4. if $C = \square$, terminate with Φ Unsatisfiable
5. If $\Phi_{i+1} = \Phi_i$ then terminate with Φ Satisfiable
6. Else

The Resolution Procedure is Not Complete

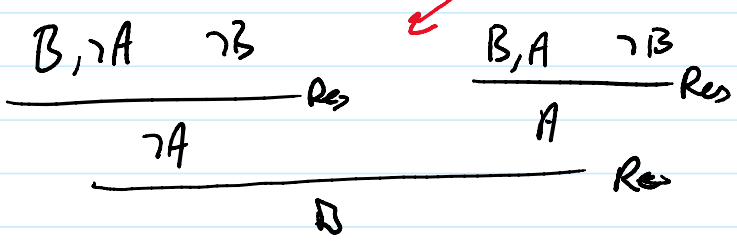
Refutation

To Make resolution Complete we use it as a refutation procedure rather than an Proof Procedure

- We instead try to derive $\square$ from the negation of the Conclusion and original clauses.
- Exploits the fact that a Sentence is Valid, iff its negation is Satisfiable

Proofs Can be written using derivation trees, or linearly

Proofs can be written using derivation trees, or linearly



1. B, ¬A
2. B, A
3. ¬B
4. ¬A Res 1,3
5. A Res 2,3
6. D Res 4,5

Soundness of Resolution Refutation

follows from soundness of Resolution. However, instead of a resolution tree showing Satisfiability we use a refutation tree to show Unsatisfiability.

if $\vdash_{res} \phi$ then $\models \phi$

- $\vdash_{res}$: Syntactic Consequence (i.e. can derive ϕ - refutation tree for $\neg\phi$)
- $\models$: Semantic consequence. ϕ is a tautology

if ∃ a refutation tree for a formula ϕ, ϕ is Satisfiable

Completeness of Refutation

if a formula ϕ is Unsatisfiable then ϕ has a refutation tree

if $\models \phi$ then $\vdash_{res} \phi$

Let ϕ be unsat. then there is a refutation tree for ϕ

Proof:

Let Φ be the CNF of ϕ

Base Case:

There exists only one Prop. Variable, P, in Φ

Since Φ is Unsat both $\{P\}, \{\neg P\} \in \Phi \xrightarrow{res} \square$

Inductive Hypothesis:

if Φ is unsat and contains n variables, then there exists a refutation tree

Step Case:

Let Φ be Unsat. containing n+1 variables
Let P be a Prop. Var. $\in \Phi$

Construct Φ' as follows:

1) Add every $C \in \Phi$ that neither contains P or $\neg P$ to Φ'

to Φ' 0

2) Let $C, D \in \Phi$ be clauses such that $P \in C$, $\neg P \in D$, then add the resolvent of C, D on P to Φ'

Since Φ is UNSat., so is Φ'

Φ' Contains at most n variables

Using the Inductive hypothesis we can !.
derive Completeness.

The DPLL Procedure

1.) Input is a set of clauses Φ, Output is a Model, $\mathcal{I}$, or UNSAT

2) If $\mathcal{I}$ models Φ then return $\mathcal{I}$

3) If Φ contains the empty clause, $\square$, backtrack

4) Unit Propagation - if $\{l\} \in \Phi$ then extend $\mathcal{I}$ with $\mathcal{I}(l) = T$.
Delete all clauses in Φ containing l, Delete all occurances of $\neg l$ in all clauses in Φ

5.) Pure literal - If l is a literal that only occurs in one Polarity in Φ, extend $\mathcal{I}(l) = T$, delete all Clauses in Φ containing l

6) Splitting - Choose a Prop var. $P \in \Phi$ not yet fixed in $\mathcal{I}$

a) - Reduce Φ and $\mathcal{I}$ extended by $\mathcal{I}(P) = T$ ⟶ Remove every Clause containing P, remove all occurances of $\neg P$

b) - Reduce Φ and $\mathcal{I}$ extended by $\mathcal{I}(P) = F$ ⟶ Remove every clause containing $\neg P$, remove all occurances of P.

First Order logic

- Propositional logic is not rich enough to express everything we want to
  - eg. Intelligent Discourse or Reasoning

- Example of a Sentence we cannot express in Prop. logic:

"Socrates is a Man, all Men are Mortal,
therefore, Socrates is Mortal"

Syntax

- Main Purpose of 1st order logic = Make statements about abstract entities in a given domain / universe

distinct entities in a given domain / universe

- Constants
  - Represent concrete individuals in our universe of discourse

- Functions
  - Maps individuals in the universe onto other individuals
    → Thereby, relating them to eachother
  - Functions have different arities (how many arguments they take)

- Variables
  - Symbolic representations of an entity that is not (yet) determined

Terms

- Let V, C, F be disjoint sets of symbols, representing Variables, Constants and functions, respectively

We can inductively define terms as follows:

1. Any Variable $\in V$ is a term
2. Any Constant $\in C$ is a term
3. If $t_1, \ldots, t_n$ are terms and $f \in F$ has arity n then $f(t_1, \ldots t_n)$ is a term
4. Nothing else is a term

Predicates

Predicates in 1st order logic replace Propositional variables in Propositional logic.

Predicates are similar to functions
- Have an arity
- View Propositional variables as Predicates with an arity of 0.

Atomic Formulae

Let P be a set of symbols representing Predicates

i) Let $P \in P$ be a Predicate taking n arguments and $t_1, \ldots, t_n$ are terms. Then $P(t_1, \ldots, t_n)$ is an atomic formula or atom

atomic formula or atom

∴ Nothing else is an atomic formula

Ground Terms & Atoms

A term is ground if it does not contain any variables

An atom is ground if it contains only ground terms

Free & bound Variables

We call a variable bound if it is either substituted by a specific value or restricted to a set of values

Quantifiers

- A way to bind variables
- We will focus on Universal, $\forall$ and existential, $\exists$
- The Scope of a quantifier is how far the influence of a quantifier reaches in our formula

Well formed formulae in First Order Predicate Logic

We define the set of WFF as follows:

i) $\top$ and $\bot$ are well-formed formulae
ii) Every atomic formula is a Wff
iii) if ϕ and ψ are wff then so are:
$\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$, $\phi \leftrightarrow \psi$

iv) if ϕ is a wff and $x \in V$ is a variable then
$\exists x.\phi$ and $\forall x.\phi$ are wff

v) Nothing else is a well formed formula

Semantics

In Prop. Logic we assigned meaning to our Prop. Vars using interpretations to map them into truth values.

In 1st order Predicate Calculus we do something similar but we must first interpret our terms. We do this by Mapping our terms into a Universe

↑
a non-empty set, U

Interpretation = Set U by a mapping of

Interpretation

- Let U be a non-empty set called Universe. An interpretation, $\mathcal{I}$, over U is a function that:

i) Maps each $c \in C$ onto an element of U

ii) Maps each $f \in \tilde{F}$ with n arguments to a Concrete function $f^{\mathcal{I}}: U^n \to U$
(where U^n = the set of n-tuples over U)

iii) Maps each $P \in \tilde{P}$ with $U \times U$ n arguments to a concrete function $P^{\mathcal{I}}: U^n \to \{T, F\}$

→ a Predicate Maps an element of U to a bool. A predicate is just a subset of the Universe

Variables

- Need special treatment

- Cannot be simply interpreted
  - Have to account $\forall \mathcal{I}, \alpha, U$

- We use a function called variable assignment

Variable Assignment - Given a universe U, and a set of variables V
We call the function $\alpha: V \to U$ a variable assignment
For a concrete variable $x \in V$ we will denote an assignment under α by $x_\alpha \in U$, i.e. $\alpha(x) = x_\alpha$

Semantics over terms

Let $\mathcal{U}$ be a non-empty universe, $\mathcal{I}$ interpretation and α a variable assignment over $\mathcal{U}$. Then we have:

i) $x \in V$ evaluates to $\mathcal{I}_\alpha(x) = \alpha(x) = x_\alpha \in \mathcal{U}$

ii) $c \in C$ evaluates to $\mathcal{I}_\alpha(c) = \mathcal{I}(c) = c_{\mathcal{I}} \in \mathcal{U}$

iii) With terms $t_1, \dots, t_n$ and $f \in F$ we evaluate $\mathcal{I}_\alpha(f(t_1, \dots, t_n))$ recursively as:

$$\mathcal{I}(f)(\mathcal{I}_\alpha(t_1), \dots, \mathcal{I}_\alpha(t_n)) = f_{\mathcal{I}}(\mathcal{I}_\alpha(t_1), \dots, \mathcal{I}_\alpha(t_n)) \in \mathcal{U}$$

Models in 1st Order Predicate Calculus

Let $P(t_1, \dots t_n)$ be an **atom**, let $\mathcal{I}$ be an interpretation over a universe $\mathcal{U}$ and α a variable assignment over $\mathcal{U}$. Then $\mathcal{I}_\alpha$ is called a **Model** for $P(t_1, \dots, t_n)$ iff $\mathcal{I}(P)(\mathcal{I}_\alpha(t_1), \dots, \mathcal{I}_\alpha(t_n)) = T$.

We write: $\mathcal{I}_\alpha \models P$ $\left(\mathcal{I}_\alpha \models P(t_1, \dots, t_n)\right)$

Semantics of formulae

Semantics of FOPL

Let $\mathcal{I}$ be an interpretation over the universe, $\mathcal{U}$.
Let $\phi, \psi \in$ WFF(FOL). Then we have:

- $\mathcal{I}_\phi(\bot) = F$ and $\mathcal{I}_\phi(\top) = T \quad \forall \phi$

- Formulae $\neg\phi, \phi \wedge \psi, \phi \vee \psi, \phi \rightarrow \psi, \phi \longleftrightarrow \psi$ are evaluated with respect to $\mathcal{I}_\phi$

- $\tilde{\mathcal{I}}_\phi \models \exists x.\phi$ if $\mathcal{I}_\phi \models \phi$ for some ϕ that maps x to some value in $\mathcal{U}$

- $\tilde{\mathcal{I}}_\phi \models \forall x.\phi$ if $\mathcal{I}_\phi \models \phi$ for assignments ϕ that map x to all possible values in $\mathcal{U}$

Normal Forms in FOPL

- We still want to end up in Clausal Normal Form but we have to convert to intermediary forms to reach it.

Prenex Normal Form

- Move all quantifiers to front of the formula

- Be careful of the scope of quantifiers when moving them

- Renaming of bound variables may be necessary

$\neg\forall x.\phi \cong \exists x.\neg\phi$ (All-Neg)	$\neg\exists x.\phi \cong \forall x.\neg\phi$ (Ex-Neg)
$(\forall x.\varphi) \vee \psi \cong \forall x.\varphi \vee \psi$ (All-Or-1)	$(\exists x.\varphi) \vee \psi \cong \exists x.\varphi \vee \psi$ (Ex-Or-1)
$(\forall x.\varphi) \wedge \psi \cong \forall x.\varphi \wedge \psi$ (All-And-1)	$(\exists x.\varphi) \wedge \psi \cong \exists x.\varphi \wedge \psi$ (Ex-And-1)
$(\forall x.\varphi) \rightarrow \psi \cong \exists x.\varphi \rightarrow \psi$ (All-Imp-Left)	$(\exists x.\varphi) \rightarrow \psi \cong \forall x.\varphi \rightarrow \psi$ (Ex-Imp-Left)
$\varphi \rightarrow (\forall x.\psi) \cong \forall x.\varphi \rightarrow \psi$ (All-Imp-Right)	$\psi \rightarrow (\exists x.\psi) \cong \exists x.\varphi \rightarrow \psi$ (Ex-Imp-Right)
$(\forall x.\varphi) \wedge (\forall x.\psi) \cong \forall x.\varphi \wedge \psi$ (All-And-2)	$(\exists x.\varphi) \vee (\exists x.\psi) \cong \exists x.\varphi \vee \psi$ (Ex-Or-2)
$(\forall x.\varphi) \vee (\forall x.\psi) \cong \forall x.\forall z.\varphi \vee \psi$ (All-Or-2)	$(\exists x.\varphi) \wedge (\exists x.\psi) \cong \exists x.\exists z.\varphi \wedge \psi$ (Ex-And-2)
$(\forall x.\varphi) \wedge (\exists x.\psi) \cong \forall x.\exists z.\varphi \wedge \psi$ (All-Ex-And)	$(\exists x.\varphi) \wedge (\forall x.\psi) \cong \exists x.\forall z.\varphi \wedge \psi$ (Ex-All-And)
$(\forall x.\varphi) \vee (\exists x.\psi) \cong \forall x.\exists z.\varphi \vee \psi$ (All-Ex-Or)	$(\exists x.\varphi) \vee (\forall x.\psi) \cong \exists x.\forall z.\varphi \vee \psi$ (Ex-All-Or)

Renaming

Renaming of variables is important in the last 6 rules above.

In order not to lose soundness when dealing with mixed quantifications

Skolem Normal Form

- Remove all quantifiers

- Done with a procedure called Skolemisation

- Replaced quantified variables with free variables

- Our Skolemised formula is not necessarily equivalent to our

- Our Skolemised formula is not necessarily equivalent to our original
  - However, Satisfiable iff the original formula was

Substitution

- Allows replacing of Variables with terms
- A Substitution is a mapping of Variables in V to terms over V, C, F
  We usually denote substitutions with small greek letters (σ, τ, μ)

  A Substitution is denoted as $[x \mapsto t]$

  A Substitution is ground if it maps Variables to ground terms

Variable Capturing ← Avoid this

- Results from careless substitution
- Make some names don't overlap / interfere
- Avoided by using renaming

Skolemisation

- Removing quantifiers from a formula in Prenex normal form
- Let ϕ be a formula in prenex form over V, F, C, P. Let $S = \{\}$

1. if ϕ has no more quantifiers, return ϕ
2. if $\phi = \forall x.\phi'$
   - 2.1. $S = S \cup \{x\}$
   - 2.2. $\phi = \phi'$
   - 2.3
3. if $\phi = \exists x.\phi'$
   - 3.1 if $S = \{\}$ then $\phi = \phi'[x \mapsto c]$ with new c in C
   - 3.2 else $\phi = \phi'[x \mapsto f(x_1, \dots, x_n)]$ with new f in F and $S = \{x_1, \dots, x_n\}$
   - 3.3.

CNF

From Skolem Normal form obtaining the CNF is trivial if the same procedure as for PropLogic.

1> proven by the same procedure as for prop logic.

First Order Resolution

- Now we can obtain CNF for 1st order formulae we need a version of Resolution for 1st order predicate calculus.

Socrates is a man, all men are mortal, therefore, Socrates is mortal

Ideally solving this with Resolution would look like:

$$\frac{Human(socrates) \qquad \neg Human(x) \vee Mortal(x)}{Mortal(socrates)} \text{ Res}$$

However, for this to work we need a means of substituting our variable x with our constant socrates during the resolution rule.

This is done using unification.

Unification

- Makes literals equal

- Given two literals, l, l' we want to compute a substitution, σ, for variables occurring in both literals that, when applied, makes them equal.

Unifiers

Any type of substitution that makes two literals equal

Eg: for our Socrates examples: $\sigma = \{[x \longmapsto socrates]\}$

- There is often more than one possible unifier to choose from for a pair of literals

- To avoid ambiguity, we define the concept of a Most general unifier or Mgu

↑ Unifier that makes least commitments on the substitution of variables (smallest set of substitutions)

Occurs Check

- Check for the occurrence of a variable in the term it is to be substituted with.

- We define the set of all variables in a term, t, to be $Var(t)$

- The occurs check is then denoted by $[x \longmapsto t]$ if $x \notin Var(t)$

Unification Algorithm

Unification Algorithm

Let σ be an substitution that is initially empty.
Consider terms s, t. We define the unification procedure
Unify(s, t, σ) recursively:

1) if $s = t$, return σ

2) if $s = f(s_1, \dots, s_n)$ and $t = f(t_1, \dots, t_n)$ then

$$\sigma_1 = \text{Unify}(s_1, t_1, \sigma),$$

$$\sigma_2 = \text{Unify}(s_2\sigma_1, t_2\sigma_1, \sigma_1)$$

$$\vdots$$

$$\sigma_n = \text{Unify}(s_n\sigma_{n-1}, t_n\sigma_{n-1}, \sigma_{n-1})$$

return σ_n

3) if $s = f(s_1, \dots, s_n)$ and $t = g(t_1, \dots, t_m)$ with $f \neq g$ or $n \neq m$ then FAIL

4) if $s = x$, and
   4.1) $x \in \text{Var}(t)$ then return $\sigma \cup \{[x \mapsto t]\}$
   4.2) else FAIL

5) if $t = x$ then Unify(x, t, σ)

The Resolution Rule

The general form of the resolution rule is therefore,

$$\frac{\{l\} \cup C_1 \quad \{\neg l'\} \cup C_2}{C_1\sigma \cup C_2\sigma} \text{Res}(l\sigma = l'\sigma)$$

↑ unify

Factoring

- Allows us to unify 2 literals of the same clause

$$\frac{\{l, l'\} \cup C}{\{l\sigma\} \cup C} \text{Factor}(l\sigma = l'\sigma)$$

Subsumption

- Used to make the resolution procedure more efficient.

- Used to remove clauses that are logically subsumed by another.

$$\frac{C \quad D}{} \text{ subsumes } (C\sigma \subseteq D)$$

$$\frac{C \quad D}{C}\ \text{subsume}\ (C\sigma \subseteq D)$$

Example

$C = \{P(x), Q(y)\}$ $D = \{R(x), P(a), Q(b)\}$

C subsumes D with substitution $\sigma = \{[x \mapsto a], [y \mapsto b]\}$

However,

$C = \{P(a), Q(y)\}$ does not subsume $D - \{R(x), P(x), Q(b)\}$
due to the constraint $\frac{a}{x}$

Tautologies

- Clauses that contain a literal & its negation,

- Since these clauses are trivially true, we can remove them from the clause set.

Note:
$\{P(x), \neg P(c)\}$ is not a tautology

Theoretical Properties

Completeness - let Φ be a set of 1st order clauses.

- If Φ is Unsat., then $\square$ can be derived by the resolution procedure

Soundness - let Φ be a set of 1st order clauses.

- if the resolution procedure derives $\square$ from Φ then Φ is Unsat.

Complexity - 1st order logic is Semi-decidable

- If we view a Problem as a decision of whether or not a particular word is part of our language, L; then we can give an effective algorithm to decide if a word is an element of L, however, our algorithm may or may not give a negative answer if a given word is not in L.

Reducing 1st order logic to this:

- we can have calculi that decide if a 1st order sentence, ϕ, follows from a set of sentences, Δ. ($\Delta \vdash \phi$) but we have no definite way of deciding if $\Delta \nvdash \phi$

$(\Delta \vdash \phi)$ but we have no definite way of deciding if $\Delta \nvdash \phi$