Unit-I Proposional Logic & First Order Logic

October 25, 2015

Satisfiability, Validity and Consequence

Definition

Let $A \in \mathfrak{F}$.

- A is satisfiable iff $v_{\mathfrak{I}}(A) = T$ for some interpretation $\mathfrak{I}$. A satisfying interpretation is a model for A.
- A is valid, denoted $\models A$, iff $v_{\mathfrak{I}} = T$ for all interpretations $\mathfrak{I}$. A valid propositional formula is also called a tautology.
- A is unsatisfiable iff it is not satisfiable, that is, if $v_{\mathfrak{I}}(A) = T$ for all interpretations $\mathfrak{I}$.
- A is falsifiable, denoted $\not\models A$, iff it is not valid, that is, if $v_{\mathfrak{I}}(A) = F$ for some interpretation v.

Satisfiability and Validity of Formulas

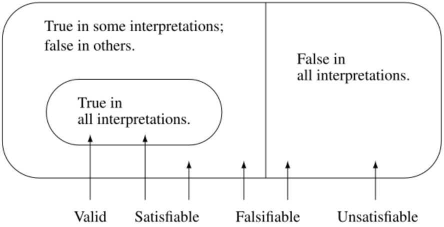


Figure: Satisfiability and validity of formulas

Satisfiability and Validity of Formulas

Theorem

Let $A \in \mathfrak{F}$. A is valid if and only if $\urcorner$ A is unsatisfiable. A is satisfiable if and only if $\urcorner$ A is falsifiable

Proof.

Let $\mathfrak{I}$ be an arbitrary interpretation. $v_{\mathfrak{I}}$ (A) = T if and only if $v_{\mathfrak{I}}$ ($\neg$A) = F by the definition of the truth value of a negation.
Since $\mathfrak{I}$ was arbitrary, A is true in all interpretations if and only if $\neg$ A is false in all interpretations, that is, iff $\neg$A is unsatisfiable.
If A is satisfiable then for some interpretation $\mathfrak{I}$, $v_{\mathfrak{I}}$ (A) = T.
By definition of the truth value of a negation, $v_{\mathfrak{I}}$ ($\neg$A) = F so that $\neg$ A is falsifiable.
Conversely, if $v_{\mathfrak{I}}$ ($\neg$A) = F then $v_{\mathfrak{I}}$ (A) = T. $\square$

Decision Procedures in Propositional Logic

Definition (Decision Procedure)

Let $\mathfrak{U} \subseteq \mathfrak{F}$ be a set of formulas. An algorithm is a decision procedure for $\mathfrak{U}$ if given an arbitrary formula A$\in \mathfrak{F}$, it terminates and returns the answer yes if A $\in \mathfrak{U}$ and the answer no if A $\notin \mathfrak{U}$. If $\mathfrak{U}$ is the set of satisfiable formulas, a decision procedure for $\mathfrak{U}$ is called a decision procedure for satisfiability, and similarly for validity.

Note

1. To decide if A is valid, apply the decision procedure for satisfiability to $\neg$ A
2. If it reports that $\neg$ A is satisfiable, then A is not valid; if it reports that $\neg$ A is not satisfiable, then A is valid. Such an decision procedure is called a refutation procedure, because we prove the validity of a formula by refuting its negation
3. Refutation procedures can be efficient algorithms for deciding validity, because instead of checking that the formula is always true, we need only search for a falsifying counterexample
4. The existence of a decision procedure for satisfiability in propositional logic is trivial, because we can build a truth table for any formula

Example for Satisfiability

- Truth table for the formula $p \rightarrow q$

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Example for Validity and Unsatisfiable

- Validity of $(p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p)$
- Proof Refer BB
- Prove that $(p \vee q) \wedge \neg p \wedge \neg q$ is unsatisfiable.
- Proof Refer BB

Satisfiability of Set of Formulas

Definition

A set of formulas U=$\{A_1,...\}$s (simultaneously) satisfiable iff there exists an interpretation $\mathfrak{I}$ such that $\mathfrak{v}_{\mathfrak{I}}(A_i) = T$ for all i. The satisfying interpretation is a model of U . U is unsatisfiable iff for every interpretation $\mathfrak{I}$, there exists an i such that $\mathfrak{v}_{\mathfrak{I}}(A_i) = F$

- Example:
  - The set $U_1 = \{p, \neg p \vee q, q \wedge r\}$ is simultaneously satisfiable by the interpretation which assigns T to each atom, while the set $U_2 = \{p, \neg p \vee q, \neg p\}$ is unsatisfiable. Each formula in U_2 is satisfiable by itself, but the set is not simultaneously satisfiable

Logical Consequence

Definition (Logical Consequence)

Let U be a set of formulas and A a formula. A is a logical consequence of U , denoted $U \models A$, iff every model of U is a model of A

- Example:
  - Let $A = (p \vee r) \wedge (\neg q \vee \neg r)$. Then A is a logical consequence of $\{p, \neg q\}$, denoted $\{p, \neg q\} \models A$, since A is true in all interpretations $\mathfrak{I}$ such that $\mathfrak{I}(p) = T$ and $\mathfrak{I}(q) = F$. However, A is not valid, since it is not true in the interpretation $\mathfrak{I}'$ where $\mathfrak{I}'(p) = F$, $\mathfrak{I}'(q)=T$, $\mathfrak{I}'(r) = T$

Theorem

Theorem

$U \vDash A$ *if and only if* $\vDash \bigwedge_i A_i \to A$.

- Example:
  - $\{p, \neg q\} \vDash (p \vee r) \wedge (\neg q \vee \neg r)$, so by Theorem, $\vDash (p \wedge \neg q) \to (p \vee r) \wedge (\neg q \vee \neg r)$

Semantic Tableaux

- The method of semantic tableaux is an efficient decision procedure for satisfiability (and by duality validity) in propositional logic
- The principle behind semantic tableaux is very simple: search for a model (satisfying interpretation) by decomposing the formula into sets of atoms and negations of atoms
- It is easy to check if there is an interpretation for each set: a set of atoms and negations of atoms is satisfiable iff the set does not contain an atom p and its negation ¬p
- The formula is satisfiable iff one of these sets is satisfiable

Decomposing Formulas into Sets of Literals

Definition (Literals)

A literal is an atom or the negation of an atom. An atom is a positive literal and the negation of an atom is a negative literal. For any atom p, $\{p, \neg p\}$ is a complementary pair of literals

- For any formula A, $\{A, \neg A\}$ is a complementary pair of formulas. A is the complement of $\neg A$ and $\neg A$ is the complement of A

Problems

1. Analyze satisfiability of the formula: $A = p \wedge (\neg q \vee \neg p)$.
   Solution Refer BB
2. Prove that $B = (p \vee q) \wedge (\neg p \wedge \neg q)$ is unsatisfiable

Theorem

A set of literals is satisfiable if and only if it does not contain a complementary pair of literals - Proof: Refer BB

SAT Solvers

Definition (SAT Solver)

A computer program that searches for a model for a propositional formula is called a SAT Solver

- Properties of Clausal Form

Definition

Let S, S' be sets of clauses. S $\approx$ S' denotes that S is satisfiable if and only if S' is satisfiable

- It is important to understand that S $\approx$ S' (S is satisfiable if and only if S' is satisfiable) does not imply that S $\equiv$ S' (S is logically equivalent to S')

Pure Literals and Renaming

Definition (Pure Literals)

Let S be a set of clauses. A Pure literal in S is a literal l that appears in atleast one clause of S, but its complement l^c does not appear in any clause of S

Definition (Renaming)

Let S be a set of clauses and U a set of atomic propositions $R_U(S)$ the renaming of S by U is obtained from S by replacing each literal l on an atomic proposition in U by l^c

Example

- If S={pqr, $\bar{p}$q, $\bar{q}\bar{r}$,r}, find R$_{p,q}$(S)

Example

- If S={pqr, $\bar{p}$q, $\bar{q}\bar{r}$,r}, find $R_{p,q}$(S)
- Solution: $R_{p,q}$(S)={$\bar{p}\bar{q}$,p$\bar{q}$,q$\bar{r}$,r},

Davis Putnam Algorithm

Input : A formula A in Clausal Form
Output: Report A is Satisfiable or Unsatisfiable
Perform the following rules repeatedly but the third rule is used only if the first two do not apply.

1. Unit Literal rule: If there is a unit clause (l), delete all clauses containing l and delete all occurrences of l^c from other clauses
2. Pure Literal rule:If there is a pure literal l, delete all clauses containg l
3. Eliminate a variable by resolution:Choose an atom p and perform all possible resolutions on clauses that clash on p and $\bar{p}$. Add these resolvents to the set of clauses and then delete all clauses containing p or $\bar{p}$

Algorithm 1: Davis Putnam Algorithm

Termination condition for the Algorithm

1. If empty clause $\square$ is produced, report the formula is unsatisfiable
2. If no more rules are applicable, report that formula is satisfiable

Example

- Consider the set of clauses $\{$p, $\bar{p}$q,$\bar{q}$r,$\bar{r}$st$\}$. Apply DP algorithm.

Example

- Consider the set of clauses {p, $\bar{p}$q,$\bar{q}$r,$\bar{r}$st}. Apply DP algorithm.
- Step(i) According to step(i) of D.P algorithm, p is a unit literal. Hence it must be deleted in all clauses containing p and delete all occurrences of p^c from other clauses. $\therefore$ the given set becomes {q,$\bar{q}$r,$\bar{r}$st}

Example

- Consider the set of clauses $\{p, \bar{p}q, \bar{q}r, \bar{r}st\}$. Apply DP algorithm.
- Step(i) According to step(i) of D.P algorithm, p is a unit literal. Hence it must be deleted in all clauses containing p and delete all occurrences of p^c from other clauses. $\therefore$ the given set becomes $\{q, \bar{q}r, \bar{r}st\}$
- Step(ii) Like the previous step, remove q and q^c. $\therefore$ the set becomes $\{r, \bar{r}st\}$

Example

- Consider the set of clauses $\{p, \bar{p}q, \bar{q}r, \bar{r}st\}$. Apply DP algorithm.
- Step(i) According to step(i) of D.P algorithm, p is a unit literal. Hence it must be deleted in all clauses containing p and delete all occurrences of p^c from other clauses. $\therefore$ the given set becomes $\{q, \bar{q}r, \bar{r}st\}$
- Step(ii) Like the previous step, remove q and q^c. $\therefore$ the set becomes $\{r, \bar{r}st\}$
- Step(iii) Like the previous step, remove r and its complement. $\therefore$ the set becomes $\{st\}$

Example

- Consider the set of clauses $\{p, \bar{p}q, \bar{q}r, \bar{r}st\}$. Apply DP algorithm.
- Step(i) According to step(i) of D.P algorithm, p is a unit literal. Hence it must be deleted in all clauses containing p and delete all occurrences of p^c from other clauses. $\therefore$ the given set becomes $\{q, \bar{q}r, \bar{r}st\}$
- Step(ii) Like the previous step, remove q and q^c. $\therefore$ the set becomes $\{r, \bar{r}st\}$
- Step(iii) Like the previous step, remove r and its complement. $\therefore$ the set becomes $\{st\}$
- No more rules are applicable for the set $\{st\}$, the set of clauses is satisfiable

Hilbert System (H)

- Gentzen System vs Hilbert System

Gentzen	Hilbert
One Axiom	Several Axioms
Many Inference Rules	Only one rule of inference

- Notations used
  - A,B,C denote arbitrary formulas in propositional logic
  - $\vdash A \rightarrow A$ means $A \rightarrow A$ can be proved

Hilbert System(H)

Definition (Axioms of H & Rules of Inference)

1. $\vdash(A\rightarrow(B\rightarrow A))$
2. $\vdash(A\rightarrow(B\rightarrow C))\rightarrow((A\rightarrow B)\rightarrow(A\rightarrow C))$
3. $\vdash(\neg B\rightarrow \neg A)\rightarrow(A\rightarrow B)$
4. Rule of Inference: (Modus Ponens)(MP) $\frac{\vdash A, \vdash A\rightarrow B}{\vdash B}$

Theorem

$\vdash A \rightarrow A$

Proof.

Refer BB $\square$

Assignment Due Date:27.10.15

1. Prove $\vdash (A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$
2. Prove $\vdash (\neg A \rightarrow A) \rightarrow A$ in H

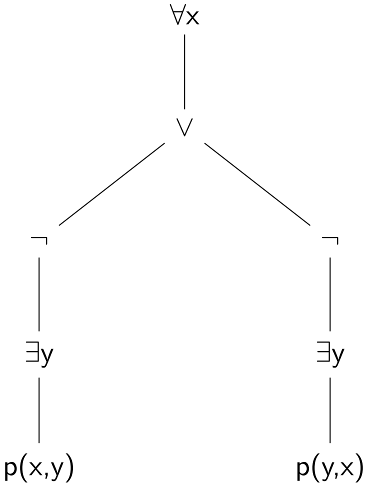
First Order Logic: Deductive Systems

Definition (First Order Logic)

An extension of propositional logic that includes predicates interpreted as relations on domains

- Notations used
  - P,A,V denotes the countable sets of predicate symbols, constant symbols and variables
  - $p^n \in P \rightarrow$ n-ary predicate
  - $n=1$ or $2 \rightarrow$ unary or binary
  - $\forall \rightarrow$ Universal Quantifier
  - $\exists \rightarrow$ Existential Quantifier

Example



First Order Logic:Deductive System

- Gentzen System:
  - Deductive System
  - One Axiom
  - 4 rules of Inference
- Two Special rules: γ and δ

γ	γ(a)
$\exists$xA(x)	A(a)
$\neg\ \forall$xA(x)	$\neg$A(a)

δ	δ(a)
$\forall$xA(x)	A(a)
$\neg\ \exists$xA(x)	$\neg$A(a)

- $\frac{U \cup \{\gamma, \gamma(a)\}}{U \cup \{\gamma\}}$
- $\frac{U \cup \{\delta(a)\}}{U \cup \{\delta\}}$

Hilbert System

Definition (Axioms of H for the First Order Logic)

1. Apart from the three axioms of H for Propositional Logic we have:
   1. $\vdash \forall x A(x) \rightarrow A(a)$
   2. $\vdash \forall x (A \rightarrow B(x)) \rightarrow (A \rightarrow \forall x B(x))$

- Rules of Inference(Modus Ponens and Generalization)
  - $\frac{\vdash A \rightarrow B, \vdash A}{\vdash B}$
  - $\frac{\vdash A(a)}{\vdash \forall x A(x)}$
- C-Rule:
  - (i) U $\vdash \exists x A(x)$ (Existential Quantifier)
  - (i+1) U $\vdash$ A(a) (C rule)
- Deduction Rule:
  - $\frac{U \cup \{A\} \vdash B}{U \vdash A \rightarrow B}$

Theorems

Theorem

$\vdash A(a) \rightarrow \exists x A(x)$

Proof.

Refer BB $\square$

Theorem

$\vdash \forall x\ A(x) \rightarrow \exists x\ A(x)$

Proof.

Refer BB $\square$

Skolem's Algorithm

- An algorithm to transform a formula A into a formula A'in clausal form $\forall x(p(x) \rightarrow q(x)) \rightarrow (\forall x p(x) \rightarrow \forall x q(x))$

Skolems Algorithm I

Input : A closed formula A of first-order logic

Output: A formula A'in clausal form such that A≈A'

1. Rename bound variables so that no variable appears in two quantifiers
   $\forall x(p(x) \rightarrow q(x)) \rightarrow (\forall y p(y) \rightarrow \forall z q(z))$
2. Eliminate all binary Boolean operators other than $\vee$ and $\wedge$
   $\neg \forall x(\neg p(x) \vee q(x)) \vee \neg \forall y p(y) \vee \forall z q(z)$
3. Push negation operators inward, collapsing double negation, until they apply to atomic formulas only. Use the equivalences:
   $\neg \forall x A(x) \equiv \exists x \neg A(x)$, $\neg \exists x A(x) \equiv \forall x \neg A(x)$

Skolems Algorithm Contd..

4. The given formula is transformed to:
   $\exists x(p(x) \wedge \neg\, q(x)) \vee \exists y \neg\, p(y) \vee \forall z q(z)$
5. Extract quantifiers from the matrix. Choose an outermost quantifier, that is, a quantifier in the matrix that is not within the scope of another quantifier still in the matrix. Extract the quantifier using the following equivalences, where Q is a quantifier and op is either $\vee$ or$\wedge$:
   $A \text{ op } QxB(x) \equiv Qx(A \text{ op } B(x))$, $QxA(x) \text{ op } B \equiv Qx(A(x) \text{ op } B)$
   Repeat until all quantifiers appear in the prefix and the matrix is quantifier-free. The equivalences are applicable because since no variable appears in two quantifiers.

Skolems Algorithm Contd..

6. In the example, no quantifier appears within the scope of another, so we can extract them in any order, for example, x, y, z: $\exists x \exists y \forall z((p(x) \wedge \neg q(x)) \vee \neg p(y) \vee q(z))$
7. Use the distributive laws to transform the matrix into CNF. The formula is now in PCNF
$\exists x \exists y \forall z((p(x) \vee \neg p(y) \vee q(z)) \wedge (\neg q(x) \vee \neg p(y) \vee q(z)))$
8. For every existential quantifier $\exists x$ in A, let $y_1, \ldots, y_n$ be the universally quantified variables preceding $\exists x$ and let f be a new n-ary function symbol. Delete $\exists x$ and replace every occurrence of x by $f(y_1, \ldots, y_n)$. If there are no universal quantifiers preceding $\exists x$, replace x by a new constant (0-ary function). These new function symbols are Skolem functions and the process of replacing existential quantifiers by functions is Skolemization

Skolem Algorithm Contd..

9. For Example, $\forall z((p(a) \vee \neg\, p(b) \vee q(z)) \wedge (\neg\, q(a) \vee \neg\, p(b) \vee q(z)))$, where a and b are the Skolem functions (constants) corresponding to the existentially quantified variables x and y, respectively
10. The formula can be written in clausal form by dropping the (universal) quantifiers and writing the matrix as sets of clauses:
$\{\{p(a), \neg\, p(b), q(z)\}, \{\neg\, q(a), \neg\, p(b), q(z)\}\}$

Algorithm 2: Skolems Algorithm

Skolem's Algorithm Example

Step	Transformation
Original formula	$\exists x \forall y p(x,y) \rightarrow \forall y \exists x p(x,y)$
Rename bound variables	$\exists x \forall y p(x,y) \rightarrow \forall w \exists z p(z,w)$
Eliminate Boolean operators	$\neg \exists x \forall y p(x,y) \vee \forall w \exists z p(z,w)$
Push negation inwards	$\forall x \exists y \neg p(x,y) \vee \forall w \exists z p(z, w)$
Extract quantifiers	$\forall x \exists y \forall w \exists z (\neg p(x,y) \vee p(z,w))$
Distribute matrix	(no change)
Replace existential quantifiers	$\forall x \forall w (\neg p(x,f(x)) \vee p(g(x,w), w))$
Write in clausal form	$\{\{\neg p(x,f(x)), p(g(x,w), w)\}\}$

Skolem's Theorem

Theorem

Let A be a closed formula. Then there exists a formula A' in clausal form such that $A \approx A'$