Logic and theory of computation

2nd lecture

24-09-2018

Laws of propositional logic

Let us extend the syntax of Boolean formulas to include the two constant atomic propositions $\top$ (true) and $\bot$ (false).Their semantics are defined by $I(\top) = T$ and $I(\bot) = F$ for any interpretation I.

- $A \vee \top \equiv \top$ and $A \wedge \bot \equiv \bot$,
- $A \vee \bot \equiv A$ and $A \wedge \top \equiv A$.
- $A \vee \neg A \equiv \top$ and $A \wedge \neg A \equiv \bot$,
- $\neg\neg A \equiv A$,
- $A \vee A \equiv A$ and $A \wedge A \equiv A$ (idempotence),
- $A \to B \equiv \neg A \vee B$,
- $A \to B \equiv \neg B \to \neg A$ (rhs: contrapositive of the lhs).

Laws of propositional logic
(cont'd)

- $A \vee B \equiv B \vee A$ and $A \wedge B \equiv B \wedge A$ (commutative laws),
- $(A \vee B) \vee C \equiv A \vee (B \vee C)$ and
  $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (associative laws),
- $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$ and
  $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$ (distributive laws),
- $\neg(A \wedge B) \equiv \neg A \vee \neg B$ and $\neg(A \vee B) \equiv \neg A \wedge \neg B$
  (De Morgan laws),
- $(A \vee B) \wedge B \equiv B$ and $(A \wedge B) \vee B \equiv B$.

Definitions of other Boolean ops:

- $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$,
- $A \oplus B \equiv (A \vee B) \wedge \neg(A \wedge B)$,
- $A \uparrow B \equiv \neg(A \wedge B)$,
- $A \downarrow B \equiv \neg(A \vee B)$.

Substitution

Substitution

Let A be a subformula of B and let A' be any formula.
$B\{A \leftarrow A'\}$, the substitution of A for A' in B, is the formula obtained by replacing all occurrences of the subtree for A in B by A'.

Example:
$B = (p \to q) \vee \neg(p \to q) \vee (\neg q \to \neg p)$.
$A = p \to q,\ A' = \neg p \vee q$
$B\{A \leftarrow A'\} = (\neg p \vee q) \vee \neg(\neg p \vee q) \vee (\neg q \to \neg p)$.
$B\{p \leftarrow \neg r\} = (\neg r \to q) \vee \neg(\neg r \to q) \vee (\neg q \to \neg\neg r)$,

Theorem

Let A be a subformula of B and let A' be a formula such that $A \equiv A'$. Then $B \equiv B\{A \leftarrow A'\}$.

Substitution

Proof

Let $B' = B\{A \leftarrow A'\}$ and I be an arbitrary interpretation. Then $v(A) = v(A')$ and we must show that $v(B) = v(B')$.

The proof is by induction on the depth d of the highest occurrence of the subtree A in B.

If $d = 0$, there is only one occurrence of A, namely B itself. Obviously, $v(B) = v(A) = v(A') = v(B')$.

If $d \neq 0$, then B is either $\neg B_1$ or $B_1 \, op \, B_2$ for some formulas B_1, B_2 and binary operator op.

In B_1 , the depth of A is less than d. By the inductive hypothesis, $v(B_1) = v(B_1') = v(B_1\{A \leftarrow A'\})$, and similarly $v(B_2) = v(B_2') = v(B_2\{A \leftarrow A'\})$.

By the definition of v, $v(B)$ depends only on $v(B_1)$ and $v(B_2)$, so $v(B) = v(B')$.

Semantic properties of a set of formulas

Satisfiability

Satisfiability of a set of formulas

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

Example:

$U_1 = \{p, \neg p \vee q, q \wedge r\}$,
$U_2 = \{p, \neg p \vee q, \neg p\}$.
$U_3 = \{p, \neg p \vee q, \neg q\}$.
Which of the three is satisfiable?
$I \models U_1$, where $I(p) = I(q) = I(r) = T$.
p and $\neg p$ can not be T simultaneously, so for all interpretations I: $I \not\models U_2$.
One can check that for all interpretations I: $I \not\models U_3$.

Semantic properties of a set of formulas

Logical consequence

Observation

If U is satisfiable, then U' is satisfiable, too, for all $U' \subseteq U$.
If U is unsatisfiable, then U'' is unsatisfiable for all $U \subseteq U''$.

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 for 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 I such that $I(p) = T$ and $I(q) = F$.
Note, that A is not valid, since it is not true in the interpretation I where $I(p) = F, I(q) = T, I(r) = T$.

Role of logical consequence in the foundation of mathematics

Theories, axiomatizability

Theory

A set of formulas T is *closed under logical consequence* iff for all formulas A, if $T \models A$ then $A \in T$. A set of formulas that is closed under logical consequence is a *theory*, the elements of T are *theorems*.

Theories are constructed by selecting a set of formulas called axioms and deducing their logical consequences.

Axiomatizable theory

Let T be a theory. T is said to be *axiomatizable* iff there exists a set of formulas U such that $T = \{A \mid U \models A\}$. The set of formulas U are the axioms of T. If U is finite, T is said to be *finitely axiomatizable*.

Examples: arithmetic (Peano), geometry (Euclid).

Decision precedure

Decision precedure

Let $\mathcal{U}$ be a set of formulas. An algorithm is a *decision procedure* for $\mathcal{U}$ if given an arbitrary formula A, it terminates and returns the answer *yes* if $A \in \mathcal{U}$ and the answer *no* if $A \notin \mathcal{U}$.

Example:
If $\mathcal{U}$ is the set of satisfiable formulas, a decision procedure for $\mathcal{U}$ is called a decision procedure for satisfiability.

decision procedure for satisfiability $\Rightarrow$ decision procedure for validity

$\neg A$ satisfiable? *yes/no* $\Rightarrow$ A valid *no/yes*

Refutation procedure: proving the validity of a formula by refuting its negation.

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.

Method of semantic tableaux

Is there a decision procedure in propositional logic for satisfiability?
YES (truth table)

Is the truth table method efficient?
NO (2^n rows)

We need more efficient decision procedure(s) for satisfiability. (And by duality for validity as well.) Efficient: in average, not in worst case.

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 and its negation simultaneously.

The formula is satisfiable iff one of these sets is satisfiable.

Method of semantic tableaux

Decomposing formulas into sets of literals

Literals, complementary pair of literals and formulas

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.

Example:

Consider the set of literals $\{\neg p, q, r, \neg r\}$.

q and r are positive literals, while $\neg p$ and $\neg r$ are negative literals.

The set contains one complementary pair of literals: $r, \neg r$.

Method of semantic tableaux

The principle of semantic tableaux by example

Example:
$A = p \wedge (\neg q \vee \neg p)$. Is A satisfiable?

- $v(A) = T \Leftrightarrow v(p) = T$ and $v(\neg q \vee \neg p) = T$
- $v(\neg q \vee \neg p) = T \Leftrightarrow v(\neg q) = T$ or $v(\neg p) = T$
- So, for an interpretation I $v_I(A) = T$ holds iff
  - either $v_I(p) = T$ and $v_I(\neg q) = T$
  - or $v_I(p) = T$ and $v_I(\neg p) = T$
- A is satisfiable iff
  - either $\{p, \neg q\}$
  - or $\{p, \neg p\}$ is satisfiable.

Theorem

A set of literals is satisfiable if and only if it does not contain a complementary pair of literals.

So A is satisfiable (and $I(p) = T$, $I(q) = F$ is the only model for A).

Method of semantic tableaux

α and β formulas

α	α_1	α_2
$\neg\neg A_1$	A_1	
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \to A_2)$	A_1	$\neg A_2$

β	β_1	β_2
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \vee B_2$	B_1	B_2
$B_1 \to B_2$	$\neg B_1$	B_2

Formulas are classified according to their principal operator. If it is a negation, the classification takes into account both the negation and the principal operator of the unnegated formula.
α-formulas are conjunctive and are satisfiable only if both subformulas α_1 and α_2 are satisfied.
β-formulas are disjunctive and are satisfied even if only one of the subformulas β_1 or β_2 is satisfiable.

Method of semantic tableaux

Algorithm for constructing a semantic tableaux

Input: A formula F of propositional logic.
Output: A semantic tableau $\mathcal{T}$ for F all of whose leaves are marked.

Algorithm ConstructionSemanticTableaux(F; $\mathcal{T}$)

---

let $\mathcal{T}$ be a tree of a single node labelled by $\{F\}$, and let the node be unmarked
while there is an unmarked leaf **do**
    choose an unmarked leaf ℓ of label $U(\ell)$
    if $U(\ell)$ is a set of literals **then**
        if $U(\ell)$ contains a complementary pair of literals **then**
            mark ℓ closed by $\times$
        else
            mark ℓ open by $\odot$
    else

Method of semantic tableaux

Algorithm for constructing a semantic tableaux (cont'd)

Algorithm ConstructionSemanticTableaux($F; \mathcal{T}$) – cont'd

10: Select a non-literal formula from $U(\ell)$.
11: **if** the selected formula A has α-subformulas A_1 and A_2 **then**
12: Create a new node ℓ' as a child of ℓ
13: $U(\ell') \leftarrow (U(\ell) - \{A\}) \cup \{A_1, A_2\}$.
14: (In the case that A is $\neg\neg A_1$, there is no A_2.)
15: **if** the selected formula B has β-subformulas B_1 and B_2 **then**
16: Create two new nodes ℓ' and ℓ'' as children of ℓ.
17: $U(\ell') \leftarrow (U(\ell) - \{B\}) \cup \{B_1\}$
18: $U(\ell'') \leftarrow (U(\ell) - \{B\}) \cup \{B_2\}$.

Method of semantic tableaux

Termination

Completed/open/closed tableau

Completed tableau: the construction terminated. A completed tableau is *closed* if all its leaves are marked closed. Otherwise it is *open*.

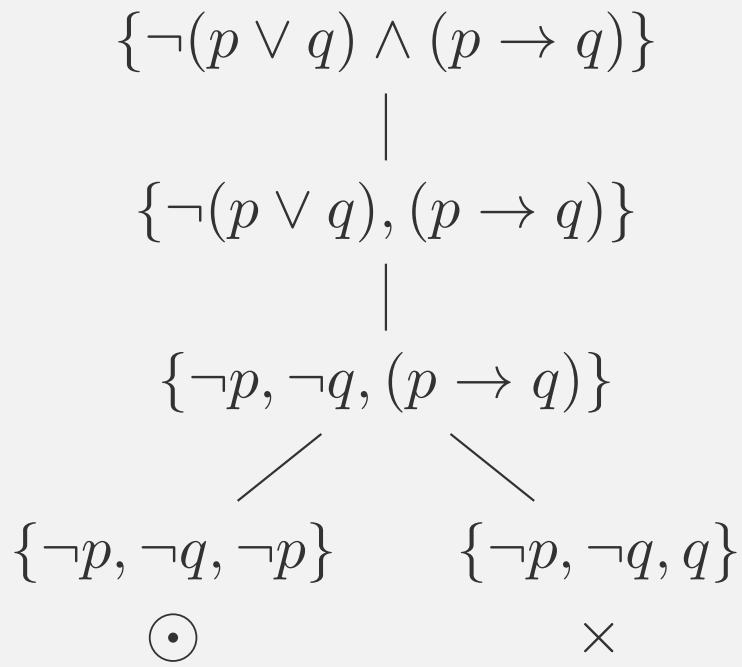
Theorem (Termination of the Construction)

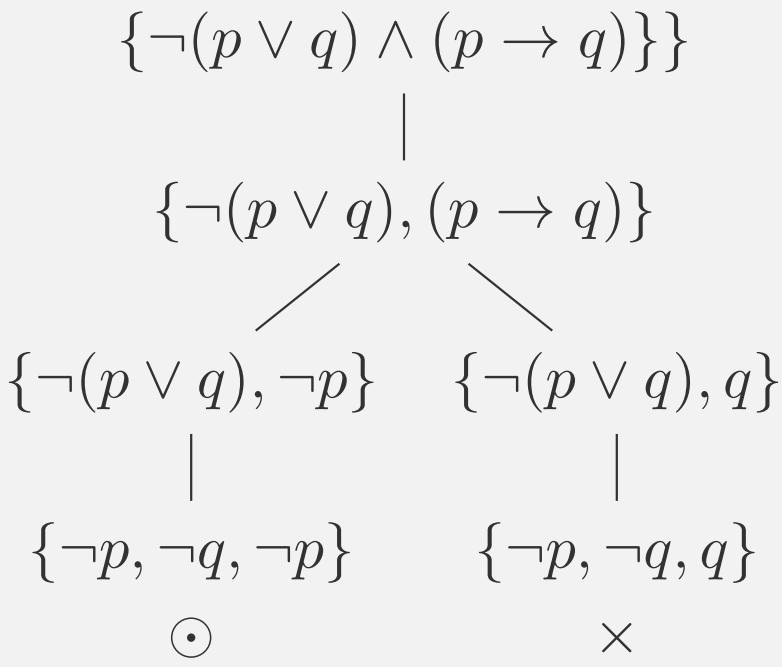
The construction of a tableau for any formula terminates. When the construction terminates, all the leaves are marked $\times$ or $\odot$.

Method of semantic tableaux

Uniqueness

Is the tableau unique? NO, example:





Method of semantic tableaux

Soundness and completeness

Theorem (Soundness and completeness)

Let $\mathcal{T}$ be a completed tableau for a formula A. A is unsatisfiable if and only if $\mathcal{T}$ is closed.

Corollary 1

A is satisfiable if and only if $\mathcal{T}$ is open.

Corollary 2

A is valid if and only if the tableau for $\neg A$ closes.

Corollary 3

The method of semantic tableaux is a decision procedure for validity in propositional logic.