Truth tables

Heinz-Dieter Ebbinghaus, Jörg Flum, Wolfgang Thomas, *Mathematical logic*, Section III.2, XI.4

Kamal Lodaya

Bharat Gyan Vigyan Samiti Karnataka, IISc

January 2025

1. Introducing propositional logic
2. Expressiveness
3. Decision problems

Introducing PL (Leibniz 17th c., 1704; Boole 1854)

Propositional logic (PL) over symbols Pr (propositional variables) has simple syntax. Take each symbol and connective to have length 1.

$$A ::= p \in Pr \mid (\neg A) \mid (A \vee B) \mid (A \wedge B) \mid (A \rightarrow B) \mid (A \leftrightarrow B)$$

Let $\mathit{false} \stackrel{\text{def}}{=} p_0 \wedge (\neg p_0)$, $\mathit{true} \stackrel{\text{def}}{=} \neg \mathit{false}$, $A \wedge B \stackrel{\text{def}}{=} \neg((\neg A) \vee (\neg B))$, $A \rightarrow B \stackrel{\text{def}}{=} (\neg A) \vee B$

A propositional assignment s is a function assigning a boolean value $p[s]$ in $\{T, F\}$ to every propositional variable p in Pr.

This is lifted to formulas: every Boolean operation has a truth table (EFT, Section III.2) giving a truth value $A[s]$ to the formula A.

Given an assignment s and formula A, the truth value $A[s]$ in $\{T, F\}$ can be given recursively, using the notation $s \models A$ (s satisfies A, or s is a model of A) for $A[s] = T$. s is a model of theory Th (set of formulas) if s satisfies every formula in Th.

$$
\begin{array}{lll}
s \models p & \text{iff} & p[s] = T \\
s \models \neg A & \text{iff} & \text{not } (s \models A) \\
s \models A \vee B & \text{iff} & s \models A \text{ or } s \models B \\
s \models A \wedge B & \text{iff} & s \models A \text{ and } s \models B \\
s \models A \rightarrow B & \text{iff} & (\text{if } s \models A \text{ then } s \models B) \\
s \models A \leftrightarrow B & \text{iff} & (s \models A \ \text{ iff } \ s \models B)
\end{array}
$$

Exercise

Evaluate $q \wedge (\neg(p \rightarrow r))$ *over* $p[s] = F, q[s] = T, r[s] = T$.

Lemma (Coincidence)

For symbols Pr, a Pr-formula A and Pr-assignment s, whether $s \models A$ depends only on propositional variables occurring in A.

Corollary

The truth table for A is finite and has 2^n rows if A has n propositional variables.

Exercise (Independence of negation)

Show that positive formulas ($\wedge, \vee$) have monotone truth tables. That is, changing an input variable from F to T cannot change formula value from T to F. How about $\wedge, \vee, \rightarrow$?

We want to add two n-bit numbers, the result may be $n+1$ bits.

Exercise (Half adder)

Given two bits x, y, give a truth table determining their sum and carry bits r, c. Which Boolean functions are these?

Exercise (Full adder)

Given two bits x, y and an incoming carry bit z, give a truth table determining their sum and outgoing carry bits r, c.

Exercise (Multiplier)

Given 4-bit numbers X, Y, show that their 8-bit product P can be determined using school arithmetic.

$$
\begin{array}{ccccccccc}
 & & & & & X0Y3 & X0Y2 & X0Y1 & X0Y0 \\
+ & & & & X1Y3 & X1Y2 & X1Y1 & X1Y0 & \\
+ & & & X2Y3 & X2Y2 & X2Y1 & X2Y0 & & \\
+ & & X3Y3 & X3Y2 & X3Y1 & X3Y0 & & & \\
= & P7 & P6 & P5 & P4 & P3 & P2 & P1 & P0
\end{array}
$$

Boolean functions are representable

Theorem (Emil Post 1921)

Given finite Pr, a function g from Pr-assignments to $\{T, F\}$, there is a Pr-formula A whose truth table is the function g.

Consider three cases to prove this theorem.

1. $g(s)$ is not T (that is, F) for every assignment s. In this case the required formula is $false$ (or $p \wedge (\neg p)$).
2. $g(s)$ is T for a unique model s_0, and F otherwise. In this case the required formula is $\overline{\mathrm{s_0}} = (\bigwedge_{p[s_0]=T} p) \wedge (\bigwedge_{p[s_0]=F} \neg p)$.

   This captures the assignment s_0.
3. $g(s)$ is T for $s_1, \ldots, s_n$ for some bound n, since the number of models over a finite symbol set is finite by the Coincidence Lemma. In this case the required formula is $\bigvee_{i=1}^{n} \overline{\mathrm{s_i}}$, where $\overline{\mathrm{s_i}}$ is defined for model s_i as above.

Definition

A formula A is valid ($\models A$) if for every assignment s, $s \models A$.

Exercise (Double negation, De Morgan, Distributivity)

Show that the following formulas are valid: $(\neg\neg A) \leftrightarrow A$;
$\neg(A \vee B) \leftrightarrow ((\neg A) \wedge (\neg B))$; $\neg(A \wedge B) \leftrightarrow ((\neg A) \vee (\neg B))$;
$(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$;
$(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$

Negation normal form formula: $\neg$ only for atomic subformulas.

Definition

Satisfiable formula ($Sat\ A$): for some assignment s, $s \models A$.

Exercise (Duality)

Show that A is valid if and only if $\neg A$ is not satisfiable, and A is satisfiable if and only if $\neg A$ is not valid.

SAT Question: How much time does it take to check if $Sat\ A$?
Question (Cook 1971, Levin 1973): Can one do better?

Definition

A literal is either a propositional symbol p or its negation $\neg p$. A formula is in disjunctive normal form (DNF) if it is a disjunction of ≥ 1 conjunctions of literals. A formula is in conjunctive normal form (CNF) if it is a conjunction of ≥ 1 disjunctions of literals.

Theorem

Every formula has a logically equivalent one which is in DNF. Every formula A has a logically equivalent one which is in CNF.

1. Follows from the fact that every formula has a truth table. By the proof of Post's theorem, truth table can be seen as a formula in DNF.
2. First find the DNF equivalent, say B, of $\neg A$. Then $\neg B \leftrightarrow \neg(\neg A) \leftrightarrow A$. Use Double Negation and De Morgan's laws to transform $\neg B$ for B in DNF to an equivalent CNF.

Checking satisfiability

A literal is always satisfiable.

A formula which is a conjunction of literals is satisfiable if it does not have contradictory literals of the form p and $\neg p$. This can be checked by going through the formula in time linear in length of the formula.

A formula in DNF is satisfiable if one of its disjuncts is satisfiable. This can be checked by going through one disjunct after another, again in linear time.

A formula in CNF is satisfiable if one disjunct is satisfied in every one of its conjuncts, such that no contrary literals of the form p and $\neg p$ are chosen in different conjuncts. This can be checked by trying all possibilities of selecting disjuncts, which can be done in time exponential in length of the formula.

Conversion to conjunctive normal form (Tseitin 1968)

Theorem (Richard Karp 1972)

There is a polynomial time algorithm reducing satisfiability of a PL formula to satisfiability of a PL formula in CNF.

Proof idea: From $A = (P_1 \wedge Q_1) \vee (P_2 \wedge Q_2) \vee \cdots \vee (P_n \wedge Q_n)$ to CNF, naively applying Distributivity of Or over And, conversion to CNF blows up formula length exponentially.

(Tseitin 1968) linear blowup, use fresh variables $r_1, \ldots, r_n$:

$$B = (r_1 \vee \cdots \vee r_n) \wedge ((r_1 \to (P_1 \wedge Q_1)) \wedge ((r_2 \to (P_2 \wedge Q_2)) \wedge \cdots \wedge (r_n \to (P_n \wedge Q_n))$$

Sat A to *Sat B*: Suppose A is sat. Assign to every r_i the truth value of $(P_i \wedge Q_i)$. One of the r_i in B is assigned T, so B is sat.

Sat B to *Sat A*: If B is sat, then one r_i in its first conjunct is assigned T. By implication, $(P_i \wedge Q_i)$ is T. Then A is sat.

Exercise

Polynomial time reduction from CNFSat to 3CNFSat?

PHP2 (Cook and Reckhow 1979; Haken 1985)

Given 3 pigeons and 2 pigeonholes.
Let p_{ij} stand for pigeon i in pigeonhole j (Fitting, Section 4.5).

Propositional pigeonhole principle:

$$((p_{11} \vee p_{12}) \wedge (p_{21} \vee p_{22}) \wedge (p_{31} \vee p_{32}))$$
$$\to ((p_{11} \wedge p_{21}) \vee (p_{21} \wedge p_{31}) \vee (p_{11} \wedge p_{31})$$
$$\vee (p_{12} \wedge p_{22}) \vee (p_{22} \wedge p_{32}) \vee (p_{12} \wedge p_{32}))$$

Negated, negations pushed to literals:

$$(p_{11} \vee p_{12}) \wedge (p_{21} \vee p_{22}) \wedge (p_{31} \vee p_{32})$$
$$\wedge (\neg p_{11} \vee \neg p_{21}) \wedge (\neg p_{21} \vee \neg p_{31}) \wedge (\neg p_{11} \vee \neg p_{31})$$
$$\wedge (\neg p_{12} \vee \neg p_{22}) \wedge (\neg p_{22} \vee p_{32}) \wedge (\neg p_{12} \vee \neg p_{32})$$

Negation, block form (set of sets of literals), $O(n^3)$ clauses for PHP_n:

$$\langle [p_{11}, p_{12}]; [p_{21}, p_{22}]; [p_{31}, p_{32}]; [\neg p_{11}, \neg p_{21}]; [\neg p_{21}, \neg p_{31}]$$
$$; [\neg p_{11}, \neg p_{31}]; [\neg p_{12}, \neg p_{22}]; [\neg p_{22}, \neg p_{32}]; [\neg p_{12}, \neg p_{32}] \rangle$$