Truth tables

Heinz-Dieter Ebbinghaus, Jörg Flum, Wolfgang Thomas,
Mathematical logic, Section III.2, VIII.3

Kamal Lodaya

January 2023

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

First-order logic (FOL) syntax over symbols (Co, Fu, Pr, Re):

$$
\begin{aligned}
t ::= \quad & x \in V \mid c \in Co \mid f(t_1, \ldots, t_n),\ f \in Fu_n \\
A ::= \quad & p \in Pr \mid R(t_1, \ldots, t_n),\ R \in Re_n \mid t_1 \equiv t_2 \mid true \mid false \\
& \mid (\neg A) \mid (A \vee B) \mid (A \wedge B) \mid (A \rightarrow B) \mid (A \leftrightarrow B) \\
& \mid \exists x A \mid \forall x A
\end{aligned}
$$

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

First-order logic (FOL) syntax over symbols (Co, Fu, Pr, Re):

$$\begin{array}{rl} t ::= & x \in V \mid c \in Co \mid f(t_1, \ldots, t_n),\ f \in Fu_n \\ A ::= & p \in Pr \mid R(t_1, \ldots, t_n),\ R \in Re_n \mid t_1 \equiv t_2 \mid true \mid false \\ & \mid (\neg A) \mid (A \vee B) \mid (A \wedge B) \mid (A \rightarrow B) \mid (A \leftrightarrow B) \\ & \mid \exists x A \mid \forall x A \end{array}$$

Propositional calculus (PL) over symbols Pr (propositional variables) has simpler syntax.

$$\begin{array}{rl} A ::= & p \in Pr \mid true \mid false \\ & \mid (\neg A) \mid (A \vee B) \mid (A \wedge B) \mid (A \rightarrow B) \mid (A \leftrightarrow B) \end{array}$$

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

First-order logic (FOL) syntax over symbols (Co, Fu, Pr, Re):

$$\begin{array}{ll} t ::= & x \in V \mid c \in Co \mid f(t_1, \ldots, t_n),\ f \in Fu_n \\ A ::= & p \in Pr \mid R(t_1, \ldots, t_n),\ R \in Re_n \mid t_1 \equiv t_2 \mid true \mid false \\ & \mid (\neg A) \mid (A \vee B) \mid (A \wedge B) \mid (A \rightarrow B) \mid (A \leftrightarrow B) \\ & \mid \exists x A \mid \forall x A \end{array}$$

Propositional calculus (PL) over symbols Pr (propositional variables) has simpler syntax.

$$\begin{array}{ll} A ::= & p \in Pr \mid true \mid false \\ & \mid (\neg A) \mid (A \vee B) \mid (A \wedge B) \mid (A \rightarrow B) \mid (A \leftrightarrow B) \end{array}$$

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, page 29) giving a truth value $A[s]$ to the formula A.

Model checking (Alfred Tarski 1935)

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

s is a model of A when $s \models A$ (assignments called models).

s is a model of theory Th if s satisfies every formula in Th.

Model checking (Alfred Tarski 1935)

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

s is a model of A when $s \models A$ (assignments called models).

s is a model of theory Th if s satisfies every formula in Th.

Exercise

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

Model checking (Alfred Tarski 1935)

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

s is a model of A when $s \models A$ (assignments called models).
s is a model of theory Th if s satisfies every formula in Th.

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.

So for formula A in which n propositional variables occur, the truth table for A is finite and has 2^n rows.

Implication truth table

In mathematics, we say that if n is a prime > 2, then n is odd.

Implication truth table

In mathematics, we say that if n is a prime > 2, then n is odd.

Case $n = 3$: if 3 is a prime > 2 (T), then 3 is odd (T). The implication is T.

Case $n = 4$: if 4 is a prime > 2 (F), then 4 is odd (F). The implication is T.

Case $n = 9$: if 9 is a prime > 2 (F), then 9 is odd (T). The implication is T.

On the other hand, saying that if n is a prime, then n is odd gives:
Case $n = 2$: if 2 is a prime (T), then 2 is odd (F). This implication is F.

Implication truth table

In mathematics, we say that if n is a prime > 2, then n is odd.

Case $n = 3$: if 3 is a prime > 2 (T), then 3 is odd (T). The implication is T.

Case $n = 4$: if 4 is a prime > 2 (F), then 4 is odd (F). The implication is T.

Case $n = 9$: if 9 is a prime > 2 (F), then 9 is odd (T). The implication is T.

On the other hand, saying that if n is a prime, then n is odd gives:
Case $n = 2$: if 2 is a prime (T), then 2 is odd (F). This implication is F.

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?

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.

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.

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.

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

Formula validity and satisfiability

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: $(\neg\neg A) \leftrightarrow A$;
$\neg(A \vee B) \leftrightarrow ((\neg A) \wedge (\neg B))$; $\neg(A \wedge B) \leftrightarrow ((\neg A) \vee (\neg B))$;
$(A \wedge (B \vee C)) \leftrightarrow ((A \wedge B) \vee (A \wedge C))$;
$(A \vee (B \wedge C)) \leftrightarrow ((A \vee B) \wedge (A \vee C))$

Definition

Formula A is satisfiable ($Sat\ A$) if there is some assignment s such that $s \models A$.

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.

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.

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 $p \wedge (\neg p)$).

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 $A_0 = (\bigwedge_{p[s_0]=T} p) \wedge (\bigwedge_{p[s_0]=F} \neg p)$.

   This captures the assignment s_0.

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 $A_0 = (\bigwedge_{p[s_0]=T} p) \wedge (\bigwedge_{p[s_0]=F} \neg p)$.

   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 $\bigvee_{i=1}^{n} A_i$, where A_i is defined for model s_i as above.

Normal forms

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.

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.

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.

A literal is always satisfiable.

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 the size of the formula.

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 the size 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.

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 the size 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 the size 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.

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 size exponentially.

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 size exponentially.

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

$$\begin{aligned} B = \quad & (r_1 \vee \cdots \vee r_n) \wedge ((r_1 \rightarrow (P_1 \wedge Q_1)) \wedge \\ & ((r_2 \rightarrow (P_2 \wedge Q_2)) \wedge \cdots \wedge (r_n \rightarrow (P_n \wedge Q_n)) \end{aligned}$$

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 size exponentially.

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

$$\begin{aligned} B = \quad & (r_1 \vee \cdots \vee r_n) \wedge ((r_1 \rightarrow (P_1 \wedge Q_1)) \wedge \\ & ((r_2 \rightarrow (P_2 \wedge Q_2)) \wedge \cdots \wedge (r_n \rightarrow (P_n \wedge Q_n)) \end{aligned}$$

Equisatisfiability left to right: Suppose A is satisfiable.
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 satisfiable.

Equisatisfiability right to left: Suppose B is satisfiable.
Then one r_i in its first conjunct is assigned T.
By implication, $(P_i \wedge Q_i)$ is T. Then A is satisfiable.