

Decision Procedures for Verification

Part 1. Propositional Logic (1)

24.10.2022

Viorica Sofronie-Stokkermans

sofronie@uni-koblenz.de

Part 1: Propositional Logic

Literature (also for first-order logic)

Schöning: Logik für Informatiker, Spektrum

Fitting: First-Order Logic and Automated Theorem Proving, Springer

Part 1: Propositional Logic

Propositional logic

- logic of truth values
- decidable (but NP-complete)
- can be used to describe functions over a finite domain
- important for hardware applications (e.g., model checking)

1.1 Syntax

- propositional variables
- logical symbols
 - ⇒ Boolean combinations

Propositional Variables

Let Π be a set of **propositional variables**.

We use letters P, Q, R, S , to denote propositional variables.

Propositional Formulas

F_{Π} is the set of propositional formulas over Π defined as follows:

F, G, H	$::=$	\perp	(falsum)
		\top	(verum)
		$P, P \in \Pi$	(atomic formula)
		$\neg F$	(negation)
		$(F \wedge G)$	(conjunction)
		$(F \vee G)$	(disjunction)
		$(F \rightarrow G)$	(implication)
		$(F \leftrightarrow G)$	(equivalence)

Notational Conventions

- We omit brackets according to the following rules:
 - $\neg > \wedge > \vee > \rightarrow > \leftrightarrow$ (binding precedences)
 - \vee and \wedge are associative and commutative

1.2 Semantics

In **classical logic** (dating back to Aristoteles) there are “only” two truth values “true” and “false” which we shall denote, respectively, by 1 and 0.

There are **multi-valued logics** having more than two truth values.

Valuations

A propositional variable has no intrinsic meaning. The meaning of a propositional variable has to be defined by a valuation.

A Π -valuation is a map

$$\mathcal{A} : \Pi \rightarrow \{0, 1\}.$$

where $\{0, 1\}$ is the set of truth values.

Truth Value of a Formula in \mathcal{A}

Given a Π -valuation \mathcal{A} , the function $\mathcal{A}^* : \Sigma\text{-formulas} \rightarrow \{0, 1\}$ is defined inductively over the structure of F as follows:

$$\mathcal{A}^*(\perp) = 0$$

$$\mathcal{A}^*(\top) = 1$$

$$\mathcal{A}^*(P) = \mathcal{A}(P)$$

$$\mathcal{A}^*(\neg F) = 1 - \mathcal{A}^*(F)$$

$$\mathcal{A}^*(F \rho G) = B_\rho(\mathcal{A}^*(F), \mathcal{A}^*(G))$$

with B_ρ the Boolean function associated with ρ

For simplicity, we write \mathcal{A} instead of \mathcal{A}^* .

Truth Value of a Formula in \mathcal{A}

Example: Let's evaluate the formula

$$(P \rightarrow Q) \wedge (P \wedge Q \rightarrow R) \rightarrow (P \rightarrow R)$$

w.r.t. the valuation \mathcal{A} with

$$\mathcal{A}(P) = 1, \mathcal{A}(Q) = 0, \mathcal{A}(R) = 1$$

(On the blackboard)

1.3 Models, Validity, and Satisfiability

F is **valid** in \mathcal{A} (\mathcal{A} is a **model** of F ; F holds under \mathcal{A}):

$$\mathcal{A} \models F :\Leftrightarrow \mathcal{A}(F) = 1$$

F is **valid** (or is a **tautology**):

$$\models F :\Leftrightarrow \mathcal{A} \models F \text{ for all } \Pi\text{-valuations } \mathcal{A}$$

F is called **satisfiable** iff there exists an \mathcal{A} such that $\mathcal{A} \models F$.

Otherwise F is called **unsatisfiable** (or **contradictory**).

A set N of formulae is **satisfiable** iff there exists an \mathcal{A} such that $\mathcal{A} \models F$ for all $F \in N$.

Otherwise N is called **unsatisfiable** (or **contradictory**).

Example

$$F = (A \vee C) \wedge (B \vee \neg C)$$

A	B	C	$(A \vee C)$	$\neg C$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$
0	0	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	1	0
0	1	1	1	0	1	1
1	0	0	1	1	1	1
1	0	1	1	0	0	0
1	1	0	1	1	1	1
1	1	1	1	0	1	1

Let $\mathcal{A} : \{A, B, C\} \rightarrow \{0, 1\}$ with $\mathcal{A}(A) = 0$, $\mathcal{A}(B) = 1$, $\mathcal{A}(C) = 1$.

$$\mathcal{A} \models (A \vee C), \quad \mathcal{A} \models (B \vee \neg C)$$

$$\mathcal{A} \models (A \vee C) \wedge (B \vee \neg C)$$

$$\mathcal{A} \models \{(A \vee C), (B \vee \neg C)\}$$

1.3 Models, Validity, and Satisfiability

Examples:

$F \rightarrow F$ and $F \vee \neg F$ are **valid** for all formulae F .

Obviously, every **valid** formula is also **satisfiable**

$F \wedge \neg F$ is **unsatisfiable**

The formula P is **satisfiable**, but not **valid**

Example

$$F = (A \vee C) \wedge (B \vee \neg C)$$

A	B	C	$(A \vee C)$	$\neg C$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$
0	0	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	1	0
0	1	1	1	0	1	1
1	0	0	1	1	1	1
1	0	1	1	0	0	0
1	1	0	1	1	1	1
1	1	1	1	0	1	1

F is not valid:

$$\mathcal{A}_1(F) = 0 \text{ f\u00fcr } \mathcal{A}_1 : \{A, B, C\} \rightarrow \{0, 1\} \text{ mit } \mathcal{A}(A) = \mathcal{A}(B) = \mathcal{A}(C) = 0.$$

F is satisfiable:

$$\mathcal{A}_2(F) = 1 \text{ f\u00fcr } \mathcal{A} : \{A, B, C\} \rightarrow \{0, 1\} \text{ mit } \mathcal{A}(A) = 0, \mathcal{A}(B) = 1, \mathcal{A}(C) = 1.$$

Entailment and Equivalence

F entails (implies) G (or G is a consequence of F), written $F \models G$, if for all Π -valuations \mathcal{A} , whenever $\mathcal{A} \models F$ then $\mathcal{A} \models G$.

F and G are called **equivalent** if for all Π -valuations \mathcal{A} we have $\mathcal{A} \models F \Leftrightarrow \mathcal{A} \models G$.

Example

$$F = (A \vee C) \wedge (B \vee \neg C) \quad G = (A \vee B)$$

Check if $F \models G$

A	B	C	$(A \vee C)$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$	$(A \vee B)$
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Example

$$F = (A \vee C) \wedge (B \vee \neg C) \quad G = (A \vee B)$$

Check if $F \models G$

A	B	C	$(A \vee C)$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$	$(A \vee B)$
0	0	0	0	1	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	1
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	0	0	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Example

$$F = (A \vee C) \wedge (B \vee \neg C) \quad G = (A \vee B)$$

Check if $F \models G$ Yes, $F \models G$

A	B	C	$(A \vee C)$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$	$(A \vee B)$
0	0	1	1	0	0	0
0	0	0	0	1	0	0
0	1	1	1	1	1	1
0	1	0	0	1	0	1
1	0	1	1	0	0	1
1	0	0	1	1	1	1
1	1	1	1	1	1	1
1	1	0	1	1	1	1

Example

$$F = (A \vee C) \wedge (B \vee \neg C) \quad G = (A \vee B)$$

Check if $F \models G$ Yes, $F \models G$

... But it is not true that $G \models F$ (Notation: $G \not\models F$)

A	B	C	$(A \vee C)$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$	$(A \vee B)$
0	0	1	1	0	0	0
0	0	0	0	1	0	0
0	1	1	1	1	1	1
0	1	0	0	1	0	1
1	0	1	1	0	0	1
1	0	0	1	1	1	1
1	1	1	1	1	1	1
1	1	0	1	1	1	1

Entailment and Equivalence

F entails (implies) G (or G is a consequence of F), written $F \models G$, if for all Π -valuations \mathcal{A} , whenever $\mathcal{A} \models F$ then $\mathcal{A} \models G$.

F and G are called **equivalent** if for all Π -valuations \mathcal{A} we have $\mathcal{A} \models F \Leftrightarrow \mathcal{A} \models G$.

Proposition 1.1:

F entails G iff $(F \rightarrow G)$ is valid

Proposition 1.2:

F and G are equivalent iff $(F \leftrightarrow G)$ is valid.

Entailment and Equivalence

Extension to sets of formulas N in the “natural way”, e.g., $N \models F$ if for all Π -valuations \mathcal{A} : if $\mathcal{A} \models G$ for all $G \in N$, then $\mathcal{A} \models F$.

Validity vs. Unsatisfiability

Validity and unsatisfiability are just two sides of the same medal as explained by the following proposition.

Proposition 1.3:

$$F \text{ valid} \Leftrightarrow \neg F \text{ unsatisfiable}$$

Hence in order to design a theorem prover (validity checker) it is sufficient to design a checker for unsatisfiability.

Q: In a similar way, entailment $N \models F$ can be reduced to unsatisfiability. How?

Validity vs. Unsatisfiability

Validity and unsatisfiability are just two sides of the same medal as explained by the following proposition.

Proposition 1.4:

$$N \models F \Leftrightarrow N \cup \{\neg F\} \text{ unsatisfiable}$$

Hence in order to design a theorem prover (validity/entailment checker) it is sufficient to design a checker for unsatisfiability.

Checking Unsatisfiability

Every formula F contains only finitely many propositional variables. Obviously, $\mathcal{A}(F)$ depends only on the values of those finitely many variables in F under \mathcal{A} .

If F contains n distinct propositional variables, then it is sufficient to check 2^n valuations to see whether F is satisfiable or not.
 \Rightarrow truth table.

So the satisfiability problem is clearly decidable (but, by Cook's Theorem, NP-complete).

Nevertheless, in practice, there are (much) better methods than truth tables to check the satisfiability of a formula. (later more)

Checking Unsatisfiability

The satisfiability problem is clearly decidable (but, by Cook's Theorem, NP-complete).

For sets of propositional formulae of a certain type, satisfiability can be checked in polynomial time:

Examples: 2SAT, Horn-SAT (will be discussed in the exercises)

Dichotomy theorem. Schaefer [Schaefer, STOC 1978] identified six classes of sets S of Boolean formulae for which $SAT(S)$ is in PTIME. He proved that all other types of sets of formulae yield an NP-complete problem.

Substitution Theorem

Proposition 1.5:

Let F and G be equivalent formulas, let H be a formula in which F occurs as a subformula.

Then H is equivalent to H' where H' is obtained from H by replacing the occurrence of the subformula F by G .

(Notation: $H = H[F]$, $H' = H[G]$.)

Proof: By induction over the formula structure of H .

Some Important Equivalences

Proposition 1.6:

The following equivalences are valid for all formulas F, G, H :

$$(F \wedge F) \leftrightarrow F$$

$$(F \vee F) \leftrightarrow F$$

(Idempotency)

$$(F \wedge G) \leftrightarrow (G \wedge F)$$

$$(F \vee G) \leftrightarrow (G \vee F)$$

(Commutativity)

$$(F \wedge (G \wedge H)) \leftrightarrow ((F \wedge G) \wedge H)$$

$$(F \vee (G \vee H)) \leftrightarrow ((F \vee G) \vee H)$$

(Associativity)

$$(F \wedge (G \vee H)) \leftrightarrow ((F \wedge G) \vee (F \wedge H))$$

$$(F \vee (G \wedge H)) \leftrightarrow ((F \vee G) \wedge (F \vee H))$$

(Distributivity)

Some Important Equivalences

Proposition 1.7:

The following equivalences are valid for all formulas F, G, H :

$$(F \wedge (F \vee G)) \leftrightarrow F$$

$$(F \vee (F \wedge G)) \leftrightarrow F$$

(Absorption)

$$(\neg\neg F) \leftrightarrow F$$

(Double Negation)

$$\neg(F \wedge G) \leftrightarrow (\neg F \vee \neg G)$$

$$\neg(F \vee G) \leftrightarrow (\neg F \wedge \neg G)$$

(De Morgan's Laws)

$$(F \wedge G) \leftrightarrow F, \text{ if } G \text{ is a tautology}$$

$$(F \vee G) \leftrightarrow \top, \text{ if } G \text{ is a tautology}$$

(Tautology Laws)

$$(F \wedge G) \leftrightarrow \perp, \text{ if } G \text{ is unsatisfiable}$$

$$(F \vee G) \leftrightarrow F, \text{ if } G \text{ is unsatisfiable}$$

(Tautology Laws)

1.4 Normal Forms

We define **conjunctions** of formulas as follows:

$$\bigwedge_{i=1}^0 F_i = \top.$$

$$\bigwedge_{i=1}^1 F_i = F_1.$$

$$\bigwedge_{i=1}^{n+1} F_i = \bigwedge_{i=1}^n F_i \wedge F_{n+1}.$$

and analogously **disjunctions**:

$$\bigvee_{i=1}^0 F_i = \perp.$$

$$\bigvee_{i=1}^1 F_i = F_1.$$

$$\bigvee_{i=1}^{n+1} F_i = \bigvee_{i=1}^n F_i \vee F_{n+1}.$$

Literals and Clauses

A **literal** is either a propositional variable P or a negated propositional variable $\neg P$.

A **clause** is a (possibly empty) disjunction of literals.

Literals and Clauses

A **literal** is either a propositional variable P or a negated propositional variable $\neg P$.

A **clause** is a (possibly empty) disjunction of literals.

Example of clauses:

\perp

P

$\neg P$

$P \vee Q \vee R$

$P \vee \neg Q \vee \neg R$

$P \vee P \vee \neg Q \vee \neg R \vee R$

the empty clause

positive unit clause

negative unit clause

positive clause

clause

allow repetitions/complementary literals

CNF and DNF

A formula is in **conjunctive normal form (CNF, clause normal form)**, if it is a conjunction of disjunctions of literals (or in other words, a conjunction of clauses).

A formula is in **disjunctive normal form (DNF)**, if it is a disjunction of conjunctions of literals.

Warning: definitions in the literature differ:

- are complementary literals permitted?

- are duplicated literals permitted?

- are empty disjunctions/conjunctions permitted?

CNF and DNF

Checking the validity of CNF formulas or the unsatisfiability of DNF formulas is easy:

A formula in CNF is valid, if and only if each of its disjunctions contains a pair of complementary literals P and $\neg P$.

Conversely, a formula in DNF is unsatisfiable, if and only if each of its conjunctions contains a pair of complementary literals P and $\neg P$.

On the other hand, checking the unsatisfiability of CNF formulas or the validity of DNF formulas is known to be coNP-complete.