Verification of Reactive Systems
Lecture Notes

Filip Niksic
fniksic@mpi-sws.org

April 30, 2014

Continuing with the preliminaries for the course, in this lecture we cover basics of propositional logic. We start by defining the syntax of propositional logic, namely what propositional formulas *are*. Then we define the semantics of propositional logic, namely what propositional formulas *mean*.

The semantics is given in terms of *interpretations*—functions mapping propositional variables to truth values—which allow us to interpret the formula as being either true or false. Related notions are *satisfiability*—whether a formula is true under at least one interpretation—and *validity*—whether a formula is true under all interpretations. For checking satisfiability and validity, we introduce the *semantic tableaux*.

Next, we introduce the *logical implication* and *logical equivalence* of formulas. We show how to transform a given formula into equivalent formulas that have a certain syntactic form: *negational normal form (NNF)*, *disjunctive normal form (DNF)* or *conjunctive (or clausal) normal form (CNF)*. We observe that the transformation into CNF might introduce an exponential blow-up in the size of the formula. However, if we drop the requirement of the resulting formula being equivalent to the initial one, and instead only require it to be *equisatisfiable*, then there is a more efficient transformation into CNF.

Finally, we introduce the notion of Craig's interpolation and prove the interpolation lemma.

1 Syntax

The object of interest in propositional logic are propositional formulas, which are nothing but words of a certain form. The basic building blocks for these words are *propositional variables*:

$$\Pi = \{p, q, r, \ldots\}$$

We usually assume Π is a countably infinite set.

Propositional formulas F are built from propositional variables using the following production rule:

$$F ::= \bot \mid \top \mid p \mid (F_1 \wedge F_2) \mid (F_1 \vee F_2) \mid (F_1 \to F_2) \mid (F_1 \leftrightarrow F_2) \mid (\neg F)$$

The rule says that a formula is either one of the constants $\bot$ (false) and $\top$ (true), a propositional variable, a *conjunction*, *disjunction*, *conditional* or a *biconditional* of two formulas, or a *negation* of a formula.

To reduce the number of parentheses in formulas, we adopt the following precedence of the connectives, from highest to lowest: $\neg$, $\wedge$, $\vee$, $\to$ and $\leftrightarrow$ (the last two have the same precedence). We also adopt the left associativity of $\wedge$ and $\vee$, and the right associativity of $\to$ and $\leftrightarrow$.

Example 1. Here are several formulas in two versions. Fully parenthesized:

$$((p \wedge q) \to (p \to q))$$
$$((p \to (q \to r)) \to ((\neg r) \to ((\neg q) \to (\neg p))))$$
$$((p \to (q \vee r)) \to (p \to r))$$

Parenthesized using the introduced precedence and associativity rules:

$$p \wedge q \to p \to q$$
$$(p \to q \to r) \to \neg r \to \neg q \to \neg p$$
$$(p \to q \vee r) \to p \to r$$

2 Semantics

We give meaning to formulas using *interpretations*. An interpretation is a partial function

$$I \colon \Pi \rightharpoonup \{\mathsf{false}, \mathsf{true}\},$$

which assigns truth values to propositional variables. Given an interpretation I, we define the truth of a formula F under I (denoted $I \vDash F$) using the following inductive definition:

$I \nvDash \bot$	
$I \vDash \top$	
$I \vDash p$	if and only if $I(p) = \mathsf{true}$ for $p \in \mathrm{dom}(I)$
$I \vDash F \wedge G$	if and only if $I \vDash F$ and $I \vDash G$
$I \vDash F \vee G$	if and only if $I \vDash F$ or $I \vDash G$
$I \vDash F \to G$	if and only if $I \vDash F$ implies $I \vDash G$
$I \vDash F \leftrightarrow G$	if and only if $I \vDash F$ implies $I \vDash G$ and $I \vDash G$ implies $I \vDash F$
$I \vDash \neg F$	if and only if $I \nvDash F$

Example 2. Let $I(p) = \mathsf{true}$ and $I(q) = \mathsf{false}$. Then

$$I \nvDash p \wedge q$$
$$I \vDash p \wedge q \to p \vee \neg q$$

We say that F is *satisfiable* if there is an interpretation I such that $I \vDash F$. We say that F is *valid* if $I \vDash F$ for all interpretations I. We say that F is *refutable* if there is an interpretation I such that $I \nvDash F$.

Theorem 1. *F is valid if and only if $\neg F$ is unsatisfiable. F is valid if and only if F is irrefutable.*

Example 3. Here is a list of several formulas, along with their status with respect to satisfiability, validity and refutability.

$p \wedge q$	satisfiable, not valid, refutable
$p \wedge q \to p \vee \neg q$	satisfiable, valid, irrefutable
$p \wedge \neg(q \to p)$	unsatisfiable, not valid, refutable
$p \wedge \neg p$	unsatisfiable, not valid, refutable
$p \vee \neg p$	satisfiable, valid, irrefutable

3 Semantic Tableaux

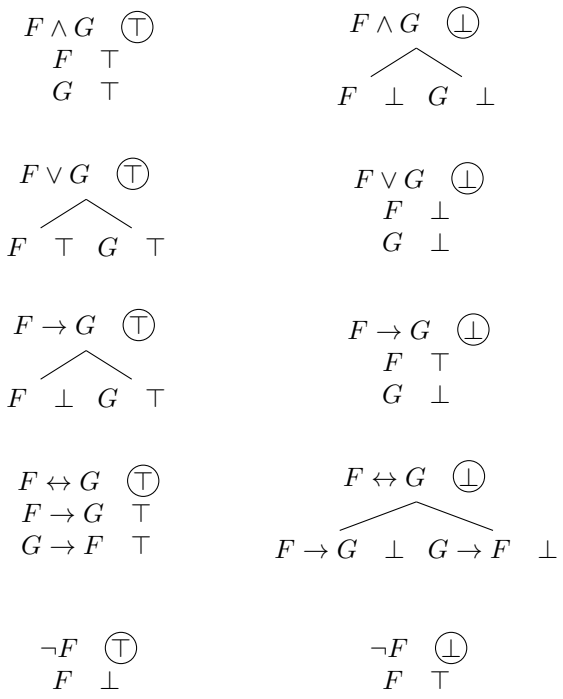


Figure 1: Rules for deconstructing formulas.

One way to check satisfiability or validity of a formula is to evaluate it under all possible interpretations on the propositional variables that appear in the formula. We can list all the interpretations in a *truth table*.

Example 4. We check the satisfiability and validity of $p \wedge q \to p \vee \neg q$ using a truth table.

p	q	$p \wedge q$	$p \vee \neg q$	$p \wedge q \to p \vee \neg q$
false	false	false	true	true
false	true	false	false	true
true	false	false	true	true
true	true	true	true	true

From the table, we can see that the formula is satisfiable: true appears at least once in the last column, namely, there is an interpretation that makes the formula true. Furthermore, the formula is valid: only true appears in the last column, namely, all interpretations make the formula true.

Instead of exhaustively listing all the interpretations, we can check for satisfiability, validity or refutability using *semantic tableaux*. In semantic tableaux, we start by asserting that the formula is either true or false, depending on whether we are looking for an interpretation that satisfies it or refutes it:

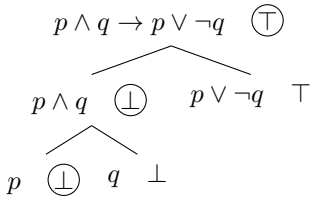
$$F \quad \top \qquad \text{or} \qquad F \quad \bot$$

We continue by successively applying the rules for deconstructing formulas (Figure 3). Once we are finished with a formula, we circle the truth constant next to it to keep track of which formulas are still pending. An important thing to note is that we have to carry over each unprocessed formula to every sub-branch of the tableau.

Once we get all the way to propositional variables, the expansion stops. The constant next to a variable tells us whether the interpretation should map the variable to true or false. If we get a contradiction, i.e. a propositional variable is supposed to be mapped to both true and false, we say that the branch is *closed*. We mark a closed branch with $\times$. If the branch is not closed, it is *open*, and we can read out an interpretation that satisfies or refutes the initial formula.

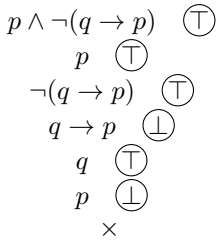
If the tableau is fully expanded and all the branches are closed, the interpretation satisfying or refuting the formula does not exist.

Example 5. Let us check the satisfiability of $p \wedge q \to p \vee \neg q$.



The leftmost branch is open, so we can use it to construct a satisfying interpretation: $I(p) = \mathsf{false}$, $I(q)$ can be anything, say $I(p) = \mathsf{true}$.

Example 6. Let us check the satisfiability of $p \wedge \neg(q \to p)$.



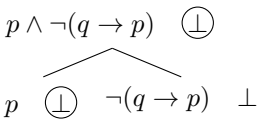
The only branch of the tableau is closed, therefore the formula is not satisfiable.

Example 7. Let us check the validity of $p \wedge q \to p \vee \neg q$. The formula is valid if it is irrefutable, so let us check refutability.

$$
\begin{array}{rl}
p \wedge q \to p \vee \neg q & \textcircled{\bot} \\
p \wedge q & \textcircled{\top} \\
p \vee \neg q & \textcircled{\bot} \\
p & \textcircled{\top} \\
q & \textcircled{\top} \\
p & \textcircled{\bot} \\
\neg q & \bot \\
\multicolumn{2}{c}{\times}
\end{array}
$$

The only branch of the tableau is closed, therefore the formula is irrefutable, i.e. valid.

Example 8. Let us check the validity of $p \wedge \neg(q \to p)$. The formula is valid if it is irrefutable, so let us check refutability.



The leftmost branch is open, so we can use it to construct a refuting interpretation: $I(p) = \mathsf{false}$, $I(q)$ can be anything, say $I(p) = \mathsf{true}$. As the formula is refutable, it is not valid.

4 Logical Implication and Equivalence

We say that F *implies* G (denoted $F \Rightarrow G$) if for all interpretations I, $I \vDash F$ implies $I \vDash G$. Formulas F and G are *equivalent* (denoted $F \Leftrightarrow G$) if F implies G and G implies F.

Note that $\Rightarrow$ and $\Leftrightarrow$ are not connectives of propositional logic, as we present it here. Likewise, $F \Rightarrow G$ and $F \Leftrightarrow G$ are not propositional formulas. Logical implication and equivalence are concepts of the meta-theory we are using to reason about propositional logic. However, they are closely related to conditional and bi-conditional.

Theorem 2. *$F \Rightarrow G$ if and only if $F \to G$ is valid. $F \Leftrightarrow G$ if and only if $F \leftrightarrow G$ is valid.*

Example 9. Here is a list of several useful equivalences. They are stated using propositional variables p, q and r, but they remain valid even if we replace the variables with arbitrary formulas F, G and H.

We start with a couple of equivalences that allow us to eliminate conditional and bi-conditional from the formulas:

$$
\begin{aligned}
&p \to q \Leftrightarrow \neg p \vee q \\
&p \leftrightarrow q \Leftrightarrow (p \to q) \wedge (q \to p)
\end{aligned}
$$

De Morgan's laws:

$$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$$
$$\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$$

Double negation:

$$\neg\neg p \Leftrightarrow p$$

Distributivity:

$$p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$$
$$(p \vee q) \wedge r \Leftrightarrow (p \wedge r) \vee (q \wedge r)$$
$$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$$
$$(p \wedge q) \vee r \Leftrightarrow (p \vee r) \wedge (q \vee r)$$

The following theorem says that if we replace a subformula in F by an equivalent formula, the obtained top-level formula F' remains equivalent to F.

Theorem 3 (Replacement Theorem)**.** *Let F be a propositional formula that contains G as a subformula. Let G' be a formula such that $G \Leftrightarrow G'$. Then $F \Leftrightarrow F[G'/G]$, where $F[G'/G]$ is a formula obtained by replacing zero or more occurences of G in F by G'.*

5 Normal Forms

We say that a formula is in *negational normal form (NNF)* if it does not contain conditionals ($\rightarrow$) and bi-conditionals ($\leftrightarrow$), and if it only contains negations at the inner-most level, attached to propositional variables. By successively applying the Replacement Theorem, and using the equivalences from Example 9 (elimination of conditionals, De Morgan's laws and elimination of double negation), every formula can be transformed into an equivalent one that is in NNF.

Example 10. We transform formula $p \rightarrow \neg(q \rightarrow p)$ into an equivalent NNF formula as follows.

$$\begin{aligned}
p \rightarrow \neg(q \rightarrow p) &\Leftrightarrow \neg p \vee \neg(q \rightarrow p) && \text{elimination of } \rightarrow \\
&\Leftrightarrow \neg p \vee \neg(\neg q \vee p) && \text{elimination of } \rightarrow \\
&\Leftrightarrow \neg p \vee (\neg\neg q \wedge \neg p) && \text{De Morgan's law} \\
&\Leftrightarrow \neg p \vee (q \wedge \neg p) && \text{elimination of double negation}
\end{aligned}$$

By going further, we can use the distributivity laws to transform the formula into *disjunctive normal form (DNF)*:

$$\bigvee_{1 \leq i \leq m} \bigwedge_{1 \leq j \leq n_i} l_{ij}\,,$$

where each l_{ij} is a *literal*—either one of the constants $\bot$ and $\top$, a propositional variable, or a negation of a propositional variable.

Alternatively, we can use the distributivity laws to transform the formula into *conjunctive (or clausal) normal form (CNF)*:

$$\bigwedge_{1 \le i \le m} \bigvee_{1 \le j \le n_i} l_{ij} \,.$$

Disjunctions of literals are called *clauses*.

Example 11. The final formula in Example 10 is already in DNF. By applying the distributivity law, we transform it into an equivalent CNF formula:

$$\neg p \vee (q \wedge \neg p) \Leftrightarrow (\neg p \vee q) \wedge (\neg p \vee \neg p) \,.$$

5.1 Equisatisfiability and Tseitin Transformation

Note: This material was not covered in the lecture.

Converting a formula into an equivalent CNF formula can cause an exponential blow-up in the size of the formula. Consider the following formula:

$$(p_0 \wedge p_1) \vee (p_2 \wedge p_3) \vee \ldots \vee (p_{2n} \wedge p_{2n+1}) \,.$$

The formula clearly has $\mathcal{O}(n)$ size. However, after conversion into CNF, the formula has $\mathcal{O}(2^n)$ size:

$$(p_0 \vee p_2 \vee \ldots \vee p_{2n}) \wedge (p_1 \vee p_2 \vee \ldots \vee p_{2n}) \wedge \ldots \wedge (p_1 \vee p_3 \vee \ldots \vee p_{2n+1}) \,.$$

Sometimes we do not need to get an equivalent formula in CNF, but rather an *equisatisfiable* one—one that is satisfiable if and only if the initial formula is. In that case, there is a transformation, called *Tseitin transformation*, that can achieve this with only a linear increase in the size of the formula.

Given a formula F, for each non-atomic (i.e. not $\bot$, $\top$ or a propositional variable) subformula G of F we introduce a fresh propositional variable p_G. Consider the following formula:

$$p_F \wedge \bigwedge_{\substack{G = G_1 \circ G_2 \\ \circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}}} (p_G \leftrightarrow p_{G_1} \circ p_{G_2}) \wedge \bigwedge_{G = \neg G_1} (p_G \leftrightarrow \neg p_{G_1}) \,.$$

The formula is equisatisfiable to F, and it has size linear in the size of F (as the number of subformulas in F is linear in the size of F). Furthermore, each of the small subformulas $p_G \leftrightarrow p_{G_1} \circ p_{G_2}$ and $p_G \leftrightarrow \neg p_{G_1}$ has a constant transformation into an equivalent CNF.

Example 12. In this example we show a constant transformation into CNF of each small subformula that appears as part of Tseitin transformation.

$$\begin{aligned}
p_G \leftrightarrow p_{G_1} \wedge p_{G_2} &\Leftrightarrow (\neg p_G \vee p_{G_1}) \wedge (\neg p_G \vee p_{G_2}) \wedge (p_G \vee \neg p_{G_1} \vee \neg p_{G_2}) \\
p_G \leftrightarrow p_{G_1} \vee p_{G_2} &\Leftrightarrow (\neg p_G \vee p_{G_1} \vee p_{G_2}) \wedge (p_G \vee \neg p_{G_1}) \wedge (p_G \vee \neg p_{G_2}) \\
p_G \leftrightarrow p_{G_1} \rightarrow p_{G_2} &\Leftrightarrow (\neg p_G \vee \neg p_{G_1} \vee p_{G_2}) \wedge (p_G \vee p_{G_1}) \wedge (p_G \vee \neg p_{G_2}) \\
p_G \leftrightarrow p_{G_1} \leftrightarrow p_{G_2} &\Leftrightarrow (\neg p_G \vee \neg p_{G_1} \vee p_{G_2}) \wedge (\neg p_G \vee p_{G_1} \vee \neg p_{G_2}) \\
&\qquad \wedge (p_G \vee \neg p_{G_1} \vee \neg p_{G_2}) \wedge (p_G \vee p_{G_1} \vee p_{G_2}) \\
p_G \leftrightarrow \neg p_{G_1} &\Leftrightarrow (\neg p_G \vee \neg p_{G_1}) \wedge (p_G \vee p_{G_1})
\end{aligned}$$

Example 13. Let us apply Tseitin transformation to $F = p \to \neg(q \to p)$. As F has three non-atomic subformulas (including F itself), we introduce three new propositional variables: s, t and u. In the first step we get a formula equisatisfiable to F:

$$s \wedge (s \leftrightarrow (p \to t)) \wedge (t \leftrightarrow \neg u) \wedge (u \leftrightarrow (q \to p)) .$$

Replacing each small subformula with an equivalent one in CNF (see Example 12), we get the final CNF formula equisatisfiable to F:

$$\begin{aligned} s \wedge (\neg s \vee \neg p \vee t) \wedge (s \vee p) \wedge (s \vee \neg t) \\ \wedge (\neg t \vee \neg u) \wedge (t \vee u) \\ \wedge (\neg u \vee \neg q \vee p) \wedge (u \vee q) \wedge (u \vee \neg p) \end{aligned}$$

While in this particular case the final formula is longer than the one obtained in Example 11, for larger examples the linear asymptotic complexity kicks in, and formulas produced using Tseitin transformation become much shorter.

6 Craig's Interpolation

Note: This material was not covered in the lecture.

Theorem 4 (Craig's Interpolation Lemma)**.** *Let F and G be formulas such that $F \wedge G$ is unsatisfiable. Then there exists a formula H such that:*

- $\mathrm{var}(H) \subseteq \mathrm{var}(F) \cap \mathrm{var}(G)$
- $F \Rightarrow H$
- *$H \wedge G$ is unsatisfiable.*

Proof. Let us assume without loss of generality that F and G are in DNF, namely that $F = F_1 \vee \ldots \vee F_m$ and $G = G_1 \vee \ldots \vee G_n$.

If F is unsatisfiable, we can take $H = \bot$, and if G is unsatisfiable, we can take $H = \top$. Therefore, let us assume that both F and G are satisfiable. Without loss of generality, we can assume that all the disjuncts $F_1, \ldots, F_m, G_1, \ldots, G_n$ are satisfiable.

For fixed i and j, it is not difficult to see that $F_i \wedge G_j$ is unsatisfiable. We claim that there are literals L in F_i and L' in G_j that share the same propositional variable and satisfy $\neg L \Leftrightarrow L'$. For suppose there are no such literals. Let $F_i = F_{i1} \wedge \ldots \wedge F_{ik}$, $G_j = G_{j1} \wedge \ldots \wedge G_{jl}$ and assume that for all s and t, $\neg F_{is} \not\Leftrightarrow G_{jt}$. Then we can define an interpretation I such that $I \vDash F_{is}$ and $I \vDash G_{jt}$. But then $I \vDash F_i \wedge G_j$, contradicting unsatisfiability of $F_i \wedge G_j$.

Let H_{ij} be the literal in F_i such that $\neg H_{ij}$ is equivalent to a literal in G_j. We define

$$H := \bigvee_{1 \le i \le m} \bigwedge_{1 \le j \le n} H_{ij} .$$

It is not difficult to verify that H has the desired properties. □

Formula H from the theorem is called *Craig's interpolant* of F and G.

Example 14. Let us find Craig's interpolant for the formulas p and $\neg(q \to p)$. Transforming the second formula into DNF, we get $q \wedge \neg p$. Since both formulas have a single disjunct, the interpolant consists of a single literal. According to the proof of the theorem, that disjunct is p and $H = p$ is the interpolant.