Logic for Computer Science - Week 3
The Semantics of Propositional Logic

Ștefan Ciobâcă

November 30, 2017

1 Reminder

So far we have seen:

1. an informal introduction to logic, concentranting on identifying valid arguments;
2. that natural language is not appropriate for the study of logic, because of inherent ambiguities;
3. a definition of a formal language called PL (for propositional logic), consisting of propositional variables linked together with logical connectives.

In our definition of PL, we have only discussed syntactic matters, i.e. what formulae are correct from a grammar point of view, but we have never said what each formula *means*.

In the present lecture, we will discuss how we can assign meaning to individual formulae. In other words, we will give semantics to propositional formulae. Reminder: semantics means meaning.

2 Boolean Values and Boolean Functions

The set $B = \{0, 1\}$ is called the set of boolean values (also called truth values). The value 0 denotes falsehood and the value 1 denotes truth.

The function $\_ : B \to B$ is called *logical negation* and is defined as follows: $\overline{0} = 1$ and $\overline{1} = 0$.

The function $+ : B \times B \to B$ is called *logical disjunction* and is defined as follows: $0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 1$.

The function $\cdot : B \times B \to B$ is called *logical conjunction* and is defined as follows: $0 \cdot 0 = 0, 0 \cdot 1 = 0, 1 \cdot 0 = 1, 1 \cdot 1 = 1$.

The tuple $(B, +, \cdot, \_)$ is a *boolean algebra*.

3 Assignments

A *truth assignment* (or simply *assignment* from hereon) is any function $\tau : A \to B$. In other words, an assignment is a function that associates to any propositional variable a truth value.

Example 3.1. *Let $\tau_1 : A \to B$ be a function defined as follows:*

1. $\tau_1(\mathtt{p}) = 1$;
2. $\tau_1(\mathtt{q}) = 0$;
3. $\tau_1(\mathtt{r}) = 1$;
4. $\tau_1(a) = 0$ *for all* $a \in A \setminus \{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$.

As it is a function from A to B, τ_1 is a truth assignment.

Example 3.2. *Let $\tau_2 : A \to B$ be a function defined as follows:*

1. $\tau_2(\mathtt{p}) = 0$;
2. $\tau_2(\mathtt{q}) = 0$;
3. $\tau_2(\mathtt{r}) = 1$;
4. $\tau_2(a) = 1$ *for all* $a \in A \setminus \{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$.

As it is a function from A to B, τ_2 is also a truth assignment.

Example 3.3. *Let $\tau_3 : A \to B$ be a function defined as follows:*

1. $\tau_3(a) = 0$ *for all* $a \in A$.

As it is a function from A to B, τ_3 is also a truth assignment.

3.1 Truth Value of A Formula in An Assignment

The truth value of a formula φ in an assignment τ is denoted by $\hat{\tau}$ and is defined recursively as follows:

$$\hat{\tau}\Big(\varphi\Big) = \begin{cases} \overline{\tau(\varphi),} & \text{if } \varphi \in A; \\ \hat{\tau}\Big(\varphi'\Big), & \text{if } \varphi = \neg\varphi' \text{ and } \varphi' \in PL; \\ \hat{\tau}\Big(\varphi_1\Big) \cdot \hat{\tau}\Big(\varphi_2\Big), & \text{if } \varphi = (\varphi_1 \wedge \varphi_2) \text{ and } \varphi_1, \varphi_2 \in PL; \\ \hat{\tau}\Big(\varphi_1\Big) + \hat{\tau}\Big(\varphi_2\Big), & \text{if } \varphi = (\varphi_1 \vee \varphi_2) \text{ and } \varphi_1, \varphi_2 \in PL. \end{cases}$$

In fact $\hat{\tau} : PL \to B$ is a function called *the homomorphic extension* of the assignment $\tau : A \to B$ to the entire set of formulae PL, but do not feel obligated to recall this name.

Here is an example of how to compute the truth value of the formula $(\mathtt{p} \vee \mathtt{q})$ in the truth assignment τ_1:

$\hat{\tau_1}\Big((\mathtt{p} \vee \mathtt{q})\Big) = \hat{\tau_1}\Big(\mathtt{p}\Big) + \hat{\tau_1}\Big(\mathtt{q}\Big) = \tau_1(\mathtt{p}) + \tau_1(\mathtt{q}) = \mathtt{1} + \mathtt{0} = \mathtt{1}.$

We conclude that the truth value of $(\mathtt{p} \vee \mathtt{q})$ in $\hat{\tau_1}$ is 1.

Here is another example, where we compute the truth value of the formula $\neg(\mathtt{p} \wedge \mathtt{q})$ in the truth assignment τ_1:

$\hat{\tau_1}\Big(\neg(\mathtt{p} \wedge \mathtt{q})\Big) = \overline{\hat{\tau_1}\Big((\mathtt{p} \wedge \mathtt{q})\Big)} = \overline{\hat{\tau_1}\Big(\mathtt{p}\Big) \cdot \hat{\tau_1}\Big(\mathtt{q}\Big)} = \overline{\tau_1(\mathtt{p}) \cdot \tau_1(\mathtt{q})} = \overline{\mathtt{1} \cdot \mathtt{0}} = \overline{\mathtt{0}} = \mathtt{1}.$

We conclude that the truth value of the formula $\neg(\mathtt{p} \wedge \mathtt{q})$ in τ_1 is 1.

Here is yet another example, where we compute the truth value of the formula $\neg\neg q$ in the truth assignment τ_2:

$\hat{\tau_2}\Big(\neg\neg\mathtt{q}\Big) = \overline{\hat{\tau_2}\Big(\neg\mathtt{q}\Big)} = \overline{\overline{\hat{\tau_2}\Big(\mathtt{q}\Big)}} = \overline{\overline{\tau_2(\mathtt{q})}} = \overline{\overline{\mathtt{0}}} = \overline{\mathtt{1}} = \mathtt{0}.$

So the truth value of $\neg\neg\mathtt{q}$ in τ_2 is 0.

Remark 3.1. *Important!*

It does not make sense to say "the truth value of a formula". It makes sense to say "the truth value of a formula in an assignment".

Also, it does not make sense to say "the formula is true" or "the formula is false". Instead, it makes sense to say "this formula is true/false in this assignment".

This is because a formula could be true in an assignment but false in another. For example, the formula $\neg\neg\mathtt{p}$ is true in τ_1, but false in τ_2.

An *assignment τ satisfies φ* if $\hat{\tau}\Big(\varphi\Big) = 1$. Instead of *$\tau$ satisfies φ*, we may equivalently say any of the following:

1. τ is a model of φ;
2. τ is true of φ;
3. φ holds in/at τ;
4. τ makes φ true;

We write $\tau \models \varphi$ (and read: τ is a model of φ or τ satisfies φ, etc.) iff $\hat{\tau}\Big(\varphi\Big) = 1$. We write $\tau \not\models \varphi$ (and read: τ is not a model of φ or τ does not satisfy φ) iff $\hat{\tau}\Big(\varphi\Big) = 0$.

Definition 3.1. *The relation $\models$, between assignments and formulae, is called the* satisfaction relation. *The relation is defined by: $\tau \models \varphi$ iff $\hat{\tau}\Big(\varphi\Big) = 1$.*

Example 3.4. *The assignment τ_1 defined in the previous examples is a model of $\neg(\mathtt{p} \wedge \mathtt{q})$.*

The assignment τ_1 is not a model of $(\neg\mathtt{p} \wedge \mathtt{q})$.

4 Satisfiability, Validity, Equivalence, Logical Consequence

4.1 Satisfiable Formulae

Definition 4.1. *A formula φ is* satisfiable *if, by definition, there exists at least an assignment τ such that $\tau \models \varphi$ (i.e., if there exists at least a model of φ).*

Example 4.1. *The formula $(\mathtt{p} \vee \mathtt{q})$ is satisfiable, since it has a model (for example τ_1 above).*

Example 4.2. *The formula $\neg\mathtt{p}$ is also satisfiable: for example, the assignment τ_3 above makes the formula true.*

Example 4.3. *The formula $(\mathtt{p} \wedge \neg\mathtt{p})$ is not satisfiable, since it is false in any assignment.*

Proof. Let us consider an arbitrary assignment $\tau : A \to B$.

We have that $\hat{\tau}\Big((\mathtt{p} \wedge \neg\mathtt{p})\Big) = \hat{\tau}\Big(\mathtt{p}\Big) \cdot \hat{\tau}\Big(\neg\mathtt{p}\Big) = \tau(\mathtt{p}) \cdot \overline{\hat{\tau}\Big(\mathtt{p}\Big)} = \tau(\mathtt{p}) \cdot \overline{\hat{\tau}\Big(\mathtt{p}\Big)} = \tau(\mathtt{p}) \cdot \overline{\tau(\mathtt{p})}$.

But $\tau(\mathtt{p})$ can be either 0 or 1:

1. in the first case ($\tau(\mathtt{p}) = 0$), we have that $\hat{\tau}\Big((\mathtt{p} \wedge \neg\mathtt{p})\Big) = \ldots = \tau(\mathtt{p}) \cdot \overline{\tau(\mathtt{p})} = \mathtt{0} \cdot \overline{\mathtt{0}} = \mathtt{0} \cdot \mathtt{1} = \mathtt{0}$;
2. in the second case ($\tau(\mathtt{p}) = 1$) , we have that $\hat{\tau}\Big((\mathtt{p} \wedge \neg\mathtt{p})\Big) = \ldots = \tau(\mathtt{p}) \cdot \overline{\tau(\mathtt{p})} = \mathtt{1} \cdot \overline{\mathtt{1}} = \mathtt{1} \cdot \mathtt{0} = \mathtt{0}$.

Therefore, in any case, we have that $\hat{\tau}\Big((\mathtt{p} \wedge \neg\mathtt{p})\Big) = \mathtt{0}$. But τ was chosen arbitrarily, and therefore the result must hold for any assignment τ: $(\mathtt{p} \wedge \neg\mathtt{p})$ is false in any assignment, which means that it is not satisfiable.

□

A formula that is not satisfiable is called a *contradiction*.

Example 4.4. *As we have seen above, $(\mathtt{p} \wedge \neg\mathtt{p})$ is a contradiction.*

4.2 Valid Formulae

Definition 4.2. *A formula φ is* valid *if, by definition, any assignment τ has the property that $\tau \models \varphi$ (any assignment is a model of the formula).*

A valid formula is also called a *tautology*.

Example 4.5. *The formula $(\mathtt{p} \vee \neg\mathtt{p})$ is valid, because it is true in any assignment: let τ be an arbitrary assignment; we have that $\hat{\tau}\Big((\mathtt{p} \vee \neg\mathtt{p})\Big) = \tau(\mathtt{p}) + \overline{\tau(\neg\mathtt{p})}$, which is either $0 + 1$ or $1 + 0$, which is 1 in any case.*

Example 4.6. *The formula $\mathtt{p}$ is not valid (because there is an assignment (for example τ_3) that makes it false).*

4.3 Contingent Formulae

Definition 4.3. *A formula that is neither a contradiction nor a tautology is called* contingent*.*

Example 4.7. $(\mathtt{p} \wedge \neg\mathtt{p})$ *is a contradiction.*
$\mathtt{p}$ *is contingent.*
$(\mathtt{p} \vee \neg\mathtt{p})$ *is a tautology.*

4.4 Equivalence

Definition 4.4. *We say that two formulae $\varphi_1, \varphi_2 \in PL$ are* equivalent *and we write $\varphi_1 \equiv \varphi_2$ if, for any assignment $\tau : A \to B$, $\hat{\tau}(\varphi_1) = \hat{\tau}(\varphi_2)$.*

Intuitively, formulae that are equivalent have the same meaning (express the same thing).

Example 4.8. *At the start of the course, someone asked whether* $\mathtt{p}$ *and* $\neg\neg\mathtt{p}$ *are equal. Of course not, I said, since one has* 1 *symbol and the other* 3 *symbols from the alphabet.*

However, we are now ready to understand the relation between them: $\mathtt{p} \equiv \neg\neg\mathtt{p}$*. In other words, even if they are not equal, they are equivalent: they express the same thing.*

To prove $\mathtt{p} \equiv \neg\neg\mathtt{p}$*, we have to show they have the same truth value in any assignment. Let τ be an arbitrary assignment. We have that* $\hat{\tau}\Big(\neg\neg\mathtt{p}\Big) = \overline{\overline{\tau(\mathtt{p})}} = \tau(\mathtt{p}) = \hat{\tau}\Big(\mathtt{p}\Big)$*. In summary,* $\hat{\tau}\Big(\neg\neg\mathtt{p}\Big) = \hat{\tau}\Big(\mathtt{p}\Big)$*. As τ was chosen arbitrarily, it follows that* $\hat{\tau}\Big(\neg\neg\mathtt{p}\Big) = \hat{\tau}\Big(\mathtt{p}\Big)$ *for any assignment τ and therefore p is equivalent to $\neg\neg p$.*

Example 4.9. *The following equivalence holds: $p \vee q \equiv \neg(\neg p \wedge \neg q)$ (check it).*

Here are two more equivalences, known as De Morgan's laws:

Theorem 4.1. *For any formulae $\varphi_1, \varphi_2 \in PL$, we have that:*

1. $\neg(\varphi_1 \vee \varphi_2) = (\neg\varphi_1 \wedge \neg\varphi_2)$*;*
2. $\neg(\varphi_1 \wedge \varphi_2) = (\neg\varphi_1 \vee \neg\varphi_2)$*.*

4.5 Semantical Consequence

Definition 4.5. *Let $\Gamma = \{\varphi_1, \ldots, \varphi_n, \ldots\}$ be a set of formulae. We say that φ is a* semantical consequence *of Γ and we write $\Gamma \models \varphi$, if is any model of all formulae in Γ is a model of φ as well.*

We also say that φ is a logical consequence of Γ or that φ is a tautological consequence of Γ instead of φ is a semantical consequence of Γ.

Example 4.10. *Let* $\Gamma = \{\mathtt{p}, (\neg\mathtt{p} \vee \mathtt{q})\}$. *We have that* $\Gamma \models \mathtt{q}$.

Indeed, let τ *be a model of* $\mathtt{p}$ *and of* $(\neg\mathtt{p} \vee \mathtt{q})$. *As* τ *is a model of* $\mathtt{p}$, *by definition, we have that* $\tau(p) = 1$.

As τ *is a model of* $(\neg\mathtt{p} \vee \mathtt{q})$, *it follows that* $\hat{\tau}\Big((\neg\mathtt{p} \vee \mathtt{q})\Big) = \mathtt{1}$. *But* $\hat{\tau}\Big((\neg\mathtt{p} \vee \mathtt{q})\Big)\overline{\tau(\mathtt{p})} + \tau(\mathtt{q})$. *But* $\tau(p) = 1$, *and therefore* $\hat{\tau}\Big((\neg\mathtt{p} \vee \mathtt{q})\Big) = \mathtt{0} + \tau(\mathtt{q}) = \tau(\mathtt{q})$. *This means that* $\tau(\mathtt{q}) = 1$.

This means that τ *is a model of* $\mathtt{q}$. *We assumed that* τ *is a model of* $\mathtt{p}$ *and of* $(\neg\mathtt{p} \vee \mathtt{q})$ *and we show that necessarily* τ *is a model of* $\mathtt{q}$. *But this is exactly the definition of* $\{\mathtt{p}, (\neg\mathtt{p} \vee \mathtt{q})\} \models \mathtt{q}$, *what we had to show.*

Remark 4.1. *We sometimes write* $\varphi_1, \ldots, \varphi_n \models \varphi$ *instead of* $\{\varphi_1, \ldots, \varphi_n\} \models \varphi$.

Example 4.11. *We have that* $\mathtt{p}, (\mathtt{p} \vee \mathtt{q}) \not\models \neg\mathtt{q}$, *that is* $\neg\mathtt{q}$ *is not a logical consequence of* $\mathtt{p}, (\mathtt{p} \vee \mathtt{q})$. *To show this "unconsequence", it is sufficient to find a model of* $\mathtt{p}$ *and* $(\mathtt{p} \vee \mathtt{q})$ *that is not a model of* $\mathtt{q}$. *Any assignment* τ *with* $\tau(\mathtt{p}) = 1$ *and* $\tau(\mathtt{q}) = 0$ *will do.*

5 Application 1

John writes the following code:

```
if (((year % 4 == 0) && (year % 100!= 0)) || (year%400 == 0))
   printf("%d is a leap year", year);
else
   printf("%d is not a leap year", year);
```

Jill simplifies the code:

```
if (((year % 4 != 0) || (year % 100 == 0)) && (year%400 != 0))
   printf("%d is not a leap year", year);
else
   printf("%d is a leap year", year);
```

Is Jill right? It is difficult to tell just by looking at the code, but we can use our logic-fu to model the problem above and to determine whether the two programs behave in the same manner.

First of all, we will "translate" the conditions in the if-else statements into propositional logic. We identify the "atomic propositions" and we replace them with propositional variables as follows:

1. the propositional variable `p` will stand for `(year % 4 == 0)`;
2. the propositional variable `q` will stand for `(year % 100 = 0)`;
3. the propositional variable `r` will stand for `(year % 400 == 0)`.

Taking into account the translation key above, we can see that John's condition is, in propositional logic speak, $((\mathtt{p} \wedge \mathtt{q}) \vee \mathtt{r})$.

Jill's formula is, in propositional logic speak, $((\neg\mathtt{p} \vee \neg\mathtt{q}) \wedge \neg\mathtt{r})$.

Also notice that the branches in the two programs are reversed (the if branch of John's program is the else branch of Jill's program and vice-versa). In order for the two programs to have the same behaviour, it is sufficient for the negation of John's formula to be equivalent to Jill's formula. Is this the case? I.e., does the equivalence

$$\neg((\mathtt{p} \wedge \mathtt{q}) \vee \mathtt{r}) \equiv ((\neg\mathtt{p} \vee \neg\mathtt{q}) \wedge \neg\mathtt{r}) \text{ hold?}$$

By applying De Morgan's laws, we can see that the equivalence does hold and therefore the two programs have the same behaviour. So Jill's changes do not break the program – they are correct.

6 Conditionals and Equivalences

There are two more important connectives in propositional logic: the conditional and the equivalence.

We use the notation $(\varphi_1 \to \varphi_2)$ for $(\neg\varphi_1 \vee \varphi_2)$. We read $(\varphi_1 \to \varphi_2)$ as "φ_1 implies φ_2".

Similarly we use $(\varphi_1 \leftrightarrow \varphi_2)$ for $((\varphi_1 \to \varphi_2) \wedge (\varphi_2 \to \varphi_1))$.

Example 6.1. *We have that $(p \to p)$ is valid. Why? The formula $(p \to p)$ is simply a notation for $(\neg p \vee p)$, which is easily seen to be valid.*

You can think of the connectives $\to$ and $\leftrightarrow$ as being like macros in the C language. Whenever you see $(\varphi_1 \to \varphi_2)$ on a sheet of paper, you very quickly unfold it in your mind and read it as $(\neg\varphi_1 \vee \varphi_2)$.