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

1 Reminder

We recall the truth value of a formula in an assignment.

We will work with the following examples of assignments.

Example 1.1. *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 1.2. *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 1.3. *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.

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

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

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 $\neg\neg\mathtt{q}$ 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 1.1. *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 1.1. *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 1.4. *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 $(\neg\mathtt{p} \wedge \mathtt{q})$.

2 Satisfiability

Definition 2.1. *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 2.1. *The formula $(\mathtt{p} \vee \mathtt{q})$ is satisfiable, since it has a model (for example τ_1 above).*

Example 2.2. *The formula $\neg\mathtt{p}$ is also satisfiable: for example, the assignment τ_3 above makes the formula true.*

Example 2.3. *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 $\hat{\tau}\Big((\mathtt{p} \wedge \neg\mathtt{p})\Big) = \hat{\tau}\Big(\mathtt{p}\Big) \cdot \hat{\tau}\Big(\neg\mathtt{p}\Big) = \tau(\mathtt{p}) \cdot \overline{\hat{\tau}\Big(\mathtt{p}\Big)} = \tau(\mathtt{p}) \cdot \overline{\tau(\mathtt{p})}$.
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 2.4. *As we have seen above,* $(\mathtt{p} \wedge \neg\mathtt{p})$ *is a contradiction.*

3 Valid Formulae

Definition 3.1. *A formula* φ *is* valid *if, by definition, any assignment* τ *has the property that* $\hat{\tau}\Big(\varphi\Big) = 1$ *(any assignment is a model of the formula).*

We sometimes write $\models \varphi$ instead of φ is valid.
A valid formula is also called a *tautology*.

Example 3.1. *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 3.2. *The formula* $\mathtt{p}$ *is not valid (because there is an assignment (for example* τ_3*) that makes it false).*

4 Contingent Formulae

Definition 4.1. *A formula that is neither a contradiction nor a tautology is called* contingent.

Example 4.1. $(\mathtt{p} \wedge \neg\mathtt{p})$ *is a contradiction.*
$\mathtt{p}$ *is contingent.*
$(\mathtt{p} \vee \neg\mathtt{p})$ *is a tautology.*

5 Equivalence

Definition 5.1. *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 5.1. *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 5.2. *The following equivalence holds:* $(\mathtt{p} \vee \mathtt{q}) \equiv \neg(\neg\mathtt{p} \wedge \neg\mathtt{q})$ *(check it).*

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

Theorem 5.1. *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. $\neg(\varphi_1 \wedge \varphi_2) = (\neg\varphi_1 \vee \neg\varphi_2)$.

6 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 $\mathtt{p}$ will stand for `(year % 4 == 0)`;
2. the propositional variable $\mathtt{q}$ will stand for `(year % 100 = 0)`;
3. the propositional variable $\mathtt{r}$ 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.

7 Semantical Consequence

Definition 7.1. *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 7.1. *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 $\tau(\mathtt{p}) = 1$.

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

Remark 7.2. *When* $n = 0$*, the notation above allows us to we write* $\models \varphi$ *instead of* $\{\} \models \varphi$*. This is consistent will the notation for validity, since a formula that is a logical consequence of the empty set is valid (and vice-versa).*

Example 7.2. *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.*

8 The relation between implications and semantical consequence

There is a strong link between implications and the concept of semantical consequence, link which is formalized in the following theorem.

Theorem 8.1 (The relation between implications and logical consequences)**.** *For any two formulae* $\varphi_1, \varphi_2 \in PL$*, we have that* $\varphi_1 \models \varphi_2$ *if an only if the formula* $\varphi_1 \to \varphi_2$ *is valid.*

The following more general theorem also holds:

Theorem 8.2 (The generalized relation between implications and logical consequence)**.** *For any formulae* $\varphi_1, \varphi_2, \ldots, \varphi_n, \varphi \in PL$*, we have that* $\varphi_1, \varphi_2, \ldots, \varphi_n \models \varphi$ *if and only if the formula* $(((\varphi_1 \wedge \varphi_2) \wedge \ldots) \wedge \varphi_n) \to \varphi$ *is valid.*

A similar relation exists between the double implication logical connective and the concept of semantical equivalence:

Theorem 8.3 (The relation between double implication and semantic equivalence)**.** *For any two formulae* $\varphi_1, \varphi_2 \in PL$*, we have that* $\varphi_1 \equiv \varphi_2$ *if and only if the formula* $\varphi_1 \leftrightarrow \varphi_2$ *is valid.*

9 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 9.1. *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 $(\neg\varphi_1 \vee \varphi_2)$.

10 Application 2

The following puzzle is from the book *Peter Smith. An Introduction to Formal Logic*. Either the butler or the cook committed the murder. The victim died from poison if the cook did the murder. The butler did the murder only if the victim was stabbed. The victim didn't die from poison.

Does it follow that the victim was stabbed?

We associate to every atomic proposition a propositional variable as follows:

1. For the proposition "the butler commited the murder" the propositional variable $\mathtt{p}$;
2. For the proposition "the cook commited the murder" the propositional variable $\mathtt{q}$;
3. For the proposition "the victim died of posion" the propositional variable $\mathtt{r}_1$;
4. For the proposition "the victim was stabbed" the propositional variable $\mathtt{r}_2$.

The hypotheses of the puzzle are modeled as propositional formulae in propositional logic as follows:

1. $((\mathtt{p} \vee \mathtt{q}) \wedge \neg(\mathtt{p} \wedge \mathtt{q}))$ (exclusive disjunction between $\mathtt{p}$ and $\mathtt{q}$);
2. $(\mathtt{q} \to \mathtt{r}_1)$;
3. $(\mathtt{p} \to \mathtt{r}_2)$ (mind the direction of the implication);
4. $\neg\mathtt{r}_1$.

The question of the puzzle is simply the formula $\mathtt{r}_2$.

To answer the puzzle with yes or no, it is sufficient to check whether

$$\left\{((\mathtt{p} \vee \mathtt{q}) \wedge \neg(\mathtt{p} \wedge \mathtt{q})), (\mathtt{q} \to \mathtt{r}_1), (\mathtt{p} \to \mathtt{r}_2), \neg\mathtt{r}_1\right\} \models \mathtt{r}_2.$$

The logical consequences does hold (left as an exercise to the reader).