Table of Contents

Syntax and Semantics

> You may skip this chapter, but it is suggested to skim it for the notations.

Introduction

Theorems and definitions we prove and define here are actually metatheorems and metadefinitions in our metalanguage. You may assume our metalangauge is first-order or second-order logic in which we define propositional logic which is sometimes called **zeroth-order logic**. Note that we haven't yet shown propositional logic is a subset of our metalanguage. It is just, kind of, nonsense syntax at this moment.

Def. Symbols

Syntactically, we will use the symbols

- $\bot$ called **bottom** or simply **bot** to denote falsity *in our syntactic model*,
- $\top$ called **top** to denote truth *in our syntactic model*,
- $\neg$ called **negation** to denote *negation*,
- $\to$ called **material implication** to denote *implication*,

Semantically, we will use the symbols

- $\mathbb{T}$ called **true** to denote *semantic truth*, and
- $\mathbb{F}$ called **false** to denote *semantic falsity*.

Def. *The* Set of Propositional Formulas

> For simplicity, you may want to consider a formula as a string of symbols such that it is an element of the set $\mathrm{Form}(\mathcal{P})$ given below. There is many ways to define the set of propositional formulas, we'll stick with this one.

Given a (possibly empty) countable set of **atomic formulas** (or **prime formulas**) $\mathcal{P}$, the set $\mathrm{Form}(\mathcal{P})$ of **formulas** of propositional logic is the *smallest set*, defined *inductively*, such that:

1. $\mathcal{P} \subseteq \mathrm{Form}(\mathcal{P})$,
2. If $F \in \mathrm{Form}(\mathcal{P})$, then $\neg F \in \mathrm{Form}(\mathcal{P})$,
3. If $F, G \in \mathrm{Form}(\mathcal{P})$, then $(F \wedge G)\mathrm{Form}(\mathcal{P})$

> This definition of formulas until here suffice as we can define other operators from $\neg$ and $\wedge$. Indeed, we will sometimes use only these when proving theorems (or even defining concepts). In practice, we also further assume:

4. If $F, G \in \mathrm{Form}(\mathcal{P})$, then $(F \vee G) \in \mathrm{Form}(\mathcal{P})$
5. If $F, G \in \mathrm{Form}(\mathcal{P})$, then $(F \to G) \in \mathrm{Form}(\mathcal{P})$

> With just the NAND operator $\bar{\wedge}$ alone, it is possible to build other logical operators, so that (2)-(3) can be reduced to single statement with NAND.

When we refer an object F as a (propositional) **formula**, we simply mean it is an element of $\mathrm{Form}(\mathcal{P})$ for some set of atomic variables $\mathcal{P}$.

Sometimes, we will omit the use of brackets for readibility, so that, for example, $A \wedge B$ should be understood as the formula $(A \wedge B)$.

Def. Syntactic Identity

With $=$, we will denote the **syntactic identity**. That is, if F and G are strings and are formulas, then $F = G$ will simply denote they are strings of symbols of same lenght with same symbols in each place. Notice that this is a *semantic defintion* for our syntactic model.

Def. Operators

Given the logical operators $\neg$ and $\wedge$ and the atomic variable $\bot$, we can *define*, for formulas F and G

- $\top \triangleq \neg\bot$,
- $(F \vee G) \triangleq \neg(\neg F \wedge \neg G)$, and
- $(F \to G) \triangleq (\neg F \vee Q)$

> Notice that we have just defined syntactic abbreviation, didn't prove anything.

Def. Subformula

Let $F \in \mathrm{Form}(\mathcal{P})$ be a formula. Then the set $\mathrm{Sub}(F)$ of **subformulas** of F is the smallest set, such that

- $F \in \mathrm{Sub}(F)$,
- If $F = \neg G$, then $G \in \mathrm{Sub}(F)$.
- If $F = (G \wedge H)$, then $G, H \in \mathrm{Sub}(F)$.

> Here, we have just defined subformula for (1)-(3) above, it should be clear what a subformula is for an extended definition.

The set $\mathrm{Sub}^{+}(F)$ of **proper subformulas** is defined as $\mathrm{Sub}(F) \setminus \{F\}$.

Thm. Principle of Induction on Formulas

Since we are going to use induction heavily, we will prove that if ϕ is any property which

1. Holds for all atomic formulas,
2. If holds for F, it also holds for $\neg F$, and
3. If holds for F and G, it also holds for $(F \wedge G)$,

Then ϕ holds for all formulas (of propositional logic).

▶ **Proof**

Def. Formation Sequence for Propositional Formulas

A finite sequence $(F_0, ..., F_n)$ is called a **formation sequence** for the formula F if $F = F_n$ and for all $i \leq n$, either

1. F_i is atomic formula, or
2. there exists $j, k < i$ such that, either
    1. $F_i = \neg F_j$,
    2. $F_i = (F_j \wedge F_k)$.

Example

For example, for atomics $p_0, p_1, p_2 \in \mathcal{P}$ the sequence

$$(p_0, p_1, (p_1 \wedge p_0), \neg(p_1 \wedge p_0))$$

is a formation sequence for $\neg(p_1 \wedge p_0)$, as is

$$(p_0, p_1, p_0, (p_1 \wedge p_0), (p_0 \wedge p_1), \neg(p_1 \wedge p_0))$$

Thm. Existence of Formation Sequence

Every propositional formula has a formation sequence.

Def. Valuation

Given a set $\mathcal{P}$ of atomic variables, a **valuation** (or **assignment**) v is a function

$$v : \mathcal{P} \to \mathcal{V} = \{\mathbb{T}, \mathbb{F}\}$$

which simply assigns semantic **true** or **false** to atomic variables.

So from now on, depending on context, we may assume $\bot \in \mathcal{P}$ and always define $v(\bot) = \mathbb{F}$, thus $v(\top) = \mathbb{T}$.

> Ponder what would happen if it were the case that $\mathcal{V} = [0, 1]$ where $\mathbb{F} = 0 \in \mathbb{R}$ and $\mathbb{T} = 1$. Obviously, most of our definitions and theorems wouldn't make sense in this setting.

Def. Evaluation

Let F be a formula and v some valuation. We will abuse notation and also use $v(F)$ for the **evaluation** of a formula F, defined naturally.

If $v(F) = \mathbb{T}$, then we say v **models** F denoted by $v \models F$. Equivalently, we say F **holds** under v.

> Notice that this is the first time we have defined the sign $\models$.

Def. Satisfiability

A formula F is said to be **satisfiable** if there exists an valuation v such that $v \models F$. Otherwise, it is called **unsatisfiable**.

Def. Tautology and Contradiction

- A formula F is said to be a **tautology** if it holds under all valuations. This is denoted by $\models F$.
- Similarly, a formula is said to be a **contradiction** if it holds under no valuation. Denoted $\not\models F$
- A formula which is satisfiable but not a tautology is called **contingent**.

Def. Entails

We say a formula F **entails** the formula G denoted by $F \models G$ if every valuation which models F also models G. In this case, we also say G is a **consequence** of F.

> Notice how we are **overloading** (and will keep on overloading) the infix notation $\models$ with valuations, formulas etc. So the elements left or right handside of "models" sign is context-dependent.

Def. Semantic Equivalence

Let F and G be formulas. If they entail each other, then we say they are **equivalent**. This is denoted by $F \equiv G$.

Exercise

$(F \wedge G) \equiv (G \wedge F)$

Exercise (Distributivity Rules)

- $(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$
- $(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H))$

Def. *A* Set of Formulas

From now on, when we say Γ is **a set of formulas**, we will mean $\Gamma \subseteq \mathrm{Form}(\mathcal{P})$. So it is not *the* set of formulas, but rather *a* set of formulas.

> We will make use of the letters such as Γ, Δ to usually denote a set of formulas and uppercase latin letters $A, B, F, G, \ldots$ et cetera to denote formulas.

Moreover, we'll overload the $\models$ notation further in respect to this notion. Let Γ be a set for formulas such that $\Gamma = \{ F_0, F_1, ... \}$, then we say

- a valuation v **models** Γ denoted by $v \models \Gamma$ if $v \models F_i$ for each i,
- Γ **entails** (a formula) G denoted by $\Gamma \models G$ if for every valuation v such that $v \models \Gamma$ implies $v \models G$,
- Γ is **satisfiable** if there exists a valution v such that $v \models F_i$ for each i, and **unsatisfiable** otherwise.

Now, let's look at some basic semantic properties of propositional logic.

Thm. *Semantic* Modus Ponens

Let Γ be a set of formulas and A, B formulas. If $\Gamma \models A$ and $\Gamma \models (A \to B)$, then $\Gamma \models B$.

Thm. ?

Let Γ be a set of formulas. If Δ is satisfiable, then so is every finite subset of it.

Thm. *Semantic* Monotonicity

Let Γ and Δ be a set of formulas such that $\Gamma \subseteq \Delta$ and A a formula. If $\Gamma \models A$ then $\Delta \models A$.

Thm. *Semantic* Transitivity

Let Γ and Δ be a set of formulas and A, B formulas. If $\Gamma \models A$ and $\Delta \cup \{A\} \models B$, then $\Gamma \cup \Delta \models B$.

Thm. *Semantic* Deduction Theorem

Let Γ be a set of formulas and A a formula. Then, $\Gamma \models A$ if and only if $\Gamma \cup \{\neg A\}$ is unsatisfiable.

Derivations

In this section we will define $\vdash$ notation called the *derives* which is pretty similar to $\models$ except *models* was being used semantically whereas $\vdash$ will be used syntatically. More precisely,

We want to formally define a syntactic model such that given a set of assumptions (or axioms; whatever they are) we want to *derive* (reach) further truths (of the model) based on these assumptions. Of course, those truths of our model should also somehow correspond to truths in our metamodel (or metalanguage). It wouldn't really make sense otherwise. So, all it takes is to blindly follow the syntactic rules of our syntactic model to reach our semantic truths. In essence, that is what a **derivation system** is. We *derive* further truths from assumptions (which we usually assume to be the truth) with syntactic wizardry.

With $\models$ we were able to keep track of entailment. That is, given a set of formulas Δ, we said Δ *entails* F written as $\Delta \models F$. It simply meant that for each valuation which holds for the set Δ also holds for F. In a nutshell, here (within our metalanguage) we are just saying Δ implies the formula F (whatever a formula is) since each valuation which is *true* for Δ is also *true* for F. Philosophically, we gave the meaning of being true in a form of boolean truths. We will further explore this concept later on.

In propositional logic, with $\vdash$ we want to do something similar, but syntatically. From a set of formulas Δ we want to be able to arrive *true* formulas F such that they are also (correspondingly) true in our metatheory.

Def. Derives

Let Γ be a set of formulas and F, G, and H formulas. Then we *define* (for our propositional logic model) the **derives** operator $\vdash$ by the (basic) **rules of derivations** which are

Premise	Conclusion	Name
$G \in \Gamma$	$\Gamma \vdash G$	Assumption
$\Gamma \vdash G$ and $\Gamma \subseteq \Delta$	$\Delta \vdash G$	Monotonicity
$\Gamma \vdash G$	$\Gamma \vdash \neg\neg G$	$\neg\neg$-Introduction
$\Gamma \vdash F$ and $\Gamma \vdash G$	$\Gamma \vdash (F \wedge G)$	$\wedge$-Introduction
$\Gamma \vdash (F \wedge G)$	$\Gamma \vdash F$	$\wedge$-Elimination

We can extend these rules of derivations in any way we want (as long as it makes sense which we will define later), for example to include $\vee$ and $\rightarrow$.

Def. Syntactic Proof

A **syntactic proof** (sometimes called **formal proof**, or simply a **proof** if not to be confused with a proof in our metalanguage) in propositional logic is a finite *sequence* of statements of the form $\Gamma \vdash F$ where Γ is a set of formulas and F is a formula.

> Notice how a syntactic proof is our "witness" to the claim that F is derivable (in our model) from Δ by following some syntactic rules.

We say F can be **derived** from Γ if there exists a formal proof with the final step $\Gamma \vdash F$.

Example

Let $\Gamma = \{\, (\neg F \vee G) \,\}$ and derive $F \to G$ from Γ.

▶ **Solution**