Week 3

Notes on the language of proposition logic

Instructor: Matteo de Ceglie

19 March 2020

Today's class will be divided in two part. In this first part I will re-introduce the language of propositional logic that you already know from last week. This time, however, I will go in more formal details, giving you the proper definition of the language and of formulas. This second definition is quite important, since with it you can evaluate if a formula is well-formed (or "grammatical") and, moreover, which one is the main connective of the formula. In the second part we will return to the connectives, and introduce the method of the truth tables.

As you remember from last week notes, we said that the formal language of propositional logic is composed of names $(a, b, c, \dots)$, variables $(x, y, z, \dots)$, properties $(P(a), Q(x), \dots)$ and relations $(R(a, b), T(x, y, a), \dots)$, connectives $(\neg, \wedge, \vee, \rightarrow, \leftrightarrow)$ and parenthesis. With this blocks we can translate from natural language to the symbolic language, for example the following sentence:

If John sits near Mary and she is in love with him, then they are not siblings.

can be translated like this:

$$(S(j, m) \wedge L(m, j)) \rightarrow \neg B(j, m),$$

where $S(x, y)$ means "x sits near y", $L(x, y)$ means "x loves y" and $B(x, y)$ means "x is a sibling of y". However, our goal is to gradually move away from the natural language, and be able to evaluate and manipulate purely symbolic formulas and arguments. To do this, we need more precise rules and definitions for the syntax of propositional logic, to distinguish between formulas with an actual meaning and meaningless strings of symbols (that are not formulas).

The first definition to consider is the definition of the *language*:

Definition 1 (Language of Propositional Logic)**.** The language of propositional logic is composed of:

- a *basic alphabet*;
- an *extended alphabet.*

The basic alphabet of propositional logic is build with the following symbols:

- an infinite number of symbols for *names*: $a, b, c, \dots$;
- an infinite number of symbols for *variables*: $x, y, z, \dots$;
- an infinite number of *unary* (i.e. with just one place in the parenthesis) symbols for *properties* (predicates): $P, R, Q, \dots$;
- an infinite number of symbols with no restriction of arity (i.e. the number of places in the parenthesis) for *relations*: $T, S, M, \dots$;

The extended alphabet of propositional logic is the basic alphabet plus the following symbols:

- the *logical connectives*: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), $\rightarrow$ (implication) and $\leftrightarrow$ (biconditional);
- the *punctuation* symbols: "," (comma), (,) (parenthesis), and "." (point).

Nothing else is part of the language.

This is all is needed to do logic. The basic alphabet gives us symbols to represent the *content* of our discourse, while the extended alphabet the way to connect and organise that content. Obviously this language is not enough: there are no quantifiers and no symbols to do anything special (for example, there is no symbol to express equality!). Quantifiers will be added in first order logic, and special symbols are usually added in extended language (for example, the language of arithmetic is the language of first order logic plus the equality symbol $=$ and symbols for the basic operations).

Also, note that in what follows (these notes and the following classes) we will also use the Greek letters $\alpha, \beta, \gamma, \dots$ as variables for "formulas" (for example, I will write "let ϕ and ψ be formulas). However, these are *not* part of the language, but are only a shortcut in our "meta-language" that we use.

Now that we have our basic language we can define what a *formula* is. To do this, we will give an *inductive* definition: this kind of definition starts with the most basic case and build all the other cases from the previous ones. First of all we will defined the set of *atomic* formulas. These are the most simple formulas we can produce.

Definition 2 (Atomic formulas)**.** The set of all *atomic formulas* of propositional logic is formed by the following:

1. a name (a) is an atomic formula;
2. a variable (x) is an atomic formula;
3. a property with a name or a variable $(P(a), P(x))$ is an atomic formula;
4. a relation with names or variables $(R(a, b), Q(a, x, y))$ is an atomic formula;
5. nothing else is an atomic formula.

For example, $Q(x)$ is an atomic formula, while Q alone or $Q(R)$ are not. From these atomic formulas we can define the set of *well-formed* formulas, i.e. all the formulas in propositional logic that are "grammatical" and actually mean something. This one too is an inductive definition.

Definition 3 (Well-formed formulas)**.** The set of all *well-formed* formulas of propositional logic is formed by the following:

1. All atomic formulas are well-formed formulas;
2. If ϕ is a well-formed formula, then $\neg\phi$ is a well-formed formula;
3. If ϕ and ψ are well-formed formulas, then $\phi \wedge \psi$ is a well-formed formula;
4. If ϕ and ψ are well-formed formulas, then $\phi \vee \psi$ is a well-formed formula;
5. If ϕ and ψ are well-formed formulas, then $\phi \to \psi$ is a well formed formula;
6. If ϕ and ψ are well-formed formulas, then $\phi \leftrightarrow \psi$ is a well formed formula;
7. Nothing else is a well formed formula.

This completes our definition of the language and the syntax of propositional logic.

What we can do with all of this? We can now check if a string of symbols is actually a well-formed formula of propositional logic or not. For example. the following

$$(P(x) \wedge R(a, b)) \to (a \vee (F(x) \leftrightarrow \neg y))$$

is a well-formed formula, while

$$(P(x) \wedge R(Q, b))\neg \to (a \vee \vee(F(x) \leftrightarrow \neg y))$$

it is not.