Propositional Logic Review

Alan Fern, afern@eecs.oregonstate.edu

February 26, 2020

1 Propositional Logic Review

I will assume that you understand the basic concepts of propositional logic. However, we will begin with a quick review. If you have not studied propositional logic previously, I strongly recommend that you read Chapter 7 of *Artificial Intelligence: A Modern Approach* by Russell & Norvig.

There are two key components involved in defining any logic: syntax and semantics. The syntax specifies what strings constitute legal statements in the logic and the semantics provide a way for assigning a meaning to the legal strings. Whenever you are introduced to a new logic it is important to first gain a clear understanding of its syntax and semantics. We will go through these formalities for propositional logic as a simple illustrative example.

1.1 Syntax

The syntax of propositional logic is defined relative to a finite set of symbols $X = \{X_1, \ldots, X_n\}$. We will call these symbols propositions. A well formed formula (wff) of propositional logic is any string that can be constructed according to the following recursive rules:

- a single propositional symbol is a wff (e.g. X_1),
- if ϕ_1 and ϕ_2 are wffs then so is any logical combination (involving $\wedge$, $\vee$, or $\Rightarrow$) of ϕ_1 and ϕ_2. (e.g. $X_1 \Rightarrow X_3$),
- if ϕ is a wff then so is $\neg\phi$, and
- if ϕ is wff then so is (ϕ)

No other strings besides the wffs are legal strings of propositional logic.

1.2 Semantics

In any logic, the semantics is defined by specifying the *models* of the logic and how to assign meaning to the legal strings with respect to those models. Intuitively, each possible model can be viewed as specifying a possible world within which a string (e.g. a wff) can be evaluated in. The evaluation of a string in a model is often referred to as the *interpretation* of the string with respect to the model. Thus, when learning about the semantics of a logic the key questions to ask are: 1) What are the models?, and 2) How are strings interpreted in those models?

1.2.1 What are the models?

A model in propositional logic with respect to a set of propositions $X = \{X_1, \ldots, X_n\}$ is simply a truth assignments to the propositions in X. For example, if our set of propositions is $\{P, Q\}$, then a model might be $\langle P = \text{true}, Q = \text{true}\rangle$. Intuitively, if we think of the propositions as representing boolean properties of worlds, then each model specifies the values of those properties in that particular world.

1.2.2 Interpretations

Given a model M and a well formed formula ϕ, the interpretation (or meaning) of ϕ with respect to M is either **true** or **false**, depending on whether ϕ evaluates to true or false under the truth assignment dictated by M. One common notation used to denote the interpretation of ϕ under M is ϕ^M (but other notational forms exist).

As an example, if we start with a well formed formula

$$\phi = (\text{BobIsRich} \Rightarrow \text{BobHasNiceCar})$$

and a model for the propositions in those formulas

$$M_1 = \langle \text{BobIsRich} = \textbf{true}, \text{BobHasNiceCar} = \textbf{true}\rangle$$

then

$$\phi^{M_1} = \textbf{true}\ .$$

However, under the model

$$M_2 = \langle \text{BobIsRich} = \textbf{true}, \text{BobHasNiceCar} = \textbf{false}\rangle$$

we have that

$$\phi^{M_2} = \textbf{false}\ .$$

When it is the case that $\phi^M = \textbf{true}$ then we often say that ϕ is satisfied by M. If a formula is satisfied by all models then we say that a formula is a *valid* formula or a tautology. For example, $X_1 \vee \neg X_1$ is valid. If no model satisfies a formula then we say that it is *unsatisfiable*. There are formulas that are neither valid or unsatisfiable, i.e. they are true in some models and false in others as in the above example.

1.3 Entailment

One reason we use logic is to determine when one formula necessarily follows from another, i.e. when one formula *entails* another. In particular, if we think of a set of formulas as encoding our knowledge of a world or system, then it is of interest to determine which other formulas (or statements) can be soundly inferred from that knowledge. For example, in a logic that can encode number theory, we might be interested in whether Fermat's last theorem follows from formulas that "define" numbers.

Entailment : Given two formulas KB (for knowledge base) and ϕ we say that KB entails ϕ denoted as $KB \models \phi$ iff for all models M, if $KB^M = \textbf{true}$ then $\phi^M = \textbf{true}$.

That is, a formula ϕ is entailed by KB if it is true in all of the worlds or models in which KB is true. Thus, if KB entails ϕ, then – as long as we are willing to believe that KB is true in "our world of interest" – we should also believe that ϕ is true in that world. Here our world of interest might be the wumpus world from your textbook, the world of natural numbers, a description of a computer system, etc. Note that the notion of entailment is not tied in any way to a particular logic. As long as we understand the models of a logic and how to interpret formulas as true or false, the above definition of entailment is well defined.

As a simple example in propositional logic we have the following entailment relationship

$$P \wedge (P \Rightarrow Q) \models Q$$

. That is, Q must be true in any world where both P is true and "P implies Q" is true.

1.4 Deductive Inference

Remember that entailment is strictly a semantic concept. In AI and computer science we are interested in computing when an entailment relationship is true and also computing the entailed formulas of a given KB. Such computation is often called **deductive inference**.

A common approach to deductive inference is to combine a set of logical inference rules with a search procedure. An example of a logical inference rule is

$$\frac{P, P \Rightarrow Q}{Q}$$

which means that if our current KB contains the formulas P and $P \Rightarrow Q$ we can add the formula Q to the KB. We require that inference rules be **sound**; in other words, they should only add formulas that are entailed by the KB. Given a set of inference rules, we can cast deductive inference as a search problem, where we start with KB and search through sequences of rule applications until either deriving the target formula (i.e. the query) or some stopping condition is met (e.g. total running time).

When such a procedure discovers a sequence of rule applications that derive the formula ϕ from KB we say that the procedure has found a proof of ϕ from KB. We often use the notation

$$KB \vdash \phi$$

to denote that our procedure can prove ϕ from KB.

Note that inference and proof are purely syntactic concepts. Deductive inference procedures merely apply syntactic transformations to strings without regard to the meaning of the strings. Of course, we *do* want there to be a link between the syntactic notion of proof and the semantic notion of entailment. This link is formalized via the notions of soundness and completeness.

Soundness An inference algorithm is sound iff whenever $KB \vdash \phi$ then $KB \models \phi$.
That is, the algorithm only proves semantically entailed formulas.

Completeness An inference algorithm is **complete** iff whenever $KB \models \phi$ then $KB \vdash \phi$.
That is, the algorithm is able to prove all entailed formulas of a knowledge base.

So long as our inference rules are sound, the above search based inference approach will be sound. Completeness is a more difficult issue. In order to achieve completeness we must use a complete

search procedure, and in addition ensure that our inference rules are rich enough to prove any entailed sentence — i.e. whenever $KB \models \phi$ there should be a sequence of rules starting with KB that result in ϕ.

For propositional logic there are sound and complete sets of inference rules. In practice, and in this course, however, we will be content with a restricted notion of completeness, that allows us to get by with simpler rule sets. This is known as refutation completeness, as we now discuss.

1.5 Refutation Completeness

It is easy to show that whenever $KB \models \phi$ that the formula $KB \wedge \neg\phi$ is unsatisfiable. That is, by the definition of entailment there are no models that satisfy KB and the negation of ϕ. Thus, we can reduce the problem of testing entailment to testing unsatisfiability. An inference procedure is said to be refutation complete if it can decide the unsatisfiability of any given formula. Given a refutation complete inference procedure, it is possible to test any entailment query; however, such procedures do not necessarily need to be able to enumerate all consequences of KB, as is required by the more strict requirement of completeness.

For propositional logic there are many algorithms for deciding (un)satisfiability, including brute-force truth-table enumerations. Any such procedure provides us with a refutation complete inference procedure for propositional logic.

Of particular importance is the resolution inference procedure for testing satisfiability. As described in your text the generic resolution inference procedure combines a single inference rule (called resolution) with a complete search. This procedure is guaranteed to discover a contradiction, i.e. discover a sequence of resolution rule applications that "prove false", whenever the original formula is unsatisfiable.

In upcoming lectures we will generalize the resolution inference procedure to handle first-order logic. Thus, it would be a good idea to review resolution for propositional logic as covered in Chapter 7 of your book.

1.6 Complexity

For unrestricted propositional logic, deductive inference is NP-complete. That is to say that it is widely believed that there is no efficient algorithm. In order to obtain polynomial time inference we must sacrifice completeness. One way to do this is by restricting attention to a subclass of propositional formulas (e.g. Horn theories as will be discussed in upcoming lectures). Managing the trade-off between completeness and efficiency is one of the fundamental dilemmas when applying logical reasoning.