CS 245 - Logic and Computation **Fall 2016**

Lecture 2: September 13, 2016

Lecturer: Kevin Lanctot *Notes By: Harsh Mistry*

2.1 Propositional Logic

What is a proposition?

A ***proposition*** is a declarative sentence that is either true or false.

2.1.1 English to Propositional Logic

- $\neg p$: Not P
- $p \wedge q$: P and Q
- $p \vee q$: P or Q
- $p \implies q$: P then Q
- $p \iff q$: P if and only if Q

Examples

1. She is clever and hard working : $P \wedge Q$
2. He is clever but not hardworking : $P \wedge Q$
3. If he does not study then he will fail : $(\neg S) \implies F$
4. He must study hard; otherwise he will fail : $(\neg S) \implies F$
5. He will fail unless he studies hard : $F \vee S$
6. He will not fail only if he studies hard : $(\neg F) \implies S$

Advanced Examples

1. If it rains. he will be at home; otherwise he will go to the market or to school.
   $(R \implies) \wedge ((\neg R) \implies (M \vee S))$
2. If the sum of two numbers is even if an only if both numbers are even or both numbers are odd.
   $S \iff (E \vee O)$

Note : Some sentences are not propositions, as not all sentences evaluate to true or false.

2.1.2 Aspects of Logic

Propositional Logic is a form of **symbolic** logic. By extension symbolic logic is formalized by the following.

- **Syntax :** The statements we consider.
- **Semantics :** The meaning of the statement.
- **Proof Procedures :** Can we prove the given statement?

2.1.3 Syntax

In propositional logic, simple **atomic propositions** are the basic building blocks. These atomic propositions can be connected to form **compound propositions**.

Questions to consider

- Does a given sequence of propositions form a valid argument?
- Can all propositions in a given set be true simultaneously?

Propositions are represented by formulas. A formula consists of a sequence of symbols. The three kinds of symbols are :

- Propositional Variables : **p, q, r**
- Connectives : $\urcorner$, $\wedge, \vee,$ $\Longrightarrow$, $\Longleftrightarrow$
- Punctuation : **'(' and ')'**

2.1.3.1 Expressions

Meta-Symbols

We often use a letter that is not formally a symbol in order to namean expression. For example, we might denote a expression as α This is an example of a **meta-symbol**. It is **NOT** a symbol!

- Two expression $\alpha and \beta$ are equal if and only if they are the are same length
- We write $\alpha\beta$ to mean the concatenation of two expressions.

Definition 2.1 *Concatenation : If α is an expression of length i and β is an expression of length j then $\alpha\beta$ is an expression of length $i + j$. We have*

$$\text{The } kth \text{ symbol of } \alpha\beta \text{ is } \begin{cases} \text{the } kth \text{ symbol of } \alpha \textbf{ if } k \leq 1 \\ \text{the } (k-i)^{th} \text{ symbol of } \beta \textbf{ if } k > i \end{cases}$$

2.1.3.2 Well-formed formula

Let P be a set of propositional variables. We define the set of well-formed formulas over p inductively as follows.

1. A expression consisting of a single symbol of P is a well-formed formula
2. If α is a well-formed formula, then $(\neg\alpha)$ is a well formed formula
3. If α and β are well formed then, $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \implies \beta),$ and $(\alpha \iff \beta)$ are well-formed
4. Nothing else is well-formed

2.1.3.3 Kinds of Formulas

- A propositional variable is called an atom
- $(\neg\alpha)$: Negation
- $(\alpha \wedge \beta)$: Conjunction
- $(\alpha \vee \beta)$: Disjunction
- $(\alpha \implies \beta)$: Implication
- $(\alpha \iff \beta)$: Equivalence

2.2 Semantics of Propositional Logic

The semantics of logic describes how to interpret the well-formed formulas of the logic. Since semantics of propositional logic is compositional, the meaning of the whole formula derives from the meaning of its parts.

2.2.1 Valuations

Definition 2.2 *A* ***truth valuation*** *is a function with the set of all proposition symbols as domain and F, T as range. Basically, a truth valuation assigns a value to every propositional variable.*

2.2.2 Semantics of Connectives

A connective represents a function from truth values to truth values. The two types of connectives are : **Unary** and **binary**.

- Unary connectives map one value to one value.
- Binary connectives map two values to one value.