Lecture 1: The Formulas of Propositional Logic

1 Logic in Computer Science

Historically logic developed within the domains of mathematics and philosophy, but in this class we will concentrate on the application of logic in computer science. Throughout the course we will make references to areas that use logic heavily. These include **Circuit Design** (also known as Digital Logic), **Computability** (a mathematical investigation of what can and cannot be computed) and **Complexity Theory** (a refinement of computability–what is feasibly computable). In addition, logic is also used in Database Theory, Design Verification, Security, Programming Languages and Semantics, AI Reasoning, and Common Sense reasoning, software engineering (providing specifications to programmers), and more.

2 Basics of Propositional Logic

One of the key ideas of mathematics is that of an abstraction. Numbers allow us to abstract away the "what" of a problem and concentrate on the arithmetic. Similarly, in logic we abstract away some of the details and concentrate on the underlying form. Two millennia ago Aristotle realized that truth follows from form. For example, consider the following syllogism:

All *Greeks are humans.*
All *humans are mortal.*
Therefore all *Greeks are mortal.*

While all sentences in this example have a clear meaning, this need not be so. The following argument is also valid, although meaningless:

All *Toves are Bojums.*
All *Bojums are slithy.*
Therefore all *Toves are slithy.*

Both examples share the same form. If we replace "Greeks" by p, "humans" by q, and "mortal" by r, we expose the form of the syllogism:

If p ***then*** q.
If q ***then*** r.
Therefore If p ***then*** r.

This syllogism has a name: transitivity of implication. An implication is shown in the statement ***If*** p ***then*** q, which means p ***implies*** q.

p and q above are called symbolic variables. A **symbolic variable** reflects some underlying quantity which is, at this point, unknown. The possible values of these variables are the truth values. Symbolic variables obey the **The Law of the Excluded Middle** which says that variables are either true or false, and there is nothing in between. This is a very sharp world. Consider the sand heap paradox. Start with a heap of sand. Take the sand away one grain at a time. Initially, the heap will seem to remain a heap, although we are moving in induction steps of one grain at a time. This is known as *false dichotomy*, where induction seems to be failing, as after enough steps, eventually the heap of sand will stop being a heap of sand, but we will not be able to tell precisely when. This world true and false is idealized. We abstract away the gray cases.

This abstraction corresponds to the digital abstraction in electrical engineering. In electrical engineering voltages above some threshold indicate 1, and below another threshold indicate 0. One might say computer science begins when you have a circuit you can pretend is digital.

These symbolic variables are propositions. A symbolic proposition is a property that can be either true or false. They are assertions of truth about the world. In the first part of the course we will regard propositions as indivisible units without intrinsic structure. This paradigm is called **propositional logic**, also known as **sentential logic**.

In order to create formulas we use **propositional connectives**. The connectives can be classified according to the number of propositions that they apply to:

- *Unary:* **not** $\neg$
- *Binary:* **and** $\wedge$, **or** $\vee$, **xor** $\oplus$, **implies** $\rightarrow$, **equivalent** $\leftrightarrow$
- *Ternary:* **if-then-else**

As a shorthand, sometimes we will use $\circ$ to indicate any binary connective.

The set of atomic propositions is called Prop. Prop, the set of connectives and the left "(" and right parentheses ")" form the alphabet of propositional logic. Using this alphabet we now can write strings of alphabetic symbols, or **expressions**. For example:

$(p \wedge q \rightarrow r)$

$)q \rightarrow (r \wedge$

The first expression is ambiguous and this ambiguity can be resolved by using more parentheses. Intuitively, there is something wrong with the second expression, but it still fits the definition of an expression. We will use the term **formulas** to indicate expressions with the "right" form. This intuitive notion can be formalized as following:

Definition 1 *The set of formulas* FORM *is the smallest set with the following properties:*

1. PROP $\subseteq$ FORM
2. *If* $\varphi \in$ FORM*, then* $(\neg\varphi) \in$ FORM
   If φ *and* $\psi \in$ FORM*, then* $(\varphi \circ \psi) \in$ FORM*.*

The second property in the definition is called the **closure** property. Also note that in some texts the formulas are called **well-formed formulas** or **wffs**.

In order to be sure that FORM is well defined, we need to show that there exists a unique smallest set that has the above properties. This task is the subject of the next lemma.

Lemma 1 (Smallest set) *There is a unique smallest set of expressions that contains* PROP *and satisfies* ***closure****.*

Proof: Let FORM$' = \bigcap \{ P : \text{PROP} \subseteq P$ and P satisfies closure $\}$.
Claim: FORM$'$ is the smallest set with these two properties (closure and containing PROP).

1. PROP $\subseteq$ FORM$'$: PROP is a subset of FORM$'$ because FORM$'$ is the intersection of sets that contain PROP.

   FORM$'$ satisfies closure: Suppose that φ and $\psi \in$ FORM$'$. From the way we constructed FORM$'$ follows that φ and ψ are in all expression sets P that satisfy closure and contain PROP. In particular, closure implies that $(\neg\varphi)$ and $(\varphi \circ \psi)$ must also be in all these sets P, and consequently, also in their intersection. This means that $(\neg\varphi)$ and $(\varphi \circ \psi)$ are in FORM$'$, therefore FORM$'$ satisfies closure.

2. FORM$'$ is the smallest set with these properties: To show this, we need to show that FORM$'$ is a subset of every set that has the two properties. To that end, suppose that P is a set that contains PROP and satisfies closure. We know, FORM$' = P_1 \cap P_2 \cap P_3 ...$ Now, if $X = Y \cap Z$, then it implies that, $X \subseteq Y$. From the way we constructed FORM$'$, the intersection of two or more sets is always a subset of them all, including the smallest P. Since, the intersection is FORM$'$, therefore FORM$' \subseteq P$.

$\Diamond\Diamond\Diamond$

Thus we have shown that there is a smallest set (namely FORM$'$) that contains PROP and satisfies closure. Therefore FORM $=$ FORM$'$.

The previous definition of **Form** is non-constructive, which can make proofs about **Form** inconvenient. Alternatively, we can define **Form** inductively.

3 Defining Form by induction

Definition 2 Form'*:*

1. $Form_0 = Prop$
2. $Form_{i+1} = Form_i \cup \{(\neg\varphi)\text{: } \varphi \in Form_i\} \cup \{(\varphi \circ \psi)\text{: } \varphi, \psi \in Form_i\}$.
3. $Form' = \bigcup_{i=0}^{\infty} Form_i$

Form will refer to the first definition and Form' will refer to the inductive definition. This inductive definition "builds from below." The question now is this: Are both of these definitions equivalent? The answer is yes, as is shown by the next lemma.

Lemma 2 *Form = Form'*

Proof To prove equivalence is to prove containment in both directions. Therefore we can prove that Form = Form', if we can show that Form $\subseteq$ Form' and Form' $\subseteq$ Form.

1. $Form \subseteq Form'$. We will show that $Form'$ satisfies the two key properties: contains PROP, and satisfies closure.

   - $Prop \subseteq Form_0, Form_i \subseteq Form_{i+1} \Rightarrow Prop \subseteq Form_i$
     Therefore $Prop \subseteq Form'$.
   - Let $\varphi, \psi \in Form'$. From the way we build $Form'$ it follows that there must exist i and j, such that $\varphi \in Form_i$ and $\psi \in Form_j$. Let $k = max(i, j)$. Clearly $\varphi, \psi \in Form_k$.
     $(\neg\varphi) \in Form_{k+1} \Rightarrow (\neg\varphi) \in Form'$ and
     $(\varphi \circ \psi) \in Form_{k+1} \Rightarrow (\varphi \circ \psi) \in Form'$

     We have shown that $Form'$ satisfies closure and contains PROP. Since $Form$ is a subset of every set with these two properties, it follows that $Form \subseteq Form'$.

2. $Form' \subseteq Form$.
   We need to show that if P contains PROP and satisfies closure, then $Form' \subseteq P$. We do this by using induction on i.

   Base: $Form_0 = Prop \subseteq P$

   Induction:
   $Form_i \subseteq P$
   $Form_{i+1} = Form_i \cup \{(\neg\varphi) : \varphi \in Form_i\}$
   $\cup \{(\varphi \circ \psi) : \varphi, \psi \in Form_i\}$
   By closure: $Form_{i+1} \subseteq P$

∎

In mathematical writing, there is a tricky balance between rigor and readability. You can imagine a proof that is precise to every detail and can be checked by a machine. But if it looked too arcane to the human eye, it would be useless in easily conveying the information to the reader. (Think about trying to read assembly versus reading Java or OCAML). Of course, if a proof lacks the necessary details it can't be used for anything. This balance if something you should always be thinking about when you write proofs. The previous proof is a good example of how to prove simple things rigorously.