CM20019 – Complementary Course Notes

Sheet 2

October 13, 2006

1 Propositional Logic

A language is a set of formulae. Let's define the language of propositional logic.

The alphabet $\mathcal{A}$ underlying the language of propositional logic consists of:

- a finite or countably infinite set of nullary *relation symbols* of the form $p, q, \ldots$
- the set of *connectives* $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$,
- the *punctuation symbols* '(' and ')'.

Sometimes the relation symbols are also called predicate symbols, propositional variables or atoms.

$\mathcal{A}_R$ denotes the set of relation symbols occurring in $\mathcal{A}$. No meaning is attached to the symbols of an alphabet: they are just symbols.

1.1 Definition. The set of *propositional logic formulas* (also called propositional sentences) is the minimal set satisfying the following conditions:

- Each propositional variable is a formula.
- If F and G are formulas, then $\neg F$, $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$, and $(F \leftrightarrow G)$ are formulas.

Formulas of the form p or $\neg p$ are often called *literals*. Furthermore, formulas are literals denoted by $F, G, \ldots$ and parenthesis are omitted whenever possible using the following precedence hierarchy to order the connectives:

$$\neg > \wedge > \vee > \{\leftarrow, \rightarrow\} > \leftrightarrow .$$

Using this precedence hierarchy a formula like $((p \rightarrow q) \leftrightarrow (\neg p \vee q))$ can simply be written as $p \rightarrow q \leftrightarrow \neg p \vee q$.

The language of propositional logic formulae is a language of structured objects. Please remember that two important principles apply on structured objects, for two different tasks: the *principle of structural induction* and *recursion*. The former is applied when you want to show that a set of structured objects satisfies a certain property. The latter is applied to define a certain function on a structured object. They take the following forms, for example in the context of (the structured definition of) *propositional logic formulae*:

Principle of Structural Induction:
Every *formula of propositional logic* has a certain property E provided that:

- Basis step: Every propositional variable has property E.
- Induction steps:
  If F is a propositional logic formula and it has property E so does $\neg F$.
  If F and G are propositional logic formulae and they have property E so does $(F \circ G)$, where $\circ \in \{\wedge, \vee, \leftarrow, \rightarrow, \leftrightarrow\}$.

Principle of Structural Recursion:
There is one and only one function h defined on the set of *propositional logic formulae*, such that:

- Basis step: The value of h is specified explicitly on propositional variables.
- Induction steps:
  The value of h on $\neg F$ is specified in terms of the value of h on F.
  The value of h on $(F \circ G)$ is specified in terms of the values of h on F and on G, where $\circ \in \{\wedge, \vee, \leftarrow, \rightarrow, \leftrightarrow\}$.

1.2 Definition. Let H be a formula. The set of subformulas $\mathcal{T}(H)$ is the smallest set satisfying the following conditions:

- $H \in \mathcal{T}(H)$,
- If $\neg F \in \mathcal{T}(H)$, then $F \in \mathcal{T}(H)$,
- If $(F \circ G) \in \mathcal{T}(H)$, then $F, G \in \mathcal{T}(H)$, where $\circ \in \{\wedge, \vee, \leftarrow, \rightarrow, \leftrightarrow\}$.

So far we have defined a formal language (or a syntax), but no meaning has been assigned to formulae. The *semantics* of propositional classical logic is given by assigning the truth values t and f to the relation symbols in the formula, and depending on the meaning assigned to the binding connectives. Because the truth values are two, we speak of a two-valued logic.

An *interpretation* I can conveniently be represented by a subset of $\mathcal{A}_{\mathcal{R}}$ with the understanding that p is mapped to t under I iff p occurs in I. In the following, we will take the liberty to identify I with such a subset.

1.3 Definition. The relation $I \models F$ (I models F) between interpretations and formulas is inductively defined as follows:
$I \models p$ iff $I(p) = t$ (iff $p \in I$),
$I \models \neg F$ iff $I \not\models F$,
$I \models F_1 \wedge F_2$ iff $I \models F_1$ and $I \models F_2$,
$I \models F_1 \vee F_2$ iff $I \models F_1$ or $I \models F_2$,
$I \models F_1 \rightarrow F_2$ iff $I \not\models F_1$ or $I \models F_2$,
$I \models F_1 \leftrightarrow F_2$ iff $I \models F_1 \rightarrow F_2$ and $I \models F_2 \rightarrow F_1$.

Observe that $I \models \neg p$ iff $I(p) = f$.

In other words, the connectives are interpreted as functions from the set of truth values into the set of truth values. $\neg$ is a unary function such that $\neg(t) = f$ and $\neg(f) = t$. The other (binary) connectives are interpreted as follows:

F	G	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$
t	t	t	t	t	t
t	f	f	t	f	f
f	t	f	t	t	f
f	f	f	f	t	t

1.4 Definition. An interpretation I for F is said to be a *model* for F iff $I \models F$. Let $\mathcal{F}$ be a set of formulas; I is a model for $\mathcal{F}$ iff I is a model for each $G \in \mathcal{F}$. In other words, sets of formulas are interpreted as conjunctions of formulas. One should observe that in case $\mathcal{F} = \emptyset$, i.e., if a conjunction of formulas is empty, each interpretation I is a model for $\mathcal{F}$.

1.5 Definition. Given a formula F, it is said to be:

- *valid* or a *tautology* iff for all interpretations I we find that $I \models F$,
- *satisfiable* iff there exists an interpretation I such that $I \models F$,
- *falsifiable* iff there exists an interpretation I such that $I \not\models F$,
- *unsatisfiable* iff for all interpretations I we find that $I \not\models F$.

1.6 Example. Consider the formula $F = (p_1 \vee p_2) \wedge (q_1 \vee q_2)$, and the interpretations $I_1 = \{p_1, q_1\}$ and $I_2 = \{q_1, q_2\}$. Then $I_1 \models F$ and $I_2 \not\models F$. I_1 is a model for F. F is satisfiable in I_1 and falsifiable in I_2. The formula $p \vee \neg p$ is valid, whereas the formula $p \wedge \neg p$ is unsatisfiable.

1.7 Definition. A set of formulas $\mathcal{F}$ *logically entails* G (or G is a *logical consequence* of $\mathcal{F}$ or G is a *theorem* of $\mathcal{F}$), in symbols $\mathcal{F} \models G$, iff each model for $\mathcal{F}$ is also a model for G. Finally, a satisfiable set of formulas together with all its theorems is said to be a *theory*.

1.8 Definition. Two formulas F and G are said to be (semantically) equivalent, in symbols $F \equiv G$, if for all interpretations I we find that $I \models F$ iff $I \models G$ holds. This can be rephrased by saying that the formula $F \leftrightarrow G$ is a tautology.

How can we determine that a formula G follows logically from a given set $\mathcal{F}$ of formulas? A very simple method is *truth tabling*, where systematically all possible combinations of truth values for the various relation symbols occurring in $\mathcal{F}$ are generated. The logical consequence relation can then easily be checked by verifying that each model for $\mathcal{F}$ is also a model for G. One should observe that if $\mathcal{F}$ is finite, then truth tabling can be performed in finite time and, hence, checking logical entailment is decidable in propositional logic. Unfortunately, the number of combinations of truth values grows exponentially with the number of relation symbols.

Yet, we can do better than truth tabling in most cases, thanks to the following theorems: the notion of logical consequence is ultimately linked to the notion of validity.

1.1 Theorem. *(Deduction Theorem)* $\mathcal{F} \cup \{G\} \models H$ *iff* $\mathcal{F} \models G \rightarrow H$.

(Prove it as exercise)

Moreover we have the following equivalence laws (semantical equivalence):

$$\begin{array}{rcl}
(F \wedge F) & \equiv & F \\
(F \vee F) & \equiv & F \quad (idempotency) \\
 & & \\
(F \wedge G) & \equiv & (G \wedge F) \\
(F \vee G) & \equiv & (G \vee F) \quad (commutativity) \\
 & & \\
((F \wedge G) \wedge H) & \equiv & (F \wedge (G \wedge H)) \\
((F \vee G) \vee H) & \equiv & (F \vee (G \vee H)) \quad (associativity) \\
 & & \\
((F \wedge G) \vee F) & \equiv & F \\
((F \vee G) \wedge F) & \equiv & F \quad (absorption) \\
 & & \\
(F \wedge (G \vee H)) & \equiv & ((F \wedge G) \vee (F \wedge H)) \\
(F \vee (G \wedge H)) & \equiv & ((F \vee G) \wedge (F \vee H)) \quad (distributivity) \\
 & & \\
\neg\neg F & \equiv & F \quad (double\ negation) \\
 & & \\
\neg(F \wedge G) & \equiv & (\neg F \vee \neg G) \\
\neg(F \vee G) & \equiv & (\neg F \wedge \neg G) \quad (deMorgan) \\
 & & \\
(F \vee G) & \equiv & F, \text{ if } F \text{ tautology} \\
(F \wedge G) & \equiv & G, \text{ if } F \text{ tautology} \quad (tautology) \\
 & & \\
(F \vee G) & \equiv & G, \text{ if } F \text{ unsatisfiable} \\
(F \wedge G) & \equiv & F, \text{ if } F \text{ unsatisfiable} \quad (unsatisfiability) \\
 & & \\
(F \leftrightarrow G) & \equiv & (F \rightarrow G) \wedge (G \rightarrow F) \quad (equivalence) \\
 & & \\
(F \rightarrow G) & \equiv & (\neg F \vee G) \quad (implication)
\end{array}$$

Repeatedly applying the deduction theorem and suitable equivalence laws above leads to the observation that

$$\{F_1, \ldots, F_n\} \models F \text{ iff } \emptyset \models F_1 \wedge \cdots \wedge F_n \rightarrow F.$$

(In the latter case, we often omit $\emptyset$ and simply write $\models F_1 \wedge \cdots \wedge F_n \rightarrow F$). Observe that $\emptyset$ is true under all interpretations. Hence, in order to show that F is a logical consequence of $F_1, \ldots, F_n$ we may show that $F_1 \wedge \cdots \wedge F_n \rightarrow F$ is valid.

There is also a link between validity and unsatisfiability.

1.2 Theorem. *F is valid iff $\neg F$ is unsatisfiable.*

(Proof left as exercise).

This tells us that in order to show the validity of $F_1 \wedge \cdots \wedge F_n \rightarrow F$ we may show that $\neg(F_1 \wedge \cdots \wedge F_n \rightarrow F)$ or, equivalently, $(F_1 \wedge \cdots \wedge F_n \wedge \neg F)$ is unsatisfiable.

Some calculi (resolution, semantic tableaux, for example) can be used to determine the unsatisfiability of a given formula. Many calculi do not operate

on general formulas but on restricted formulas only. Such restricted formulas are often called *normal forms*.

Intuitively, it should be possible to replace a subformula G of F by an equivalent one without changing the meaning of F.

By $F[G]$ we denote a formula F, in which the occurrences of the formula G play an important role; G may occur several times in F or may not occur at all. By $F[G/H]$ we denote the formula that has been obtained from F by replacing all occurrences of G by H. The following holds:

1.3 Theorem. *If the formulae G and H are semantically equivalent, so are $F[G]$ and $F[G/H]$.*

Various normal forms can be defined.

- A formula is in *negation normal form* iff it is built solely by literals, conjunctions and disjunctions.
- A formula is in *conjunctive normal form* iff it has the form $F_1 \wedge \cdots \wedge F_n$, for $n \geq 0$, where each of F_i, for $1 \leq i \leq n$ is a disjunction of literals.
- A formula is in *disjunctive normal form* iff it has the form $F_1 \vee \cdots \vee F_n$, for $n \geq 0$, where each of F_i is a conjunction of literals.

Moreover the set notation of formulae in conjunctive normal form is often called *clause form*: for example, given the conjunctive normal form

$$(L_{11} \vee \cdots \vee L_{1n_1}) \wedge \cdots \wedge (L_{m1} \vee \cdots \vee L_{mn_m}),$$

its clausal form is

$$\{\{L_{11}, \ldots, L_{1n_1}\}, \ldots \{L_{m1}, \ldots, L_{mn_m}\}\}.$$

The following algorithm produces conjunctive normal forms (there are many algorithms):
Input: A propositional logic formula F .
Output: A propositional logic formula G in conjunctive normal form which is equivalent to F.

1. Eliminate all equivalence signs using the equivalence law.
2. Eliminate all implication signs using the implication law.
3. Eliminate all negation signs except those in front of atoms using the de Morgan and the double negation laws.
4. Distribute all disjunctions over conjunctions using the second distributivity, the commutativity and the associativity laws.

Note that after the step 3 the formula is in negation normal form. The negation normal form can be transformed into disjunctive normal form if in the fourth step the conjunctions are distributed over the disjunctions (application of commutativity and associativity laws might be needed).