Lecture 7. Logic. Section 1: Statement Logic.

1. Statement Logic

1.0. Goals

Our next big topic is Logic. We begin with Statement Logic and then consider Predicate Logic. We presuppose that you know something about Logic. Our goals here are to discuss some basic notions:

- Logical Languages as examples of Formal Languages
- Syntax and Semantics
- Models and Model-theoretic Semantics
- The Principle of Compositionality
- Axioms and Theories
- Algebra and Logic.

1.1. Syntax of Statement Logic

One of the main features of formal languages is a well-defined Syntax and Semantics. Syntax deals with the structure of expressions (and formulas) of language. Semantics deals with the meaning of expressions of language.

The syntax of Statement Logic is very simple. Formulas of Statement Logic are built of elements of two main kinds: atomic statements and logical connectives, symbols of logical operations.

We assume an infinite set \( \text{Atom} \) of atomic statements represented by the symbols \( p, q, r, s, \ldots \), with primes or subscripts added as needed, \( \text{Atom} = \{p, q, r, s, \ldots\} \)

Several logical connectives are used: the unary connective \( \neg \) (negation) and the binary connectives \( \wedge \) (conjunction), \( \vee \) (disjunction), \( \rightarrow \) (conditional) and \( \leftrightarrow \) (biconditional).

Examples of formulas:

\( \neg \psi \) (negation)
\( 
\varphi \land \psi 
\) (conjunction)
\( 
\varphi \lor \psi 
\) (disjunction)
\( 
\varphi \implies \psi 
\) (conditional)
\( 
\varphi \iff \psi 
\) (biconditional)

1.2. Semantics of Statement Logic

The semantics of statement logic is nearly as simple as its syntax. First of all we consider two truth values: 1 (true) and 0 (false). Let us denote the set of truth values by \( D_1 = \{0, 1\} \). We are thus working with two-valued logic. Systems with more than two values have also been studied. But they will not concern us here. [See notes at end under “Supplementary Reading.”]

Let us denote the set of truth values by \( D = \{0, 1\} \). The principle of compositionality is a homomorphism.

Syntactic Rules:

1. Any atomic statement is a formula.
2. If \( \varphi \) and \( \psi \) are formulas then \( \neg \varphi \), \( \varphi \land \psi \), \( \varphi \lor \psi \), \( \varphi \implies \psi \) and \( \varphi \iff \psi \) are also formulas.
3. There are no other formulas (in Statement Logic).

Note. Sometimes some other symbols are used with the same names (and meaning): \( \text{\sim} \) for negation, \( \text{\&} \) for conjunction (these symbols are used in PtMW), \( \text{\equiv} \) for conditional (called also implication) and \( \text{\equiv} \) or \( \text{\sim} \) for biconditional (called also equivalence symbols).

Examples of formulas: \( p, q, (p \lor q), \neg(p \rightarrow (p \land q)) \)

Notes. Atomic formulas (atomic statements) are considered as logical representatives of simple declarative sentences of natural language. The connectives \( \land, \lor, \rightarrow \) and \( \iff \) are intended as counterparts of natural language connectives, for example, English and, or, if ... then, and if and only if, when used to conjoin declarative sentences. – is a unary operator, its English counterpart is not or it is not the case that. But the relation between (statement) logic and natural language is not uncontroversial. [On first sight, it would seem that there are major discrepancies between the interpretations of some of the logical connectives and of their natural language counterparts. With careful attention to the distinction between semantics and pragmatics, as urged and exemplified in the work of Grice, one can make a much more plausible case for a close semantic correspondence between the logical connectives and their natural language counterparts. But it is still controversial; see, for instance, Angelika Kratzer’s work on conditionals.]

1.3. Logical equivalence and laws of statement logic

The principle of compositionality is a homomorphism.

Supplementary Reading

Homework 7, due Oct 10.

values of these constituents by truth-functional\ properties of the connective used in the formula. These truth-functional properties of connectives are usually given in the form of truth tables.

Below we give the truth tables for the five connectives used in our formulas. In the following, \( \phi \) and \( \psi \) stand for any arbitrary formulas, atomic or complex.

### 1.2.1. Negation

Negation reverses the truth value of the statement to which it is attached. For any formula \( \phi \), if \( \phi \) is true, then \( \neg \phi \) is false, and, conversely, if \( \phi \) is false, then \( \neg \phi \) is true. This is summarized in the truth table below.

<table>
<thead>
<tr>
<th>( \phi )</th>
<th>( \neg \phi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### 1.2.2. Conjunction

The result of logical conjunction is true iff both of its conjuncts are true. It corresponds to the meaning of English and conjoining two declarative sentences.

The truth table for the logical connective \( \land \) is given below:

<table>
<thead>
<tr>
<th>( \phi )</th>
<th>( \psi )</th>
<th>( (\phi \land \psi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Note that \( \phi \) and \( \psi \) are variables denoting any formulas whatsoever and there are four rows in the table corresponding to the four ways of assigning two truth values independently to two statements.

### 1.2.3. Disjunction

The logical connective \( \lor \) has the following truth table:

<table>
<thead>
<tr>
<th>( \phi )</th>
<th>( \psi )</th>
<th>( (\phi \lor \psi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### 1.2.4. The Conditional

The truth table for the conditional is shown below:

<table>
<thead>
<tr>
<th>( \phi )</th>
<th>( \psi )</th>
<th>( (\phi \rightarrow \psi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

So the formula \( (\phi \rightarrow \psi) \) is false only in the case when its antecedent \( \phi \) is true and the consequent \( \psi \) is false. This table mirrors the use of conditionals in mathematics (in inferences in proofs); its correspondence to natural language if-then is controversial.

### 1.2.5. The Biconditional

The truth table for the biconditional is shown below:

<table>
<thead>
<tr>
<th>( \phi )</th>
<th>( \psi )</th>
<th>( (\phi \leftrightarrow \psi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The biconditional corresponds to if and only if, abbreviated as iff; as with the conditional, its correspondence to natural language if and only if is controversial (if one considers if and only if a part of “natural” natural language at all).

### 1.2.6. Truth value of complex formula

The truth tables provide a general and systematic method of computing the truth value of any arbitrary complex statement. The number of lines in the truth table (for a given formula) is determined by the requirement that all possible combinations of truth values of atomic statements must be considered. In general, there are \( 2^n \) lines when there are \( n \) atomic statement in the formula. The order of evaluating the constituent statements is from the most deeply embedded ones to the outermost. So to construct a truth table for \( (\neg(\phi \rightarrow \psi) \leftrightarrow (\phi \land \psi)) \) one would proceed as follows:
1.4. Logical equivalence and laws of statement logic

Two statements are logically equivalent if they have the same truth value for any possible assignment of truth values to their atomic parts. To denote logical equivalence between two arbitrary statements \( \phi \) and \( \psi \) we write \( \phi \equiv \psi \). Note that “double arrow” is not a new connective for statements, but rather a metalinguage symbol expressing logical equivalence. Read \( \phi \equiv \psi \) as “\( \phi \) if and only if \( \psi \)” and \( \phi \equiv \psi \) as “the statement \( \phi \) is logically equivalent to the statement \( \psi \)”.

It is a property of statement logic that if a biconditional statement is a tautology, the two constituent statements so connected are logically equivalent. For example, \( \neg(p \lor q) \) and \( \neg p \land \neg q \) are logically equivalent, and \( \neg(p \lor q) \iff (\neg p \land \neg q) \) is a tautology. Logically equivalent statements are important because they can freely replace one another in any statement without affecting its truth value.

Below we write down some most frequently used “laws” of equivalency. But first let us add to the set Atom of atomic statements two constants: True and False and let us for convenience use for them the same symbols as for truth values, 1 and 0.

Laws of statement logic

1. Idempotent Laws
   (a) \( (\phi \lor \phi) \iff \phi \)
   (b) \( (\phi \land \phi) \iff \phi \)

2. Commutative Laws
   (a) \( (\phi \lor \psi) \iff (\psi \lor \phi) \)
   (b) \( (\phi \land \psi) \iff (\psi \land \phi) \)

3. Associative Laws
   (a) \( ((\phi \lor \psi) \lor \chi) \iff (\phi \lor (\psi \lor \chi)) \)
   (b) \( ((\phi \land \psi) \land \chi) \iff (\phi \land (\psi \land \chi)) \)

4. Distributive Laws
   (a) \( (\phi \lor (\psi \land \chi)) \iff ((\phi \lor \psi) \land (\phi \lor \chi)) \)
   (b) \( (\phi \land (\psi \lor \chi)) \iff ((\phi \land \psi) \lor (\phi \land \chi)) \)

5. Identity Laws
   (a) \( (\phi \lor 0) \iff \phi \)
   (b) \( (\phi \lor 1) \iff 1 \)
   (c) \( (\phi \lor 0) \iff 0 \)
   (d) \( (\phi \lor 1) \iff \phi \)

6. Complement Laws
   (a) \( (\phi \lor \neg \phi) \iff 1 \)
   (b) \( (\neg \phi) \iff \neg \phi \) (double negation)
   (c) \( (\phi \lor \neg \phi) \iff 0 \)

7. DeMorgan’s Laws
   (a) \( (\neg (\phi \lor \psi)) \iff (\neg \phi \land \neg \psi) \)
   (b) \( (\neg (\phi \land \psi)) \iff (\neg \phi \lor \neg \psi) \)

8. Conditional Laws
   (a) \( (\phi \iff \psi) \iff (\neg \phi \iff \neg \psi) \)
   (b) \( (\phi \iff \psi) \iff (\neg \phi \iff \neg \psi) \)
2) logical connectives $\neg$, $\land$, $\lor$. As we have seen we can restrict ourselves to these three and consider formulas with other connectives as abbreviations;

3) the brackets ( and ).

We will do it recursively. We will begin with a basic set $\text{Atom}$ of atomic statements. Then we will define some operations on the set $\text{Form}$ of formulas and with the help of these operations we will construct the set $\text{Form}$. We will name our operations by symbols of logical connectives.

(1) Basis of recursion: every symbol of $\text{Atom}$ (considered as one-symbol word) is a (atomic) formula.

(2) Recursive rules:

- $\neg$ if $\varphi$ is a formula, then $\neg \varphi$ is a formula (considered as the result of prefixing $\neg$ to $\varphi$).
- $\land$ if $\varphi$ and $\psi$ are formulas, then $(\varphi \land \psi)$ is a formula (considered as the result of concatenating $(\varphi, \land, \psi)$ in that order).
- $\lor$ if $\varphi$ and $\psi$ are formulas, then $(\varphi \lor \psi)$ is a formula (considered as the result of concatenating $(\varphi, \lor, \psi)$ in that order).

These syntactic rules define the set of words $\text{Form}$. But we can consider the set of formulas as an algebra $\text{Form}$ in the signature $\Omega_{BA} = \{0, 1, \neg, \land, \lor\}$ defined on the set $\text{Form}$ of words as the carrier of our algebra. The operations of $\Omega_{BA}$ are defined in a natural way: constants 0 and 1 “mark” corresponding atomic statements, the unary operation $\neg$ applied to formula $\varphi$ gives the formula $\neg \varphi$ and the binary operations $\land$ and $\lor$ applied to formulas $\varphi$ and $\psi$ give formulas $(\varphi \land \psi)$ and $(\varphi \lor \psi)$ respectively.

Note that the algebra $\text{Form}$ has a very special property: any formula is either an atomic formula, or a unique result of some operation applied to its “constituent” subformulas. I.e., no formula is structurally ambiguous; each can be derived in only one way. [This can be proved by induction, but we won’t try to prove it here.]

2.2. Semantics as a homomorphism

Now we define the Semantics of Statement Logic as a homomorphism $\sigma: \text{Form} \rightarrow T$ (where $T$ is the Boolean algebra of truth values considered in Lecture 4).

The homomorphism $\sigma$ is the mapping $\sigma: \text{Form} \rightarrow \{0, 1\}$ defined in the following way:

- $\sigma(0) = 0, \sigma(1) = 1$. On the set of variable statements $\{p, q, r, s, \ldots\}$, $\sigma$ is defined in some arbitrary way.

- On complex formulas $\sigma$ is defined in such a way that it will secure a homomorphism. For any formulas $\varphi$ and $\psi$, we have:

\[ \sigma (\varphi \land \psi) = \sigma (\varphi) \land \sigma (\psi), \quad \sigma (\varphi \lor \psi) = \sigma (\varphi) \lor \sigma (\psi), \quad \sigma (\neg \varphi) = \neg \sigma (\varphi) \]

1 Later we will study the important notion of proof by induction. At that time, this would be a good example to practice on.
The syntactic rules (1) and (2) (algebra Form) can be used in assigning to each formula of SL a unique derivation tree (similar to a Phrase-Structure tree). The semantics given as homomorphism $\sigma$ in rules in (a) and (b) will then give each tree a compositional interpretation, following the derivation node by node from the bottom up. Below is one illustration.

The derivation of the statement $\neg(p \lor q)$ is presented in (2-5). Its compositional interpretation is as given in (2-6), where it assumed that the truth values of the atomic statements are given: 0 for $p$ and 1 for $q$.

(2-5) \hspace{2cm} (2-6)
\[\neg(p \lor q) \quad \neg \]
\[
\begin{array}{c}
(p \lor q) \\
\lor \\
q \\
p \\
\end{array}
\]
\[
\begin{array}{c}
\neg \\
\lor \\
0 \\
\end{array}
\]

Supplementary Reading.
1. If you can read German, a great thing to look at is Godehard Link’s small but superb (and very algebraic) Montague Grammar textbook, which I can lend you. He works through statement logic, first-order predicate logic, then various enrichments with type theory and tense and modality, and for each fragment he includes an explicitly formulated algebraic statement of the syntax and semantics and shows the homomorphism between them.
2. If you are interested in vagueness/context-dependence and systems with more than two truth-values, you might be interested in the contrast between linear systems of truth values (seeing truth values as numbers (real or rational) in the interval $[0,1]$) and truth values which have a Boolean structure. There are arguments about why truth values should form a Boolean algebra in: Kamp and Partee (1995) Prototype theory and compositionality. Cognition 57:129-191. In the context of this lecture, we could recast those arguments in terms of the need for a homomorphism between the syntactic algebra and the semantic one. It’s interesting to see what goes wrong when you try to make the semantic algebra something that doesn’t have a Boolean structure. But this should wait until we come back later to constructing a Boolean algebra based on the syntactic algebra Form.

Homework 7, due Oct 10.

I. Exercises from PtMW:
Chapter 6, pp. 129, 130. #3 (a,h,e); 4(a,b).
Chapter 13, pp. 365, 366. #1ai, 1bi.

II. Show that the algebra Form is not a Boolean algebra.