

Lecture 7. Logic. Section1: Statement Logic.

1. Statement Logic.....	1
1.0. Goals.....	1
1.1. Syntax of Statement Logic	1
1.2. Semantics of Statement Logic	2
1.2.1. Negation	3
1.2.2. Conjunction	3
1.2.3. Disjunction	3
1.2.4. The Conditional.....	4
1.2.5. The Biconditional.....	4
1.2.6. Truth value of complex formula.....	4
1.3. Tautologies, contradictions and contingencies.....	5
1.4. Logical equivalence and laws of statement logic	6
2. Compositionality and a compositional account of statement logic: Semantics as a homomorphism.....	7
2.0. The principle of Compositionality.....	7
2.1. Formulas as Algebra.....	7
2.2. Semantics as a homomorphism	8
Supplementary Reading.....	9
Homework 7, due Oct 10.....	10

Reading: Chapter 5, Chapter 6: 6.1 – 6.4, Chapter13: 13.1, 13.1.1 of PTMW, pp. 87 – 134, 315-321.

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 *Atom* of *atomic statements* represented by the symbols p, q, r, s, \dots , with primes or subscripts added as needed, $Atom = \{p, q, r, s, \dots\}$

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

Note. Sometimes some other symbols are used with the same names (and meaning): “ \sim ” for negation, “ $\&$ ” for conjunction (these symbols is used in PtMW), “ \supset ” for conditional (called also *implication*) and “ \equiv ” or “ \approx ” for biconditional (called also *equivalence symbols*).

Syntactic Rules:

1. Any atomic statement is a formula.
2. If φ and ψ are formulas then $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ and $(\varphi \leftrightarrow \psi)$ are also formulas.
3. There are no other formulas (in Statement Logic).

[Compare the formulation in PtMW p.97; the ‘informal’ version above reflects common usage, in which we use the connectives and parentheses in our descriptive metalanguage as names of the same symbols in the object language, the language we are describing.]

Examples of formulas: $p, q, (p \vee q), \neg(p \rightarrow (p \wedge q))$

Notes. Atomic formulas (atomic statements) are considered as logical representatives of simple declarative sentences of natural language. The connectives $\wedge, \vee, \rightarrow$ and \leftrightarrow 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. \neg 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.]

1.2. 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_t , $D_t = \{1, 0\}$. 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 note at end under “Supplementary Reading”.]

Each atomic statement is assumed to have assigned to it one of the two truth values. Each complex formula also receives a truth value, which is determined by

- (1) the truth values of its syntactic components, and
- (2) the syntactic structure of the formula, i.e., its connectives and their arrangements in the formula.

So, semantics of every formula is defined compositionally, with the help of its syntax. As we see, syntactically every complex formula is built from its immediate constituents with the help of some connective. The truth value of the formula is determined by truth

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, ϕ and ψ 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 ϕ , if ϕ is true, then $\neg\phi$ is false, and, conversely, if ϕ is false, then $\neg\phi$ is true. This is summarized in the truth table below

ϕ	$\neg\phi$
1	0
0	1

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 \wedge is given below:

ϕ	ψ	$(\phi \wedge \psi)$
1	1	1
1	0	0
0	1	0
0	0	0

Note that ϕ and ψ 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 \vee has the following truth table:

ϕ	ψ	$(\phi \vee \psi)$
1	1	1
1	0	1
0	1	1
0	0	0

Thus the disjunction of two statements is true if at least one of the disjuncts is true; it is false only if both are false.

1.2.4. The Conditional

The truth table for the conditional is shown below:

ϕ	ψ	$(\phi \rightarrow \psi)$
1	1	1
1	0	0
0	1	1
0	0	1

So the formula $(\phi \rightarrow \psi)$ is false only in the case when its *antecedent* (ϕ) is true and the *consequent* (ψ) 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:

ϕ	ψ	$(\phi \leftrightarrow \psi)$
1	1	1
1	0	0
0	1	0
0	0	1

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 $(\neg(p \rightarrow q) \leftrightarrow (p \wedge r))$ one would proceed as follows:

$$(c) (\varphi \rightarrow \psi) \Leftrightarrow \neg(\varphi \wedge \neg\psi)$$

9. Biconditional Laws

$$(a) (\varphi \leftrightarrow \psi) \Leftrightarrow ((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)) \quad (b) (\varphi \leftrightarrow \psi) \Leftrightarrow ((\neg\varphi \wedge \neg\psi) \vee (\varphi \wedge \psi))$$

Note that the first seven kinds of laws are similar to the first seven kinds of set-theoretical laws (Lecture 1), as well as to the axioms for Boolean algebras (Lecture 4, Section 2). We will discuss this similarity later.

These laws also demonstrate that some connectives could be represented with the help of others. For example, all connectives can be represented in terms of combinations with negation and conjunction, or with negation and disjunction. There exist two binary connectives, “Sheffer stroke” and “Quine’s dagger”, each of which has the property that all the other connectives can be represented with the help of it (see Appendix BI in PTMW, p. 238).

2. Compositionality and a compositional account of statement logic: Semantics as a homomorphism

2.0. The principle of Compositionality

Above, describing the syntax and the semantics of statement logic, we claimed that we do it compositionally. The principle of compositionality plays a central role in the relation of semantics to syntax in formal languages. The principle of compositionality was introduced by Gottlob Frege and is also called Frege’s principle.

The Principle of Compositionality. The meaning of a complex expression is a function of the meaning of its parts and of the syntactic rules by which they are combined.

Note that in this form this principle is rather vague. To formulate it in a more exact way we need more precise notions of *meaning*, of *part*, and of *syntactic rules*, as well as permitted *functions*.

Below we formulate these notions for statement logic. Note that it can be done in many ways. Our approach here is different than in PtMW Chapter 13. In this Section we will show that it is possible to give to this Principle an algebraic meaning in terms of *homomorphism*.

2.1. Formulas as Algebra

Let us reformulate the syntax of the statement logic. We will define the set *Form* of formulas of SL as a set of *words (strings)* built of symbols of three kinds:

1) the set of symbols of atomic statements $Atom = \{0, 1\} \cup \{p, q, r, s, \dots\}$. Here 1 and 0 are constants representing the constantly true and false statement respectively, and p, q, r, s, \dots are *variable statements*;

2) logical connectives \neg, \wedge, \vee . 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 *Atom* of atomic statements. Then we will define some operations on the set *Form* of formulas and with the help of these operations will construct the set *Form*. We will name our operations by symbols of logical connectives.

(1) **Basis of recursion:** every symbol of *Atom* (considered as one-symbol word) is a (atomic) formula.

(2) **Recursive rules:**

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

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

Note that the algebra **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.¹]

2.2. Semantics as a homomorphism

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

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

- (a) $\sigma(0) = 0, \sigma(1) = 1$. On the set of variable statements $\{p, q, r, s, \dots\}$, σ is defined in some arbitrary way.
- (b) On complex formulas σ is defined in such a way that it will secure a homomorphism. For any formulas φ and ψ , we have:

¹ Later we will study the important notion of proof by induction. At that time, this would be a good example to practice on.

$$\sigma(\neg\phi) = \neg\sigma(\phi)$$

$$\sigma(\phi \vee \psi) = \sigma(\phi) \vee \sigma(\psi)$$

$$\sigma(\phi \wedge \psi) = \sigma(\phi) \wedge \sigma(\psi)$$

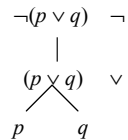
The rule (b) is possible because of the property of the algebra **Form** mentioned above. We say that the mapping σ given on atomic formulas is *extended to complex ones*.

Note: Unless you remember to read \wedge and \vee as operations on the algebras **Form** and **T**, rule (b) will probably look peculiar, as if it's mapping syntax onto syntax. That's because we're used to reading $(\phi \vee \psi)$ as a syntactic formula, not as an expression denoting the result of applying the syntactic (on the left-hand side) or semantic (on the right-hand side) operation \vee to the arguments ϕ, ψ . If it helps, you could represent the signature for yourself as $\{0, 1, \text{Op}_{\neg}, \text{Op}_{\vee}, \text{Op}_{\wedge}\}$.

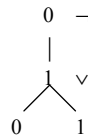
The syntactic rules (1) and (2) (algebra **Form**) can be used in assigning to each formula of SL a unique derivation tree (similar to a Phrase-Structure tree). The semantics given as homomorphism σ in rules in (a) and (b) will then give each tree a compositional interpretation, following the derivation node by node from the bottom up. Below is one illustration.

The derivation of the statement $\neg(p \vee q)$ 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)



(2.6)



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,b,e); 4(a,b).

Chapter 13, pp. 365, 366. #1ai, 1bi.

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