Sentential Logic – Chapter 1

1 Informal Remarks on Formal Languages

Remark 1.1 (Mini-lesson 1.0.1: Why formal languages?)**.** We want a symbolic language into which we can translate English sentences. Unlike natural languages (English, Chinese, ...), a formal language has precise formation rules and avoids ambiguity. In this section we look informally at the ingredients such a language should have.

Example 1.2 (Atomic sentences and negation)

Let

$$\mathbf{K} \quad \text{mean "Traces of potassium were observed."}$$

Then

$$(\neg \mathbf{K})$$

is read "It is not the case that traces of potassium were observed." Here $\neg$ is the *negation symbol*.

We deliberately treat "Traces of potassium were not observed" as $\neg\mathbf{K}$ rather than introducing a brand new atomic symbol such as $\mathbf{J}$.

Remark 1.3 (Decomposing vs. rebranding)**.** Whenever possible we prefer to break an English sentence into simpler atomic parts, rather than replacing the whole sentence by a new atomic symbol. This is what makes logical structure visible.

Example 1.4 (Conjunction and conditional)

Let

$$\mathbf{C} \quad \text{mean "The sample contained chlorine."}$$

Then we may translate:

- "If traces of potassium were observed, then the sample did not contain chlorine." as
$$(\mathbf{K} \to (\neg \mathbf{C})).$$
- "The sample contained chlorine, and traces of potassium were observed." as
$$(\mathbf{C} \wedge \mathbf{K}).$$

The symbol $\wedge$ is the *conjunction symbol* ("and"); $\to$ is the *conditional symbol* ("if … then …").

Example 1.5 (Disjunction and "neither … nor")

Using $\vee$ as the *disjunction symbol* ("or", inclusive):

- "Either no traces of potassium were observed, or the sample did not contain chlorine." becomes
$$((\neg \mathbf{K}) \vee (\neg \mathbf{C})).$$
- "Neither did the sample contain chlorine, nor were traces of potassium observed." can be rendered in two equivalent ways:
$$(\neg(\mathbf{C} \vee \mathbf{K})) \quad \text{or} \quad ((\neg \mathbf{C}) \wedge (\neg \mathbf{K})).$$

The relationship between these two formulas will be analyzed later.

Remark 1.6 (Truth of compounds from truth of atoms)**.** Once we know the truth values of the atomic sentences (here **K** and **C**), the truth values of the compound sentences above are completely determined by the connectives.

For instance, if the chemist reports that she *did* observe traces of potassium and the sample *did not* contain chlorine, then the four compound sentences above are respectively: true, false, true, false.

TABLE I Truth values of two equivalent translations

$\mathbf{K}$	$\mathbf{C}$	$(\neg(\mathbf{C} \vee \mathbf{K}))$	$((\neg \mathbf{C}) \wedge (\neg \mathbf{K}))$
F	F	T	T
F	T	F	F
T	F	F	F
T	T	F	F

Remark 1.7 (Precision vs. expressiveness)**.** A formal language lets us avoid the vagueness and ambiguity of natural language, but at a price: its expressive power is quite limited compared with everyday speech. We will have to be very explicit about what can and cannot be said.

Definition 1.8 (Describing a formal language)**.** To describe a formal language we give:

1. its *alphabet* (the set of basic symbols);
2. its *formation rules*, specifying which finite symbol strings are *well-formed formulas* (wffs);
3. a translation scheme between English and the formal language, assigning meanings to some wffs.

Remark 1.9 (Syntax vs. semantics)**.** Only item (3) assigns meanings. Items (1)–(2) are purely syntactic. In principle, someone who knows only (1) and (2) could still manipulate wffs according to the rules—without any understanding of what the formulas *mean*.

Example 1.10 (Mini-lesson 1.0.2: Computer languages as formal languages)

Modern computer languages offer familiar examples of formal languages.

- In one language a typical wff is a binary string, for example

  $$0110101101010001111100010000011110 10.$$

- In another, a wff might be

  `STEP#ADDIMAX, A.`

  where # is a special "blank" symbol.

- In `C++` we find wffs such as

  `while(*s++);`

Each has a precise syntax and a fixed (though restricted) way of translating to English. The computer, however, knows only the syntax; it blindly follows rules for symbol manipulation.

Remark 1.11 (Our attitude vs. the computer's)**.** We *could* treat logic like a computer language and manipulate formulas mechanically, ignoring meaning. But for us the intended English interpretations will be crucial for understanding why the rules are sensible and how to use them.

2 The Language of Sentential Logic

Definition 2.1 (Alphabet of sentential logic)**.** We fix an infinite list of distinct objects called *symbols*. The alphabet of sentential logic consists of the symbols listed in Table II. No symbol itself is a finite sequence of other symbols.

TABLE II Symbols of sentential logic

Symbol	Verbose name	Remarks
(	left parenthesis	punctuation
)	right parenthesis	punctuation
$\neg$	negation symbol	English: not
$\wedge$	conjunction symbol	English: and
$\vee$	disjunction symbol	English: or (inclusive)
$\rightarrow$	conditional symbol	English: if
$\leftrightarrow$	biconditional symbol	English: if and only if
$\mathbf{A}_1$	first sentence symbol	
$\mathbf{A}_2$	second sentence symbol	
...		
$\mathbf{A}_n$	nth sentence symbol	
...		

Remark 2.2 (Logical vs. nonlogical symbols)**.** The connectives

$$\neg,\ \wedge,\ \vee,\ \rightarrow,\ \leftrightarrow$$

and the parentheses are the *logical symbols*. Their role in translation never changes. The $\mathbf{A}_n$ are *sentence symbols* (or parameters, or nonlogical symbols); their intended English meanings can vary from one context to another.

Remark 2.3 (How many sentence symbols?)**.** We have chosen countably many sentence symbols

$$\mathbf{A}_1, \mathbf{A}_2, \ldots.$$

A more economical alternative would be to use a single symbol $\mathbf{A}$ and a "prime" symbol to get $\mathbf{A}, \mathbf{A}', \mathbf{A}'', \ldots$, reducing the size of the alphabet.

A more generous alternative would be to allow an arbitrary (even uncountable) set of sentence symbols. Much of what is said in this chapter would continue to be applicable in that case; the exceptions are primarily in Section 1.7.

Remark 2.4 ("Sentential" vs. "propositional")**.** Some authors prefer to speak of *proposition symbols* and *propositional logic*, reserving the word "sentence" for specific utterances and "proposition" for what such a sentence asserts.

Remark 2.5 (What are symbols, really?)**.** We call these objects "symbols" without committing ourselves to their ontological nature. They could be sets, numbers, marbles, or elements of a universe of linguistic objects. The printed strings "$\mathbf{A}_{243}$", "$\rightarrow$" are *names* of those symbols, not the symbols themselves.

In the last case, it is conceivable that the symbols are actually the same things as the names we use for them. Another possibility, which will be explained in the next chapter, is that the sentence symbols are themselves formulas in another language.

Remark 2.6 (Unique decomposition of strings)**.** We assume no symbol is itself a finite sequence of other symbols. Thus, if

$$\langle a_1, \ldots, a_m \rangle = \langle b_1, \ldots, b_n \rangle$$

with each a_i and b_j a symbol, then $m = n$ and $a_i = b_i$ for all i. This guarantees that any finite sequence of symbols is uniquely decomposable into its individual symbols. (Compare Chapter 0, Lemma 0A.)

Definition 2.7 (Expressions and concatenation)**.** An *expression* is a finite sequence of symbols. We name expressions by writing the symbols consecutively. For example

$$(\neg \mathbf{A}_1)$$

denotes the sequence $\langle (, \neg, \mathbf{A}_1,) \rangle$.

If α and β are expressions, then $\alpha\beta$ is the expression obtained by concatenating the sequence for β after that for α.

Example 2.8 (Concatenation)

Let

$$\alpha = (\neg \mathbf{A}_1), \qquad \beta = \mathbf{A}_2.$$

Then

$$(\alpha \to \beta)$$

stands for the expression

$$\big((\neg \mathbf{A}_1) \to \mathbf{A}_2\big).$$

Example 2.9 (Translations involving law and economics)

Let $\mathbf{A}, \mathbf{B}, \ldots, \mathbf{Z}$ be the first 26 sentence symbols (e.g. $\mathbf{E} = \mathbf{A}_5$). Examples of translations:

- "The suspect must be released from custody." $\rightsquigarrow$ $\mathbf{R}$.
- "The evidence obtained is admissible." $\rightsquigarrow$ $\mathbf{E}$.
- "The evidence obtained is inadmissible." $\rightsquigarrow$ $(\neg\mathbf{E})$.
- "The evidence obtained is admissible, and the suspect need not be released from custody." $\rightsquigarrow$ $(\mathbf{E} \wedge (\neg\mathbf{R}))$.
- "Either the evidence obtained is admissible, or the suspect must be released from custody (or both)." $\rightsquigarrow$ $(\mathbf{E} \vee \mathbf{R})$.
- "Either the evidence obtained is admissible, or the suspect must be released from custody, but not both." $\rightsquigarrow$ $\big((\mathbf{E} \vee \mathbf{R}) \wedge \neg(\mathbf{E} \wedge \mathbf{R})\big)$.
- "The evidence obtained is inadmissible, but the suspect need not be released from custody." $\rightsquigarrow$ $((\neg\mathbf{E}) \wedge (\neg\mathbf{R}))$.

The last expression contrasts with $((\neg\mathbf{E}) \vee (\neg\mathbf{R}))$, which translates "Either the evidence obtained is inadmissible or the suspect need not be released from custody." We intend always to use $\vee$ for the inclusive "or" (i.e. "and/or").

Example 2.10 ("If", "iff")

- "If wishes are horses, then beggars will ride." $\rightsquigarrow$ $(\mathbf{W} \to \mathbf{B})$.
- "Beggars will ride if and only if wishes are horses." $\rightsquigarrow$ $(\mathbf{B} \leftrightarrow \mathbf{W})$.
- "This commodity constitutes wealth iff it is transferable, limited in supply, and either productive of pleasure or preventive of pain." $\rightsquigarrow$ $\big(\mathbf{W} \leftrightarrow (\mathbf{T} \wedge (\mathbf{L} \wedge (\mathbf{P} \vee \mathbf{Q})))\big)$.

Note that the same symbol (e.g. $\mathbf{W}$) may be used with different English meanings in different examples.

Remark 2.11 (Object language vs. English)**.** Do not confuse an English sentence such as "Roses are red" with a formula such as $\mathbf{R}$ that *represents* it. The English sentence is meaningful and (we hope) either true or false. The formula $\mathbf{R}$ is, by itself, just a string of symbols that can receive various interpretations.

Example 2.12 (Ill-formed expression)

Some expressions, such as

$$((\to \mathbf{A}_3,$$

cannot reasonably be translations of any English sentence. We will call such expressions *ill-formed*.

Definition 2.13 (Well-formed formulas via closure)**.** We want to specify exactly which expressions are *well-formed formulas* (wffs), i.e. grammatically correct expressions.

Informally, we require:

(a) every sentence symbol is a wff;

(b) if α and β are wffs, then so are $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \to \beta)$, $(\alpha \leftrightarrow \beta)$;

(c) nothing is a wff unless it must be by repeated use of (a) and (b).

Formally, define the following *formula-building operations* on expressions:

$$\begin{aligned}
\mathcal{E}_{\neg}(\alpha) &= (\neg\alpha),\\
\mathcal{E}_{\wedge}(\alpha,\beta) &= (\alpha \wedge \beta),\\
\mathcal{E}_{\vee}(\alpha,\beta) &= (\alpha \vee \beta),\\
\mathcal{E}_{\to}(\alpha,\beta) &= (\alpha \to \beta),\\
\mathcal{E}_{\leftrightarrow}(\alpha,\beta) &= (\alpha \leftrightarrow \beta).
\end{aligned}$$

A wff is any expression that can be built from sentence symbols by applying these operations finitely many times.

Example 2.14 (A complex wff and its tree)

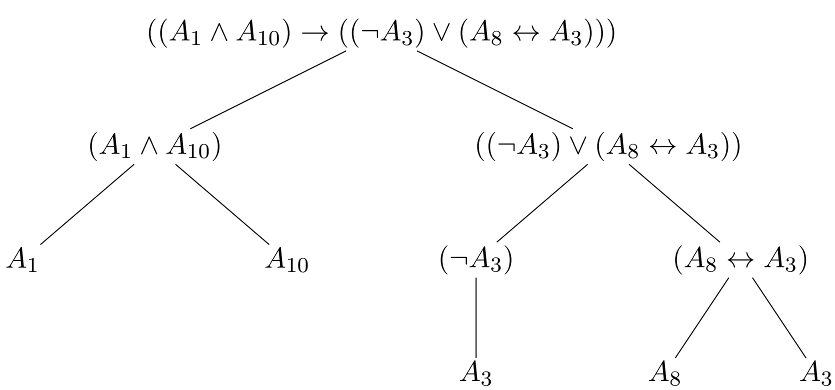
The expression

$$\Big((\mathbf{A}_1 \wedge \mathbf{A}_{10}) \to \big((\neg\mathbf{A}_3) \vee (\mathbf{A}_8 \leftrightarrow \mathbf{A}_3)\big)\Big)$$

is a wff. It is built from the four sentence symbols $\mathbf{A}_1, \mathbf{A}_{10}, \mathbf{A}_3, \mathbf{A}_8$ by five applications of the formula-building operations. Its *ancestral tree* records this construction step by step (Figure **??**).

At the other extreme, each sentence symbol (e.g. $\mathbf{A}_3$) is a wff whose ancestral tree has only one vertex.

We do not count the empty sequence as being "built up from the sentence symbols".



Remark 2.15 (Building by closing under operations)**.** This sort of construction—taking some basic building blocks (here the sentence symbols) and "closing" under some operations (here the five formula-building operations)—occurs frequently both in logic and in other branches of mathematics. In Section 1.4, we will examine this kind of construction in a more general setting.

Definition 2.16 (Construction sequence)**.** A *construction sequence* is a finite sequence $\langle \varepsilon_1, \dots, \varepsilon_n \rangle$ of expressions such that for each $i \leq n$ at least one of the following holds:

- ε_i is a sentence symbol;
- $\varepsilon_i = \mathcal{E}_\neg(\varepsilon_j)$ for some $j < i$;
- $\varepsilon_i = \mathcal{E}_\Box(\varepsilon_j, \varepsilon_k)$ for some $j < i$, $k < i$, where $\Box$ is one of $\wedge, \vee, \rightarrow, \leftrightarrow$.

An expression α is a wff iff it appears as the last term of some construction sequence. We may think of ε_i as the expression produced at stage i of the building process.

> **Remark 2.17** (From trees to sequences)**.** The construction sequence for our complex example above is obtained by "flattening" its ancestral tree into a linear order: lower nodes come earlier, higher nodes later.

Definition 2.18 (Closure under an operation)**.** A set S is *closed* under a one-place function f if $x \in S$ implies $f(x) \in S$, and closed under a two-place function g if $x, y \in S$ imply $g(x, y) \in S$. We say S is closed under the five formula-building operations when this holds for each of them.

Proposition 2.19 (Induction principle on wffs)

Let S be a set of wffs that contains all sentence symbols and is closed under all five formula-building operations. Then S is the set of *all* wffs.

> **Remark 2.20** (Method 1.1.1: Proof by structural induction)**.** To prove a statement about all wffs, it suffices to:
>
> - verify it for all sentence symbols, and
> - show that if it holds for α (and for β when relevant), then it also holds for each of $(\neg\alpha)$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$, $(\alpha \leftrightarrow \beta)$.
>
> This is just an application of Proposition **??**.

Proof of Proposition **??** *(tree version).* Take any wff α. It is built from sentence symbols by finitely many applications of the formula-building operations; this is recorded in its ancestral tree. Starting from the leaves (sentence symbols in S) and moving upward, closure of S under each operation shows that every node of the tree lies in S. In particular, the root α lies in S. $\square$

Second proof of Proposition **??** *(sequence version).* Let α be any wff. Then α is the last term of some construction sequence $\langle \varepsilon_1, \dots, \varepsilon_n \rangle$. Use strong numerical induction on i to show that $\varepsilon_i \in S$ for all $i \leq n$. The induction step checks that whenever all earlier terms are in S, the rules defining construction sequences plus the closure of S under the operations guarantee that $\varepsilon_i \in S$. Hence $\alpha \in S$. $\square$

Example 2.21 (Balanced parentheses)

Claim. Any expression with more left parentheses than right parentheses is not a wff.

Idea. Start from sentence symbols (no parentheses) and observe that each formula-building operation adds parentheses only in matched pairs. So wffs always have equally many left and right parentheses.

Formally, let S be the set of expressions with equal numbers of left and right parentheses. Then:

- all sentence symbols lie in S;
- each formula-building operation preserves membership in S.

By Proposition **??**, all wffs lie in S, so any expression with unequal numbers of left and right parentheses cannot be a wff.

Remark 2.22 (Formulas only grow)**.** Each operation $\mathcal{E}_\square(\alpha, \beta)$ produces an expression that contains α (and β) as contiguous segments, plus extra symbols; in particular, the result is longer than either input. Thus every building step strictly increases length.

A useful consequence: if a wff φ does not contain the symbol $\mathbf{A}_4$, then no construction sequence for φ ever needs to use $\mathbf{A}_4$. (See Exercise **??**.)

Exercises

Exercise 2.23 (Exo 1.1)**.** Give three English sentences together with translations into our formal language. Choose sentences with interesting logical structure, and make each translation contain at least 15 symbols.

Exercise 2.24 (Exo 1.2)**.** Show that there are no wffs of length 2, 3, or 6, but that any other positive length occurs.

Exercise 2.25 (Exo 1.3)**.** Let α be a wff. Let c be the number of occurrences of binary connective symbols $(\wedge, \vee, \rightarrow, \leftrightarrow)$ in α, and let s be the number of occurrences of sentence symbols in α. For example, if α is $(\mathbf{A} \rightarrow (\neg\mathbf{A}))$ then $c = 1$ and $s = 2$. Use Proposition **??** (structural induction) to show that $s = c + 1$.

Exercise 2.26 (Exo 1.4)**.** Suppose we have a construction sequence ending in a wff φ and that φ does not contain the symbol $\mathbf{A}_4$. If we delete from the sequence all expressions that contain $\mathbf{A}_4$, show that what remains is still a legal construction sequence.

Exercise 2.27 (Exo 1.5)**.** Let α be a wff that does not contain the negation symbol $\neg$.

(a) Show that the length of α (number of symbols) is odd.

(b) Show that more than a quarter of the symbols of α are sentence symbols.

Hint. Use structural induction to show that the length of α is of the form $4k + 1$ and the number of sentence symbols is $k + 1$ for some integer k.