Notes on Knowledge Representation and Reasoning

Riccardo Lo Iacono

January 16, 2026

Contents.

- 1 - Introduction to Knowledge Representation and Reasoning.

Knowledge Representation and Reasoning (KRR) is a branch of Artificial Intelligence focused on understandig how knowledge, whose meaning will be soon clarified, can be represented symbolically and manipulated automatically by reasoning programs.

Formally, knowledge is defined as the relationship existing between a *knower* and a proposition; that is, the idea expressed by a simple declarative sentence (eg. "Mark will come to the party"). To represent and manipulate knowledge, which is our end goal, we have to understand what reasoning is. Let $p_1, p_2, \dots, p_n$ be a sequence of premises; we define *reasoning* as the process which, from the premises $p_1, \dots, p_n$ allows one to draw a conclusiion. Later on we discuss the logic formalization of reasoning.

Throughout this notes we use a "logic language", later discussed, known as First Order Logic (FOL). Similarly to any other language it has its own:

- *syntax*: the set of symbols and rules of the language;
- *semanic*: the meaning of syntactically correct sentences; and
- *pragmatics*: how contex and habits contribute to the meaning.

The next two sections cover all the type of reasoning. Precisely, we first discuss of all those aspects of the human reasoning that lead to an incorrect reasoning, we then focus on the three main types of correct reasoning.

- 1.1 - Sources of incorrect reasoning.

At the source of an incorrect (human) reasoning there are mainly three causes: *fallacies*, *biases* and *paradoxes*. Each of these groups a set of causes that we discuss briefly.

Remark. The lists we provide for each (fallacies, biases and paradoxes) is not exhaustive at all. More on these can be found easily online.

- 1.1.1 - Fallacies.

The concept of *fallacy* is straightforward. It is a argument that, despite sounding convincing, is invalid.

Example

Consider the sentece "A pencil isn't an animal. All animal feeds. Therefore a pencil doesn't feed". Though both premisies are true, and therefore the conclusion, the overall sentece is non-sensical.

We can distinguish several types of fallacies, some of which are listed below.

- Fallacy of composition (compositio): attributes to an entity as a whole some property valid for its parts.

  > "The product is on sale at a price of 10,000 euros, payable in small installments of only 100 euros each. Therefore the total price is moderate."

- Fallacy of division (divisio): attributes to the individual parts a property that is valid for the whole entity

  > "A truck is heavy. Therefore each of its component is heavy."

- Fallacy of relevance: derives a conclusion from premisies that are irrelevant.

  It has different forms. We list them below.

  - Argumentum ad hominem: discredits a thesis in order to deduce its falsity.
  - Tu quoque: the one proposing a thesis does the opposite of what he wants to prove.
  - Ad vericundum: the thesis is supposed to be true just because credit is given to those proposing it.
  - Ad populum: accepts a thesis simply because it's accepts by the majority.
  - Straw Man: creates a distorted version of a thesis, to refute it.
  - Ad ignorantiam: supports a thesis by refuting the arguments in favor of the opposite thesis.
  - Petitio principi: the thesis itself is hidden within the premisies.
  - False dichotomy: present two solution to the problem, as if these are the only ones.
  - Non causa pro causa: fix two peculiar characteristics and derive that the first causes the second one.

- 1.1.2 - biases.

We as humans are subject to biases, which are systematic pattern deviation from the norm or rationality in the mental process of judgement. As for fallacies, we can distinguish several forms of bias; and interestringly enough biases are at the base of some of the fallacies we discussed previously.

As done for fallacies, we list some of the most common types of biases, to give the reader an idea of what these are concretely.

- Confirmation bias: is a mental process that consists in selecting the information possessed in order to place greater attention, and therefore greater credibility, on those that confirm one's beliefs.
- Belief bias: is the tendency to judge the correctness of an argument based on the correctness of the conclusion.
- Availability bias (studied by Kahneman[1]and Tversky) is the tendency that leads us to make judgments on the basis of the most available examples and/or information, which most easily and vividly come to mind.

- 1.1.3 - paradoxes.

The concept of paradox is well known to everybody, though in the context of logic it's usually missleading; by this we refer to the fact that some of those that are generally considered paradoxes, are not. For instance, let us consider Epimenides's liar paradox:

"All Cretans always lie."

Since Epimenides was Cretan, he himself would be lying. Thus, the phrase "All Cretans always lie" cannot be true, cause it would contradict itself. Therefore, it's simply false.

More often then not paradoxes derive from the fact that semantic meaning is attributed to a sentence that refers to itself. This is what we call self-referentiality. This problem with self-referentiality was later understood by *Russell*.

- 1.2 - Correct forms of reasoning.

One can reach a correct reasoning through three approches: *deduction*, *abduction* and *induction*. We discuss each of these in the next few sections, but we focus maily on deduction in the rest of these notes.

We note that, in the case of formal reasoning, frequent mistakes usually concernes conditional statements, i.e., sentences of the form "If A, then B" or those that can somehow be transformed into one of this type. We call A and B, respectively, *antecedent* and *consequent*.

"If it rains, we always take the umbrella. Yesterday we took the umbrella, so yesterday it rained."

[1] For their result, Kahneman was praised with the nobel (Tversky was dead at that time). On the topic of biases, we suggest reading "Thinking, Fast and Slow" by Kahneman.

- 1.2.1 - Deduction.

Let us consider again the sentece:

> “If it rains, we always take the umbrella. Yesterday we took the umbrella, so yesterday it rained.”

Is it necessary true? What if we took the umbrella to rapair it?

This kind of reasoning is called deduction; that is, deduction is the process that from a “rule” (the conditional) derives a conclusion.

The above is an example of what we call *affirming the consequent*; that is, we deduce that A is true knowing that B is true, when there exist a conditional statements involving A and B. As its counterpart we have the *denying the antecedent* fallacy, in which we deduce the falsity of B by knowing with certainty that A is not true.

Given a conditional sentece, the two correct ways to apply deduction are called

- modus ponens: we know that A is true and deduce that B is also true.
- mudus tollens: we know that B is false and deduce that A is also false.

- 1.2.2 - Abduction and induction.

For what we have said about deduction, is easy to understand that it with logical inferences that are certain. Abduction and induction on the other hand cover probable inferences. That is, deduction differs from both, abduction and induction, by the degree of trust we put in the inference. To see such differece, see how the sentence

> “If it rains, we always take the umbrella. Yesterday we took the umbrella, so yesterday it rained.”

becomes the sentence

> “If it rains, we always take the umbrella. Yesterday we took the umbrella, so yesterday it probably rained.”

in the abduction case, and

> “If it rains, we always take the umbrella. Tomorrow it will rain, so probably tomorrow we will take the umbrella.”

in the induction one.

- 2 - Propositional logic.

At the core of Propositional Logic are atomic proposition, that are, any simple assertion. This means that, questions or meaningless sentences do not correspond to a proposition.

To build up more complex proposition we make use of connectives, that, up to a degree, have the same meaning to the English counterparts. Such connectives are: $\neg$ (not) and $\wedge$ (and). From these two, three additional connectives can be derived, though the latter are more a shorthand. Such additional connectives are: $\vee$ (or), which is equivalent to $\neg(\neg p \wedge \neg q)$, $\implies$ (implies), equivalent to $\neg p \vee q$ and $\iff$ (if and only if), which corresponds to $p \implies q \wedge q \implies p$, where p and q are atomic propositions. We use the symbol **1** and **0**, to indicate, respectively, a true and a false statement.

- 2.1 - Validity, satisfiability and contradiction.

Let $P = \{p_1, \ldots, p_n\}$ be a set of atomic formulas, let $\mathcal{F}(P)$ be the set of all formulas that can be derived from atomic propositions in P.

We say that a function $A : P \to \{\mathbf{0}, \mathbf{1}\}$ is an assignment of P; that is, A is an assignment of truth values to each atomic formula in P. The extension of A to $\mathcal{F}(P)$ is immediate. Given $F \in \mathcal{F}(P)$ and A an assignment of P, we say that A is a model of F if $A(F) = \mathbf{1}$; we write $A \models F$ to denote this concept. If $A(F) = \mathbf{1}$ for all A, we say that F is a tautology; and we write $\models F$ to denote it. Viceversa, if $A(F) = \mathbf{0}$ for all A, we say that F is a contradiction. In this case we write $\models \neg F$ to denote it. If $A(F) = \mathbf{1}$ for some A, we say that F is satisfiable.

As the reader may be aware of, the satisfiability problem, i.e., the problem of enstablishing if a given formula is true, is NP-complete[2].

- 2.2 - Consequence and equivalence.

Let F and G be two formulas; we say that G is a consequence of F if for every A such that $A \models F$, it holds $A \models G$. If it holds that F is a consequence of G and G a consequence of F, we say that F and G are equivalent; we write $F \equiv G$ to denote this.

Let $\mathcal{F} = \{F_1, F_2, \ldots\}$ be a set of formulas. Given a formula G, can we determine if $\mathcal{F} \models G$? Observe that if $\mathcal{F}$ is finite, one could simply check whether

$$\bigwedge_{i=1}^{n} \mathcal{F} \implies G,$$

that is, if $F_1 \wedge F_2 \wedge \cdots \wedge F_n \implies G$. In the case $\mathcal{F}$ is infinite such approach

[2] Throught this notes we assume the reader to have a background in complexity theory.

wont work at all. In this case we try to derive G from $\mathcal{F}$; we denote this by $\mathcal{F} \vdash G$.

- 2.3 - Rules, soundness and completeness of propositional logic.

Since Propositional Logic is, before everything, a logic it has rules which allows to deduce the truth of a sentence given another. Such rule system is a system of formal proof. In particular, we are interested in the relationship between the notion of formal proof and that of consequence; that is, we want our rule system to be *sound*.

Definition (soundness and completeness) Let $P = \{p_1, p_2, \ldots\}$ be a set ot atomic propositions, and let $\mathcal{F}(P)$. We say that our rule system is sound if every formula derived from $\mathcal{F}$ is a consequence of some proposition in $\mathcal{F}$. That is, if

$$\{G : \mathcal{F} \vdash G\} \subseteq \{G : \mathcal{F} \models G\}.$$

If it holds that

$$\{G : \mathcal{F} \models G\} \subseteq \{G : \mathcal{F} \vdash G\},$$

we say that our rule system is complete.

It can be shown that Propositional Logic is both sound and complete.

- 2.4 - Horn formulas and resolution.

Definition. Let F be a formula. We say that F is a literal if it's either an atomic formula, or the negation of an atomic formula.

Definition. Let F be a formula. We say that F is in conjunctive normal form (CNF) if it can be written as conjunction of disjunctions. That is, if

$$F = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m} L_{i,j} \right)$$

where $L_{i,j}$ is a literal.

It easy to understand that not all formulas are given in CNF. Thankfully there exists an algorithm that allows one to convert each non-CNF formulas into the equivalent CNF. We illustrate such algorithm in the following.

Let F be a non-CNF formula, to get the equivalent CNF:

Step 1: Replace each subformula of the form $G \implies H$ and $G \iff H$ with their expanded counterparts. That is, replace each $G \implies H$ with $\neg G \vee H$ and all $G \iff H$ with $(\neg G \vee H) \wedge (\neg H \vee G)$.

Step 2: Get rid of all double negations and apply De Morgan's rules. That is, replace

$$\begin{aligned}&\neg\neg G \text{ with } G,\\&\neg(G \wedge H) \text{ with } \neg G \vee \neg H, \text{ and}\\&\neg(G \vee H) \text{ with } \neg G \wedge \neg H.\end{aligned}$$

Step 3: Distibute over $\wedge$. That is, given $G \wedge (H \vee K)$ replace it with $(G \wedge H) \vee (G \wedge K)$.

If a formula F is in CNF and every disjunction has at most one positive literal, that is, all literals but one are negated, we say that F is a Horn formula. Among other things, Horn formulas admits a solution to the satisfiability problem.

- 2.4.1 - Resolution.

Resolution is a theorem-proving technique based on refutation, i.e., we try to get a contradiction. To work properly, we have to write any given formula in a clause form, that is, we have to write the formula as disjunction of literals. Therefore, each disjunction in a CNF is a clause. To ease the notation we represent clauses as sets, where the clause $\{\}$ means false.

Example

Consider $F = (\neg p \vee q) \wedge (q \vee r)$, with our notation we write F as the set of sets $\{\{\neg p, q\}, \{q, r\}\}$.

But how do we proceed? Essentially, given a formula in clause form we repeatedly apply the *resolution rule*; i.e., given two clauses $C_1 \cup \{A\}$ and $C_2 \cup \{\neg A\}$ we get $C_1 \cup C_2$. That is, we look for pairs of clauses with two opposing literals and we merge them. We proceed this way until either we have no more clauses satisfing the above, or we get the empty clause, in which case we get the wanted contradiction.

- 3 - Prepositional logic.

We now discuss the concept of Prepositional Logic which, in its most general form, is at the core of the FOL introduced in *Section 1*.

In Prepositional Logic formulas are made up of predicates, as the name suggests; which are in turns made up of n-ary relations of terms[3]. Formally speacking, a term is either a constant a, b, c, a variable x, y, z or a n-ary function of terms. In the case of Propositional Logic, formulas were made up of atomic propositions held together by connectives; similarly, formulas in Propositional Logic are held together by connectives in addiont to quantifiers $\exists$ (exists) and $\forall$ (for all), whose meaning is pretty clear. Additionally, the following equivalences hold.

$$\exists x\ (P(x) \vee Q(x)) \equiv \exists x\ P(x) \vee \exists x\ Q(x)$$
$$\forall x\ (P(x) \wedge Q(x)) \equiv \forall x\ P(x) \wedge \forall x\ Q(x)$$

Let F be a formula of the form $\forall x P(x)$, where $P(x)$ is some preposition; we say that x is a bound variable, i.e., x is within the scope of a quantifier. Viceversa, we say that a variable x is free, or unbounded, if no quantifier is associated to it.

For what regards the complexity: in [2] *Gödel* proves that FOL is not decidable, while in [1] *Church* proves that it's semi-decidable.

We now introduce some definitions needed to undestand the remainder of this section. Let F be a formula, we say that F is *closed* if it contains no free variable, we say it's *open* otherwise. A close formula is also called a *sentence*. For any given open formula one could consider the *universal closure* or the *existential closure*; in both case a quantifier is assigned to each free variable.

- 3.1 - Semantic interpretation in FOL.

So far we have discussed the syntactic aspect of FOL, but in doing so we made a semantic interpretation of the terms. To clearly untie semantic and syntax we have to introduce the concepts of *functional symbols* and *predicative symbols*, instead of functions and predicates respectively.

To define the semantic of a given formula we must fix a domain D, an interpretation function $I : D^n \to D$ and an interpretation of th thee predicates. We define this way an interpretative structure. Similarly for what discussed about Propositional Logic, given close formula F, we call *model* for F, a interpretative structure that makes F true. Again, if F admits at least a model, we say that F is satisfiable, otherwise is unsatisfiable. We say that A is valid, if it's true for any interpretative structure.

- 3.2 - Resolution in FOL.

In this section we show how the resolution algorithm presented in *Section 2.4.1* can be extended to formulas in FOL. As done in the case of Propositional Logic, we need to convert each formula into its clause form. But as we are about to show, this is not so straightforward.

The procedure we are about to show is not unique; apart from the last two steps, all other steps can be performed in any order. Since in general we deal with long formulas, we suggest as first step to remove any material condition, i.e., implications. Then move each negation inward, that is, replace each

$$\neg\forall x\ P(x) \text{ with } \exists x\ \neg P(x)$$

and

$$\neg\exists x\ P(x) \text{ with } \forall x\ \neg P(x).$$

As third step, if needed, to avoid confusion, standardize variable apart, meaning that, if the same variable is bounded to two, or more, quantifiers, one should introduce a new variables. For instance, take

$$\forall x\ \neg Q(x,z) \vee \exists x\ P(x),$$

here the second x can be replaced by the variable y to make more clear the distinction of the two parts; hence the formula becomes

$$\forall x\ \neg Q(x,z) \vee \exists y\ P(y).$$

As next step we introduce the prenex normal form (PNF), that is, a formula in the form

$$Q_1x_1\ Q_2x_2\ \cdots\ Q_nx_n\ P$$

where each Q_i is a quantifier and P is without quantifiers. For instance: $\exists x\ \neg P(x) \vee \exists y\ Q(y)$ becomes $\exists x\ \exists y\ \neg P(x) \vee Q(y)$. We then transform the formula in PNF to its skolem normal form (SNF), i.e., we remove all existential quantifiers. More specifically, each quantifiers of the form $\exists y$ is replaced by some functional symbol $f(x_1, \dots, x_n)$ where each x_i is a universal quantified variable preceding y, or a constant if none. Lastly, we remove all universal quantifiers and assume the formula as universally quantified.

Note that formulas in SNF are not equivalent to the original ones; despite this, thaks to the following theorem and the fact that the resolution algorithm is refutation-complete, the problem does not exitst.

Theorem (Skolem's theorem) *A formula is unsatisfiable if and only if the SNF of its universal closure is unsatisfiable.*

Once the formula is in SNF the conversion to CNF and clause form is straightforward.

- 3.2.1 - The resolution rule.

In the case of Propositional Logic we show how to use the resolution rule to derive the empty clause. The same principle can be extended to Prepositional Logic with a caveat; since prepositions are made up of terms, wich in turns, could be made up of other terms, i.e., they are n-dimensional functions of terms, we need a way to be sure that the prepositions we are working with refer to the same terms. To achive such requirement we use *substitution* and *unification*, described in the following. Let us note that, if both substitution and unification are used, the system can be proved to be both sound and refutation-complete.

Let $P(x, y)$ and $Q(w, z)$ be two prepositions. A substitution is an equality of the form $x = f(x)$ such that the resuting prepositions look alike; that is, we replace each occurence of a given variable with another value, in such a way that after the substitution the prepositions look alike. For instance, in the case of the prepositions P, Q above, we can make the substitutions $[x/w, y/z]$, which reads as x replaces w and y replaces z, and get $P(x, y)$ and $Q(x, y)$. In the above case, if we assume that $Q = \neg P$ and that both are clauses, we could immediately derive the empty clause.

Let C_1 and C_2 be clauses. We use unification to determine when two clauses are equivalent for variable substitutions, and can be therefore resolved. For instance, let

$$C_1 = \{P(x, z), \neg Q(y, z)\} \text{ and } C_2 = \{R(x, y, z), Q(f(x), z)\}.$$

If we consider C_1 and C_2 with no substitution, we cannot find any resolvent. On the other end, if we consider the substitution $[f(x)/y]$ we get the clauses

$$C_1 = \{P(x, y), \neg Q(f(x), z)\} \text{ and } C_2 = \{R(x, f(x), z), Q(f(x), z)\}$$

for which a resolvent, namely $\{P(x, y), R(x, f(x), z)\}$, does exist. For our purpose we we indicate substitution by σ and by $\mathrm{dom}(\sigma)$ the set of variable substituted. Is somewhat obvious that substitutions can be composed and extended to literals. Given σ a substitution of terms, we say that σ is a unifier if it makes the terms syntactically identical. Lastly, to have achive completeness we need *factoring*, that is, we apply unification to literals within the same clause.

Once all of this is applied to all clauses, our Logic can be proved to be refutation-complete.

[3] Think of a term as a name or a definite description, eg. “Socrates”, “Alice’s pen”.

References.

[1] A. Church. “A note on the Entscheidungsproblem”. In: *Journal of Symbolic Logic* 1.1 (1936), pp. 40–41. DOI: 10.2307/2269326.

[2] K. Gödel. “Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.” In: *Monatsh. f. Mathematik und Physik* 38 (1931), pp. 173–198. DOI: 10.1007/BF01700692.

Acronyms.

CNF conjunctive normal form. 6

FOL First Order Logic. 1

KRR Knowledge Representation and Reasoning. 1

PNF prenex normal form. 9

SNF skolem normal form. 9