Validity of Formulas in First-order Logic is Undecidable

Prakash Panangaden

8$^{\text{th}}$ April 2021

First-order logic is a formal language for capturing the notion of quantified formulas. I assume that you are familiar with it but I will quickly describe it and give its semantics. The syntax is defined as follows. First we define the ingredients of the language.

- There is a countable set of symbols called *variables*: $x, y, z, \ldots$.
- There is a set of symbols called *constants*: these vary from theory to theory. For example they may be the natural numbers: $0, 1, 2, 3, \ldots$ or arbitrary symbols: $a, b, c, \ldots$.
- There is a finite set of *function symbols*: $f, g, h, \ldots$. Associated with each function symbol is a positive integer called its *arity*, this represents the number of arguments that it takes.
- There is a finite set of *predicate symbols*: $P, Q, R, \ldots$. As with function symbols, each predicate symbol has an arity.

Next we define a set of *terms*, these denote “things”.

1. A constant is a term.
2. A variable is a term.
3. If f is a function symbol of arity n and $t_1, \ldots, t_n$ are terms then $f(t_1, \ldots, t_n)$ is a term.

A term with no variables is called a *ground* term.

Next we define *formulas*.

1. If t_1, t_2 are terms then $t_1 = t_2$ is a formula.

2. If P is a predicate symbol of arity n and $t_1, \ldots, t_n$ are terms then $P(t_1, \ldots, t_n)$ is a formula.

3. If ϕ is a formula then $\neg\phi$ is a formula.

4. If ϕ and ψ are formulas then $\phi \wedge \psi$, $\phi \vee \psi$ and $\phi \Rightarrow \psi$ are formulas.

5. If ϕ is a formula and x is a variable then $\forall x\ \phi$ and $\exists x\ \phi$ are formulas.

A formula as such does not mean anything; we cannot say if a formula is true or false if we do not know what it means. We have to define its meaning through an *interpretation*. In order to start we assume that we have a set D that is the collection of things that we are talking about. An interpretation $\mathscr{I}$ consists of: (i) for each constant symbol a an element d_a of D, (ii) for each function symbol f of arity n a function $f_D : D^n \longrightarrow D$ and (iii) for each predicate symbol P of arity n a subset P_D of D^n. Here D^n stands for the n-fold cartesian product. A *valuation* is a map $\rho : Var \longrightarrow D$ where Var is the collection of variables. We will write $\rho[x \mapsto d]$ for the valuation that is exactly the same as ρ except for x which is mapped to d. Given a valuation we can define what it means for a formula to be "true".

We write $\mathscr{I}, \rho \models \phi$ to mean that the formula ϕ is true with the given interpretation and valuation. The notion of truth is defined inductively. First we define the interpretation of a term inductively. We write $[\![t]\!]$ for the interpretation of a term. We assume a fixed interpretation $\mathscr{I}$ throughout and, for the interpretation of terms we assume a fixed valuation ρ.

1. $[\![a]\!] = d_a$, where d_a is the interpretation of a according to $\mathscr{I}$,

2. $[\![x]\!] = \rho(x)$

3. $[\![f(t_1, \ldots, t_n)]\!] = f_D([\![t_1]\!], \ldots, [\![t_n]\!])$, where f_D is the interpretation of f according to the interpretation $\mathscr{I}$, the function *symbol* f has the arity n.

We can now define whether a formula is true in an interpretation with a valuation: we write $\mathscr{I}, \rho \models \phi$ if the formula ϕ is true with the interpretation $\mathscr{I}$ and the valuation ρ.

1. $\mathscr{I}, \rho \models t_1 = t_2$ if $[\![t_1]\!] = [\![t_2]\!]$

2. $\mathscr{I}, \rho \models P(t_1, \ldots, t_n)$ if $([\![t_1]\!], \ldots, [\![t_n]\!]) \in P_D$; where $P_D \subseteq D^n$ is the interpretation according to $\mathscr{I}$ of the predicate symbol P which has arity n.

3. $\mathscr{I}, \rho \models \phi \wedge \psi$ if $\mathscr{I}, \rho \models \phi$ and $\mathscr{I}, \rho \models \psi$

4. $\mathscr{I}, \rho \models \neg\phi$ if $\mathscr{I}, \rho \not\models \phi$
5. $\mathscr{I}, \rho \models \exists x\ \phi$ if there is some element d of D such that $\mathscr{I}, \rho[x \mapsto d] \models \phi$
6. $\mathscr{I}, \rho \models \forall x\ \phi$ if for every $d \in D$ $\mathscr{I}, \rho[x \mapsto d] \models \phi$.

Given a formula ϕ three things could happen. It could be true in every interpretation, it could true in some interpretations and false in others and it could never be true. A formula that is true in *every* interpretation is said to be *valid.* These are statements that are true because of the meaning of the logical symbols alone. A formula that is true in some interpretation with some valuation is said to be *satisfiable*; we say that the interpretation and valuation is a *model* of the formula and that this model *satisfies* formula. The word "model" is a short way of saying "interpretation and valuation." A formula that is never true is said to be *invalid.* These are all semantic notions and have *nothing to do with axioms or provability.* We write $\models \phi$ to signify a valid formula. We also write $\Gamma \models \phi$, where Γ is a set of formulas, if every model for which all the formulas in Γ are true also satisfies ϕ. We will not talk about axioms, rules of inference and provability in this note but I assume that you are familiar with these concepts. A major theorem that links these ideas is *soundness* and *completeness.* We write $\vdash \phi$ if ϕ can be proved from the axioms and rules of logic and $\Gamma \vdash \phi$ if ϕ can be proved using the formulas in Γ as assumptions; we are not required to use all the formulas of Γ. Soundness says that $\Gamma \vdash \phi$ implies $\Gamma \models \phi$; crudely speaking "only true statements can be proved". A logical system that is not sound is worthless. Completeness says that

$$\Gamma \models \phi \Rightarrow \Gamma \vdash \phi.$$

This implies that every valid formula can be proved. It would be nice[1] if every logical system were complete; but this is not the case. First-order logic is sound and complete.

The purpose of this note is not to discuss completeness but to prove

Theorem 1. The set of valid formulas of first-order logic is computably enumerable but not decidable.

Put another way, there is no algorithm for deciding whether a formula is valid or not. There is an algorithm that searches through proofs and if a formula is valid there will be a proof of it (completeness) which the algorithm

[1] Actually the World would be a dull place, so perhaps it would not be nice.

will find. Thus we can Turing recognize[2] the set of valid formulas. However we cannot *decide* whether a formula is valid or not. If a formula is not valid at any given stage of the search for a proof we will not know whether our search algorithm will find a proof later on, or whether the formula is not valid.

We now turn to the undecidability proof. We will exhibit a mapping reduction to the Post Correspondence Problem or PCP[3] We will show that if we could solve the validity problem for FOL we could solve any instance of PCP, which we know is impossible.

The strings we consider will be over the alphabet $\{0, 1\}$. In our logical formulas we use one constant symbol, a and two unary function symbols f_0 and f_1. If we have a word $w = w_1 w_2 \dots w_k$ over our alphabet, where each w_i is 0 or 1, we write $f_w(a)$ as shorthand for $f_{w_k}(f_{w_{k-1}}(\dots f_{w_2}(f_{w_1}(a))\dots))$. For example we would write $f_{1101}(a)$ for $f_1(f_0(f_1(f_1(a))))$. *This allows us to encode words as terms.* We will also use a single binary predicate p.

Suppose that we are given an instance of PCP over the alphabet $\{0, 1\}$:

$$\{(\alpha_1, \beta_1), (\alpha_2, \beta_2), \dots, (\alpha_n, \beta_n)\}.$$

Here there are n domino types, the α_i are the strings on the top halves of the dominoes and the β_i are the strings on the bottom halves of the dominoes. Now we define a formula

$$\left\{ \bigwedge_{i=1}^{n} p(f_{\alpha_i}(a), f_{\beta_i}(a)) \quad \wedge \quad \forall x \forall y \, [p(x,y) \Rightarrow \bigwedge_{i=1}^{n} p(f_{\alpha_i}(x), f_{\beta_i}(y))] \right\} \Rightarrow \exists z \, p(z,z).$$

Before we proceed with the proof let us understand the intuition behind this formula which is hard to deconstruct at first sight. The strings that appear in the dominoes are encoded in the terms using f_0 and f_1 in the manner described above. The a is just there to start the encoding of the string. The predicate p says that the first argument and the second argument describe a pair of strings that one can obtain as the top and bottom strings of some sequence of dominoes. The first conjunct just says that if one takes *any* single domino one gets a pair that are related in this way. The second conjunct says that if one has a pair of strings that are obtained in this way

[2] Remember this means that we can tell if a formula is valid if it is, but perhaps we cannot say that it is not valid in case it is not valid.

[3] Please note that in complexity theory, the acronym PCP is used for something completely different.

then by adding *any* one domino one gets an extended pair of strings that are related. Finally the last formula outside the curly brackets says that there is a string that appears as the top and the bottom string of some sequence of dominoes. Thus this formula says that the PCP has a solution.

More formally we can prove that this formula is valid if and only if the PCP instance has a solution. This will complete the reduction of PCP to the validity problem.

First we assume that the formula is valid: this means that it is true in *every* interpretation, which means that we are free to construct a convenient interpretation. We will construct an interpretation that will establish that the PCP instance has a solution. The domain of the interpretation is $\{0,1\}^*$ and the constant a will denote ε, the empty string. The function symbol f_0 denotes the function that adds the symbol 0 to the end of any word, $f_0(w) = w0$ and $f_1(w) = w1$. The predicate symbol p is interpreted as the set of pairs of words of the form

$$(\alpha_{i_1}\alpha_{i_2}\alpha_{i_3}\ldots\alpha_{i_k}, \beta_{i_1}\beta_{i_2}\beta_{i_3}\ldots\beta_{i_k})$$

for some sequence of integers $i_1, i_2, \ldots, i_k$. This interpretation formalises what we have been saying informally in the preceding paragraph. Now since the formula ϕ is assumed to be valid it is true in this specific interpretation. The antecedents are clearly true. The first conjunct just describes the Post system and and the second conjunct is just the inductive statement of what it means to build pairs of strings with dominoes so the consequent $\exists z\ p(z,z)$ must be true, but this says precisely that the Post system has a solution.

For the reverse direction we assume that the Post system has a solution; in other words there is a sequence of integers $i_1, i_2, \ldots, i_k$ such that

$$\alpha_{i_1}\alpha_{i_2}\ldots\alpha_{i_k} = \beta_{i_1}\beta_{i_2}\ldots\beta_{i_k} = z;$$

we are naming this z in anticipation of the argument below. Now we must show that the formula ϕ is true in *every* interpretation. To show that ϕ is true means that we assume the formula in curly brackets and show that the formula $\exists z\ p(z,z)$ must follow. Now it is a simple induction on the length of sequences of integers that if we have *any* sequence of integers in the range $\{1,\ldots,n\}$, say $j_1, j_2, \ldots, j_m$ that

$$p(\alpha_{j_1}\alpha_{j_2}\ldots\alpha_{j_m}, \beta_{j_1}\beta_{j_2}\ldots\beta_{j_m}).$$

The first conjunct in the curly brackets asserts the base case and the second conjunct is precisely the inductive step. Thus, it applies to the specific sequence that arises from the solution to the Post system, in short we have $\exists z\, p(z, z)$. We have shown that the formula is valid.

The consequences are immense. There is no algorithm to check whether formulas in first-order logic are valid. We can decide validity of propositional logic using truth tables (and as far as we know there is no better way) but there is no such method for first-order predicate calculus. There are no *fully automatic* theorem-proving procedures for even a wimpy logic like first-order predicate calculus. You cannot even express induction in first-order logic. There is no hope at all for arithmetic.

For second-order predicate calculus the set of valid formulas is not even CE or co-CE. Interestingly if we allow only *unary* predicates we regain decidability but that is another story.

What can we do? An important development is *computer-assisted* proofs. Here the possibilities are unlimited. A number of major theorems have been proved this way recently. Humans and computers working together are a very powerful combination.