CS 4269/5469– Fundamentals of Logic In Computer Science

SEMESTER II, 2022-2023

Overall Notes

---

1 Motivating the Study of Logic

Professor Mathur's research interest is in the space of formal program verification. Formal program verification is the problem of determining if a program P meets a specification Φ; that is, $P \models \Phi$. This is a more complete way of verifying the correctness of programs compared to software engineering-style testing, but requires a background in mathematical logic.

Another motivation is in the realm of databases. We can model a query as a first-order logic formula, such as the formula $\phi \equiv \mathsf{Friends}(p_1, p_2)$ where the interpretation of $\mathsf{Friends}$ is $I(\mathsf{Friends}) = \{(p_1, p_2), (p_2, p_3), \ldots\}$ and the universe U is the set of all persons.

Yet another motivation is in the study of complexity theory, which discusses whether polynomial-time algorithms exist to solve a problem and what the best algorithm to solve a problem is. If we were able to encode a computational problem as a logic formulae, then determining whether the problem is solvable in polynomial time could be equivalent to determining the satisfiability of the formula.

---

2 Propositional Logic

Lemma 1 (Relevance Lemma)**.** *Let $\phi \in \mathsf{FORM}$ be a well-formed formula and v_1 and v_2 be valuations. If for every $p \in \mathsf{occurs}(\phi)$, $v_1(p) = v_2(p)$, then $v_1 \models \phi$ iff $v_2 \models \phi$.*

Proof sketch. Use structural induction on ϕ. □

We will establish some definitions and results to prove the Compactnes theorem. Assume that Γ is a set of propositional logic formulae over $(\mathcal{P}, \mathcal{C})$.

Definition 1 (Satisfiability)**.** A set Γ of propositional logic formulae over $(\mathcal{P}, \mathcal{C})$ is *satisfiable* if there is a valuation v such that $v \models \Gamma$.

Definition 2 (Finite Satisfiability)**.** A set Γ of propositional logic formulae over $(\mathcal{P}, \mathcal{C})$ is *finitely satisfiable* if every finite subset of Γ is satisfiable.

Definition 3 ($\mathcal{P}$-completeness)**.** Γ is *$\mathcal{P}$-complete* if for every proposition $p \in \mathcal{P}$, either $p \in \Gamma$ or $\neg p \in \Gamma$.

Definition 4 ($\mathcal{P}$-consistency)**.** Γ is *$\mathcal{P}$-consistent* if for every proposition $p \in \mathcal{P}$, either $p \notin \Gamma$ or $\neg p \notin \Gamma$.

Lemma 2. *If Γ is finitely satisfiable, then it is also $\mathcal{P}$-consistent.*

Lemma 3. *If Γ is finitely satisfiable and Γ is $\mathcal{P} - complete$, then Γ is satisfiable.*

Lemma 4. *If Γ is finitely satisfiable, then one of $\Gamma \cup \{\varphi\}$ or $\Gamma \cup \{\neg\varphi\}$ is finitely satisfiable.*

Theorem 1 (Compactness of Propositional Logic)**.** Let Γ be a set of propositional logic formulae over $(\mathcal{P}, \mathcal{C})$. Γ is satisfiable iff Γ is finitely satisfiable.

Proof sketch. Inductively define a set that is a superset of Γ and for every proposition p, either p or $\neg p$ is in the set depending on which makes the i^{th} set finitely satisfiable. Prove that it is finitely satisfiable. □

2.1 Modelling Computational Problems with Propositional Logic

Claim 1. *For any graph $G = (V, E)$, there is a set of formulae $\Gamma_{G,k}$ such that G is k-*colorable *iff $\Gamma_{G,k}$ is satisfiable.*

Claim 2. *For any graph $G = (V, E)$, there is a set of formulae $\Gamma_{G,k}$ such that G is has a vertex cover of size at most k iff $\Gamma_{G,k}$ is satisfiable.*

2.2 Interpolation Theorem

1. Add the 0-place connectives $\top$, $\bot$ to our language. For each wff φ and sentence symbol A, let $\varphi^A_\top$ be the wff obtained from φ by replacing A by $\top$. Similarly for $\varphi^A_\bot$. Then let $\varphi^A_* = (\varphi^A_\top \vee \varphi^A_\bot)$. Prove the following:

   (a) $\varphi \vDash \varphi^A_*$.

   (b) If $\varphi \vDash \psi$ and A does not appear in ψ, then $\varphi^A_* \vDash \psi$.

   (c) The formula φ is satisfiable iff φ^A_* is satisfiable.

   Remarks: We can think of φ^A_* as trying to say everything φ says, but without being able to use the symbol A. Parts (a) and (b) state that φ^A_* is the strongest A-free consequence of φ. The formulas φ and φ^A_* are not tautologically equivalent in general, but they are "equally satisfiable" by part (c). The operation of forming φ^A_* from φ is (in another context) called *resolution* on A.

2. (Interpolation theorem) If $\alpha \vDash \beta$, then there is some γ all of whose sentence symbols occur both in α and in β and such that $\alpha \vDash \gamma \vDash \beta$. *Suggestion*: Use the preceding exercise.

---

3 Proof Systems for Propositional Logic

A proof system provides a mechanical procedure for establishing a logical inference. We will look at two different types of proof systems for porpositional logic: the *Hilbert-style/Frege* proof system and the *Resolution* proof system.

3.1 Hilbert-style/Frege Proof System

The *Hilbert-style* proof system (also known as *Frege* proof system) allows one to establish new valid formulae. It assumes that formulae are written using just implication and $\bot$. One can prove that all propositional logic formulae can be expressed as a combination of those two operators.

The Hilbert-style proof system comprises 3 *axiom schemas* and one inference rule called *modus ponens*.

Definition 5 (Axiom Schemas)**.**

$$\frac{}{\varphi \to (\psi \to \varphi)} \quad \text{(Axiom Schema 1)}$$

$$\frac{}{(\varphi \to (\psi \to \rho)) \to ((\varphi \to \psi) \to (\varphi \to \rho))} \quad \text{(Axiom Schema 2)}$$

$$\frac{}{((\varphi \to \bot) \to \bot) \to \varphi} \quad \text{(Axiom Schema 3)}$$

$$\frac{\varphi \qquad \varphi \to \psi}{\psi} \quad \text{(Modus Ponens)}$$

An *instantiation* of an axiom schema is a formula obtained by substituting the variables φ, ψ, and ρ by some propositional logic formulae. For example, the instantiation of Axiom Schema 1 with $\varphi = (p \to q)$, $\psi = q$, and $\rho = r$ is $(p \to q) \to (q \to (p \to q))$.

Proofs in Hilbert-style proof systems are similar to proofs in usual mathematics.

Definition 6 (Proof in Hilbert-style Proof System)**.** A *proof* of a formula φ, starting from a set of formulae Γ (also known as *assumptions* or *hypotheses*) is a finite sequence $\pi = \phi_1, \phi_2, \ldots, \phi_n$ such that $\phi_n = \varphi$, and for every $i \in \{1, 2, \ldots, n\}$, either

- ϕ_i is an assumption (i.e. $\phi_i \in \Gamma$), or
- ϕ_i is an instantiation of an axiom schema, or
- ϕ_i follows from previous formulae ϕ_j and ϕ_k, with $j, k < i$, by modus ponens

If there is a proof of φ from Γ, we write $\Gamma \vdash \varphi$. In particular, when $\Gamma = \varnothing$, we write $\vdash \varphi$.

In general, there is no semantic meaning behind these axioms. That said, we can attempt to give an intuitive meaning to them:

- Axiom Schema 1: if we have proven φ, we can prove that anything else implies φ.
- Axiom Schema 2: if we have proven that φ and ψ implies ρ, we can prove that if φ implies ψ, then φ also implies ρ. This is similar to *transitivity*.
- Axiom Schema 3: if we have proven that $\neg\varphi$ leads to a contradiction, then we can prove φ.

From here on, we will use Γ to denote a set of assumptions (i.e. propositional logic formulae), and φ and ψ to denote a some propositional logic formulae.

Derivations are tedious and typically difficult to find, but the following theorem makes it easier to show that certain derivations exist. Before present it, we first state a simple observation.

Claim 3. *If* $\Gamma \vdash \varphi$ *and* $\Gamma \subseteq \Gamma'$*, then* $\Gamma' \vdash \varphi$.

Proof sketch. There is a proof $\pi = \phi_1, \phi_2, \ldots, \phi_n$ for $\Gamma \vdash \varphi$. It is also a proof for $\Gamma' \vdash \varphi$. We can establish this by structural induction on each $\phi_1, \ldots, \phi_i$. □

Now, we present the *Deduction Theorem* which makes it easier to come up with new derivations.

Theorem 2 (Deduction Theorem)**.** Let Γ be a set of assumptions. $\Gamma \vdash (\varphi \to \psi)$ iff $\Gamma \cup \{\varphi\} \vdash \psi$.

Proof sketch. We first prove the easier forward direction. Consider the proof π of $\Gamma \vdash (\varphi \to \psi)$. π is also a proof of $\Gamma \cup \{\varphi\} \vdash (\varphi \to \psi)$ by Claim 3. Then, $\pi \cdot \varphi \cdot \psi$ is a proof of $\Gamma \cup \{\varphi\} \vdash \psi$ by modus ponens.

Next, we prove the converse, which is more difficult. Consider a proof $\pi = \psi_1, \psi_2, \ldots, \psi_n$ of ψ from $\Gamma \cup \{\varphi\}$. We can inductively show that for every $1 \leq i \leq n$, $\Gamma \vdash (\varphi \to \psi_i)$. □

Next, we state some syntactic entailments. The proof of each entailment is left as an exercise.

- $\vdash \varphi \to \varphi$.
- $\vdash ((\varphi \to \bot) \to \varphi) \to \varphi$.
- $\{\bot\} \vdash \varphi$.
- $\vdash (\varphi \to \bot) \to (\varphi \to \rho)$.

3.1.1 Soundness and Completeness

Two important properties of a proof system are *soundness* and *completeness*. A proof system is *sound* if every valid formula is provable. A proof system is *complete* if every provable formula is valid. The formal definitions are as follows.

Definition 7 (Soundness)**.** A proof system is *sound* if $\Gamma \vdash \varphi$ implies $\Gamma \models \varphi$.

Definition 8 (Completeness)**.** A proof system is *complete* if $\Gamma \models \varphi$ implies $\Gamma \vdash \varphi$.

Of course, the Hilbert-style proof system is sound and complete. It is typically easier to prove soundness, so we first prove that it is sound.

Theorem 3 (Soundness of Hilbert-style Proof System)**.** Let Γ be a set of assumptions and φ be a formula. If $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$.

Proof sketch. Let $\pi = \psi_1, \psi_2, \ldots, \psi_n$ be a proof of φ from Γ. We can prove that $\Gamma \models \varphi$ by induction on $i \in [1, n]$. For the base case, we have to prove that $\Gamma \models \psi_1$ when either $\psi_1 \in \Gamma$ or when ψ_1 is an axiom instance. For the inductive step, we have to prove that $\Gamma \models \psi_i$ either when $\psi_i \in \Gamma$ or when ψ_i is an axiom instance or when ψ_i follows from modus ponens. □

Next, we want to prove that the Hilbert-style proof system is complete. However, to establish this difficult theorem, we first establish some intermediate results.

Definition 9 (Consistent and Inconsistent Sets)**.** A set of formulae Γ is *inconsistent* if there is some formula φ such that $\Gamma \vdash \varphi$ and $\Gamma \vdash (\varphi \to \bot)$. Γ is *consistent* if it is not inconsistent.

Claim 4. *Γ is inconsistent iff for every formula ρ, $\Gamma \vdash \rho$.*

Proof sketch. For the forward direction, use Modus Ponens with the proofs of φ and $(\varphi \to \bot)$ to get a proof of $\bot$, then use one of the previous syntactic entailments to get a proof of ρ.

For the other direction, simply use the definition of inconsistency. □

Now, we establish some properties about inconsistent sets.

Lemma 5. *If $\Gamma \cup \{\varphi \to \bot\}$ is inconsistent, then $\Gamma \vdash \varphi$.*

Proof sketch. Use a combination of the definition of inconsistency, the Deduction theorem, one of the syntactic entailments mentioned earlier, Claim 3, and Modus Ponens. □

Lemma 6. *If $\Gamma \cup \{\varphi\}$ is inconsistent, then $\Gamma \vdash (\varphi \to \bot)$.*

Proof sketch. Use a combination of the definition of inconsistency and the Deduction theorem. □

Corollary 1. *If* Γ *is consistent, then at least one of* $\Gamma \cup \{\varphi\}$ *or* $\Gamma \cup \{\varphi \to \bot\}$ *is consistent.*

Claim 5. *Suppose* $\Gamma \subseteq \Gamma'$. *If* Γ' *is consistent, then so is* Γ.

Lemma 7. *If* Γ *is inconsistent, then there is a finite subset* $\Gamma' \subseteq_{fin} \Gamma$ *of* Γ *such that* Γ' *is also inconsistent.*

Proof sketch. Consider the proofs π and π' of $\Gamma \vdash \varphi$ and $\Gamma \vdash (\varphi \to \bot)$ respectively. Proofs are finite sequences of formulae. Construct Γ' from the set of assumptions in π and π'. Then, prove inductively that π and π' are proofs of $\Gamma' \vdash \varphi$ and $\Gamma' \vdash (\varphi \to \bot)$ respectively. □

Definition 10. A set of formulae Γ is *negation-complete* if for every formula φ, $\Gamma \vdash \varphi$ iff $\Gamma \vdash (\varphi \to \bot)$.

Now, we prove a *weaker* version of completeness that talks about one specific set of assumptions, the empty set $\varnothing$.

Theorem 4 (Weak Completeness of Hilbert-style Proof System)**.** Let φ be a formula. If $\models \varphi$, then $\vdash \varphi$.

Proof sketch. We can prove the contrapositive; that is, if $\nvdash \varphi$ then $\not\models \varphi$. Assume that there is no proof for φ, then show that there is some valuation v such that $v \models (\varphi \to \bot)$. To do so, we inductively define a set of formulae Δ and prove, with induction, that it is consistent and negation-complete. With this, we can construct the natural valuation v and prove that $v \models (\varphi \to \bot)$. □

Now, we can prove the original (stronger) version of completeness.

Theorem 5 (Completeness of Hilbert-style Proof System)**.** Let Γ be a set of assumptions and φ be a formula. If $\Gamma \models \varphi$, then $\Gamma \vdash \varphi$.

Proof sketch. Use the Compactness theorem, Theorem 4, and repeated use of the Deduction theorem. □

3.2 Resolution Proof System

Notice that the proofs in the Hilbert-style proof system are mechanical and symbolic — proofs are constructed simply based on the patterns of formulae. However, mechanization of proofs in that proof system is difficult as there are infinitely many choices for each step of the proof. In contrast, the *Resolution proof system* has a constrained number of choices at each step of the proof, making it amenable to mechanization.

The Resolution proof system is used to establish unsatisfiability of a set of clauses. Thus, unlike the Hilbert-style proof system which proves the validity of a formula directly, the Resolution proof system proves that the negation of the formula is unsatisfiable.

Definition 11 (Literals and Clauses)**.** A *literal* ℓ is either a proposition or its negation. That is, $\ell = p$ or $\ell = \neg p$ for some proposition $p \in \mathcal{P}$.

A *clause* is a disjunction of literals, represented as a finite set $C = \{\ell_1, \ell_2, \ldots, \ell_k\}$. Note that clauses could contain both p and $\neg p$.

For proofs in the Resolution proof system, we will work only with formulae in *conjunctive normal form* (CNF). This is fine as every propositional logic formula has a logically equivalent formula in CNF.

Definition 12 (Conjunctive Normal Form)**.** A formula is in *Conjunctive Normal Form* (CNF) if it is a conjunction of clauses. That is, $\varphi = \bigwedge_{i=1}^{n} C_i$ where each $C_i = \bigvee_{j=1}^{k_i} \phi_{ij}$ is a clause and ϕ_{ij} is a literal. We think of a CNF formula as a set of clauses.

Definition 13 (Satisfiability)**.** A clause C (in our set form) is *satisfiable* if there is a valuation v such that $v \models \ell$ for some literal $\ell \in C$.

A formula in CNF is satisfiable if there is a valuation v that satisfies every clause in the formula.

Remark. *The empty clause is unsatisfiable.*

The only rule in the Resolution proof system is as follows.

Definition 14 (Resolvent)**.** Given two clauses $C' = C \uplus \{p\}$ and $D' = D \uplus \{\neg p\}$, the *resolvent* of C' and D' with respect to proposition p is the clause $C \cup D$.

Definition 15 (Refutation/Resolution Proof)**.** A *refutation* (i.e. resolution proof) of a set of clauses Γ is a finite sequence of clauses $\pi = C_1, C_2, \ldots, C_n$ such that $C_n = \{\}$ and each clause C_i is either in Γ or a resolvent of two clauses C_j and C_k $(j, k < i)$.

Now, we begin to prove the soundness and completeness of the Resolution proof system.

Theorem 6 (Soundness of Resolution)**.** Let Γ be a set of clauses over propositions $\mathcal{P}$. If there is a resolution proof of Γ, then Γ is unsatisfiable.

Proof sketch. Consider a resolution proof $C_1, C_2, \ldots, C_n$ of Γ. Prove, by induction, that if a valuation v is such that $v \models \Gamma$, then $v \models C_i$. □

Before we begin to prove the completeness of the Resolution proof system, we first establish some results.

Claim 6. *Let Γ be an unsatisfiable set of clauses. Show that, if Γ is unsatisfiable, then there is a finite set $\Gamma_0 \subseteq_{fin} \Gamma$ such that Γ_0 is unsatisfiable.*

Proof sketch. Convert Γ to a set of formulae (from a set of set of literals) using the natural interpretation. Then, invoke the Compactness theorem. □

Claim 7. *Let Γ_1 and Γ_2 be sets of clauses such that $\Gamma_1 \subseteq \Gamma_2$. If there is a resolution proof of Γ_1, then there is a resolution proof of Γ_2.*

Theorem 7 (Completeness of Resolution)**.** Let Γ be a set of clauses (over propositions $\mathcal{P}$). If Γ is unsatisfiable, then there is a resolution proof of Γ.

3.3 Compactness vs Completeness

The Compactness theorem played a crucial role in establishing the completeness of the aforementioned proof systems. It turns out that the soundness and completeness of these proof systems can be used to establish the Compactness theorem.

Claim 8. *Given that the Hilbert-style proof system is sound and complete, the Compactness theorem holds.*

Claim 9. *Given that the Resolution proof system is sound and complete, the Compactness theorem holds.*

Proof sketch. One direction of the Compactness theorem is straightforward. For the other direction, consider a proof π in either of the proof systems for a set of formulae Γ. Construct a set of formulae $\Gamma_0 \subseteq_{\text{fin}} \Gamma$ containing the formulae that are in Γ and in π. By soundness and/or completeness, show that Γ_0 is satisfiable/unsatisfiable. □

---

4 First-Order Logic

First-order Logic (FOL) is a formal language to describe and reason about *predicates* rather than propositions. A predicate is a proposition that depends on the value of some variables. To do this, first-order logic builds upon propositional logic with functions, variables, and quantification.

4.1 Motivation behind FOL

First-order logic grew out of the desire to study the foundations of mathematics in number theory and set theory. To illustrate the need for FOL, recall that we can represent the following statements as propositions in propositional logic:

$$s = \text{"Greeks are humans."}$$
$$r = \text{"Humans are mortals."}$$
$$p = \text{"Greeks are mortals."}$$

However, the limited expressiveness of propositional logic prevents us from reasoning about the elements in a universe. Thus, propositional logic cannot properly encode the following statements:

$s_1 =$ "If a person is a Greek, then he is a human." or $\forall x.\, G(x) \to H(x)$.

$s_2 =$ "There is a Greek." or $\exists x.\, G(x)$.

$s_3 =$ "There is a human." or $\exists x.\, H(x)$.

Here, G and H are *predicates* where $G(x)$ means x is a Greek and $H(x)$ means x is a human.

4.2 Syntax of FOL

We start by defining the syntax of first-order logic. First-order logic formulae are defined over a signature that identifies non-logical symbols, namely, the predicates, constants, and functions that can be used in formulae.

Definition 16 (Signature)**.** A *signature* or *vocabulary* is $\Sigma = (\mathcal{C}, \mathcal{F}, \mathcal{R})$ where

1. $\mathcal{C} = \{c_1, c_2, \ldots\}$ is a set of *constant symbols*.
2. $\mathcal{F} = \{f_1, f_2, \ldots\}$ is a set of *function symbols*. Each $f \in \mathcal{F}$ has an associated arity, denoted $\mathsf{arity}(f) \in \mathbb{N}_{\geq 0}$.
3. $\mathcal{R} = \{R_1, R_2, \ldots\}$ is a set of *relation symbols*. Each $R \in \mathcal{R}$ has an associated arity, denoted $\mathsf{arity}(R) \in \mathbb{N}_{>0}$.

Besides the signature, we also need a set $\mathcal{V} = \{x_1, x_2, \ldots\}$ of variables. We typically consider signatures Σ and variables $\mathcal{V}$ that are countable. Lastly, we also inherit the propositional connectives $\vee$ and $\neg$.

Now, we can define the set of *terms* in first-order logic.

Definition 17 (Terms)**.** A *terms* over a signature $\Sigma = (\mathcal{C}, \mathcal{F}, \mathcal{R})$ and variables $\mathcal{V}$ is given by the following BNF grammar:

$$t := x \mid c \mid f(t, t, \ldots, t)$$

where $x \in \mathcal{V}$, $c \in \mathcal{C}$, and $f \in \mathcal{F}$. The number of terms in a function f is determined by $\mathsf{arity}(f)$.

Having defined terms, we can use them to define well-formed formulae (wff) or formulae for short.

Definition 18 (Formulae)**.** A *well-formed formula (wff)* over a signature $\Sigma = (\mathcal{C}, \mathcal{F}, \mathcal{R})$ and variables $\mathcal{V}$ is given by the following BNF grammar:

$$\varphi := t = t \mid R(t, t, \ldots, t) \mid (\neg\varphi) \mid (\varphi \vee \varphi) \mid (\exists x.\, \varphi)$$

Here, t is a term, $x \in \mathcal{V}$ is a variable, and $R \in \mathcal{R}$ is a relation. The number of terms in a relation R is determined by $\mathsf{arity}(R)$.

We will additionally use the derived operators "$\wedge$", "$\rightarrow$" and the derived *universal* quantifier "$\forall$" which are obtained as follows:

1. $\varphi_1 \wedge \varphi_2 \equiv \neg((\neg\varphi_1) \vee (\neg\varphi_2))$
2. $\varphi_1 \rightarrow \varphi_2 \equiv (\neg\varphi_1) \vee \varphi_2$
3. $\forall x.\, \varphi \equiv \neg(\exists x.\, (\neg\varphi))$

From here on, when we say *formulae*, we really mean *well-formed formulae* in first-order logic. Furthermore, we will omit parantheses where the order of operations is clear. For example, we will write $\neg\varphi \vee \varphi_2$ instead of $(\neg\varphi) \vee \varphi_2$.

A formula φ is an *atomic formula* if it does not have any logical operators; that is, it is of the form $t_1 = t_2$ or $R(t_1, t_2, \ldots, t_k)$. Lastly, a *literal* is a formula that is either atomic or the negation of an atomic formula.

Example. Consider the signature $\Sigma = (\{c\}, \{f^1\}, \{<^2\})$ where f has arity 1 and $<$ has arity 2. Furthermore, suppose that $\mathcal{V} = \{x, y\}$. Then, the following are formulae:

1. $\exists x. \forall y. < (x, y) \vee x = y$
2. $\forall x. \forall y. x = y$
3. $\forall x. x = f(c)$
4. $x = f(c)$

Note that in formula 4, the variable x is not quantified. In this case, x is known as a *free variable*. On the other hand, all of the variables in formula 1 to 3 are quantified and thus, they are *bound variables*. Formulae with no free variables are known as *sentences*.

4.3 Semantics of FOL

The semantics of formulae in any logic is defined with respect to a *model*. In propositional logic, models were truth assignments to the propositions. For first-order logic, models are known as *structures* that help identify the interpretation of symbols in the signature.

Definition 19 (Structure)**.** Given a signature $\Sigma = (\mathcal{C}, \mathcal{F}, \mathcal{R})$, a *structure* $\mathcal{A}$ over Σ is a tuple (U, I) where:

- U is a non-empty set known as the *universe/domain* of the structure,
- For each constant symbol $c \in \mathcal{C}$, $I(c) \in U$ is its interpretation,
- For each function symbol $f \in \mathcal{F}$, $I(f)\colon U^{\mathsf{arity}(f)} \to U$ is its interpretation, and
- For each relation symbol $R \in \mathcal{R}$, $I(R) \subseteq U^{\mathsf{arity}(R)}$ is its interpretation.

The structure is *finite* if the universe U is finite.

Example. Consider the signature $\Sigma = (\{0\}, \{S^1\}, \{<^2\})$. Then, we have a Σ-structure $\mathcal{A} = (U, I)$ given by:

- $U = \mathbb{N}$,

- $I(0) = 0$,
- $I(s)(x) = x + 1$ for all $x \in U$, and
- $I(<) = \{(a, b) \mid a < b \text{ in the usual sense}\}$.

Now, consider the following formulae:

1. $\varphi_1 \equiv \exists x. < (0, x)$
2. $\varphi_2 \equiv < (0, x)$

As intuition leads, φ_1 is always true in $\mathcal{A}$. On the other hand, the truthiness of φ_2 depends on the *assignment* of x. This leads to following definition of assignments.

Definition 20 (Assignment)**.** For a Σ-structure $\mathcal{A}$, an *assignment* over $\mathcal{A}$ is a function $\alpha \colon \mathcal{V} \to U$ that assigns every variable $x \in \mathcal{V}$ a value $\alpha(x) \in U$.

We can then extend our choice of Σ-structure and assignment to a *valuation* function over the set of Σ-terms. A valuation gives an interpretation to the terms in the language.

Definition 21 (Valuation)**.** For a Σ-structure $\mathcal{A}$ and an assignment α over $\mathcal{A}$, a *valuation* $\mathsf{val}_{\mathcal{A},\alpha} \colon \mathsf{Terms} \to U$ is defined inductively as follows:

- For any variable $x \in \mathcal{V}$, $\mathsf{val}_{\mathcal{A},\alpha}(x) = \alpha(x)$
- For any constant $c \in \mathcal{C}$, $\mathsf{val}_{\mathcal{A},\alpha}(c) = I(c)$, and
- For any function $f^k \in \mathcal{F}$ where $k = \mathsf{arity}(f)$, we have

$$\mathsf{val}_{\mathcal{A},\alpha}(f(t_1, t_2, \ldots, t_k)) = I(f)(\mathsf{val}_{\mathcal{A},\alpha}(t_1), \mathsf{val}_{\mathcal{A},\alpha}(t_2), \ldots, \mathsf{val}_{\mathcal{A},\alpha}(t_k))$$

Before we finally define the entailment rules for formulae, we have to first define a new shorthand notation for *reassignment*. This helps us define the semantics of quantifiers.

Definition 22 (Reassignment)**.** For an assignment $\alpha \colon \mathcal{V} \to U$ over a Σ-structure $\mathcal{A} = (U, I)$, $\alpha[x \mapsto e]$ is the assignment

$$\alpha[x \mapsto u](y) = \begin{cases} \alpha(y) & \text{if } y \neq x \\ u & \text{otherwise.} \end{cases}$$

Finally, we can define the semantics of FOL formulae.

Definition 23 (Satisfaction)**.** Given a Σ-structure $\mathcal{A}$ and an assignment α, the *satisfaction* relation is a ternary relation $\models$. We write $\mathcal{A}, \alpha \models \varphi$ to mean that "φ holds in $\mathcal{A}$ under assignment α. We also write $\mathcal{A}, \alpha \not\models \varphi$ to mean that $\mathcal{A}, \alpha \models \varphi$ does not hold.

The satisfaction relation $\models$ is inductively defined as follows:

- $\mathcal{A}, \alpha \models t_1 = t_2$ iff $\mathsf{val}_{\mathcal{A},\alpha}(t_1) = \mathsf{val}_{\mathcal{A},\alpha}(t_2)$
- $\mathcal{A}, \alpha \models R(t_1, \ldots, t_k)$ iff $(\mathsf{val}_{\mathcal{A},\alpha}(t_1), \ldots, \mathsf{val}_{\mathcal{A},\alpha}(t_k)) \in I(R)$ where $k = \mathsf{arity}(R)$
- $\mathcal{A}, \alpha \models \neg\varphi$ iff $\mathcal{A}, \alpha \not\models \varphi$
- $\mathcal{A}, \alpha \models \varphi \vee \psi$ iff $\mathcal{A}, \alpha \models \varphi$ or $\mathcal{A}, \alpha \models \psi$
- $\mathcal{A}, \alpha \models \exists x.\, \varphi$ iff there exists $u \in U$ such that $\mathcal{A}, \alpha[x \mapsto u] \models \varphi$

We will now more formally define *bound* and *free* variables in a formula. We start by defining the *scope* of a quantifier.

Definition 24 (Scope)**.** Given a formula $\varphi = \exists x.\, \psi$, ψ is said to be the *scope* of the quantifier $\exists x$.

Definition 25 (Bound and Free Variables)**.** Every occurrence of the variable x in $\varphi = \exists x.\, \psi$ is called a *bound occurrence* of x in ψ. Any occurrence of x which is not bound is called a *free occurrence* of x in ψ.

The free variables in φ is denoted by $\mathsf{FVar}(\varphi)$. We go with the convention that a free variable of φ must occur in φ and thus $|\mathsf{FVar}(\varphi)|$ is bounded by the size of the formula φ.

The set of bound and free occurrences of a variable $x \in \mathcal{V}$ in a formula can be defined formally using induction on the structure of the formula, but is skipped here.

Notice that a variable may occur both bound and free. For example, in the formula $\varphi_1 \equiv R(\mathbf{x}) \to \forall x.\, T(\underline{x}, f(\underline{x}))$, the bolded variable is free but the underlined variables are bound. We can *rename* bound variables such that (a) bound variables are disjoint from free variables and (b) two bound occurrences of a variable refer to the same quantifier.

Claim 10. *For every formula φ, there is an equivalent formula ψ such that the bound and free variables of ψ are disjoint and every bound variable is in the scope of a unique quantifier.*

Example. The formula $\varphi_2 \equiv R(\mathbf{x}) \to \forall y.\, T(\underline{y}, f(\underline{y}))$ is equivalent to φ_1.

Definition 26 (Sentence)**.** A *sentence* is a formula φ that has no free variables (i.e. $\mathsf{FVar}(\varphi) = \varnothing$).

An analogous notion of the Relevance Lemma for FOL is the observation that the satisfaction of a formula depends only on the values that α assigns to the free variables of φ; the values assigned to bound variables are irrelevant. Before we prove the Relenvace Lemma for FOL, we first prove a related result on terms.

Lemma 8 (Relevance Lemma on Terms)**.** *Let t be a term and let $\mathcal{A} = (U, I)$ be a structure. If assignments α_1 and α_2 are such that $\alpha_1(x) = \alpha_2(x)$ for each variable x occurring in t, then $\mathsf{val}_{\mathcal{A},\alpha_1}(t) = \mathsf{val}_{\mathcal{A},\alpha_2}(t)$.*

Proof. We will prove by induction on the structure of t that if $\alpha_1(x) = \alpha_2(x)$ for each variable x occurring in t, then $\mathsf{val}_{\mathcal{A},\alpha_1}(t) = \mathsf{val}_{\mathcal{A},\alpha_2}(t)$.

Base Case 1. For any constant symbol $c \in \mathcal{C}$, $\mathsf{val}_{\mathcal{A},\alpha_1}(c) = I(c) = \mathsf{val}_{\mathcal{A},\alpha_2}(c)$ by definition.

Base Case 2. For any variable $x \in \mathcal{V}$, $\mathsf{val}_{\mathcal{A},\alpha_1}(x) = \alpha_1(x) = \alpha_2(x) = \mathsf{val}_{\mathcal{A},\alpha_2}(x)$ by the assumption.

Inductive Step. Assume that the inductive hypothesis holds for some terms $t_1, \ldots t_k$ for some $k \in \mathbb{N}$. Consider any function symbol $f^k \in \mathcal{F}$ where $k = \mathsf{arity}(f)$. Then, we have:

$$\begin{aligned}
\mathsf{val}_{\mathcal{A},\alpha_1}(f(t_1,\ldots,t_k)) &= I(f)(\mathsf{val}_{\mathcal{A},\alpha_1}(t_1),\ldots,\mathsf{val}_{\mathcal{A},\alpha_1}(t_k)) && \text{(by definition)}\\
&= I(f)(\mathsf{val}_{\mathcal{A},\alpha_2}(t_1),\ldots,\mathsf{val}_{\mathcal{A},\alpha_2}(t_k)) && \text{(by inductive hypothesis)}\\
&= \mathsf{val}_{\mathcal{A},\alpha_2}(f(t_1,\ldots,t_k)) && \text{(by definition)}
\end{aligned}$$

Thus, the inductive hypothesis holds for all terms t. □

Now, we can use the previous result to prove the Relevance Lemma on FOL formulae.

Lemma 9 (Relevance Lemma on Formulae)**.** *Let φ be a FOL formula and let $\mathcal{A}$ be a structure. If assignments α_1 and α_2 are such that $\alpha_1(x) = \alpha_2(x)$ for every $x \in \mathsf{FVar}(\varphi)$, then $\mathcal{A}, \alpha_1 \models \varphi$ iff $\mathcal{A}, \alpha_2 \models \varphi$.*

Proof. Without loss of generality, suppose that $\mathcal{A}, \alpha_1 \models \varphi$. We want to show that $\mathcal{A}, \alpha_2 \models \varphi$. Once we have shown this, then by swapping α_1 and α_2 in our argument, we will have also proven that if $\mathcal{A}, \alpha_2 \models \varphi$ then $\mathcal{A}, \alpha_1 \models \varphi$.

So, to proceed, we will prove by induction on the structure of φ that if $\alpha_1(x) = \alpha_2(x)$ for every $x \in \mathsf{FVar}(\varphi)$, and $\mathcal{A}, \alpha_1 \models \varphi$, then $\mathcal{A}, \alpha_2 \models \varphi$.

Base Case 1. For any atomic formula of the form $\varphi \equiv t_1 = t_2$ for some terms t_1 and t_2, we have:

$$\begin{aligned}
\mathcal{A}, \alpha_1 &\models \varphi && \text{(by assumption)}\\
\mathsf{val}_{\mathcal{A},\alpha_1}(t_1) &= \mathsf{val}_{\mathcal{A},\alpha_1}(t_2) && \text{(since } \mathcal{A}, \alpha_1 \models \varphi\text{)}\\
\mathsf{val}_{\mathcal{A},\alpha_2}(t_1) &= \mathsf{val}_{\mathcal{A},\alpha_2}(t_2) && \text{(by the previous lemma)}\\
\mathcal{A}, \alpha_2 &\models \varphi && \text{(by definition)}
\end{aligned}$$

Base Case 2. For any atomic formula of the form $\varphi \equiv R(t_1, \ldots, t_k)$ for some terms $t_1, \ldots, t_k$ where $k = \mathsf{arity}(R)$, we have:

$$\begin{aligned}
\mathcal{A}, \alpha_1 &\models \varphi && \text{(by assumption)}\\
(\mathsf{val}_{\mathcal{A},\alpha_1}(t_1),\ldots,\mathsf{val}_{\mathcal{A},\alpha_1}(t_k)) &\in I(R) && \text{(by definition)}\\
(\mathsf{val}_{\mathcal{A},\alpha_2}(t_1),\ldots,\mathsf{val}_{\mathcal{A},\alpha_2}(t_k)) &\in I(R) && \text{(by the previous lemma)}\\
\mathcal{A}, \alpha_2 &\models \varphi && \text{(by definition)}
\end{aligned}$$

Inductive Step 1. Assume that the inductive hypothesis holds for some formula ψ. Consider the formula $\varphi \equiv \neg\psi$. Then, we have:

$\mathcal{A}, \alpha_1 \models \varphi$	(by assumption)
$\mathcal{A}, \alpha_1 \not\models \psi$	(by definition)
$\mathcal{A}, \alpha_2 \not\models \psi$	(by the inductive hypothesis)
$\mathcal{A}, \alpha_2 \models \varphi$	(by definition)

Note that $\mathsf{FVar}(\varphi) = \mathsf{FVar}(\psi)$ by definition, so we can invoke the inductive hypothesis in the third step.

Inductive Step 2. Assume that the inductive hypothesis holds for some formulae ψ_1 and ψ_2. Consider the formula $\varphi \equiv \psi_1 \vee \psi_2$. Then, we have:

$\mathcal{A}, \alpha_1 \models \varphi$	(by assumption)
$\mathcal{A}, \alpha_1 \models \psi_1 \vee \psi_2$	(by definition)
$\mathcal{A}, \alpha_1 \models \psi_1$ or $\mathcal{A}, \alpha_1 \models \psi_2$	(by definition)
$\mathcal{A}, \alpha_2 \models \psi_1$ or $\mathcal{A}, \alpha_2 \models \psi_2$	(by the inductive hypothesis)
$\mathcal{A}, \alpha_2 \models \varphi$	(by definition)

Note that any free variable x in ψ_1 or ψ_2 is also free in φ. Hence, we can invoke the inductive hypothesis in the fourth step.

Inductive Step 3. Assume that the inductive hypothesis holds for some formula ψ. Consider the formula $\varphi \equiv \exists x.\, \psi$. Then, we have:

$\mathcal{A}, \alpha_1 \models \varphi$	(by assumption)
$\mathcal{A}, \alpha_1 \models \exists x.\, \psi$	(by definition)
$\exists u \in U.\ \mathcal{A}, \alpha_1[x \to u] \models \psi$	(by definition)
$\exists u \in U.\ \mathcal{A}, \alpha_2[x \to u] \models \psi$	(by the inductive hypothesis)
$\mathcal{A}, \alpha_2 \models \exists x.\, \psi$	(by definition)
$\mathcal{A}, \alpha_2 \models \varphi$	(by definition)

Note that any free variable in ψ is either free in $\exists x.\, \psi$ or bound to x. In either case, $\alpha_1[x \to u]$ and $\alpha_2[x \to u]$ agree on all free variables in ψ, and so we can invoke the inductive hypothesis in the fourth step.

Hence, we have proven that if $\mathcal{A}, \alpha_1 \models \varphi$, then $\mathcal{A}, \alpha_2 \models \varphi$. Since our argument does not assume any additional property about α_1 or α_2, by swapping α_1 and α_2 in our argument, we can conclude that if $\mathcal{A}, \alpha_2 \models \varphi$, then $\mathcal{A}, \alpha_1 \models \varphi$.

Therefore, we have proven that, assuming that $\alpha_1(x) = \alpha_2(x)$ for every $x \in \mathsf{FVar}(\varphi)$, then $\mathcal{A}, \alpha_1 \models \varphi$ iff $\mathcal{A}, \alpha_2 \models \varphi$. □

Corollary 2. *For any sentence φ and assignments α_1 and α_2, $\mathcal{A}, \alpha_1 \models \varphi$ iff $\mathcal{A}, \alpha_2 \models \varphi$.*

It follows from the Relevance Lemma that if φ is a sentence, then all variable assignments are equivalent with respect to satisfiability. Hence, for any sentence φ, we simply write $\mathcal{A} \models \varphi$ whenever $\mathcal{A}, \alpha \models \varphi$ for some assignment α.

4.4 Satisfiability and Validity of FOL Formulae

We can define the satisfiability and validity of FOL formulae similar to the way we defined them for propositional logic.

Definition 27 (Satisfiability)**.** A FOL formula φ over signature Σ is *satisfiable* if there is some structure $\mathcal{A}$ and assignment α such that $\mathcal{A}, \alpha \models \varphi$. Otherwise, φ is *unsatisfiable*.

Definition 28 (Satisfiability of a Set)**.** A set of FOL formulae Γ is *satisfiable* if there is a structure $\mathcal{A}$ and assignment α such that $\mathcal{A}, \alpha \models \varphi$ for every $\varphi \in \Gamma$. Equivalently, we write $\mathcal{A}, \alpha \models \Gamma$.

Definition 29 (Validity)**.** A FOL formula φ is said to be *valid* if for every structure $\mathcal{A}$ and assignment α, $\mathcal{A}, \alpha \models \varphi$.

Definition 30 (Logical Consequence)**.** A formula φ is a *logical consequence* of a set of formulae Γ if for each structure $\mathcal{A}$ and assignment α, $\mathcal{A}, \alpha \models \Gamma$ implies that $\mathcal{A}, \alpha \models \varphi$.

As a shorthand, when $\varnothing \models \varphi$, we write $\models \varphi$.

Definition 31 (Logical Equivalence)**.** Two formulae φ_1 and φ_2 are *logically equivalent* if for every structure $\mathcal{A}$ and assignment α, $\mathcal{A}, \alpha \models \varphi_1$ iff $\mathcal{A}, \alpha \models \varphi_2$.

Definition 32 (Equisatisfiability)**.** Two formulae φ_1 and φ_2 are *equisatisfiable* when φ_1 and φ_2 are both satisfiable or both unsatisfiable.

4.5 Normalization and Skolemization

Similar to how propositional formulae in conjunctive normal form are easier to process, FOL formulae in *prenex normal form* are also easier for automated theorem provers to process.

Definition 33 (Prenex Normal Form)**.** A formula is in *prenex normal form* if it is of the form $\varphi \equiv Q_1 x_1 \cdot Q_2 x_2 \cdots Q_k x_k \cdot \psi$ where $Q_1, \ldots, Q_k \in \{\forall, \exists\}$ are quantifiers, $x_1, \ldots, x_k \in \mathcal{V}$ and ψ is quantifier-free. ψ is called the *matrix* of the prenex normal formula formula φ.

Claim 11. *Every FOL formula is logically equivalent to a formula in prenex normal form.*

One can go even further and eliminate existential quantifiers from a formula in prenex normal form via a process called *Skolemization*. This results in an equisatisfiable formula.

Definition 34 (Skolem Normal Form)**.** A formula is in *Skolem normal form* if it is in prenex normal form with only universal quantifiers (i.e. $\forall$).

Claim 12 (Skolemization)**.** *Every FOL formula is equisatisfiable with a formula in Skolem normal form. This equisatisfiable formula can be obtained via* Skolemization*.*

4.6 Herbrand's Theorem for FOL

Herbrand's Theorem relates satisfiability of a set of FOL sentences to their propositional satisfiability. Before we state the theorem, we first state some new definitions.

Definition 35 (Ground Term)**.** A term over a signature Σ is a *ground term* if it does not contain any variables. The set of all ground terms over Σ is denoted by Terms_Σ.

Definition 36 (Ground Formula)**.** A formula is a *ground formula* if it contains no variables (and thus, also no quantifiers). The set of ground formulae over the signature Σ is denoted by FORM_Σ.

Definition 37 (Ground Atom)**.** A formula is a *ground atom* if it is both a ground formula and an atomic formula. The set of ground atoms over the signature Σ is denoted by Atoms_Σ.

Definition 38 (Ground Instantiation)**.** Let $\varphi \equiv \forall x_1 \forall x_2 \ldots \forall x_n \psi$ be a universally quantified FOL sentence over Σ and $\mathcal{V}$ in prenex normal form. A ground instantiation of φ is a formula ϕ obtained by replacing, for every $1 \leq i \leq n$, each occurrence of x_i in ψ by some ground term t_i. That is, there is a mapping $T : \mathcal{V} \to \mathsf{Terms}_\Sigma$ such that $\phi \equiv \psi[x_1 \to T(x_1)] \ldots [x_n \to T(x_n)]$.

We use $\mathsf{Ground}(\varphi)$ to denote the set $\{\psi[x_1 \to T(x_1)] \ldots [x_n \to T(x_n)] \mid T : \mathcal{V} \to \mathsf{Terms}_\Sigma\}$ of all ground instantiations of φ. For a set Γ of universally quantified FOL sentences over Σ and $\mathcal{V}$, we denote $\mathsf{Ground}(\Gamma) = \bigcup_{\varphi \in \Gamma} \mathsf{Ground}(\varphi)$.

Definition 39 (Propositional Satisfiability)**.** A *ground atomic valuation* over Σ is a mapping $\tau : \mathsf{Atoms}_\Sigma \to \{\mathtt{true}, \mathtt{false}\}$. Such a valuation can be extended to ground formulae in the natural manner given by the relation $\models_{\mathrm{Gr}}$:

$\tau \models_{\mathrm{Gr}} \psi$	iff $\tau(\psi) = \mathtt{true}$	if $\psi \in \mathsf{Atoms}_\Sigma$
$\tau \models_{\mathrm{Gr}} (\neg\varphi)$	iff $\tau \not\models_{\mathrm{Gr}} \varphi$	if φ is ground
$\tau \models_{\mathrm{Gr}} (\varphi_1 \vee \varphi_2)$	iff $\tau \models_{\mathrm{Gr}} \varphi_1$ or $\tau \models_{\mathrm{Gr}} \varphi_2$	if φ_1, φ_2 are ground

A set of ground FOL formulae Γ is *propositionally satisfiable* if there is a ground atomic valuation τ such that $\tau \models_{\mathrm{Gr}} \varphi$ for every $\varphi \in \Gamma$.

We can begin discussing Herbrand's Theorem. As a start, we will only consider FOL formulae that are universally quantified sentences that do not have the symbol "=". We will refer to them as *formulae without equality*.

Theorem 8 (Herbrand's Theorem without Equality)**.** Let Σ be a signature with at least one constant. Let Γ be a set of universally quantified FOL sentences without equality in prenex normal form. Γ is satisfiable iff $\mathsf{Ground}(\Gamma)$ is propositionally satisfiable.

We shall prove this theorem in two directions. The first direction is easier, and the second direction is more difficult. We first prove the easier direction:

Lemma 10 (Easier direction of Herbrand's Theorem without Equality)**.** *Let Σ be a signature with at least one constant. Let Γ be a set of universally quantified FOL sentences without equality in prenex normal form. If Γ is satisfiable, then $\mathsf{Ground}(\Gamma)$ is propositionally satisfiable.*

Proof. Consider a structure $\mathcal{A}$ such that $\mathcal{A} \models \Gamma$. Let $\tau_{\mathcal{A}} : \mathsf{Atoms}_\Sigma \to \{\mathtt{true}, \mathtt{false}\}$ be the ground atomic valuation defined as follows:

$$\tau_{\mathcal{A}}(\varphi) = \begin{cases} \mathtt{true} & \text{if } \mathcal{A} \models \varphi \\ \mathtt{false} & \text{otherwise} \end{cases}$$

We will first prove the following property about $\tau_{\mathcal{A}}$ that we will later use in our proof of the lemma:

Claim 13. *For any ground formula φ, $\mathcal{A} \models \varphi$ iff $\tau_{\mathcal{A}} \models_{Gr} \varphi$.*

Subproof. We will prove, by structural induction on φ, that $\mathcal{A} \models \varphi$ iff $\tau_{\mathcal{A}} \models_{\text{Gr}} \varphi$.

Base Case. Consider a ground atom φ. If $\mathcal{A} \models \varphi$, then $\tau_{\mathcal{A}}(\varphi) = \mathtt{true}$ by definition, and so $\tau_{\mathcal{A}} \models_{\text{Gr}} \varphi$. Conversely, if $\mathcal{A} \not\models \varphi$, then $\tau_{\mathcal{A}}(\varphi) = \mathtt{false}$ by definition, and so $\tau_{\mathcal{A}} \not\models_{\text{Gr}} \varphi$.

Inductive Case 1. Assume the inductive hypothesis holds for some ground formula ψ. Consider the ground formula $\varphi \equiv (\neg\psi)$.

If $\mathcal{A} \models \varphi$, then $\mathcal{A} \not\models \psi$ by definition. Then, by the inductive hypothesis, $\tau_{\mathcal{A}} \not\models_{\text{Gr}} \psi$ and so $\tau_{\mathcal{A}} \models_{\text{Gr}} \varphi$ by definition.

Conversely, if $\mathcal{A} \not\models \varphi$, then $\mathcal{A} \models \psi$ by definition. Then, by the inductive hypothesis, $\tau_{\mathcal{A}} \models_{\text{Gr}} \psi$ and so $\tau_{\mathcal{A}} \not\models_{\text{Gr}} \varphi$ by definition.

Inductive Case 2. Assume the inductive hypothesis holds for some ground formulae ψ_1 and ψ_2. Consider the ground formula $\varphi \equiv (\psi_1 \vee \psi_2)$.

If $\mathcal{A} \models \varphi$, then $\mathcal{A} \models \psi_1$ or $\mathcal{A} \models \psi_2$ by definition. Then, by the inductive hypothesis, $\tau_{\mathcal{A}} \models_{\text{Gr}} \psi_1$ or $\tau_{\mathcal{A}} \models_{\text{Gr}} \psi_2$ and so $\tau_{\mathcal{A}} \models_{\text{Gr}} \varphi$ by definition.

Conversely, if $\mathcal{A} \not\models \varphi$, then $\mathcal{A} \not\models \psi_1$ and $\mathcal{A} \not\models \psi_2$. Then, by the inductive hypothesis, $\tau_{\mathcal{A}} \not\models_{\text{Gr}} \psi_1$ and $\tau_{\mathcal{A}} \not\models_{\text{Gr}} \psi_2$ and so $\tau_{\mathcal{A}} \not\models_{\text{Gr}} \varphi$ by definition.

Hence, we have shown that $\mathcal{A} \models \varphi$ iff $\tau_{\mathcal{A}} \models_{\text{Gr}} \varphi$. ■

Now, using the aforementioned claim, we will prove that $\mathsf{Ground}(\Gamma)$ is propositionally satisfiable.

Consider any ground formula $\varphi \in \mathsf{Ground}(\Gamma)$. φ must have been instantiated from a universally quantified sentence $\varphi' \in \Gamma$. Since $\mathcal{A} \models \varphi'$, we must have $\mathcal{A} \models \varphi$. Hence, by the aforementioned lemma, $\tau_{\mathcal{A}} \models_{\text{Gr}} \varphi$. Thus, $\mathsf{Ground}(\Gamma)$ is propositionally satisfiable. □

Now, we can progress towards proving the opposite direction. The proof of the opposite direction relies on the observation that a special simple structure, called the *Herbrand structure*, satisfies Γ. We first define the Herbrand structure.

Definition 40 (Herbrand Structure)**.** Let Σ be a signature with at least one constant symbol. A Σ-Herbrand-structure is a structure $\mathcal{A}_\Sigma = (U_\Sigma, I_\Sigma)$ defined as follows:

- $U_\Sigma = \mathsf{Terms}_\Sigma$ is the set of all ground terms in Σ. It is also called the *Herbrand universe*.
- I_Σ assigns the unique "natural" interpretation to the constant and function symbols in Σ (also called the *Herbrand interpretation*). That is,
  - For every constant symbol $c \in \mathcal{C}$, $I_\Sigma(c) = c$.
  - For every k-ary function symbol $f \in \mathcal{F}$ and for all ground terms $t_1, \ldots, t_k \in U_\Sigma$, $I_\Sigma(f)(t_1, \ldots, t_k) = f(t_1, \ldots, t_k)$.

Thus, a Herbrand structure is built from syntax, with terms and function symbols being interpreted "as themselves".

Observe that I_Σ does not restrict what interpretations must be given to the relation symbols, and any meaning for symbols in $\mathcal{R}$ can be picked as part of I_Σ.

Now, we can prove the opposite direction of Herbrand's theorem without equality.

Lemma 11 (Harder direction of Herbrand's theorem without Equality)**.** *Let Σ be a signature with at least one constant symbol. Let Γ be a set of universally quantified FOL sentences without equality in prenex normal form. If $\mathsf{Ground}(\Gamma)$ is propositionally satisfiable, then Γ is satisfiable.*

Proof. Consider a ground atomic valuation τ such that $\tau \models_{\text{Gr}} \psi$ for every ground formula $\psi \in \mathsf{Ground}(\Gamma)$. Construct the Herbrand structure $\mathcal{A}^\tau_\Sigma$ such that for every k-ary relation symbol $R \in \mathcal{R}$, $I(R) = \{(t_1, \ldots, t_k) \mid t_1, \ldots, t_k \in \mathsf{Terms}_\Sigma \text{ and } \tau(R(t_1, \ldots, t_k)) = \texttt{true}\}$.

We will first prove properties about $\mathcal{A}^\tau_\Sigma$ that we will later use in our proof of the lemma:

Claim 14. *Let $\varphi \in \Gamma$ be a ground formula without equality. If $\tau \models_{Gr} \varphi$, then $\mathcal{A}^\tau_\Sigma \models \varphi$.*

Subproof. We will prove, by structural induction on φ, that if $\tau \models_{\text{Gr}} \varphi$, then $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$.

Base Case. Consider $\varphi \equiv R(t_1, \ldots, t_k)$ for some k-ary relation $R \in \mathcal{R}$ and terms $t_1, \ldots, t_k$. Note that φ is a ground atom and $t_1, \ldots, t_k$ are ground terms. If $\tau \models_{\text{Gr}} \varphi$, then $\tau(\varphi) = \texttt{true}$ and so $(t_1, \ldots, t_k) \in I(R)$. Hence, by definition, $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$.

Inductive Case 1. Assume the inductive hypothesis for some ground formula ψ. Consider $\varphi \equiv (\neg\psi)$. Note that φ is still a ground formula. If $\tau \models_{\text{Gr}} \varphi$, this means that $\tau \not\models_{\text{Gr}} \psi$. Then, by the inductive hypothesis, $\mathcal{A}^{\tau}_{\Sigma} \not\models \psi$. Hence, by definition, $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$.

Inductive Case 2. Assume the inductive hypothesis for some ground formulae ψ_1 and ψ_2. Consider $\varphi \equiv (\psi_1 \vee \psi_2)$. Note that φ is still a ground formula. If $\tau \models_{\text{Gr}} \varphi$, then $\tau \models_{\text{Gr}} \psi_1$ or $\tau \models_{\text{Gr}} \psi_2$ by definition. By the inductive hypothesis, $\mathcal{A}^{\tau}_{\Sigma} \models \psi_1$ or $\mathcal{A}^{\tau}_{\Sigma} \models \psi_2$. Hence, by definition, $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$.

Thus, we have proven that, for every ground formula φ, $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$ if $\tau \models_{\text{Gr}} \varphi$. ■

Next, we'll prove a stronger version of the claim that will be useful in our proof of the lemma.

Claim 15. *Let $\varphi \equiv \forall x_1 \ldots \forall x_k . \psi$ in Γ be a sentence without equality in prenex normal form. If τ propositionally satisfies* $\mathsf{Ground}(\varphi)$*, then $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$.*

Subproof. We will prove, by structural induction on the number of quantifiers k in φ, that if τ propositionally satisfies $\mathsf{Ground}(\varphi)$, then $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$.

Base Case. Consider the case $k = 0$. This means that $\varphi \equiv \psi$ where ψ is a matrix. Since φ is a ground formula, $\varphi \in \mathsf{Ground}(\varphi)$. If τ satisfies $\mathsf{Ground}(\varphi)$, $\tau \models_{\text{Gr}} \varphi$. By the previous claim, we can conclude that $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$.

Inductive Case. Assume the inductive hypothesis holds for some $k \in \mathbb{N}$. Consider $\varphi \equiv \forall x_1 \ldots \forall x_{k+1} . \psi$ where ψ is a matrix. If τ satisfies $\mathsf{Ground}(\varphi)$, then in particular τ satisfies $\mathsf{Ground}(\forall x_2 \ldots \forall x_{k+1} . \psi[x_1 \to t])$ for every ground term $t \in \mathsf{Terms}_{\Sigma}$. By the inductive hypothesis, for every ground term $t \in \mathsf{Terms}_{\Sigma}$, $\mathcal{A}^{\tau}_{\Sigma} \models \forall x_2 \ldots \forall x_{k+1} . \psi[x_1 \to t]$. Equivalently, there is no $t \in \mathsf{Terms}_{\Sigma}$ such that $\mathcal{A}^{\tau}_{\Sigma} \not\models \forall x_2 \ldots \forall x_{k+1} . \psi[x_1 \to t]$. Hence, by definition, $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$.

Thus, we have proven that, for every sentence φ in prenex normal form, $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$ if τ propositionally satisfies $\mathsf{Ground}(\varphi)$. ■

Now, we finally complete the proof of the lemma. Consider any sentence φ in Γ. By the assumption and by definition of $\mathsf{Ground}(\Gamma)$, $\tau \models_{\text{Gr}} \mathsf{Ground}(\varphi)$. Then, by the previous claim, $\mathcal{A}^{\tau}_{\Sigma} \models \varphi$. Hence, $\mathcal{A}^{\tau}_{\Sigma} \models \Gamma$. □

4.7 Implications of Herbrand's Theorem

The proof of Herbrand's Theorem highlights some interesting *model-theoretic* observations. We go through them in this section.

First, the proof illustrates that whenever there is a model $\mathcal{A}$ for a set Γ of universally quantified FOL sentences without equality, there is a model which is not *too large*. This is known as the *Downward Löwenheim-Skolem Theorem*.

Theorem 9 (Downward Löwenheim-Skolem Theorem)**.** Let Σ be a countable signature and let Γ be a set of universally quantified FOL sentences without equality. If Γ is satisfiable, then there is a countable structure $\mathcal{A}$ such that $\mathcal{A} \models \Gamma$.

Proof. Since Γ is satisfiable, $\mathsf{Ground}(\Gamma)$ is propositionally satisfiable. By the proof of Herbrand's Theorem, there is a Herbrand structure $\mathcal{A}^{\tau}_{\Sigma}$ that satisfies Γ. We will show that $\mathcal{A}^{\tau}_{\Sigma}$ is countable, which will then imply the Downward Löwenheim-Skolem Theorem.

It suffices to show that $U_\Sigma = \mathsf{Terms}_\Sigma$ is countable. By assumption, Σ is countable and so for any natural number $k \in \mathbb{N}$, the set of all ground terms of length k is countable (since it is isomorphic to some subset of $\mathbb{N}^k$, which is countable). Then, $\mathsf{Terms}_\Sigma = \bigcup_{k\in\mathbb{N}} \{\text{all ground terms of length } k\}$ is a countable union of countable sets and so, it is also countable. Hence, the Herbrand structure $\mathcal{A}^{\tau}_{\Sigma}$ is countable. □

4.8 Herbrand's Theorem with Equality

We have seen that in the absence of equality, a ground atomic valuation that makes all of $\mathsf{Ground}(\Gamma)$ propositionally satisfiable guarantees the existence of a structure that satisfies Γ. In the presence of equality, this is not true.

We need to convert equality to a form that is correctly interpretable. To do so, we replace any set of FOL Γ with $\Gamma^* = \mathsf{eq}(\Gamma) \cup \mathscr{E}$. We replace all instances of $x = y$ to $\mathsf{EQ}(x, y)$, thereby transforming equality into a relation symbol.

We define $\mathscr{E} = \bigcup_{i=1}^{5} \mathscr{E}_i$. $\mathscr{E}_1$ represents reflexivity, $\mathscr{E}_2$ represents symmetry, $\mathscr{E}_3$ represents transitivity, $\mathscr{E}_4$ represents preservation of equality under function application, $\mathscr{E}_5$ represents preservation of equality with respect to relations.

---

5 Primer on Countability

Definition 41 (Injection)**.** A function $f : A \to B$ is said to be *injective* if for all $a_1, a_2 \in A$, if $f(a_1) = f(a_2)$, then $a_1 = a_2$.

We aim to represent cardinality of sets in terms of injections.

Definition 42 (Countable set)**.** A set S is said to be *countable* if there exists an injection $f : S \to \mathbb{N}$.

Notice that if a set is countable, then one can assign indices which are natural numbers to the elements of the set such that no two elements of S get the same index.

This is analogous to the idea of "enumerating" the elements of S.

Claim 16 (Finite sets are countable)**.** *A finite set is countable.*

Proof sketch. Let e be an arbitary enumeration of S. Consider the function $f : S \to \mathbb{N}$ defined by $f(a) = i$ such that $i \in \mathbb{N}$ is the index of a in e; that is, $f(a) = e_i$. We can show that this is an injection □

To formally prove Claim 16, we will have to come up with an injective function from any finite set to the natural numbers. We will omit the proof here.

A more crucial observation is the following theorem.

Theorem 10. There are sets that are not countable; that is, they are *uncountable*.

Example. The set of real numbers $\mathbb{R}$ and the powerset of the natural numbers $\mathscr{P}(\mathbb{N})$ are uncountable.

Next, we will state a straightforward claim.

Claim 17. *The set of natural numbers $\mathbb{N}$ is countable.*

Proof sketch. Consider the function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(n) = n$. We can show that this is an injection. □

Another simple observation is the following.

Claim 18. *Let S be a set and $S' \subseteq S$. If S is countable, then S' is countable.*

Proof sketch. Consider an injection $f : S \to \mathbb{N}$. We can extend f to an injection $g : S' \to \mathbb{N}$ by defining $g(s) = f(s)$ for all $s \in S'$. □

Example. Some other examples of countable sets are: the set of even numbers, odd numbers, and prime numbers. These are all subsets of the natural numbers.

Notice that while there is an injection from the set of even numbers to the set of naturals, there is also an injection from the set of naturals to the set of even numbers. We will formalise this notion in the following claim.

Claim 19. *If A and B are countably infinite sets, then there is a bijection between A and B. Thus, $|A| = |B|$.*

An even more important result is the following claim:

Claim 20. *Let A and B be countably infinite sets. Then, $A \times B = \{(a, b) \mid a \in A, b \in B\}$ is countable.*

The idea of a proof for Claim 20 is to draw a grid and list the elements of A along the rows and the elements of B along the columns. Numbering the elements along the diagonals of the grid, we can then define an injection from $A \times B$ to $\mathbb{N}$.

Claim 20 yields the following corollaries:

Corollary 3. *$\mathbb{N} \times \mathbb{N}$ is countable.*

Corollary 4. *$\mathbb{Q}$ is countable.*

Proof sketch. Every rational $r \in \mathbb{Q}$ is of the form p/q where $p, q \in \mathbb{N}$. Thus, r can be represented as a pair $(p, q) \in \mathbb{N} \times \mathbb{N}$ and so $\mathbb{Q}$ is isomorphic to a subset of $\mathbb{N} \times \mathbb{N}$. By Corollary 3 and Claim 18, $\mathbb{Q}$ is countable. □

Corollary 5. *$\mathbb{Z}$ is countable.*

Proof sketch. For every integer i, we can write it as either $(0, i)$ or $(-1, i)$. Thus, $\mathbb{Z}$ is isomorphic to a subset of $\mathbb{N} \times \mathbb{N}$. By Corollary 3 and Claim 18, $\mathbb{Q}$ is countable. □

Theorem 11. There is a collection $\mathcal{C}$ of countable sets such that $\mathcal{C}$ is uncountable.

Example. The powerset of the natural numbers $\mathscr{P}(\mathbb{N})$ is an uncountable collection of countable sets.

Next, we state a more important observation:

Theorem 12. Let $\mathcal{C}$ be a collection of countable sets such that $\mathcal{C}$ is countable. Then, $\mathcal{I} = \bigcup_{S \in \mathcal{C}} S$ is countable.

Proof sketch. We can write each element $c \in \mathcal{I}$ as (i, j) where i is the index of the set $S \in \mathcal{C}$ and j is the index of the element $x \in S$. Thus, $\mathcal{I}$ is isomorphic to some subset of $\mathbb{N} \times \mathbb{N}$. By Corollary 3 and Claim 18, $\mathbb{Q}$ is countable. □

Finally, we arrive at one of the most important theorems in countability.

Theorem 13 (Uncountability of reals)**.** $\mathbb{R}$ is uncountable.

Proof sketch. The proof is by Cantor's diagonal argument. □

Theorem 14 (Uncountability of powerset of naturals)**.** $\mathscr{P}(\mathbb{N})$ is uncountable.

Proof sketch. For any set $S \in \mathscr{P}(\mathbb{N})$, we can associate a unique real number r to S where $r = 0.\ldots$ where the i^{th} decimal place is 1 if the i^{th} natural number is in S and 0 otherwise. Hence, $\mathscr{P}(\mathbb{N})$ is isomorphic to $\mathbb{R}$. By Theorem 13, $\mathscr{P}(\mathbb{N})$ is uncountable. □

Next, let's look at alphabets, strings, and languages.

Theorem 15. Let Σ be some countable alphabet (or set). Σ^* is countable, where Σ^* denotes the set of all finite strings over Σ.

Theorem 16. If S be a countable set, then the set $\mathscr{P}_{\text{fin}}(S)$ of finite subsets of S is countable.

We have previously seen that every set of strings on a countable alphabet is countable. But what about the set of all languages?

Question 1. Let Σ be a countable alphabet. A language L over Σ is a subset of Σ^*. Is the collection of all languages over Σ countable? What if Σ is finite?

---

6 Primer on Computability Theory

Computability Theory asks questions like "what is the complexity of solving a problem?", or more generally, "is the problem solvable?". We will study computability theory in the context of first-order logic (FOL), by modelling computational problems as formulae and determining if such formulae are satisfiable in polynomial time.

Here's an exercise: write a program P that takes

1. a program Q as input,
2. and an input I to Q as input.

such that P outputs "yes" if Q on input I prints "hello" as the first 5 characters, and "no" otherwise.

It turns out that it is impossible to write such a program.

Claim 21. *There is no program P that takes as input (1) a program Q and (2) an input I to Q such that P outputs "yes" if Q on input I prints "hello" as the first 5 characters, and "no" otherwise.*

Proof. Suppose, on the contrary, that there exists such a program P. Then, we can write a program P' which takes in a program Q and runs $P(Q,Q)$, then outputs "no" if $P(Q,Q)$ outputs "yes" else it outputs "no".

Now, suppose we execute $P'(P')$. If it outputs "no", then $P(P',P')$ outputs "yes", which means that $P'(P')$ outputs "hello". This is a contradiction. Otherwise, if $P'(P')$ outputs "hello", then $P(P',P')$ outputs "no", which means that $P'(P')$ does not output "hello". This is also a contradiction.

In either case, we arrive at a contradiction. Therefore, there is no such program P. □

The previous claim shows that certain computational problems that are unsolvable. Another well-known unsolvable problem is the Halting problem, which is the problem of determining whether a given program will terminate on a given input.

6.1 Turing Machines

We use Turing machines as our main model of computation. Input that is fed into a Turing machine can be modelled as strings over an alphabet. We will define some terms and notation before proceeding.

Definition 43 (Turing machine)**.** A *Turing machine* is a tuple $M = (Q, q_0, q_{\text{acc}}, q_{\text{rej}}, k, \delta, \Sigma)$ where

1. Q is a finite set of control states
2. Σ is the alphabet of tape symbols
3. $q_0 \in Q$ is the initial state
4. $q_{\text{acc}} \in Q$ is the accepting state
5. $q_{\text{rej}} \in Q$ is the rejecting state

Definition 44 (Configuration)**.** A *configuration* C of a Turing machine M is $C = (q, w_{\text{inp}} \uparrow w'_{\text{inp}}, w_{\text{WT}_1} \uparrow w'_{\text{WT}_1}, \ldots, w_{\text{WT}_k} \uparrow w'_{\text{WT}_k}, w_{\text{out}} \uparrow w'_{\text{out}})$ where w_x are finite strings over Σ representing the finite contents of the unbounded tape and $\uparrow$ refers to the pointer of the input tape, work tapes, and output tapes.

Definition 45 (Run/Computation)**.** A *run/computation* of Turing machine M on input $w \in \Sigma^*$ is $\pi = c_0, c_1, \ldots, c_m$ such that c_0 is the initial configuration and, for each i, c_{i+1} follows from c_i using δ.

Definition 46 (Accepting Run/Acceptance)**.** A run π is an *accepting run* if there is a configuration c in the run such that $c = (q_{\text{acc}}, \ldots)$ and q_{rej} is not reached before c. The input w is *accepted* by M iff the run π on M is an accepting run. Otherwise, w is rejected.

Definition 47 (Language of a Turing Machine)**.** The *language* of a Turing machine M is $L(M) = \{w \in \Sigma^* \mid w \text{ is accepted by } M\}$. A language $A \subseteq \Sigma^*$ is *recognized/accepted* by M if $A = L(M)$.

Definition 48 (Halting)**.** A Turing machine M *halts* on input w if there is a computation $\pi = c_0, c_1, \ldots, c_m$ such that $c_m = (q_{\text{acc}}, \ldots)$ or $c_m = (q_{\text{rej}}, \ldots)$. The run π is called a *halting run*.

For a list of objects $O_1, O_2, \ldots, O_k$, we will use $\langle O_1, O_2, \ldots, O_k \rangle$ to denote their binary encoding. In particular, for a Turing machine M, $\langle M \rangle$ is its encoding as a binary string. We may then define the language $L_{\text{Halt}} = \{\langle M, w \rangle \mid w \text{ on } M \text{ halts}\}$ Is there a Turing machine H such that $L(H) = L_{\text{Halt}}$?

The answer is yes: just simulate M on w. This Turing machine is known as a *universal Turing machine* since it can simulate the specification of an arbitrary Turing machine on arbitrary input.

6.2 Recursive and Recursively Enumerable Languages

Recall that there are 3 possible outcomes when a Turing machine M runs on an input string w — M may halt and accept w, M may halt and reject w, or M may not halt on w. Depending on how a Turing machine behaves, we can define two different classes of problems solvable on a Turing machine.

Definition 49 (Recursively Enumerable)**.** A language A is *recursively enumerable/semi-decidable* if there is a Turing machine M such that $A = L(M)$. We denote the set of all recursively enumerable languages as RE.

Example. L_{Halt} is recursively enumerable. That is, $L_{\text{Halt}} \in \mathsf{RE}$.

Definition 50 (Recursive/Decidable)**.** A language A is *recursive/decidable* if there is a Turing machine M that halts on *all* inputs and $A = L(M)$. We denote the set of all recursive languages as REC.

As an example, consider the language $L_{\text{Sorted}} = \{\langle l \rangle \mid l \text{ is a sorted list}\}$.

Example. $L_{\text{Sorted}} \in \mathsf{REC}$ but $L_{\text{Halt}} \notin \mathsf{REC}$.

Observe that a problem that is recursive is solvable by an algorithm that always halts. Thus, by definition, recursive languages are also recursively enumerable. This observation is equivalent to the following lemma:

Lemma 12 (Recursive Implies Recursively Enumerable)**.** *REC* $\subseteq$ *RE*.

We also define the complement of a language.

Definition 51 (Complement of a Language)**.** The complement of a language L is $\overline{L} = \Sigma^* \setminus L$.

Theorem 17 (Complement of Recursive is Recursive)**.** If $L \in \mathsf{REC}$, then $\overline{L} \in \mathsf{REC}$.

Proof sketch. Take the Turing machine M that accepts L and let M' be the Turing machine M with the only difference being that the accepting and rejecting states are swapped. Then, $\overline{L} = L(M')$ and M' halts on every input. Thus, $\overline{L} \in \mathsf{REC}$. □

The following theorem is a useful way to prove that a problem is recursive.

Theorem 18. L is recursive iff L and $\overline{L}$ are recursively enumerable. That is, $L \in \mathsf{REC}$ iff $L \in \mathsf{RE}$ and $\overline{L} \in \mathsf{RE}$.

Proof sketch. The (easy) forward direction uses Lemma 12.

The reverse direction uses a technique called *dovetailing*. Suppose that L and $\overline{L}$ are recognized by M and $\overline{M}$ respectively. Then, construct M' by running both M and $\overline{M}$ simultaneously on an input w, and accept if M accepts or reject if $\overline{M}$ accepts. Since w belongs to either L or $\overline{L}$, M' halts on all inputs and $L = L(M')$. Thus, $L \in \mathsf{REC}$. □

Not every decision problem is recursively enumerable.

Theorem 19 (Language outside RE)**.** There is a language L that is not recursively enumerable. That is, $L \notin \mathsf{RE}$.

Proof. Since $L_{\text{Halt}} \notin \mathsf{REC}$ but $L_{\text{Halt}} \in \mathsf{RE}$, by Theorem 18, $\overline{L_{\text{Halt}}} \notin \mathsf{RE}$.

Alternatively, note that there are uncountably many languages yet countably many Turing machines. □

Next, consider the language $K = \{\langle M \rangle \mid \langle M \rangle \in L(M)\}$. Using Cantor's diagonalization technique, we can establish that the complement $\overline{K}$ is not recursively enumerable.

Theorem 20. The language $\overline{K} = \{\langle M \rangle \mid \langle M \rangle \notin L(M)\}$ is not recursively enumerable. That is, $\overline{K} \notin \mathsf{RE}$.

Proof. Suppose to the contrary that $\overline{K} = L(M)$ for some Turing machine M. If $\langle M \rangle \in \overline{K}$, then $\langle M \rangle \notin L(M) = \overline{K}$, which is a contradiction. Otherwise, if $\langle M \rangle \notin \overline{K}$, then $\langle M \rangle \in L(M) = \overline{K}$, which is also a contradiction. Therefore, $\overline{K} \neq L(M)$ for any Turing machine M and so $\overline{K} \notin \mathsf{RE}$. □