Mathematical Logic

Eduardo Freire

May 2021

Nesse exercício consideraremos apenas interpretações sobre uma linguagem que contêm apenas um símbolo de relação binária p e nenhum símbolo de função. Também não especificaremos as funções de designação, já que ϕ_1, ϕ_2 e ϕ_3 não possuem variáveis livres, logo seus significados semânticos não dependem das funções de designação.

A interpretação $\mathfrak{A}_r$ cujo domínio é $\mathbb{N}$ e $p^{\mathfrak{A}_r} = \varnothing$ vacuosamente satisfaz ϕ_2 e ϕ_3. Contudo, $\mathfrak{A}_r$ não satisfaz ϕ_1, já que $(0, 0) \notin p^{\mathfrak{A}_r}$, por exemplo.

Defina agora a interpretação $\mathfrak{A}_s$ cujo domínio é $\mathbb{N}$ e $p^{\mathfrak{A}_s}$ é a relação $\leqslant$ usual nos naturais. A relação $\leqslant$ é reflexiva e transitiva, contudo não é simétrica, já que por exemplo $4 \leqslant 5$ mas $\neg(5 \leqslant 4)$. Ou seja, temos $(4, 5) \in p^{\mathfrak{A}_s}$ mas $(5, 4) \notin p^{\mathfrak{A}_s}$. Dessa forma, $\mathfrak{A}_s$ satisfaz ϕ_1 e ϕ_3, mas não ϕ_2.

Por fim, considere a relação $\mathfrak{A}_t$ cujo domínio é $\{1, 2, 3\}$ e $p^{\mathfrak{A}_t}$ é

$$\{(1, 1), (2, 2), (3, 3), (0, 1), (1, 2), (1, 0), (2, 1)\}.$$

Claramente $p^{\mathfrak{A}_t}$ é reflexiva e simétrica. Contudo, $p^{\mathfrak{A}_t}$ não é transitiva, já que $(0, 1) \in p^{\mathfrak{A}_t}$, $(1, 2) \in p^{\mathfrak{A}_t}$ mas $(0, 2) \notin p^{\mathfrak{A}_t}$. Dessa forma, $\mathfrak{A}_t$ satisfaz ϕ_1 e ϕ_2, mas não ϕ_3.

1 Propositional Logic

Definition 1.1. *Let $Vars_P := \{P_n : n \in \mathbb{N}\}$ be the set of the symbols $P_1, P_2, \ldots,$ each called a propositional variable. We define the* ***language of propositional logic*** *as $\mathcal{L}_P := Vars_P \cup \{\to, \neg\}$.*

Definition 1.2. *Let ϕ be an $\mathcal{L}_P$ string. We say that ϕ is a* ***propositional formula*** *(also called* ***p-formula****) if and only if*

1. *ϕ is a propositional variable, or*
2. *$\phi :\equiv (\alpha \to \beta)$ where α and β are propositional formulas, or*
3. *$\phi :\equiv (\neg\alpha)$ and α is a propositional formula.*

Definition 1.3. *An* ***assignment function*** *is any function with domain $Vars_P$ and codomain $\{T, F\}$. Given an assignment function s, we define the function $\bar{s}$ whose domain is the set of all p-formulas and codomain is $\{T, F\}$ as follows:*

$$\bar{s}(\phi) := \begin{cases} s(\phi) & \phi \in Vars_P, \\ F & \phi :\equiv (\neg\alpha) \text{ and } \bar{s}(\alpha) = T, \\ F & \phi :\equiv (\alpha \to \beta) \text{ and } \bar{s}(\alpha) = T \text{ and } \bar{s}(\beta) = F, \\ T & \text{otherwise.} \end{cases}$$

Also, if Σ is a set of p-formulas, we say that s ***satisfies*** *Σ if and only if $\bar{s}(\sigma) = T$ for every $\sigma \in \Sigma$. Otherwise, we say that s* ***does not satisfy*** *Σ. If there is some assignment function s' that satisfies Σ, we say that Σ is* ***satisfiable****.*

Definition 1.4. *Let ϕ be a p-formula. If $\bar{s}(\phi) = T$ for every assignment function s, we say that ϕ is a* ***tautology****. On the other hand, if $\bar{s}(\phi) = F$ for every assignment function s, we call ϕ a* ***contradiction****. In particular, we define $\top$ as the tautology $(P_1 \to (P_1 \to P_1))$ and $\bot$ as the contradiction $\neg\top$, i.e $\neg(P_1 \to (P_1 \to P_1))$.*

Definition 1.5. *Let Λ be a set of p-formulas such that for every p-formula ϕ, $\phi \in \Lambda$ if and only if*

1. *$\phi :\equiv (A \to (B \to A))$, or*
2. *$\phi :\equiv ((A \to (B \to C)) \to ((A \to B) \to (A \to C)))$, or*
3. *$\phi :\equiv ((\neg B \to \neg A) \to (A \to B))$*

where A, B, C are p-formulas. We call Λ the set of ***logical axioms****.*

Lemma 1.1. *Every $\lambda \in \Lambda$ is a tautology.*

Proof. This is trivial to check case by case, using the definition of assignment functions for p-formulas. □

Lemma 1.2. *Let α and β be p-formulas and s be an assignment function such that $\bar{s}(\alpha) = T$ and $\bar{s}(\alpha \to \beta) = T$. Then $\bar{s}(\beta) = T$.*

Proof. Assume for contradiction that $\bar{s}(\beta) = F$. Since $\bar{s}(\alpha) = T$ by assumption, it follows from the definition of $\bar{s}$ that $\bar{s}(\alpha \to \beta) = F$, which contradicts our assumption that $\bar{s}(\alpha \to \beta) = T$. Thus $\bar{s}(\beta) = T$. □

Definition 1.6. *Let Σ be a set of p-formulas and ϕ be a p-formula. We say that $\Sigma \models \phi$ if and only if every assignment function that satisfies Σ assigns ϕ to T.*

Definition 1.7. *Let Σ be a set of p-formulas and ϕ be a p-formula. We say that a finite sequence $D = (\phi_1, \phi_2, \ldots, \phi_n)$ of p-formulas whose last entry is ϕ is a* ***deduction from*** *Σ* ***of*** *ϕ if and only if for each $1 \leqslant i \leqslant n$,*

1. *$\phi_i \in \Lambda \cup \Sigma$, or*
2. *There exists $j, k < i$ such that $\phi_j :\equiv (\phi_k \to \phi_i)$.*

In this case, we write $\Sigma \vdash \phi$, read as Σ proves ϕ. If Γ is a set of p-formulas such that $\Sigma \vdash \gamma$ for every $\gamma \in \Gamma$, we write $\Sigma \vdash \Gamma$.

The following lemma has an easy proof and will be used implicitly several times.

Lemma 1.3. *Let Σ, Γ be sets of p-formulas and α, β, ϕ be p-formulas. It follows that:*

1. *If $\Sigma \vdash (\alpha \to \beta)$ and $\Sigma \vdash \alpha$, then $\Sigma \vdash \beta$,*
2. *If $\Gamma \vdash \phi$ and $\Gamma \subseteq \Sigma$, then $\Sigma \vdash \phi$,*
3. *If $\Gamma \vdash \phi$ and $\Sigma \vdash \Gamma$, then $\Sigma \vdash \phi$.*

Theorem 1.1 (Soundness Theorem)**.** *Let Σ be a set of p-formulas, ϕ be a p-formula. Then $\Sigma \vdash \phi$ implies $\Sigma \models \phi$.*

Proof. Assume that $\Sigma \vdash \phi$. We let s be an arbitrary assignment function that satisfies Σ and induct on the shortest length of deduction of ϕ. If there is a deduction of ϕ with length 1, then either $\phi \in \Lambda$ or $\phi \in \Sigma$. In the first case, ϕ is a tautology by Lemma 1.1, so $\bar{s}(\phi) = T$. The other case follows from our assumption that s satisfies Σ. Now assume inductively that if ψ is a p-formula provable from Σ such that its shortest length of deduction is less than or equal to n then $\bar{s}(\psi) = T$.

Assume that the shortest length of deduction of ϕ is $n + 1$. $\phi \notin \Sigma$ and $\phi \notin \Lambda$, since its shortest length of deduction would be 1 in that case. Thus, we have ϕ_j and ϕ_k in the deduction of ϕ such that $\phi_j :\equiv \phi_k \to \phi$. By the inductive hypothesis, $\bar{s}(\phi_j) = \bar{s}(\phi_k) = T$, so it follows from Lemma 1.2 that $\bar{s}(\phi) = T$. □

Lemma 1.4. *For every p-formula ϕ, $\vdash (\phi \to \phi)$.*

Proof. Let ϕ be a p-formula. The following is a deduction of $(\phi \to \phi)$ from $\{\}$.

(1)	$(\phi \to ((P_1 \to \phi) \to \phi))$	Ax 1
(2)	$((\phi \to ((P_1 \to \phi) \to \phi)) \to ((\phi \to (P_1 \to \phi)) \to (\phi \to \phi)))$	Ax 2
(3)	$((\phi \to (P_1 \to \phi)) \to (\phi \to \phi))$	MP 1,2
(4)	$(\phi \to (P_1 \to \phi))$	Ax 1
(5)	$(\phi \to \phi)$	MP 3,4.

□

Theorem 1.2 (Deduction Theorem)**.** *Let Σ be a set of p-formulas and θ, ϕ be p-formulas. Then, $\Sigma \vdash (\theta \to \phi) \iff \Sigma \cup \{\theta\} \vdash \phi$.*

Proof. For the forward direction, assume that $\Sigma \vdash (\theta \to \phi)$. We can use the same deduction from Σ of $(\theta \to \phi)$ to see that $\Sigma \cup \{\theta\} \vdash (\theta \to \phi)$. But clearly $\Sigma \cup \{\theta\} \vdash \theta$, so $\Sigma \cup \{\theta\} \vdash \phi$ by modus ponens.

For the converse direction, we will assume that $\Sigma \cup \{\theta\} \vdash \phi$ and induct on the shortest length of deduction of ϕ. For the base case, assume first that $\phi \in \Lambda \cup \Sigma$.

Then, $\Sigma \vdash \phi$ and $\phi \to (\theta \to \phi)$ is a logical axiom so Σ also proves it. By modus ponens, $\Sigma \vdash (\theta \to \phi)$. The last subcase of the base case is $\phi :\equiv \theta$, but we already know that $\Sigma \vdash (\theta \to \theta)$, by Lemma 1.4.

Next, assume the inductive hypothesis and let the shortest length of deduction of ϕ be $n+1$. Then, we must have ψ and $(\psi \to \phi)$ in the deduction of ϕ from $\Sigma \cup \{\theta\}$. By the inductive hypothesis (IH), $\Sigma \vdash (\theta \to (\psi \to \phi))$ an $\Sigma \vdash (\theta \to \psi)$. Then,

(1)	$\Sigma \vdash ((\theta \to (\psi \to \phi)) \to ((\theta \to \psi) \to (\theta \to \phi)))$	Ax 2
(2)	$\Sigma \vdash ((\theta \to \psi) \to (\theta \to \phi))$	MP 1,IH
(3)	$\Sigma \vdash (\theta \to \phi)$	MP 2,IH

□

Lemma 1.5. *Let ψ, ϕ be p-formulas. Then $\psi, \neg\psi \vdash \phi$.*

Proof.

(1)	$\neg\psi \to (\neg\phi \to \neg\psi)$	Ax 1
(2)	$\neg\psi$	
(3)	$(\neg\phi \to \neg\psi)$	MP 1,2
(4)	$(\neg\phi \to \neg\psi) \to (\psi \to \phi)$	Ax 3
(5)	$(\psi \to \phi)$	MP 3,4
(6)	ψ	
(7)	ϕ	MP 5,6.

□

Definition 1.8. *A set of p-formulas Σ is inconsistent if and only if there is a p-formula ϕ such that $\Sigma \vdash \phi$ and $\Sigma \vdash \neg\phi$. Σ is consistent if and only if it is not inconsistent.*

Lemma 1.6. *Let Σ be a set of p-formulas. The following statements are equivalent:*

1. *Σ is consistent.*
2. *There is a p-formula ψ such that $\Sigma \nvdash \psi$.*
3. *There is no p-formula ψ such that $\Sigma \vdash \neg(\psi \to \psi)$.*
4. *$\Sigma \nvdash \bot$.*

Proof. For the equivalence between (1) and (2), we show instead that Σ is inconsistent if and only if Σ proves every p-formula. For the forward direction, assume that Σ is inconsistent. Then there is some formula ϕ such that $\Sigma \vdash \phi$ and $\Sigma \vdash \neg\phi$. From the deductions of each of these, we can use Lemma 1.5 to produce a deduction of any formula ψ.

For the converse direction, assume that Σ proves every p-formula. Then $\Sigma \vdash P_1$ and $\Sigma \vdash \neg P_1$, so it is inconsistent.

For the equivalence between (2) and (3), assume first that there is a p-formula ψ such that $\Sigma \vdash \neg(\psi \to \psi)$. By Lemma 1.4, $\Sigma \vdash (\psi \to \psi)$. Thus, it follows from Lemma 1.5 that Σ proves every formula, thus showing that (2) is not the case. The other direction is trivial.

(4) $\implies$ (2) is trivial, and (3) $\implies$ (4) also follows easily. $\square$

Lemma 1.7. *Let Σ be a set of p-formulas. If ϕ is a p-formula such that $\Sigma \nvdash \phi$, then $\Sigma \cup \{\neg\phi\}$ is consistent.*

Proof. We prove by contrapositive, so assume that $\Sigma \cup \neg\phi$ is inconsistent. By Lemma 1.6, $\Sigma \cup \neg\phi \vdash \perp$, and the Deduction Theorem guarantees that $\Sigma \vdash (\neg\phi \to \perp)$. Then,

(1)	$\Sigma \vdash (\neg\phi \to \perp)$	Deduction Theorem
(2)	$\Sigma \vdash (\neg\phi \to \perp) \to (\top \to \phi)$	Ax 3
(3)	$\Sigma \vdash \top \to \phi$	MP 1,2
(4)	$\Sigma \vdash \top$	Ax 1
(5)	$\Sigma \vdash \phi$	MP 4,5.

$\square$

Lemma 1.8. *The following statements are equivalent:*

1. *For every set of p-formulas Γ and every p-formula ϕ, if $\Gamma \models \phi$ then $\Gamma \vdash \phi$.*
2. *Every consistent set of p-formulas is satisfiable.*

Proof. For the forward direction, assume the contrapositive of (1) and let Δ be a consistent set of p-formulas. By Lemma 1.6, $\Delta \nvdash \perp$. By assumption, $\Delta \not\models \perp$. If there was no assignment s that satisfied Δ, then $\Delta \models \perp$ would be vacuously true, so Δ must be satisfiable.

For the converse direction, assume (2) and let Γ and ϕ be such that $\Gamma \nvdash \phi$. By Lemma 1.7, $\Gamma \cup \{\neg\phi\}$ is consistent, so it is satisfied by some assignment s. Thus, $s(\neg\phi) = T$, so $s(\phi) = F$. Since s satisfies Γ but $s(\phi) = F$, it follows that $\Gamma \not\models \phi$, as wanted. $\square$

Definition 1.9. *Let Σ be a set of p-formulas. We say that Σ is complete if and only if Σ is consistent and for every p-formula ϕ, exactly one of $\phi, \neg\phi$ is in Σ.*

Lemma 1.9. *Let Σ be a complete set of p-formulas. Then, $\Sigma \vdash \phi \iff \phi \in \Sigma$ for all p-formulas ϕ.*

Proof. For the forward direction assume that $\Sigma \vdash \phi$. If $\neg\phi \in \Sigma$ then clearly $\Sigma \vdash \neg\phi$, so Σ is inconsistent, contradicting the assumption that Σ is complete. Thus $\neg\phi \notin \Sigma$, therefore $\phi \in \Sigma$. The converse direction is trivial. $\square$

Definition 1.10. *Let Σ be a set of p-formulas. We say that Σ is maximally consistent if and only if*

1. *Σ is consistent, and*
2. *For every consistent Σ', if $\Sigma \subseteq \Sigma'$ then $\Sigma' = \Sigma$.*

Lemma 1.10. *Definitions 1.9 and 1.10 are equivalent.*

Proof. Let Σ be a set of p-formulas. For the forward direction, assume that Σ is complete and that Σ' is consistent with $\Sigma \subseteq \Sigma'$. Assume for contradiction that there is some $\psi \in \Sigma'$ such that $\psi \notin \Sigma$. Since Σ is complete we can apply Lemma 1.9 to see that, $\neg\psi \in \Sigma$, so it follows by assumption that $\neg\psi \in \Sigma'$ thus Σ' is inconsistent. This contradiction means that $\Sigma' \subseteq \Sigma$, so $\Sigma' = \Sigma$.

For the converse direction, assume that Σ is maximally consistent and let ϕ be a formula such that $\Sigma \nvdash \phi$. By Lemma 1.7, $\Sigma \cup \{\neg\phi\}$ is consistent. Since $\Sigma \cup \{\neg\phi\} \subseteq \Sigma$, it follows that $\Sigma \cup \{\neg\phi\} = \Sigma$, so $\neg\phi \in \Sigma$, therefore $\Sigma \vdash \neg\phi$, as wanted. Also, since Σ is consistent, it can only prove at most one of ϕ and $\neg\phi$ for any given ϕ. ☐

Lemma 1.11. *Let Σ be a complete set of p-formulas. If s is an assignment function such that for every propositional variable p,*

$$s(p) := \begin{cases} T & p \in \Sigma \\ F & \neg p \in \Sigma, \end{cases}$$

then s is the unique assignment that satisfies Σ.

Proof. Let s be as described in the Lemma. Notice that s is well-defined, since Lemma 1.9 guarantees that for every propositional variable p either $p \in \Sigma$ or $\neg p \in \Sigma$, but not both. To see that s satisfies Σ, we show that $s(\sigma) = T \iff \sigma \in \Sigma$ by induction on the complexity of σ.

The base case is that σ is a propositional variable, but then $s(\sigma) = T \iff \sigma \in \Sigma$ follows trivially. Assume the expected induction hypothesis. If $\sigma :\equiv \neg\alpha$, then $s(\sigma) = T \iff s(\alpha) = F \iff \neg\alpha \in \Sigma \iff \sigma \in \Sigma$. The other case is $\sigma :\equiv (\alpha \to \beta)$. For the forward direction, assume that $s(\alpha \to \beta) = T$, and notice that $s(\alpha \to \beta) = T \iff s(\alpha) = F$ or $s(\beta) = T$. If $s(\alpha) = F$, then $\neg\alpha \in \Sigma$, by the inductive hypothesis. By Lemma 1.5, $\Sigma, \alpha \vdash \beta$, so the Deduction Theorem gives that $\Sigma \vdash (\alpha \to \beta)$, thus $\sigma \in \Sigma$. Next, assume that $s(\beta) = T$. Then, $\Sigma \vdash \beta$, so $\Sigma \vdash (\beta \to (\alpha \to \beta))$, thus $\Sigma \vdash (\alpha \to \beta)$.

For the converse direction, assume that $(\alpha \to \beta \in \Sigma)$. If $\neg\alpha \in \Sigma$ then $s(\alpha) = F$, so $s(\alpha \to \beta) = T$. The last case is $\alpha \in \Sigma$. Applying modus ponens, $\Sigma \vdash \beta$, so $\beta \in \Sigma$ and $s(\beta) = T$ by the inductive hypothesis, so $s(\alpha \to \beta) = T$. It follows by induction that s satisfies Σ.

Now assume that s' is another assignment that satisfies Σ and let p be an arbitrary propositional variable. If $p \in \Sigma$ then $s'(p) = T$, but also $s(p) = T$. If $p \notin \Sigma$ then $\neg p \in \Sigma$ so $s'(\neg p) = T$ and $s'(p) = F$, and we also have $s(p) = F$. Since s and s' agree on every propositional variable, they must be the same function, so that s is unique. ☐

Theorem 1.3 (Completeness Theorem)**.** *Let Σ be a set of p-formulas and ϕ be a p-formula. Then, $\Sigma \models \phi \implies \Sigma \vdash \phi$.*

Proof. If we can show that any consistent set of p-formulas is satisfiable the result follows by Lemma 1.8, so let Δ be one such set. Since $\mathcal{L}_P$ only has countably many symbols and every $\mathcal{L}_P$ string is finite, there are only countably many p-formulas. Thus, we can fix a list of the p-formulas as follows:

$$\phi_0, \phi_1, \phi_2, \ldots$$

This can be done so that every p-formula occurs in the list exactly once.

Let $\Sigma_0 := \Delta$ and define Σ_{n+1} recursively as

$$\Sigma_{n+1} := \begin{cases} \Sigma_n \cup \{\phi_n\} & \Sigma_n \vdash \phi_n \\ \Sigma_n \cup \{\neg\phi_n\} & \Sigma_n \nvdash \phi_n \end{cases}$$

We argue by induction that each Σ_n is consistent. The base case follows from the assumption that Δ is consistent, so assume that Σ_n is consistent. If $\Sigma_n \nvdash \phi_n$, $\Sigma_{n+1} = \Sigma_n \cup \{\neg\phi_n\}$ is consistent by Lemma 1.7. The other case is $\Sigma \vdash \phi_n$, but then $\Sigma_{n+1} = \Sigma_n \cup \{\phi_n\}$ is clearly consistent.

Define $\Sigma := \bigcup_{n \in \mathbb{N}} \Sigma_n$. Clearly $\Sigma_0 = \Delta \subseteq \Sigma$. Assume for contradiction that Σ is inconsistent and fix some deduction D of $\perp$ from Σ. Since D is finite, there are only finitely many assumptions (i.e elements of Σ) used in D, so that there is some $N \in \mathbb{N}$ such that Σ_N includes all of those assumptions. Thus, $\Sigma_N \vdash \perp$. But we have already shown that Σ_N must be consistent, so we have our contradiction.

Also, given any p-formula ψ, there is some natural n such that $\phi_n :\equiv \psi$, so one of ψ or $\neg\psi$ are in Σ. Since Σ is consistent, it cannot be the case that both $\psi, \neg\psi \in \Sigma$, so Σ is complete. By Lemma 1.11, there is an assignment s that satisfies Σ. Since $\Delta \subseteq \Sigma$, s also satisfies Δ, thus Δ is satisfiable and we are done. $\square$

Definition 1.11. *A set Γ of p-formulas is finitely satisfiable if and only if all of its finite subsets are satisfiable.*

Theorem 1.4 (Compactness Theorem)**.** *A set Γ of p-formulas is satisfiable if and only if it is finitely satisfiable.*

Proof. The forward direction is trivial, so we focus on the converse. Assume that Γ is not satisfiable. It follows vacuously that $\Gamma \models \perp$, so $\Gamma \vdash \perp$ by the Completeness Theorem. Since every proof is finite, there must be some $\Gamma_0 \subseteq \Gamma$ such that $\Gamma_0 \vdash \perp$. By the Soundness Theorem, $\Gamma_0 \models \perp$, therefore it is not satisfiable. $\square$

2 First-order Logic

Throughout this section $\mathcal{L}$ will denote an arbitrary first-order language.

Definition 2.1. *An $\mathcal{L}$-string t is called an $\mathcal{L}$-term if and only if*

1. $t \in Vars$, *or*
2. t *is a constant symbol of* $\mathcal{L}$, *or*
3. $t :\equiv ft_1, \dots t_n$ *where* $t_1, \dots t_n$ *are* $\mathcal{L}$*-terms and* f *is an* n*-ary function symbol from* $\mathcal{L}$.

Definition 2.2. *An* $\mathcal{L}$*-string* ϕ *is called an* $\mathcal{L}$*-formula if and only if*

1. $\phi :\equiv = t_1 t_2$, *where* t_1, t_2 *are* $\mathcal{L}$*-terms, or*
2. $\phi :\equiv Rt_1, \dots, t_n$ *where* $t_1, \dots t_n$ *are* $\mathcal{L}$*-terms and* R *is an* n*-ary relation symbol from* $\mathcal{L}$, *or*
3. $\phi :\equiv (\alpha \to \beta)$, *where* α, β *are* $\mathcal{L}$*-formulas, or*
4. $\phi :\equiv (\neg \alpha)$, *where* α *is a* $\mathcal{L}$*-formula, or*
5. $\phi :\equiv (\forall x)(\alpha)$, *where* x *is a variable and* α *is an* $\mathcal{L}$*-formula.*

Definition 2.3. *We say that* $\mathfrak{A}$ *is an* $\mathcal{L}$*-structure if and only if* $\mathfrak{A}$ *is a (possibly infinite) tuple* $(A, c_1^{\mathfrak{A}}, c_2^{\mathfrak{A}}, \dots, f_1^{\mathfrak{A}}, f_2^{\mathfrak{A}}, \dots, R_1^{\mathfrak{A}}, R_2^{\mathfrak{A}}, \dots)$ *where* A *is a nonempty set,* $c_1, c_2, \dots$ *are all of the constant symbols of* $\mathcal{L}$, $f_1, f_2 \dots$ *are the functions symbols of* $\mathcal{L}$ *and similarly for* $R_1, R_2, \dots$. *We also require that:*

1. *For each constant symbol* $c \in \mathcal{L}$, $c^{\mathfrak{A}} \in A$;
2. *For each* n*-ary function symbol* $f \in \mathcal{L}$, $f^{\mathfrak{A}} : A^n \to A$, *i.e* $f^{\mathfrak{A}}$ *is a function from* A^n *to* A.
3. *For each* n*-ary relation symbol* $R \in \mathcal{L}$, $R^{\mathfrak{A}} \subseteq A^n$.

Definition 2.4. *Let* $\mathfrak{A}$ *be an* $\mathcal{L}$ *structure. An assignment function is a function with domain* $Vars$ *and codomain* A. *Also, for every assignment function* $s : Vars \to A$ *every* $a \in A$ *and every* $x \in Vars$, *we define the function* $s[x|a] : Vars \to A$ *as*

$$s[x|a](v) = \begin{cases} a, & \text{if } v = x \\ s(v) & \text{if } v \neq x. \end{cases}$$

Definition 2.5. *Let* $\mathfrak{A}$ *and* $\mathfrak{B}$ *be* $\mathcal{L}$*-structures. We say that* $\mathfrak{A}$ *is isomorphic to* $\mathfrak{B}$ *if and only if there is a bijection* $f : A \to B$ *such that*

1. $f(c^{\mathfrak{A}}) = c^{\mathfrak{B}}$ *for every constant symbol* $c \in \mathcal{L}$,
2. $f(g^{\mathfrak{A}}(a_1, \dots, a_n)) = g^{\mathfrak{B}}(f(a_1), \dots, f(a_n))$ *for every* n*-ary function symbol* $g \in L$ *and every* $(a_1, \dots, a_n) \in A^n$,
3. $(a_1, \dots, a_n) \in R^{\mathfrak{A}} \iff (f(a_1), \dots, f(a_n)) \in R^{\mathfrak{B}}$ *for every* n*-ary relation symbol* $R \in \mathcal{L}$ *and every* $(a_1, \dots, a_n) \in A^n$.

Definition 2.6. *Let* $\mathfrak{A}$ *and* $\mathfrak{B}$ *be* $\mathcal{L}$*-structures. We say that* $\mathfrak{A}$ *is elementarily equivalent to* $\mathfrak{B}$ *if and only if given any* $\mathcal{L}$*-formula* ϕ, $\mathfrak{A} \models \phi \iff \mathfrak{B} \models \phi$.

Lemma 2.1. *Let* $\mathfrak{A}$ *and* $\mathfrak{B}$ *be isomorphic* $\mathcal{L}$*-structures. Given an isomorphism* $f : A \to B$ *and an assignment* $s : Vars \to B$, *we have*

$$\mathfrak{B} \models \phi[s] \iff \mathfrak{A} \models \phi[f^{-1} \circ s].$$

Proof. Fix an isomorphism $f : A \to B$. Throughout the lemma we will write v instead of $f^{-1} \circ s$. We will show by induction on the complexity of ϕ that for all assignments $s : Vars \to B$, $\mathfrak{B} \models \phi[s] \iff \mathfrak{A} \models \phi[v]$.

For the base case, assume $\phi :\equiv Rt_1 \dots t_n$ where R is an n-ary relation symbol and $t_1, \dots, t_n$ are $\mathcal{L}$-terms. Then

$$\begin{aligned}
&\mathfrak{B} \models Rt_1 \dots t_n[s] && \iff \\
&(\bar{s}(t_1), \dots, \bar{s}(t_n)) \in R^{\mathfrak{B}} && \iff \\
&(f(f^{-1}(\bar{s}(t_1))), \dots, f(f^{-1}(\bar{s}(t_n)))) \in R^{\mathfrak{B}} && \iff \\
&(\bar{v}, \dots, \bar{v}(t_n)) \in R^{\mathfrak{A}} && \iff \\
&\mathfrak{A} \models \phi[v].
\end{aligned}$$

The cases $\phi :\equiv \alpha \vee \beta$ and $\phi :\equiv (\neg\alpha)$ are straightforward. For the case $\phi :\equiv \forall x\psi$, assume the inductive hypothesis and notice that

$$\begin{aligned}
&\mathfrak{B} \models \forall x\psi[s] && \iff \\
&\mathfrak{B} \models \psi[s[x|b]] \text{ for every } b \in B. && \iff \\
&\mathfrak{A} \models \psi[f^{-1} \circ s[x|b]] \text{ for every } b \in B. && \iff \\
&\mathfrak{A} \models \psi[(f^{-1} \circ s)[x|a]] \text{ for every } a \in A. && \iff \\
&\mathfrak{A} \models \forall x\psi[v].
\end{aligned}$$

The result follows by induction.

□

Theorem 2.1. *If $\mathfrak{A}$ and $\mathfrak{B}$ are isomorphic $\mathcal{L}$-structures, then they are elementarily equivalent.*

Proof. Let $\mathfrak{A}$ and $\mathfrak{B}$ be isomorphic $\mathcal{L}$-structures and assume that $\mathfrak{A} \models \phi$. Let $s : Vars \to B$ be an arbitrary assignment function into $\mathfrak{B}$. By Lemma 2.1, we have $\mathfrak{B} \models \phi[s] \iff \mathfrak{A} \models \phi[f^{-1} \circ s]$, where $f : A \to B$ is an isomorphism. Since $\mathfrak{A} \models \phi[s']$ for any assignment s', the result follows. The converse follows similarly. □

3 Computability Theory

Definition 3.1. *We define $\mathcal{O} : \varnothing \to \mathbb{N}$ as the function with no arguments that returns 0. $\mathcal{S} : \mathbb{N} \to \mathbb{N}$ is such that $\mathcal{S}(x) = x + 1$ for every $x \in \mathbb{N}$. For each $n \in \mathbb{N}$ we define the projection function $\mathcal{I}_i^n : \mathbb{N}^n \to \mathbb{N}$ for each $1 \leqslant i \leqslant n$ as $\mathcal{I}_i^n(x_1, x_2, \dots, x_i, \dots, x_n) = x_i$ for all $x_1, \dots x_n \in \mathbb{N}$.*

The functions above are collectively called the initial functions.

Definition 3.2. *We define the set of computable functions as follows:*

1. *The initial functions are computable.*

2. *If h is a computable function of arity m (possibly 0) and $g_1, \ldots, g_m$ are functions of arity n, then $f(x_1, \ldots x_n) = h(g_1(\underset{\sim}{x}), \ldots, g_m(\underset{\sim}{x}))$.*
3. *If g is a computable function of arity n and h is a computable function of arity $n+2$, then the function f given by*

$$f(\underset{\sim}{x}, 0) = g(\underset{\sim}{x})$$
$$f(\underset{\sim}{x}, y+1) = h(\underset{\sim}{x}, y, f(\underset{\sim}{x}, y))$$

   is a computable function.
4. *If g is a computable function of arity $n+1$, then $f(\underset{\sim}{x}, y) = (\mu i \leqslant y)(g(\underset{\sim}{x}, i))$ is computable.*

4 Exercises

Exercise 7.3.8.

(a) The statement clearly holds for the initial functions, so assume inductively that $f(\underset{\sim}{x}) = h(g_1(\underset{\sim}{x}), \ldots, g_m(\underset{\sim}{x}))$ where g and h meet the inductive hypothesis. Then $f(\underset{\sim}{x}) \leqslant g_i(\underset{\sim}{x}) + K_h \leqslant x_j + K_g + K_h$. The result follows by setting $K := K_g + K_h$.

(b) The result follows easily if f is of rank 0, so assume that it is not. Then $f()$

Exercise 7.3.9.

(a) To show that $A(y, x)$ is a natural number we induct on y. The base case is straightforward, so assume that $A(y, x)$ is defined for all x. To show that $A(y+1, x)$ is defined for all x, we now induct on x. For the base case, $A(y+1, 0) = 2$ by definition, so assume that $A(y+1, x)$ is defined. Then $A(y+1, x+1) = A(y, A(y+1, x))$ by definition. But $A(y+1, x)$ is defined by the second inductive hypothesis therefore $A(y, A(y+1, x))$ is defined by the first inductive hypothesis.

It is easy to see that $A(1, x) = 2x + 2$ and $A(2, x) = 2^{x+2} - 2 > 2^x$ by induction.

5 Turing Machines

Definition 5.1. *We will denote the set $\{0, 1\}$ by S, $\{-1, 1\}$ by D, and any non-empty string will be called a state.*

Definition 5.2. *A Turing Machine is a tuple (Q, T) where Q is a set of states containing the string A but not containing H, and $T : Q \times S \to Q \cup \{H\} \times S \times D$ is a transition function.*

Definition 5.3. *Given a Turing Machine* $\mathrm{TM} = (Q, T)$ *we define a function* $\sigma_{\mathrm{TM}} : S^{\mathbf{Z}} \times \mathbb{N} \to S^{\mathbf{Z}} \times Q \cup H \times \mathbf{Z}$ *called* TM*'s step function inductively. Fix some* $I_0 : \mathbf{Z} \to S$*. For the base case, let* $\sigma_{\mathrm{TM}}(I_0, 0) := (I_0, A, 0)$*. Now assume that* $\sigma_{\mathrm{TM}}(I_0, n) = (I_n, q_n, h_n)$ *is defined. If* $q_n = H$*, then let* $\sigma_{\mathrm{TM}}(I_0, n+1) := \sigma_{\mathrm{TM}}(I_0, n)$*. Otherwise, let* $T(q_n, I_n(h_n)) = (q_{n+1}, s_{n+1}, d_{n+1})$ *and let* I_{n+1} *be the same function as* I_n*, except possibly at* h_n*, where we set* $I_{n+1}(h_n) = s_{n+1}$*. Then define* $\sigma_{\mathrm{TM}}(I_0, n+1) := (I_{n+1}, q_{n+1}, h_n + d_{n+1})$*.*

Definition 5.4. *We say that a Turing Machine* TM *halts on input* I *if and only if we have* $\sigma_{\mathrm{TM}}(I, n) = (I', H, z)$ *for appropriate* I' *and* z *and some* $n \in \mathbb{N}$*. In that case we also say that* TM *halts in* n *steps.*

Notice that the step function of a Turing Machine that halts on some input is eventually constant at that input, but the converse is not always true.