Project for 236331 (Fall 2021): lower bound of the size of reduction sequences in Feferman-Vaught theorem

Shengyu Huang

May 31, 2022

1 Introduction

Let τ be a fixed and finite relational vocabulary. We use $\mathrm{FOL}(\tau)$ for the set of τ-sentences in first-order logic and $\mathrm{FOL}^q(\tau)$ for sentences in $\mathrm{FOL}(\tau)$ of quantifier rank at most $q \in \mathbb{N}$.

We define the size (or length) $|\varphi|$ of a first-order formula φ as the number of vertices of φ's syntax tree. In this report, we will investigate the size of a reduction sequence in a computational proof of Feferman-Vaught theorem [3] as reproduced below.

Theorem 1.1. *[5, Theorem 1.6] For two τ-structures $\mathcal{A}, \mathcal{B}$ and a first-order logic formula θ of quantifier rank $q \in \mathbb{N}$, one can compute effectively a reduction sequence and a boolean function $B_\theta : \{0,1\}^{2m} \to \{0,1\}$ to determine whether $\mathcal{A} \sqcup \mathcal{B} \models \theta$.*

Specifically, a reduction sequence is a sequence of formulas of the form $\langle \psi_1^{\mathcal{A}}, ..., \psi_m^{\mathcal{A}}, \psi_1^{\mathcal{B}}, ..., \psi_m^{\mathcal{B}} \rangle$, where $\psi_i^{\mathcal{A}}, \psi_i^{\mathcal{B}} \in FOL^q(\tau)$ for $i \in \{1, ..., m\}$.

For the boolean function B_θ, we have $\mathcal{A} \sqcup \mathcal{B} \models \theta$ iff $B_\theta(b_1^{\mathcal{A}}, ..., b_m^{\mathcal{A}}, b_1^{\mathcal{B}}, ..., b_m^{\mathcal{B}}) = 1$, where $b_j^{\mathcal{A}} = 1$ iff $\mathcal{A} \models \psi_j^{\mathcal{A}}$ and $b_j^{\mathcal{B}} = 1$ iff $\mathcal{C} \models \psi_j^{\mathcal{B}}$.

We show two ways of constructing a reduction sequence in this section, one inductively and one using Hintikka sentences. We define the size of a reduction sequence as the size of the biggest formula in the sequence. In Section 4, we will derive a lower bound of the size of reduction sequences and show that using Hintikka sentences to construct a reduction sequence, although a naive approach, is the best we can do in some cases. In Section 3, we introduce a class of finite trees, upon which the lower bound in Section 4 is derived. Because the lower bound would be phrased in terms of elementary functions, we will in the next section first introduce the definitions of primitive recursive functions and elementary functions.

Constructing a reduction sequence inductively The proof of Theorem 1.1 given in [5] constructs a reduction sequence inductively. For an atomic formula

θ, a reduction sequence can be very short. Consider the same example in [5], where we are given two ordered graphs $G_1 = (V_1, E_1, <_1)$ and $G_2 = (V_2, E_2, <_2)$. The disjoint union $G = (V, E, <)$ of G_1 and G_2 is the ordered sum of these two graphs with $V = V_1 \sqcup V_2$, $E = E_1 \sqcup E_2$, and $< \, = \, <_1 \sqcup <_2 \sqcup (V_1 \times V_2)$. Since free variables are involved in the inductive construction, we assume $z : Vars \to V$ is an assignment of the free variables to the vertices.

If $\theta = E(u, v)$, then a reduction sequence is simply $\langle E_1(u, v), E_2(u, v)\rangle$ and the boolean function is $b_1^{G_1} \vee b_1^{G_2}$, where $b_1^{G_1} = 1$ iff $E_1(u, v)$ holds and $b_1^{G_2} = 1$ iff $E_2(u, v)$ holds. With $\theta = E(u, v)$, only the cases where $z(u)$ and $z(v)$ are both in V_1 or V_2 are relevant.

Constructing a reduction sequence using Hintikka sentences In class, we adopted a more naive approach to prove Theorem 1.1 by using Hintikka sentences. Since there are only finitely many Hintikka sentences of quantifier at most q, we can enumerate them as $\{h_1^q, ..., h_\alpha^q\}$. By Feferman-Vaught theorem, there is a function $g : [\alpha]^2 \to [\alpha]$ such that $\mathcal{A} \models h_i^q$ and $\mathcal{B} \models h_j^q$ iff $\mathcal{A} \sqcup \mathcal{B} \models h_{g(i,j)}^q$.

If θ is a Hintikka sentence h_k^q for $\mathcal{A} \sqcup \mathcal{B}$, a reduction sequence for θ can be given by combining two sequences:

$$(h_i^q(\mathcal{A}) : \mathcal{A} \models h_i^q \text{ and } \exists j g(i, j) = k) \text{ and } (h_j^q(\mathcal{B}) : \mathcal{B} \models h_j^q \text{ and } \exists i g(i, j) = k).$$

The boolean function B_θ is

$$\bigvee_{(i,j)\in F_k} (h_i^q(\mathcal{A}) \wedge h_j^q(\mathcal{B})), \text{ where} F_k = \{(i, j) \in [\alpha]^2 : g(i, j) = k\}.$$

If θ is not a Hintikka sentence, it can be written as a disjunction of Hintikka sentences $\bigvee_{i, h_i^q \to \theta} h_i^q$. We can then proceed by having a reduction sequence for each h_i^q and piece all reduction sequences together.

2 Primitive recursive and elementary functions

In the following text, we denote $\mathbf{0}$ the zero function, i.e, $\mathbf{0}(x) = 0$ for all x, and U_i^n $(1 \le i \le n)$ the projection function, where $U_i^n(x_1, x_2, ..., x_n) = x_i$.

Definition 2.1 (Primitive recursive function)**.** The class *primitive recursive functions* is the smallest class of functions that includes $\mathbf{0}$, the successor function $x + 1$, $\mathbf{U}_i^n$ and is closed under substitution and recursion.

There are nevertheless recursive (or equivalently, total computable) functions that are not primitive recursive. One famous example is the Ackerman function. One version of Ackerman function is defined as

$$\begin{aligned} A(0, n) &= n + 1 \\ A(m + 1, 0) &= A(m, 1) \\ A(m + 1, n + 1) &= A(m, A(m + 1, n)). \end{aligned}$$

An important subclass of primitive recursive functions is the class of elementary functions. Roughly speaking, elementary functions can be obtained by iteration of the operations of ordinary arithmetic. We define it as follows.

Definition 2.2 (Elementary function)**.** The class of elementary functions is the smallest class such that it includes $x + 1$, U_i^n, $x - y$, $x + y$, xy and is closed under substitution and the operations of forming bounded sums and bounded products. Specifically, if $f(\vec{x}, z)$ is an elementary function, then $\sum_{z<y} f(\vec{x}, z)$ and $\prod_{z<y} f(\vec{x}, z)$ are also elementary, where

$$\sum_{z<y} f(\vec{x}, z) = \begin{cases} 0 & y = 0 \\ \sum_{z<y-1} f(\vec{x}, z) + f(\vec{x}, y) & \text{otherwise} \end{cases} \tag{1}$$

and

$$\prod_{z<y} f(\vec{x}, z) = \begin{cases} 1 & y = 0 \\ \prod_{z<y-1} f(\vec{x}, z) \cdot f(\vec{x}, y) & \text{otherwise} \end{cases} \tag{2}$$

Definition 2.3 (Tower function)**.** For $x, y \in \mathbb{N}$,

$$\text{Tower}(x, y) = \begin{cases} y & x = 0 \\ 2^{\text{Tower}(x-1,y)} & \text{otherwise} \end{cases} \tag{3}$$

For succinctness, we abbreviate Tower$(x, 1)$ as Tower(x) for now on.

Theorem 2.1. *If $f(\vec{x})$ is elementary, there is a number k such that for all x, $f(\vec{x}) \leq Tower(k, max(\vec{x}))$, where $max(\vec{x})$ is the same as the L-infinity norm of $\vec{x}$.*

Theorem 2.1 can be proved by induction. We skip the proof here for brevity. The reader can seek [1, Chapter 12, Theorem 4.7] for a detailed proof.

Theorem 2.2. *The function $f(x) := Tower(x, x)$ is primitive recursive but not elementary.*

Proof. Tower(x, x) is obviously a primitive recursive function. To see that f is not elementary, notice that $f(k+1) = \text{Tower}(k+1, k+1) > \text{Tower}(k, k+1)$ for every k. Therefore, there is no k such that $f(x) \leq \text{Tower}(k, x)$ for all x. □

3 Encoding numbers by trees

Encoding numbers by trees gives us some small first-order formulas that will come in handy for the proof in Section 4. In this section, we will develop several first-order logic formulas that are satisfied iff a tree is indeed a valid encoding of some natural number or the number encoded by the tree has some intuitive properties.

For every number $n \in \mathbb{N}$, we define a tree $\mathcal{T}(n)$ as follows.

- $\mathcal{T}(0)$ is the node-node tree.
- For $n \geq 1$ the tree $\mathcal{T}(n)$ is obtained by creating a new root and attaching it to every tree $\mathcal{T}(i)$ such that the i-th bit in the binary representation of n is 1.

Recall that the height of a tree $\mathcal{T}(n)$ is the length of the longest path in $\mathcal{T}(n)$. Observe that we can encode numbers as big as Tower(h) with a tree of height at most h. In fact, the other way is also true, i.e., if the height of a tree $\mathcal{T}(n)$ is at most h, n must be less than or equal to Tower(h).

Lemma 3.1. *[4, Lemma 10.20] For all $h, n \geq 0$, height$(\mathcal{T}(n)) \leq h \iff n \leq$ Tower(h).*

We now introduce three first-order formulas $encoding_h(x)$, $succ_h(x,y)$, and $max_h(x)$ that describe the properties of the tree structures defined above.

In the following text, we use E to denote a binary relation symbol and view trees as being directed from the root to leafs. For a directed graph $\mathcal{A} = (A, E^{\mathcal{A}})$ and an $a \in A$, we use A_a to be the set of all vertices b such that there is a path from a to b. $\mathcal{A}_a$ is the induced substructure of $\mathcal{A}$ with universe A_a. Note that there is always a path from a node to itself.

Intuitively, $encoding_h(x)$ holds, if a tree with x as its root is a valid encoding of some number $n <$ Tower(h). By Lemma 3.1, it also implies that such a tree has a height strictly less than h. The meanings for $succ_h(x,y)$ and $max_h(x)$ are self-explanatory, and we will give their formal definitions in Lemma 3.4 and Lemma 3.3, respectively.

Lemma 3.2. *For all structures $\mathcal{A} = (A, E^{\mathcal{A}})$ and $x \in A$ there is a first-order formula $encoding_h(x)$ of length $O(h^2)$ for every $h \geq 0$ such that $\mathcal{A} \models encoding_h(x)$ iff $\mathcal{A}_x$ is isomorphic to $\mathcal{T}(i)$ for some $i \in \{0, ..., \text{Tower}(h) - 1\}$.*

Proof. For an arbitrary structure $\mathcal{A} = (A, E^{\mathcal{A}})$ and $x \in A$, we define $encoding_h(x)$ inductively as follows.

When $h = 0$, Tower$(h) = 1$. We want $\mathcal{A} \models encoding_0(x)$ iff $\mathcal{A}_x$ is isomorphic to $\mathcal{T}(0)$. Since $\mathcal{T}(0)$ is the one-node tree, we can simply choose $encoding_0(x) := \neg\exists y E(x,y)$.

For $h \geq 1$, we define $encoding_h(x) := \forall y\big(E(x,y) \to encoding_{h-1}(y)\big) \wedge \forall y \forall y'\big((E(x,y) \wedge E(x,y') \wedge \neg y = y') \to \neg eq_{h-1}(y,y')\big)$.

We introduce $eq_{h-1}(y,y')$ to express the notion of equality. Formally, for every structures $\mathcal{A} = (A, E^{\mathcal{A}})$ and $x, y \in A$ if there are $m, n <$ Tower(h) such that $\mathcal{A}_x$ and $\mathcal{A}_y$ are isomorphic to $\mathcal{T}(m)$ and $\mathcal{T}(n)$, respectively, then $\mathcal{A} \models eq_h(x,y) \iff m = n$. In addition, the length of $eq_h(x,y)$ is $O(h)$ for every $h \geq 1$ [4, Lemma 10.21].

Based on the definition of $encoding_h(x)$, we know there exists a constant $c > 0$ such that $|encoding_h| \leq |encoding_{h-1}| + |eq_{h-1}| + c$. Solving this recurrence equation gives us

$$|encoding_h| \leq |encoding_0| + \sum_{i=0}^{h-1} |eq_i| + c \cdot h = O(h^2).$$

□

In a similar way, we can show the existence of $succ_h(x, y)$ and $max_h(x)$ and derive the upper bound for their formula lengths. We refer readers to the original paper [3] for more detailed proofs.

Lemma 3.3. *[2, Lemma 3.4] For all structures $\mathcal{A} = (A, E^{\mathcal{A}})$ and $x, y \in A$ there is a first-order formula $succ_h(x, y)$ of size $O(h^3)$ for every $h \geq 0$ such that $\mathcal{A} \models succ_h(x, y)$ iff $m + 1 = n$, where $m, n < Tower(h)$ and $\mathcal{A}_x$ and $\mathcal{A}_y$ are isomorphic to $\mathcal{T}(m)$ and $\mathcal{T}(n)$, respectively.*

Lemma 3.4. *[2, Lemma 3.4] For all structures $\mathcal{A} = (A, E^{\mathcal{A}})$ and $x \in A$ there is a first-order formula $max_h(x)$ of size $O(h^4)$ for every $h \geq 0$ such that $\mathcal{A} \models max_h(x)$ iff $\mathcal{A}_x$ is isomorphic to $\mathcal{T}(Tower(h) - 1)$.*

4 Size of reduction sequences in Feferman-Vaught theorem

Recall that $\mathrm{FOL}(\tau)$ denote the set of τ-sentences in first-order logic and $\mathrm{FOL}^q(\tau)$ denote the sentences of quantifier rank at most $q \in \mathbb{N}$ in $\mathrm{FOL}(\tau)$. For a class of τ-structures K, let $\mathrm{Th}(K)$ be the set of sentences in $\mathrm{FOL}(\tau)$ that are true in all $\mathcal{U} \in K$. We write $\mathrm{Th}(\mathcal{U})$ if $K = \{\mathcal{U}\}$. Similarly to $\mathrm{FOL}^q(\tau)$, we use $\mathrm{Th}^q(\mathcal{U})$ to denote the sentences in $\mathrm{Th}(\mathcal{U})$ of quantifier rank at most q. Additionally, sentences in $\mathrm{Th}(\mathcal{U})$ that are of length at most l are written as $\mathrm{Th}_l(\mathcal{U})$.

For two structures $\mathcal{A}$ and $\mathcal{B}$, constructing a reduction sequence for θ of quantifier rank at most $q \in \mathbb{N}$ requires us to look at sentences in $\mathrm{Th}^q(\mathcal{A})$ and $\mathrm{Th}^q(\mathcal{B})$. Since there are only finitely many Hintikka sentences of quantifier rank q, it follows that there is a function f such that we can construct the reduction sequence for θ using formulas in $\mathrm{Th}_{f(|\theta|)}(\mathcal{A})$ and $\mathrm{Th}_{f(|\theta|)}(\mathcal{B})$.

In Section 1's Theorem 1.1, we used Hintikka sentences to construct a reduction sequence for a formula θ of length n. This makes $f(n) = O(\mathrm{Tower}(n))$, but maybe surprisingly, this upper bound is essentially tight for some cases. Specifically, the size of a reduction sequence for a formula θ is not bounded by any elementary function. By Theorem 2.1, it means $f(|\theta|)$ is not bounded by $O(\mathrm{Tower}(k))$ for any *fixed* k.

Theorem 4.1. *[2, Theorem 6.1] Let $\mathfrak{T}$ denote the class of all finite trees. There is no elementary function f such that the following holds for all trees $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathfrak{T}$ and $n \geq 1$: if $Th_{f(n)}(\mathcal{A}) = Th_{f(n)}(\mathcal{B})$, then $Th_n(\mathcal{A} \sqcup \mathcal{C}) = Th_n(\mathcal{B} \sqcup \mathcal{C})$.*

Proof. Assume f elementary for contradiction. For some $h \in \mathbb{N}$, we will give a first-order formula φ_h and $\mathcal{A}, \mathcal{B}, \mathcal{C} \in \mathfrak{T}$ such that $\mathcal{A} \sqcup \mathcal{C} \models \varphi_h$ and $\mathcal{B} \sqcup \mathcal{C} \not\models \varphi_h$ while $\mathrm{Th}_{f(|\varphi_h|)}(\mathcal{A}) = \mathrm{Th}_{f(|\varphi_h|)}(\mathcal{B})$.

Specifically,

$$\varphi_h := \forall x\big(encoding_h(x) \to \big(max_h(x) \vee \exists y\ succ_h(x, y)\big)\big).$$

Intuitively, φ_h says for all nodes x in a finite tree structure $\mathcal{A}$ with a height strictly less than h, if $\mathcal{A}_x$ is a valid coding of some number $n \in \{0, ..., \text{Tower}(h) - 1\}$, $\mathcal{A}_x$ either encodes the largest number, i.e., $\text{Tower}(h) - 1$ (c.f. Lemma 3.4 and Lemma 3.1), or there exists some $y \in \mathcal{A}$ such that the number encoded by $\mathcal{A}_y$ is the successor of the number encoded by $\mathcal{A}_x$.

Directly following from Lemma 3.2, Lemma 3.3, and Lemma 3.4, $|\varphi_h| \leq c \cdot h^4$ for some $c \geq 1$ and for all $h \geq 0$. We fix an h such that $|\text{Th}_{f(|\varphi_h|)}(\mathfrak{T})| \leq \text{Tower}(h) - 1$. Since there are only exponentially many first-order sentences of a given length and f is bounded by $\text{Tower}(k)$ for some $k \in \mathbb{N}$ (c.f. Theorem 2.1), we are assured to find such an h.

Construction of $\mathcal{A}, \mathcal{B} \in \mathfrak{T}$ such that $\textbf{Th}_{f(|\varphi_h|)}(\mathcal{A}) = \textbf{Th}_{f(|\varphi_h|)}(\mathcal{B})$ We continue to use $\mathcal{T}(j)$ to denote a finite tree that encodes the natural number $j \in \mathbb{N}$ as introduced in Section 3.

In order to find two finite tree structures $\mathcal{A}, \mathcal{B}$ that satisfy the same sentences of length up to $f(|\varphi_h|)$, first we construct a set of structures $\mathcal{F} := \bigcup_{i=0}^{\text{Tower}(h)-1} \mathcal{F}_i$, where $\mathcal{F}_i := \{\mathcal{T}(j) \mid i \leq j \leq \text{Tower}(h) - 1\}$ for $0 \leq i \leq \text{Tower}(h) - 1$. Next, we construct a set of new trees $\mathcal{U} := \bigcup_{i=0}^{\text{Tower}(h)-1} \mathcal{U}_i$, where $\mathcal{U}_i$ connects a new root to all the roots of trees in $\mathcal{F}_i$.

Since $|\mathcal{U}| = \text{Tower}(h)$ and $|\text{Th}_{f(\varphi_h|)}(\mathfrak{T})| \leq \text{Tower}(h) - 1$, by pigeonhole principle, there are two numbers a, b with $0 \leq a < b \leq \text{Tower}(h) - 1$ such that $\text{Th}_{f(|\varphi_h|)}(\mathcal{U}_a) = \text{Th}_{f(|\varphi_h|)}(\mathcal{U}_b)$. In other words, $\mathcal{A} = \mathcal{U}_a$ and $\mathcal{B} = \mathcal{U}_b$.

Find a $C \in \mathfrak{T}$ such that $\mathcal{A} \sqcup \mathcal{C} \models \varphi_h$ and $\mathcal{B} \sqcup \mathcal{C} \not\models \varphi_h$ We simply set $C = \mathcal{T}(a-1)$ and show that $\mathcal{U}_a \sqcup \mathcal{T}(a-1) \models \varphi_h$ and $\mathcal{U}_b \sqcup \mathcal{T}(a-1) \not\models \varphi_h$.

To see $\mathcal{U}_a \sqcup \mathcal{T}(a-1) \models \varphi_h$, we first check the root of $\mathcal{U}_a$ and then rest of the nodes. Because $\mathcal{T}(\text{Tower}(h) - 1) \in \mathcal{F}_a$, the root of $\mathcal{U}_a$ now encodes a number bigger than $\text{Tower}(h) - 1$ and hence the root cannot satisfy $\mathit{encoding}_h(x)$. In other words, the root of $\mathcal{U}_a$ satisfies $\mathit{encoding}_h(x) \to \big(\mathit{max}_h(x) \vee \exists y\ \mathit{succ}_h(x, y)\big)$. The rest of the nodes in $\mathcal{U}_a$ are nodes in $\mathcal{F}_a$. Observe that φ_h holds for every $\mathcal{F}_i$ with $i \in \{0, ..., \text{Tower}(h) - 1\}$. Since $\mathcal{F}_a \sqcup \mathcal{T}(a-1) = \mathcal{F}_{a-1}$, $\mathcal{F}_a \sqcup \mathcal{T}(a-1) \models \varphi_h$. Overall, $\mathcal{U}_a \sqcup \mathcal{T}(a-1) \models \varphi_h$

To see $\mathcal{U}_b \sqcup \mathcal{T}(a-1) \not\models \varphi_h$, notice that the root of $\mathcal{T}(a-1)$ encodes a number strictly less than the maximum $\text{Tower}(h) - 1$. In addition, there does not exist a subtree isomorphic to $\mathcal{T}(a)$ in $\mathcal{U}_b \sqcup \mathcal{T}(a-1)$. Hence, the root of $\mathcal{T}(a-1)$ in $\mathcal{U}_b \sqcup \mathcal{T}(a-1)$ not does satisfy φ_h.

Wrap-up We have shown $\mathcal{U}_a \sqcup \mathcal{T}(a-1) \models \varphi_h$ and $\mathcal{U}_b \sqcup \mathcal{T}(a-1) \not\models \varphi_h$. Namely, $\text{Th}_{|\varphi_h|}(\mathcal{U}_a \sqcup \mathcal{T}(a-1)) \neq \text{Th}_{|\varphi_h|}(\mathcal{U}_b \sqcup \mathcal{T}(a-1))$, while $\text{Th}_{f(|\varphi_h|)}(\mathcal{U}_a) = \text{Th}_{f(|\varphi_h|)}(\mathcal{U}_b)$. Therefore, the assumption that f is elementary cannot hold.

□

References

[1] Nigel Cutland, *Computability: An introduction to recursive function theory*, Cambridge university press, 1980.

[2] Anuj Dawar, Martin Grohe, Stephan Kreutzer, and Nicole Schweikardt, *Model theory makes formulas large*, International Colloquium on Automata, Languages, and Programming, Springer, 2007, pp. 913–924.

[3] Solomon Feferman and Robert L Vaught, *The first order properties of products of algebraic systems*, Journal of Symbolic Logic **32** (1967), no. 2.

[4] Jörg Flum and Martin Grohe, *Parameterized complexity theory*, Springer, 2006.

[5] Johann A Makowsky, *Algorithmic uses of the feferman–vaught theorem*, Annals of Pure and Applied Logic **126** (2004), no. 1-3, 159–213.