A friendly tour of Parikh's theorem

Caleb Koch

May 2018

Abstract

Parikh's theorem [Par66] is a result in compatibility about the relationship between context free languages and regular languages. Since its publication in the 1960s, researchers have extended, generalized, reproved, and developed the theorem in various interesting directions. The purpose of this work is to introduce Parikh's theorem and its various proofs while giving a high level overview of related work. Some things we'll look at include a simplified proof from [Gol77], the generalization of the theorem due to Pilling [Pil73], and the further generalization to commutative Kleene algebras due to [HK99].

Contents

1 Preliminaries **1**
- 1.1 Purpose . . . 1
- 1.2 Definitions . . . 2

2 Parikh's theorem **3**
- 2.1 Parikh's original proof . . . 3
- 2.2 Simplified proof using strong pumping lemma . . . 4
- 2.3 Alternative proof by NFA construction . . . 5
- 2.4 Other alternative proofs and generalizations . . . 8

3 Pilling's result **9**

4 Hopkins & Kozen's Generalization **10**
- 4.1 Background on polynomials in commutative Kleene algebras . . . 11
- 4.2 Proof of main theorem . . . 12

1 Preliminaries

1.1 Purpose

This work is meant to serve as an introduction to Parikh's theorem and related work. The theorem itself is already covered in at least three standard texts in computability theory [HU90, LP97, Har78]; however there is not to the author's knowledge a comprehensive survey discussing the research prompted by the result. This paper serves to fill that gap by providing a brief introduction to these studies. We assume only a basic (perhaps first year)

background in computability. Elaborate proof details are largely left out in favor of proof sketches. The precise details of all results can be found in the literature cited.

1.2 Definitions

Parikh's theorem originated in a research report from 1961 and was published five years later in 1966. The result shows that looking merely at the number of occurrences of letters in words of some context free language (CFL) renders that language indistinguishable from a regular language. One immediate consequence is that CFLs over a single alphabet are regular.

Throughout we use the following notational conventions.

- $\Sigma = \{a_1, ..., a_n\}$ is a fixed finite alphabet for some $n > 0$.
- For a word $w \in \Sigma^*$, we denote $\#_i(w)$ as the number of occurrences of a_i in w.
- $G = (N, \Sigma, R, S)$ is a context free grammar (CFG) where N is the set of nonterminals, Σ is the set of terminals, R is the set of rules, and $S \in N$ is the start symbol.
- For a CFG G, $L(G)$ is the context free language (CFL) of G.
- m is the number of nonterminals in G, i.e. $m = |N|$.
- A one-step derivation of G is denoted with $\Rightarrow$, a multi-step derivation is denoted with $\overset{*}{\Rightarrow}$ (a derivation $S \Rightarrow S'$ is used for strings $S, S' \in (N \cup \Sigma)^*$ to denote the application of a single rule to some nonterminal in S to derive the new string S').
- For an automaton, we use δ to denote the transition function for a single letter and $\hat{\delta}$ to denote the inductive extension of δ to arbitrary words.

For the main result we begin with a few preliminary definitions. Throughout we assume $\Sigma = \{a_1, ..., a_n\}$ is a fixed finite alphabet for some $n > 0$. For a word $w \in \Sigma^*$, let $\#_i(w)$ be the number of occurrences of a_i in w.

Definition 1.1. [The Parikh map] The map $\Psi : \Sigma^* \to \mathbb{N}^n$ given by

$$\Psi : w \mapsto \begin{bmatrix} \#_1(w) \\ \vdots \\ \#_n(w) \end{bmatrix}$$

is the Parikh map.

For a given word $w \in \Sigma^*$, $\Psi(w)$ is sometimes called the *commutative image* or the *Parikh image* of w. The commutative image or Parikh image of a language L is simply $\Psi(L)$, the image of the map Ψ restricted to L. A Parikh vector is simply a vector in $\mathbb{N}^n$ which is the image of some word in Σ

.

We can regard $(\mathbb{N}^n, +)$ as a commutative monoid with identity 0. The submonoid generated by $v_1, ..., v_m \in \mathbb{N}^n$ is given by $\langle v_1, ..., v_m \rangle = \{c_1 v_1 + ... + c_m v_m : c_i \in \mathbb{N}, i \in [n]\}$ where $[n] = \{1, ..., n\}$.

Definition 1.2. [A (semi)linear set] A coset of the form

$$u + \langle v_1, ..., v_m \rangle = \{u + c_1 v_1 + ... + c_m v_m : c_i \in \mathbb{N}, i \in [n]\}$$

for $u \in \mathbb{N}$ is a linear set. If $S_1, ..., S_p$ are linear sets then

$$S = \bigcup_{i=1}^{p} S_i$$

is a semilinear set.

Claim 1.3. For every semilinear set S over a commutative alphabet $\{a_1, ..., a_n\}$, there is some regular set R such that $\Psi(R) = S$.

Proof. For $v \in \mathbb{N}^n$ we note $\Psi^{-1}(v) = \{a_1^{(v)_1} \cdots a_n^{(v)_n}\}$ is a singleton and thus regular (here $(v)_i$ is the ith entry of v). For a linear set $S_i = u_i + \langle v_1, ..., v_m \rangle$ we observe

$$\Psi\left(\Psi^{-1}(u_i)\Psi^{-1}(v_1)^* \cdots \Psi^{-1}(v_m)^*\right) = S_i$$

where $\Psi^{-1}(u_i)\Psi^{-1}(v_1)^* \cdots \Psi^{-1}(v_m)^*$ is regular since it is the concatenation of the regular sets $\Psi^{-1}(u_i), \left(\Psi^{-1}(v_1)\right)^*, ..., \left(\Psi^{-1}(v_m)\right)^*$. And thus for some semilinear $S = \cup_{i=1}^{p} S_i$ where $S_i = u_i + \langle v_{i_1}, ..., v_{i_m} \rangle$ we can take the union of sets of the form $\Psi^{-1}(u_i)\Psi^{-1}(v_{i_1})^* \cdots \Psi^{-1}(v_{i_m})^*$ and apply Ψ to that union to get S. □

2 Parikh's theorem

Theorem 2.1 (Parikh's theorem)**.** *Let L be a CFL, then $\Psi(L)$ is semilinear. Equivalently by Claim 1.3, $\Psi(L)$ is equal to the Parikh image of some regular set.*

Another way of stating the theorem is that ignoring the order of letters or considering Σ as a commutative alphabet, L is isomorphic to a regular set. The original proof relies heavily on the structure of parse trees and has a similar flavor to the proof the pumping lemma for CFLs. We give a sketch.

2.1 Parikh's original proof

Proof sketch of Theorem 2.1. Let $G = (N, \Sigma, R, S)$ be a context free grammar (CFG) in Chomsky normal form. Let $L = L(G)$ be the associated CFL. For a parse tree t we define $w(t)$ to be the string in $N \cup \Sigma$ taken from the leaves in t (starting from the leftmost leaf and ending at the rightmost leaf). Intuitively we want to associate a coset of the form $u + \langle v_1, ..., v_m \rangle$ with the Parikh image of some subset of L. To do this, we think about a class of parse trees which we associate with u and another class which we associate with $\langle v_1, ..., v_m \rangle$.
Formally, for $A \in N$ we define R_A to be the set $w(t)$ such that

(1) A is the vertex of t and no symbol in $N \cup \Sigma$ appears more than $|N \cup \Sigma|$ times in t
(2) A is in $w(t)$ and is the only nonterminal in $w(t)$

Let T_A be defined as the set $w(t)$ such that (1) holds AND $w(t)$ contains only terminal symbols AND every symbol $N \cup \Sigma$ appears in t. T_A and R_A are both finite sets by condition (1) and can thus be derived from G. The argument proceeds by constructing a semilinear set from T_A and R_A in a direct way: for $w \in R_A$ simply remove the nonterminal symbol and apply Ψ, for $w \in T_A$ just directly apply Ψ since there aren't any nonterminals. This particular case only accounts for strings for which A appears in the derivation of that string, so one has to argue that L can be written as a finite union of subsets of L, each of which can be described in this manner. For a complete proof see [Par66] or for an even more complete treatment see [Koz97, Theorem H.1 pg 202].

□

2.2 Simplified proof using strong pumping lemma

There's a nice simplification of Theorem 2.1 due to Goldstine [Gol77] which leverages the pumping lemma for CFLs.

Theorem 2.2 (Pumping lemma for CFLs)**.** *Let L be a CFL. There is an integer $p > 0$ such that all $z \in L$ with $|z| \geq p$ can be written as $z = uvwxy$ with $|vwx| \leq p$, $vx \neq \varepsilon$ (either v or x is not the empty string) and $uv^k wx^k y \in L$ for all $k \geq 0$. The integer p is the pumping length.*

The basic intuition for the lemma is that if L is infinite, then there must be some nonterminal which can be repeatedly applied in a derivation to get arbitrarily many new strings. A proof of the pumping lemma can be found in [HU90, Theorem 7.18]. Goldstine uses a slightly strong version of Theorem 2.2:

Theorem 2.3 (Strong pumping lemma)**.** *Let L be a CFL with corresponding CFG $G = (N, \Sigma, R, S)$. There is an integer $p > 0$ such that for any $k \geq 1$, if $z \in L$ and $|z| \geq p^k$ then $z = uv_1...v_k w x_k...x_1 y$ and there is a derivation*

$$S \overset{*}{\Rightarrow} uAy \overset{*}{\Rightarrow} uv_1 A x_1 y \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} uv_1...v_k w x_k...x_1 y$$

for some nonterminal $A \in N$ and each $v_i x_i \neq \varepsilon$ and $|v_1...v_k w x_k...x_1| \leq p^k$.

Goldstine's proof of Theorem 2.1. Let $G = (N, \Sigma, R, S)$ be a CFG for L and let p be the pumping length. The main approach is to construct a regular set R such that $\Psi(L) = \Psi(R)$. Fix $U \subseteq N$ where $S \in U$. Let $L_U \subseteq L$ be those words in L whose derivation uses all the nonterminals in U (and no nonterminals outside U). We'll show $\Psi(L_U)$ is semilinear which is sufficient because L can be written as a union of sets of the form L_U. Let $k = |U|$ and define

$$F = \{z \in L_U : |z| < p^k\}$$
$$G = \{vx : 1 \leq |vx| \leq p^k \text{ and } \exists A \in U \text{ s.t. } A \overset{*}{\Rightarrow} vAx\}.$$

Then $\Psi(L_U) = \Psi(FG^*)$. This proves that $\Psi(L_U) \subseteq \Psi(FG^*)$ and $\Psi(FG^*) \subseteq \Psi(L_U)$ are both inductive. Clearly if $z \in L_U$ and $|z| < p^k$ then $z \in F \subseteq FG^*$. Likewise if $z \in F$ then $z \in L_U$ by definition.

The key idea for the inductive step in showing $\Psi(L_U) \subseteq \Psi(FG^*)$ for some $z \in L_U$ with $|z| \geq p^k$ is to look at the derivation given by the strong pumping lemma. Write

$z = uv_1...v_kwx_k...x_1y$. Then use the fact that all k nonterminals in U must appear in the derivation of z by the construction of L_U. A simple pigeonhole argument shows that at least one nonterminal is repeated and can thus be deleted in this derivation. The deletion removes a production of the form $A \stackrel{*}{\Rightarrow} v_iAx_i$. So we get some new $|z'| < |z|$ which inductively satisfies $\Psi(z') \in \Psi(FG^*)$. And since $v_ix_i \in G$ and $\Psi(z) = \Psi(z'v_ix_i)$ we have $\Psi(z) \in \Psi(FG^*)$.

The other direction $\Psi(FG^*) \subseteq \Psi(L_U)$ is a bit easier. The main idea is to take $z = z'vx \in FG^*$ where $z' \in FG^*$ and $vx \in G$ and inductively $\Psi(z') = \Psi(\hat{z})$ for some $\hat{z} \in L_U$. There is some nonterminal $A \in U$ such that $A \stackrel{*}{\Rightarrow} vAx$. Look at the derivation of $\hat{z}$ which includes A somewhere by the construction of L_U. Apply the production rule $A \stackrel{*}{\Rightarrow} vAx$ to that occurrence of A in the derivation of $\hat{z}$, and this gives a new word $z'' \in L_U$ which satisfies

$$\Psi(z'') = \Psi(\hat{z}vx) = \Psi(z'vx) = \Psi(z).$$

And so $\Psi(z) \in \Psi(L_U)$.

It remains to show that $\Psi(FG^*)$ is semilinear. F and G are both finite sets, so we can write $F = \{x_1, ..., x_r\}$ and $G = \{y_1, ..., y_m\}$. Then define $X_i = \Psi(x_i) + \langle \Psi(y_1), ..., \Psi(y_m) \rangle$ and we have $\Psi(FG^*) = \cup_{i=1}^{r} X_i$. □

There are some immediate corollaries of Parikh's theorem that are worth mentioning.

Corollary 2.4 (of Theorem 2.1)**.** *If $\Sigma = \{a\}$, a single letter alphabet, and $G = (N, \Sigma, R, S)$ is a CFG. Then the language L associated with G is regular.*

Proof. Let T be the regular set satisfying $\Psi(T) = \Psi(L)$. Given $a^i \in T$ we have $[i] = \Psi(a^i) \in \Psi(L)$ and thus $a^i \in L$. Likewise for $a^i \in L$ we have $[i] \in \Psi(T)$ and so $a^i \in T$. Thus $T = L$. Essentially, there is a one-to-one correspondence between the word and the Parikh vector so by virtue of having the same Parikh image, T and L must be the same. □

2.3 Alternative proof by NFA construction

An alternative, simplified proof to Parikh's theorem is given in [EGKL11]. The authors there give an explicit nondeterministic automaton (NFA) which accepts a regular set whose Parikh image is the same as $\Psi(L)$ for the CFL L of interest. The automaton has $O(2^n)$ states which is shown in [LP12] to be optimal.

Intuitively the construction involves thinking about the state space as the space of possible Parikh vectors and the alphabet as chucks of possible letters. We want the transitions in the NFA to mimic the derivations, or at least the Parikh image of the derivations, that lead to a word in the language. The trickiest part in the construction is defining the transition function of the automaton. It involves the notion of a *step* in the production rules of the grammar.

Definition 2.5 (Step)**.** Let $G = (N, \Sigma, P, S)$ be a CFG and let $A \to \gamma$ be an arbitrary production rule in G. Then (α, β) is a step, denoted $\alpha \Rightarrow \beta$, if there exist $\alpha_1, \alpha_2 \in (N \cup \Sigma)^*$ such that $(\alpha, \beta) = (\alpha_1 A \alpha_2, \alpha_1 \gamma \alpha_2)$.

One can think of a step as a derivation which does not interfere with the outermost letters of the string (hence the overlap in the notation $\Rightarrow$). For a given production rule, there are infinitely many steps containing that rule - one can just keep adding a single letter to α_1. We are interested however in the production rule associated with a given step. Given $\alpha \Rightarrow \beta$, the tuple $(\alpha_1, \alpha_2, A, \gamma)$ satisfying $(\alpha, \beta) = (\alpha_1 A \alpha_2, \alpha_1 \gamma \alpha_2)$ and $A \to \gamma \in P$ is unique since A is a single nonterminal (at least as long as we assume wlog that the CFG G doesn't have any redundant rules such as $A \to A$, $B \to B$ in which case e.g. the rule $AB \to AB$ would have multiple rules associated with it).

Example 2.6. Suppose the CFG G is given by $N = \{S, E\}$ and $\Sigma = \{a, b\}$ with production rules

$$S \to aSa|bSb|aEb|bEa$$
$$E \to aE|bE|\varepsilon.$$

Then the step $aaESbb \Rightarrow aaaESbb$ is uniquely associated with the rule $E \to aE$ where $\alpha_1 = aa$ and $\alpha_2 = Sbb$.

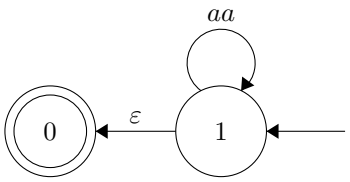
We wish to associate an automata transition with each rule $\alpha \Rightarrow \beta$. We define a projection homomorphism $p_\Sigma : (N \cup \Sigma)^* \to \Sigma^*$ which simply removes the nonterminals from a given string. So for example, using the above grammar $p_\Sigma(aSa) = p_\Sigma(EaaS) = aa$. We define p_N as the analogous projection onto N (e.g. $p_N(aSa) = S$). The last ingredient is the Parikh image of a string of nonterminals. We can modify the definition of Ψ to account for this change by simply considering $\Psi : N^* \to \mathbb{N}^m$ and then proceed in the same manner as in Definition 1.1. Again using the above example, we have $\Psi(SSSES) = \begin{pmatrix} 4 & 1 \end{pmatrix}$. It should be clear from context whether the domain of interest is the set of strings of nonterminals or of terminals. Hence we abuse notation and use Ψ when referring to both the Parikh image of nonterminal strings and of terminal strings. Notice then we have both $\Psi \circ p_N : (N \cup \Sigma)^* \to \mathbb{N}^m$ and $\Psi \circ p_\Sigma : (N \cup \Sigma)^* \to \mathbb{N}^n$. These maps are the tools we need for relating steps and Parikh vectors. Since we are construction an NFA $(Q, \Sigma, \delta, q_0, F)$ we can think of the transition function δ and $\delta \subseteq Q \times \Sigma \times 2^Q$. If we want δ to model a step in a CFG G where the states are Parikh vectors, then one intuitive definition is $\delta = \{(\Psi \circ p_N(\alpha), p_\Sigma(\gamma), \{\Psi \circ p_N(\beta)\}) : \alpha \to \beta$ is a step with associated rule $A \to \gamma\}$. This transition turns out to give the desired NFA as long as we define the state space carefully and ensure that we restrict the steps we consider (since there are infinitely many steps but $|\delta|$ is finite). As such, we restrict the state space to be an enumeration of all possible vectors in $\mathbb{N}^n$ that are smaller (in terms of $|\cdot|$, the sum of the entries of the vector) than the largest possible Parikh vector of interest. We thus start with the following definition.

Definition 2.7. Let $G = (N, \Sigma, R, S)$ be a CFG and let $k \geq 1$ be arbitrary. The k-Parikh automaton of G is the NFA $M_k = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = \{v \in \mathbb{N}^m : |v| \leq k\}$
- $\delta = \{(\Psi \circ p_N(\alpha), p_\Sigma(\gamma), \{\Psi \circ p_N(\beta)\}) :$
  $\alpha \to \beta$ is a step with associated rule $A \to \gamma$ and
  $\Psi \circ p_N(\alpha), \Psi \circ p_N(\beta) \in Q\}$
- $q_0 = \Psi \circ p_N(S)$
- $F = \{\Psi \circ p_N(\varepsilon)\} = \{\begin{pmatrix} 0 & \cdots & 0 \end{pmatrix}\}$

Note that we impose the condition $k \geq 1$ to ensure that $q_0 \in Q$ (since $|q_0| = 1$). An alternative way to ensure this inclusion is to define $Q = \{v \in \mathbb{N}^n : |v| \leq k\} \cup \{\Psi \circ p_N(S)\}$ and allow $k \geq 0$. Either way will work for our purposes. Note also that $p_\Sigma(\gamma)$ in the labeled transition of δ may be in Σ^* rather than Σ. We can without loss of generality interpret these as regular expressions over Σ which can thus be converted into NFAs with transitions over Σ. So for example, we may have $\Sigma = \{a, b\}$ and states s_1, s_2 such that $\delta(s_1, ba) = s_2$ which we can just interpret as $\delta(s_1, b) = s_3$ and $\delta(s_3, a) = s_2$ where s_3 is a new state. Graphically this transition is just $s_1 \xrightarrow{b} s_3 \xrightarrow{a} s_2$. Note also that the restriction that $\Psi \circ p_N(\alpha), \Psi \circ p_N(\beta) \in Q$ ensures that δ is finite.

Example 2.8. Consider the CFG G given by $N = \{S\}, \Sigma = \{a\}$ and the production rules $S \to aSa|\varepsilon$. Hence, $L(G)$ is the language of all even length strings in Σ^*. Suppose we want to form the 1-Parikh automaton of G. In construction M_k, we have found it easiest to inspect each production rule of G individually and consider all possible steps that could contain that production rule. Of course we only want to consider steps small enough to be in the state space of M_k. In this case $k = 1$ so the inspection is simple. We denote the states $[0], [1]$ by simply 0 and 1. For $S \to aSa$ there is only one step which is the rule itself (note here that $\alpha_1, \alpha_2 = \varepsilon$). Thus we get the transition $\delta(1, aa) = 1$. Similarly the only step for $S \to \varepsilon$ is the rule itself. This gives the transition $\delta(1, \varepsilon) = 0$. We thus get the automaton:

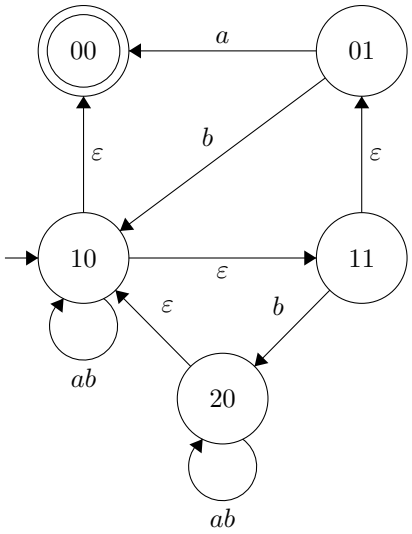


which clearly accepts the language $(aa)^*$. In this case, the context free language we started out with is regular so the example is a bit uninteresting.

Consider a more sophisticated grammar, this time over nonterminals S, E and letters a, b. The rules are given by

$$S \to aSb|\varepsilon|ES$$
$$E \to aE|a|Sb.$$

In this case, $k = 2$ will give us a Parikh automaton, whose language is Parikh equivalent to that of the context free language. We give the automaton below. We write xy for $x, y \in \{0, 1\}$ to denote the vector $\begin{pmatrix} x & y \end{pmatrix}$.



Using this notion of the Parikh automaton, the authors prove the following result which implies Parikh's theorem.

Theorem 2.9. *Let G be a CFG and $L(G)$ be the corresponding CFL. Then there exists a constant c, depending on G, such that the language of the c-Parikh automaton M_c and $L(G)$ are Parikh equivalent.*

Theorem 2.9 implies Parikh's theorem since the language of M_c is regular and so $L(G)$ is Parikh equivalent to a regular language. The authors give an explicit c in terms of m (the number of nonterminals) and the degree of G, defined as $d = \max\{p_N(\gamma) : A \to \gamma \in R\} - 1$. Specifically, they show that $c = md + 1$ is large enough. The proof is in two parts, first showing that $\Psi(L(M_c)) \subseteq \Psi(L(G))$ then showing the reverse inclusion. The proof of $\Psi(L(M_c)) \subseteq \Psi(L(G))$ starts by showing that for any state q of M_c (where c is arbitrary), if $q \in \hat{\delta}(S, \sigma)$ for some $\sigma \in \Sigma^*$, then there is a derivation $S \stackrel{*}{\Rightarrow} \alpha$ such that $\Psi \circ p_N(\alpha) = q$ and $\Psi \circ p_\Sigma(\alpha) = \Psi \circ p_\Sigma(\sigma)$. Now suppose $\omega \in L(M_c)$. Then for some $S \stackrel{*}{\Rightarrow} \alpha$ we have $\Psi \circ p_N(\alpha) = q_f = (0 \cdots 0)$. Thus α contains only terminals and is thus in $L(G)$. And so σ and α are Parikh equivalent and so $\Psi(\omega) \in \Psi(L(G))$ (we note by definition ω also only contains terminals so we don't have to worry about applying p_Σ to it).

For the proof of the second direction, the key idea is that $\Psi(L(G)) \subseteq \Psi(L_{md+1}(G))$ where $L_{md+1}(G)$ is the set of words derivable in G using no more than $md+1$ nonterminals at any one step in the derivation. The proof of this fact uses parse trees and is rather involved. It is similar to the aforementioned proof of Parikh's theorem or the proof of Parikh's theorem in [Koz97] which looks at the size of parse trees relative to the Parikh image of their yield (i.e. the word formed from the tree). The result of Theorem 2.9 then follows since straightforward induction shows that $\Psi(L_k(G)) \subseteq \Psi(L(M_k)$.

2.4 Other alternative proofs and generalizations

A fully equational proof of the theorem is given in [AEI02] which involves computing fixed points. An early generalization of the theorem is due to Greibach who shows that a particular substitution operator on sets preserves semilinearity of the sets [Gre72]. A subsequent generalization is due to Pilling [Pil73].

3 Pilling's result

Pilling observes in [Pil73] that the language of a context free grammar can be viewed as the least solution to a set of equations. In particular let $N = \{A_1, ..., A_k\}$ and $G = (N, \Sigma, R, A_1)$ be a CFG. We can think of a production rule as $A_i \to h(A_1, ..., A_k)$ where $h(A_1, ..., A_k)$ is some string in $(N \cup \Sigma)^*$ (this notation makes substitution easier). Let A be the vector $\begin{pmatrix} A_1 & \cdots & A_k \end{pmatrix}$. Thus elements of R are pairs of the form $(A_i, h(A))$. We then let $X_1, ..., X_k$ be variables and define

$$f_i(X_1, ..., X_k) = \sum_{(A_i, h(A)) \in R} h(X_1, ..., X_k).$$

Then we concern ourselves with solutions to the system of equations given by

$$\begin{aligned} X_1 &= f_1(X_1, ..., X_k) \\ X_2 &= f_2(X_1, ..., X_k) \\ &\vdots \\ X_k &= f_k(X_1, ..., X_k). \end{aligned}$$

The language generated by G is the smallest set $L \subseteq \Sigma^*$ such that there exists sets X_i, $i \neq 1$ and $L = f_1(L, X_2, ..., X_k)$ and $X_i = f_i(L, ..., X_k)$ (here "smallest" means that if L' is an alternative solution then $L \subseteq L'$). As Pilling shows, once we find L the other variables can be solved sequentially. The ultimate goal is to show that there is a minimal solution to this system and that that solution is regular. In particular, Pilling shows

Theorem 3.1 (Pilling's generalization)**.** *A system of regular equations over a commutative alphabet has a minimal regular solution.*

Before getting further into the theorem an example may be helpful.

Example 3.2. Let $\Sigma = \{a, b\}$ and let $S \to AB$ and $A \to aaA \mid \varepsilon$ and $B \to bB \mid \varepsilon$. Then the language generated by this CFG is given as the least solution to

$$\begin{aligned} X_1 &= X_2 X_3 \\ X_2 &= aaX_2 + \varepsilon \\ X_3 &= bX_3 + \varepsilon \end{aligned}$$

which in this case happens to be given by $X_1 = (aa)^* b^*, X_2 = (aa)^*, X_3 = b^*$.

The proof of Pilling's theorem leverages the following lemma which allows us to solve the system of equations one variable at a time.

Lemma 3.3. *Consider the system of k equations above. Suppose we let f_i be arbitrary functions of regular sets. Let $1 \leq r \leq k$. Turn the equational system into a system of inequalities by writing*

$$\begin{aligned} f_1(X_1, ..., X_k) &\subseteq X_1 \\ &\vdots \\ f_r(X_1, ..., X_k) &\subseteq X_r. \end{aligned}$$

Then we can construct functions $g_i(X_{r+1}, ..., X_k)$ for $i = 1, ..., r$ such that $f_i(X_1, ..., X_k) \subseteq X_i$ implies $g_i(X_{r+1}, ..., X_k) \subseteq X_i$ and $g_i(X_{r+1}, ..., X_k)$ is a solution of f_i meaning that if $g_i(X_{r+1}, ..., X_k) = X_i$ then $f_i(X_1, ..., X_k) = X_i$.

This lemma allows us to solve $f_1(X_1, ..., X_k) = X_1$ by first considering $X_2, ..., X_k$ as fixed. Once we get this solution, by the lemma we'll be able to substitute it in as the *least* solution and we can then move on to the second variable. The proof of this lemma involves constructing a sequence $(X_{1N}, X_{2N}, ..., X_{rN})$ of approximating variables which we substitute into $f_i(X_1, ..., X_k)$ and take the sum over all N to form g_i. This approximation assumes $*-$continuity (Pilling's result is generalized in [HK99] by removing this assumption). The proof of the main theorem then becomes solving a particular case which we use in induction.

Proof of theorem 3.1. We want to compute the minimal solution to

$$X_1 = f_1(X_1, ..., X_k).$$

We can write it as

$$X_1 = E + F(X_1)X_1$$

where E is independent of X_1. We thus have $F(X_1)^*E \subseteq X_1$. Now since $E \subseteq X_1$ we have $F(E) \subseteq F(X_1)$ and so combining this equation with the one above we have $F(E)^*E \subseteq X_1$ (by the axioms of the * operator). Thus $F(E)^*E$ at least as small as any solution to $X_1 = f_1(X_1, ..., X_k)$. Hence if we show it is a solution, it must be a minimal one. We compute

$$\begin{aligned} E + F(F(E)^*E)F(E)^*E &= E + F(E)^*F(E)F(E)^*E \\ &= E + F(E)^*F(E)E \\ &= F(E)^*E \end{aligned}$$

using the fact that $F(E)^*F(E)^* = F(E)^*$. Thus $F(E)^*E$ is a minimal regular solution. We can solve for the other variables using this computation as the inductive step. The lemma ensures that we can solve the whole system by solving one variable at a time. □

Parikh's theorem is a special case of Pilling's theorem since a context free language is given as a minimal solution to a system of equations of the form in Theorem 3.1. And thus by the theorem if we ignore the order of letters in words, we get a regular set.

4 Hopkins & Kozen's Generalization

Hopkins and Kozen [HK99] generalize Pilling's result further. Their work applies to all commutative Kleene algebras regardless of whether $*$-continuity holds. Specifically let K be a commutative Kleene algebra and let $x_1, x_2, ..., x_n$ be variables over K. Then Hopkins and Kozen's generalization states that for polynomials $f_1, ..., f_n$ over $x_1, x_2, ..., x_n$ the system $f_i \leq x_i$ has a unique minimal solution.

We let $K[x_1, ..., x_n]$ denote polynomials in K which are essentially regular expressions over K and $x_1, ..., x_n$.

Theorem 4.1. *Let $f_1, ..., f_n$ be arbitrary polynomials over $x_1, x_2, ..., x_n$ with coefficients in K. Then the system*

$$\begin{aligned} f_1(x_1, ..., x_n) &\leq x_1 \\ f_2(x_1, ..., x_n) &\leq x_2 \\ &\vdots \\ f_n(x_1, ..., x_n) &\leq x_n \end{aligned}$$

has a unique minimal solution.

4.1 Background on polynomials in commutative Kleene algebras

Let $\mathbf{x} = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}$, then this system states that $f_i(\mathbf{x}) \leq x_i$. It turns out that the minimal solution to this system is most concisely given with the notion of a derivative on polynomials in $K[x]$. Specifically, the derivative is a special type of "differential operator."

Definition 4.2. A map $D : K \to K$ is a differential operator if for all $x, y \in K$

$$\begin{aligned} D(x + y) &= D(x) + D(y) \\ D(xy) &= yD(x) + xD(y) \\ D(x^*) &= x^* D(x) \\ D(0) &= D(1) = 0. \end{aligned}$$

A differential operator D from K to K has the useful property that it can be uniquely extended to a differential operator $D : K[\mathbf{x}] \to K[\mathbf{x}]$, given a map from $\mathbf{x} \to K$ (i.e. a way to evaluate $x_1, .., x_n$ in K). Let x_i be some variable in $\mathbf{x}$. We can thus define the derivative $\frac{\partial}{\partial x_i} : K[\mathbf{x}] \to K[\mathbf{x}]$ as the differential operator extending a differential on K.

Definition 4.3 (Derivative)**.** Let $\frac{\partial}{\partial x_i}$ be a differential operator defined initially on K and $\mathbf{x}$ as

$$\begin{aligned} &\frac{\partial x_i}{\partial x_i} = 1 \quad \frac{\partial x_i}{\partial x_j} = 0, \quad j \neq i \\ &\frac{\partial a}{\partial x_i} = 0. \end{aligned}$$

then extended uniquely to a map $\frac{\partial}{\partial x_i} : K[\mathbf{x}] \to K[\mathbf{x}]$.

For a polynomial $f \in K[\mathbf{x}]$ we write f' for $\frac{\partial f}{\partial x_i}$ and $f'(e)$ for evaluating the polynomial $f' = \frac{\partial f}{\partial x_i}$ at $x_i \mapsto e$. We can also extend this notion naturally to a notion of a Jacobian matrix. Specifically let $\mathbf{f} = \begin{pmatrix} f_1 & \cdots & f_m \end{pmatrix}$ be a vector where $f_i \in K[\mathbf{x}]$. Then $\frac{\mathbf{f}}{\mathbf{x}}$ can be thought of as the $m \times n$ matrix whose i, jth entry is $\frac{\partial f_i}{\partial x_j}$. Now we're in good shape to prove the main theorem, but first we need a lemma.

Lemma 4.4 (Taylor's theorem for commutative KA)**.** *Let $f, g \in K[x]$ then $f(x + g) = f(x) + f'(x + g)g$.*

4.2 Proof of main theorem

The resemblance to the traditional Taylor's theorem comes when we substitute 0 for x to get $f(g) = f(0) + f'(g)g$. It is this form that will be most useful. The proof of Lemma 4.4 is a direct computation by inducting on the structure of f [HK99, Theorem 3.3].

Proof of Theorem 4.1. The proof is by induction on n. Let $n = 1$, then the minimal solution to $f(x) \leq x$ is given by

$$f'(f(0))^* f(0).$$

Let $b = f(0)$ and $c = f'(b)$ then the minimal solution is c^*b. We remark briefly that this solution is of the same form as that in theorem 3.1 if we let $f' = F$ and $f(0) = E$.

The proof proceeds by first showing that c^*b is in fact a solution. Note that since $f(x)$ can be viewed as simply a substitution into a regular expression we have a monotonicity property: if $a \leq b$ then $f(a) \leq f(b)$ and in particular if $ac \leq bc$ then $f(a)c \leq f(b)c$ We wish to show that $f(c^*b) \leq c^*b$ or equivalently (by commutativity) $f(bc^*) \leq bc^*$. Since bc^* is a polynomial by Lemma 4.4 we can write $f(bc^*) = f(0) + f'(bc^*)bc^*$. Thus, write

$$\begin{aligned} f(bc^*) &= b + f'(bc^*)bc^* \\ &\leq b + f'(b)bc^* \\ &= f(0) + cf(0)c^* \\ &= f(0)(1 + cc^*) \\ &= f(0)c^* \\ &= bc^* \end{aligned}$$

where $b + f'(bc^*)bc^* \leq b + f'(b)bc^*$ since $bc^*bc^* \leq bbc^*$, i.e.

$$bc^*bc^* = bbc^*c^* = bbc^*$$

so certainly $bc^*bc^* \leq bbc^*$ and by the above property of polynomials this inequality gives $f(bc^*)bc^* \leq f(b)bc^*$.

The next step is to show minimality. That is, let y be an arbitrary solution, then we need to show that $c^*b \leq y$. Now since $c^*b \leq y \Leftrightarrow b + cy \leq y$, it is sufficient to observe that since $0 \leq y$ we have $b = f(0) \leq f(y)$ and $f(y) \leq y$ by assumption (y is a solution) thus $b \leq y$ and in particular $c = f'(b) \leq f'(y)$. So we can write

$$\begin{aligned} b + cy &\leq b + f'(y)y \\ &= f(y) \qquad \text{Lemma 4.4} \\ &\leq y. \end{aligned}$$

The proof proceeds inductively using this step as the base case. It is important to observe that the solution is derivable directly from the axioms of Kleene algebra so the least solution holds (and its minimality is preserved) under homomorphic images. This fact is used in the inductive step below to show that minimality is preserved when iteratively solving the system.

In their paper, Hopkins and Kozen show the ingredients of the inductive step in a two dimensional system (for $n = 2$). We give a fully general inductive step here for the sake of completeness. The core ideas are the same.

Suppose there is a system of inequalities in $K[x_1, ..., x_n]$

$$\begin{aligned} f_1(x_1, ..., x_n) &\leq x_1 \\ &\vdots \\ f_n(x_1, ..., x_n) &\leq x_n. \end{aligned}$$

Consider $f_n(x_1, ..., x_n) \leq x_n$ as a one dimensional system in the single variable x_n. We use the inductive assumption to find a least solution $h_n(x_1, ..., x_{n-1})$ to this particular inequality. Then we compute the least solution to $f_{n-1}(x_1, ..., x_{n-1}, h_n(x_1, ..., x_{n-1})) \leq x_{n-1}$ where f_{n-1} is treated as a one dimensional system in x_{n-1}. We label the new solution as $h_{n-1}(x_1, ..., x_{n-2})$. Proceed in this fashion until we get a solution h_1 (which is independent of all other variables $x_1, ..., x_n$) satisfying $f_1(h_1, h_2(h_1, ...), ..., h_n(h_1, ...)) \leq h_1$. Ignoring substitutions, this inequality can simply be written as $f_1(h_1, ..., h_n) \leq h_1$. We thus have

$$\begin{aligned} f_1(h_1, ..., h_n) &\leq h_1 \\ &\vdots \\ f_n(h_1, ..., h_n) &\leq h_n(\cdots). \end{aligned}$$

For example, if $n = 3$, we would get

$$\begin{aligned} f_1(h_1, h_2(h_1), h_3(h_1, h_2(h_1))) &\leq h_1 \\ f_2(h_1, h_2(h_1), h_3(h_1, h_2(h_1))) &\leq h_2(h_1) \\ f_3(h_1, h_2(h_1), h_3(h_1, h_2(h_1))) &\leq h_3(h_2(h_1)) \end{aligned}$$

or using simplified notation

$$\begin{aligned} f_1(h_1, h_2, h_3) &\leq h_1 \\ f_2(h_1, h_2, h_3) &\leq h_2 \\ f_3(h_1, h_2, h_3) &\leq h_3. \end{aligned}$$

It remains then to establish minimality of the solutions. The key idea is to combine the fact that inductively the solutions are minimal with the fact that they remain minimal under homomorphic images. Indeed if $(a_1, ..., a_n)$ a different solution then $h_2(a_1) \leq a_2$ and $h_1 \leq a_1$ so $h_2(h_1) \leq h_2(a_1) \leq a_2$. One can proceed in this fashion to show $(h_1, ..., h_n) \leq (a_1, ..., a_n)$ (for a more detailed treatment in the $n = 2$ case see [HK99, Proof of theorem 1.1, pg 9], specifically the last last two paragraphs). □

Corollary 4.5 (Parikh's theorem 2.1)**.** *Let L be a CFL over a commutative alphabet, then L is isomorphic to a regular language.*

Proof. Simply represent L as the solution to a system of "regular equations" or equivalently polynomials over a commutative K. The above theorem states that there exists a minimal solution to this system in the commutative K and is hence some regular set. □

The above proof guarantees that a minimal solution exists, but it only gives us an explicit solution in the one dimensional case. However, it turns out with the right definitions, the general solution mimics the form of the one dimensional case.

Theorem 4.6. *Let K be a commutative algebra and $\mathbf{x} = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}$, and $\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) & \cdots & f_n(\mathbf{x}) \end{pmatrix}$. Define*

$$\begin{aligned} \mathbf{a}_0 &= \mathbf{f}(\mathbf{x}) \\ \mathbf{a}_{k+1} &= \frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{a}_k)^* \mathbf{a}_k. \end{aligned}$$

Let $N = (7(3^n) - 5)/2$. Then $\mathbf{a}_N$ is the least solution to

$$\mathbf{f}(\mathbf{x}) \leq \mathbf{x}$$

and is minimal under homomorphic images as in Theorem 4.1.

Proof idea. The main idea is to induct on n with Theorem 4.1 serving as the base case. The inductive step considers the Jacobian matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$, splitting it into blocks based on an arbitrary partitioning of n. By considering the systems formed by the blocks themselves, one can leverage the inductive hypothesis to obtain the desired result. See [HK99, Theorem 5.1] for the computation. The value for N arises naturally in the proof as the solution to the recurrence

$$\begin{aligned} N(1) &= 1 \\ N(n+1) &= 3N(n) + 5. \end{aligned}$$

□

References

[AEI02] Luca Aceto, Zoltán Esik, and Anna Ingólfsdóttir. A fully equational proof of Parikh's theorem. *RAIRO-Theoretical Informatics and Applications*, 36(2):129–153, 2002.

[EGKL11] Javier Esparza, Pierre Ganty, Stefan Kiefer, and Michael Luttenberger. Parikh's theorem: A simple and direct automaton construction. *Information Processing Letters*, 111(12):614–619, 2011.

[Gol77] Jonathan Goldstine. A simplified proof of Parikh's theorem. *Discrete Mathematics*, 19(3):235–239, 1977.

[Gre72] Sheila A Greibach. A generalization of Parikh's semilinear theorem. *Discrete Mathematics*, 2(4):347–355, 1972.

[Har78] M. A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1978.

[HK99] Mark W. Hopkins and Dexter C. Kozen. Parikh's theorem in commutative Kleene algebra. In *Logic in Computer Science, 1999. Proceedings. 14th Symposium on*, pages 394–401. IEEE, 1999.

[HU90] John E. Hopcroft and Jeffrey D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1990.

[Koz97] Dexter C. Kozen. Automata and computability, undergraduate texts in computer science, 1997.

[LP97] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1997.

[LP12] Giovanna J. Lavado and Giovanni Pighizzini. Parikh's theorem and descriptional complexity. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 361–372. Springer, 2012.

[Par66] Rohit J. Parikh. On context-free languages. *Journal of the ACM (JACM)*, 13(4):570–581, 1966.

[Pil73] DL Pilling. Commutative regular equations and Parikh's theorem. *Journal of the London Mathematical Society*, 2(4):663–666, 1973.