AUTOMATA AND NON-COMMUTATIVE POLYNOMIALS

CONTENTS

1 FINITE FIELDS

2 NON-COMMUTATIVE POLYNOMIALS

A non-commutative polynomial is an expression over a finite set of variables and involving addition (which is commutative) and multiplication (which is associative but is not commutative). Moreover, multiplication distributes over addition. In this writeup we will fix the variables to be $\Sigma = \{x, y\}$. Some examples are given below

1. $p_1 ::= xyxx + xyy + xxy$
2. $p_2 ::= 2xyxx + 3x + xxyy$

Unlike a commutative polynomial, $xyx \neq xxy$ for non-commutative polynomial. Unless otherwise stated all polynomials in this writeup is non-commutative.

A polynomial with no addition operation is called a monomial. For eg, $xyxx$, xx are monomials. We will denote monomials by u, v, w. Since multiplication distributes over addition, any polynomial can be written as a sum of

monomials. In other words any polynomial P is equivalent to

$$\mathrm{P} ::= \sum_{w \in \mathrm{S}} w$$

where S is a *multiset* of monomials.

Given two matrices M_x and M_y of same dimension and a monomial w, we evaluate w over M_x and M_y as follows:

$$[\mathrm{M}_x, \mathrm{M}_y](z) ::= \mathrm{M}_z, \quad \text{if } z \in \{x, y\}$$

$$[\mathrm{M}_x, \mathrm{M}_y](zx) ::= [\mathrm{M}_x, \mathrm{M}_y](z) \times \mathrm{M}_x$$

We extend this to evaluate a polynomial over M_x and M_y.

$$[\mathrm{M}_x, \mathrm{M}_y](\mathrm{P}) ::= \sum_{w \in \mathrm{S}} [\mathrm{M}_x, \mathrm{M}_y](w)$$

Let w be a monomial. The degree of w (denoted by $\deg(w)$) is the length of w. The degree of a polynomial $\mathrm{P} = \sum_{w \in \mathrm{S}} w$ (denoted by $\deg(\mathrm{P})$) is the maximum of degrees of its monomials. That is

$$\deg(\mathrm{P}) = \max\{\deg(w) \mid w \in \mathrm{S}\}$$

The following lemma is equivalent to Lemma 4.1 in [2].

2.1 LEMMA. *Let p_1 and p_2 be two polynomials where $\deg(p_1)$ and $\deg(p_2)$ are less than d and such that $p_1 \neq p_2$. Let $\mathbb{F}$ be a finite field of size at least $3d$. Then for "random matrices" $\mathrm{M}_x \in \mathbb{F}^{d \times d}$ and $\mathrm{M}_y \in \mathbb{F}^{d \times d}$,*

$$\mathtt{Prob}([\mathrm{M}_x, \mathrm{M}_y](p_1) \neq [\mathrm{M}_x, \mathrm{M}_y](p_2)) \text{ is high}$$

For two different polynomials of degree less than d, the lemma implies that there are matrices M_x and M_y that "distinguishes" them.

We can "view" any finite language as a polynomial. For a language $\mathrm{L} = \{xyxy, xxyy, xyx\}$ the corresponding polynomial is $xyxy + xxyy + xyx$. Thus the definitions evaluation of a language, degree of a language etc follows naturally from that of polynomials. Moreover, the above lemma can also be seen as talking about languages.

Consider finite languages L_1 and L_2 of degree d. Then Lemma 2.1 shows that there are lots of matrices $\mathrm{M}_x \in \mathbb{F}^{d \times d}$ and $\mathrm{M}_y \in \mathbb{F}^{d \times d}$ such that evaluating L_1 and L_2 over the matrices result in different matrices.

Lemma 2.1 can be strengthened if the number of words in a language of degree d is bounded by some $poly(d)$. Let us state the stronger lemma for this special case [1].

2.2 LEMMA. *Let L_1 and L_2 be two languages of degree d and $|\mathrm{L}_1|, |\mathrm{L}_2| \leq \mathrm{poly}(d)$. Let us assume $\mathrm{L}_1 \neq \mathrm{L}_2$. Consider a finite field $\mathbb{F}$ of at least size $3d$. Then for*

"random matrices" $M_x \in \mathbb{F}^{\log d \times \log d}$ *and* $M_y \in \mathbb{F}^{\log d \times \log d}$,

$$\texttt{Prob}([M_x, M_y](L_1) \neq [M_x, M_y](L_2)) \textit{ is high}$$

The above lemma is not really important to us.

3 WEIGHTED AUTOMATA AND MATRIX EVALUATIONS

A weighted automata consists of a matrix M_x for every letter x, an initial vector σ and final vector f. For our purpose we can view the initial and final vector to be vectors with exactly a single 1 and the others are all zeros. A weighted automata $\mathcal{W}$ "runs" over a word w and returns an element of a field. We will denote this by $\mathcal{W}(w)$. We extend this to runs over a finite language L as follows.

$$\mathcal{W}(L) = \sum_{w \in L} \mathcal{W}(w)$$

The following lemma follows from Lemma 2.1.

3.1 LEMMA. *Let* L_1 *and* L_2 *be two finite languages of degree at most* d*. Let* $L_1 \neq L_2$*. Consider a field* $\mathbb{F}$ *of size at least* $3d$*. Then there exists a weighted automata* $\mathcal{W}$ *over* $\mathbb{F}$ *whose the number of states is* d *and such that* $\mathcal{W}(L_1) \neq \mathcal{W}(L_2)$*.*

Proof. From Lemma 2.1 we know that there is are matrices M_x and M_y such that $[M_x, M_y](L_1) \neq [M_x, M_y](L_2)$. Let these two matrices differ at the location (i, j). We know construct the weighted automata where each letters x and y are assigned M_x and M_y respectively. The initial vector σ has 1 at the i^{th} position and zero elsewhere. The final vector f has 1 at the j^{th} position and zero elsewhere. □

4 SOME QUESTIONS TO THINK ABOUT!

The following questions are worth thinking about.

1. We are interested in an "efficient" algorithm that takes as input a DFA that accepts a finite language $L \subseteq \{x, y\}^*$ and two matrices M_x and M_y and returns the matrix $[M_x, M_y](L)$.

2. We are interested in an "efficient" algorithm that takes as input a number n, a DFA that accepts a language $L \subseteq \{x, y\}^*$ and two matrices M_x and M_y and returns the matrix $[M_x, M_y](L \cap \Sigma^{\leq n})$.

3. (designing automata hashing) We can think about $[M_x, M_y](L)$ as a hash function. The probability that two languages collide is low. To boost the probability, one can think about multiple matrix pairs (M_x, M_y) and compute a hash vector. This hashing can be used to give a randomized equivalence checking of automata. Obviously this is not good if have to just compare two machines unless we can do all this in linear time. Recall

that equivalence of automata can be done in $O(n \log n)$. A randomized linear time algorithm will be a good improvement. On the other hand, if we have to compare many automatas will hashing help?

4. We know that with high probability $[M_x, M_y](L_1) \neq [M_x, M_y](L_2)$ if the languages are different. If someone gives the matrices M_x, M_y and $[M_x, M_y](L)$ and promises that L is regular, can we return a regular language L' where $[M_x, M_y](L') = [M_x, M_y](L)$. We can boost this confidence by considering many pairs of (M_x, M_y).

5. Lemmas 2.1 and 2.2 talk about all languages. Can we make a stronger claim about regular languages (at least as strong as Lemma 2.2)?

6. Can we recast this M_x and M_y matrices into normal DFA. Maybe rather than taking a weight q when we take letter x from a state, we can go to a state which stands for q^{th} element of the field. Since it is a finite field of size $3d$, the machine blows up only by that much.

REFERENCES

[1] V. Arvind, P. Mukhopadhyay, and S. Raja. Randomized polynomial time identity testing for noncommutative circuits, https://arxiv.org/abs/1606.00596.

[2] Andrej Bogdanov and Hoeteck Wee. More on noncommutative polynomial identity testing, https://www.cse.cuhk.edu.hk/~andrejb/pubs/ncpit.pdf.