Energy-Based Subclasses of Regular Languages

by

Burak Çetin

Submitted to the Department of Computer Engineering in partial fulfillment of the requirements for the degree of Bachelor of Science

Undergraduate Program in Computer Engineering

Boğaziçi University

Spring 2022

Energy-Based Subclasses of Regular Languages

APPROVED BY:

Prof. A. C. Cem Say ....

(Project Supervisor)

DATE OF APPROVAL: . .

ACKNOWLEDGEMENTS

I would like to thank my project advisor A. C. Cem Say for his guidance on this project. And to my family and friends for their support.

ABSTRACT

Energy-Based Subclasses of Regular Languages

On the theoretical study of simple computational models like DFA's (Discrete Finite Automata) and the languages that are recognized by these auotomata there are still a lot of avenues for exploration. One such topic is the amount of maximum incoming transitions to a state per symbol in the alphabet. This metric relates to the energy spent during calculation for the more complex versions of these automata. In this work we take some initial steps for a better knowledge of language classes that can be generated based on this metric. And give some proofs about closure properties that these energy classes do or do not have.

ÖZET

Düzenli Dillerin Enerji Tabanlı Alt Sınıfları

SDM (Sonlu Durum Makinesi) gibi basit hesaplama modelleri hakkında hala inceleme yapılmamış birçok konu mevcut. Bunlardan biri tek bir duruma aynı sembol ile yapılan geçişlerin mininum sayısıdır. Bu ölçüt bu tür makinelerin daha karmaşık versiyonlarında hesaplama sürecinde harcanan enerjiyi etkileyen bir etmen. Bu çalışmada bu metrik üzerinden tanımlanan bazı düzenli dil alt ailelerini tanımlayacağız. Sonra da bu ailelerin farklı özellikleri taşıyıp taşımadığı konusunda bazı ispatlar vereceğiz.

TABLE OF CONTENTS

1. INTRODUCTION AND MOTIVATION

On the study of one of the simplest computational models like DFA's (Discrete Finite Automata) and the familiy of regular languages there are still gaps in our knowledge. A lot of subfamilies of the regular languages can be constructed based on different properties that the languages can hold. There are a lot recent papers looking into such families of languages. In this work we are focusing on a specific property of a regular languages, what is the minimum number of the amount of incoming transitions to a state in a DFA recognizing this language for any symbol. A formal and more clear definition of this property will be given later. For introductory purposes it is important to point out that this metric cannot be determined directly from the minimal DFA for a language. The reason is that one can generate a DFA recognizing the same language as the minimal DFA with some computationally redundant states. As a result the search for which regular language falls under which family under the minimal incoming transitions per state requirement is a novel question that cannot be easily answered using known computational theory.

2. DEFINITIONS AND EXAMPLES

For clarity, the definition of a Deterministic Finite Automata and relevant concepts are provided here.

Definition 1. *A DFA is a quintuple* (Q, Σ, f, s, A) *where*
Q *is the set of states;*
Σ *is the input alphabet;*
$f : Q \times \Sigma \to Q$ *is the complete transition function;*
$s \in Q$ *is the starting state;*
$A \subseteq Q$ *is the set of accepting states.*

It is important to point out that the transition function covers the domain. As a result the computation will never halt in this machine until the entire input string is inserted into the machine.

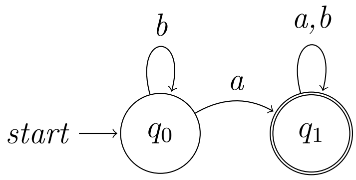
Definition 2. *For any regular language* L *with alphabet size* n*,* $L \in E_i, i \in \mathbb{Z}^+$ *holds when*

- *There is a DFA that recognizes* L *with no states that have more than* i *incoming transitions for any symbol in the input alphabet.*

As proven in the article [1], E_j for $j > n$ is the entire family regular languages. Meaning that there cannot be a language requiring more than $n+1$ incoming transitions per symbol to a state. From this definition it also follows that for $i < j$, $E_i \subseteq E_j$.

As we will focus on binary regular languages, any future mention of the families E_i will be based on an alphabet size $n = 2$.

Example 1. *Define the language* L_a *to be the language of strings containing* a *as a substring.* L_a *can be recognized with the following DFA:*



To determine which E_i families this language falls in we can check the maximum incoming transitions per symbol for a state. The state $q1$ has 2 incoming a-transitions. Existence of this DFA implies this language is in E_2. Keep in mind that this by itself is not enough to claim that L_{aa} is not in E_1. We did not prove there cannot be a DFA that can recognize the language with no more than 1 incoming transition per symbol per state. To be able to prove that $L_a \notin E_1$ we will make some useful definitions.

Definition 3. *For any DFA M, two states of M are related by the relation $\sim_M$ if and only if the language recognized by M after changing the initial state to be these states is the same language.*

This relation between the states of a DFA will be really useful in future proofs.

Theorem 1. *$\sim_M$ is an equivalence relation.*

We omit the proof as it is easy to see that this relation is reflexive, symmetric and transitive.

Definition 4. *For any DFA M define the equivalence class of the state reached by inputting the string ω to M be G_ω by $\sim_M$.*

Now we are ready to show that $L_a \notin E_1$ by contradiction.

Proof. Assume $L_a \in E_1$. Then there is a DFA M recognizing the language with no more than one incoming transition per symbol per state. Consider the states of machine that fall in G_ϵ and G_a. These classes are clearly distinct and non-empty since G_ϵ consists

of reject states and contains the initial state and G_a consists of accept states and the machine must have one such accept state. Now consider any state q in G_ϵ. Name the state that we would end up if we were to input a to q, r. Now by definition of the machine r would be a state that no matter what string you would input into it, you would end up in an accept state. This is the exact same behaviour with the states in G_a. This means that any state in G_ϵ a-transitions into a state in G_a. In a similar way we can see that any state in G_a also a-transitions to a state in G_a.
Now we know that the states in G_a have a total of at least $|G_a| + |G_\epsilon|$ states which is bigger than $|G_a|$. Then there must be at least one state in G_a with more than one incoming a-transition, which contradicts the definition of the machine. $\square$

Our hope is that the given example language L_a and the proof that it is in E_2 but not in E_1 will shed an extra light into the proofs of our results.

3. RESULTS

We begin an investigation into the binary regular language families E_1,E_2 and E_3 by considering if they are closed under some common regular language operations or not.

3.1. Complements

Theorem 2. *E_1 is closed under complements.*

Proof. $L \in E_1$ implies there is an FSM M recognizing L such that M has no states with more than 1 incoming transition per symbol. By changing accept states of M to reject states and vice versa an FSM recognizing $\overline{L}$ can be constructed without altering the transitions in any way. $\overline{L} \in E_1$. □

Theorem 3. *E_2 is closed under complements.*

Proof. Assume otherwise. Then there is a language $L \in E_2$ such that $\overline{L} \notin E_2$. Similarly, changing the accept states and reject states of the machine recognizing L, one can construct a machine recognizing $\overline{L}$ without altering the transitions of the machine. Then there is an FSM recognizing $\overline{L}$ with no more than 2 incoming transition per symbol for all states. This shows that $L \in E_2$ which contradicts the assumption. □

As E_3 is the entire family of binary regular languages, it is closed under complements trivially.

3.2. Intersections

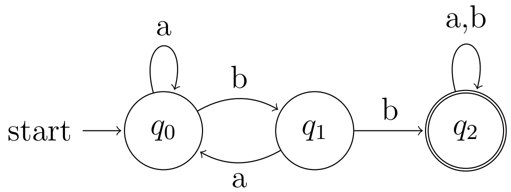
Theorem 4. *E_1 is closed under intersections.*

Proof. For any two binary languages O and P in E_1 there are, by definition, DFAs $M_O = (Q_O, \{a, b\}, f_O, s_O, A_O)$ and $M_P = (Q_P, \{a, b\}, f_P, s_P, A_P)$ recognizing them with no more than one incoming transition per symbol to a state. Construct a DFA $M = (Q_O \times Q_P, \{a, b\}, f_O \times f_P, (s_O, s_P), A_O \cap A_P)$. It is easy to see that M is constructed in such a way that it recognizes $O \cap P$. If we assume E_1 to be not closed M must have a state (q_{O1}, q_{P1}) with more than one incoming transition per symbol. Without loss of generality M must contain unique states (q_{O2}, Q_{P2}) and (q_{O3}, Q_{P3}) that transition to (q_{O1}, q_{P1}) with symbol a. This implies in machine M_O the states q_{O2} and q_{O3} both transition into state q_{O1}. Which contradicts the definition of M_O. □

Let L_1 be the language of strings that have an occurrence of bb. Let L_2 be the language of strings that do not end with a.

Theorem 5. *L_1 is in E_2 and not in E_1.*

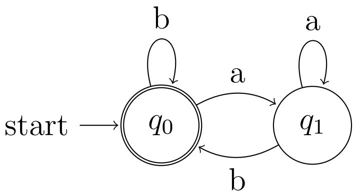
Proof. The following DFA recognizes L_1 with a maximum of 2 incoming transitions for the same symbol. Therefore $L_1 \in E_2$.



Assume $L_1 \in E_1$. Then there is a DFA, M recognizing L_1 with exactly one incoming transition per state per symbol. Let q_i be the sequence of states generated by inputting b repeatedly to M. Since there are finitely many states, this sequence must contain repetitions and by definition all q_i, except q_0, already has an incoming b transition. As a result the first state that appears twice in this sequence must be q_0, creating a closed loop. This loop must have at least 3 unique states, 2 of which are not accepting states, since the string bb must be accepted but the strings ϵ and b must not. Then a string of b symbols with a length equal to the number of states in this loop will end in q_0 which is a reject state. Then M cannot recognize L_1. □

Theorem 6. *L_2 is in E_2 and not in E_1.*

Proof. The following DFA recognizes L_2 with a maximum of 2 incoming transitions for the same symbol. Therefore $L_2 \in E_2$.



For any DFA M recognizing L_2 consider the sets of states G_ϵ and G_a. If a state in G_ϵ were to be the starting state, the resulting language recognized would contain ϵ. If a state in G_a were to be the starting state, the resulting language would not contain ϵ. Then these are distinct equivalency classes.

Observe that if a state from G_a would become the starting state, the machine would accept the strings that end with b. If we were to insert a string of the form $b\omega$ to a state in G_a it would be accepted if and only if ω is ϵ or ends with b. This is the accepting behaviour of the initial state of the machine and as a result of any state in $G_{epsilon}$. Then any state in G_a transitions to a state in G_ϵ with the symbol b.

Observe that if we insert a string of the form $b\omega$ to a state in G_ϵ, it would be accepted if and only if ω is ϵ or ends with b. This also is the same accepting behaviour of states in G_ϵ. Then any state in G_ϵ transitions to a state in $G_{epsilon}$ with the symbol b.

Then the states in G_ϵ receive at least $|G_\epsilon| + |G_a|$ transitions with the symbol b. As $|G_\epsilon| + |G_a|$ is strictly larger than $|G_\epsilon|$, there must be a state in G_ϵ receiving more than one transition with the same symbol. $L_2 \notin E_1$. □

Theorem 7. *E_2 is not closed under intersections.*

Proof. It is proven that $L_1 \cap L_2$ not in E_2 here [1] at section 4. □

3.3. Unions

Theorem 8. *E_1 is closed under unions.*

Proof. As E_1 is closed under both intersections and complements it is also closed under unions by a simple application of De Morgan's rule. □

Theorem 9. *E_2 is not closed under unions.*

Proof. As E_i are closed under complements $\overline{L_1 \cap L_2}$ is not in E_2. By De Morgan's law $\overline{L_1} \cup \overline{L_2}$ is also not in E_2. Since E_2 is closed under complements $\overline{L_1}$ and $\overline{L_2}$ are in E_2. □

3.4. Concatenations

Using the same examples from the last section we can conclude that E_2 is not closed under concatenation.

Theorem 10. *E_2 is not closed under concatenation.*

Proof. Start by showing $L_1L_2 = L_1 \cap L_2$.
For any string $\omega \in L_1L_2$, $\omega = \omega_1\omega_2$ where $\omega_1 \in L_1$ and $omega_2 \in L_2$. By definition, ω_2 does not end with a. Then ω also does not. $\omega \in L_1$. Similarly, ω_1 contains an occurrence of bb. Then ω does as well.
Take any string $\omega \in L_1 \cup L_2$. By definition of L_2, $\epsilon \in L_2$. $L_1 \cup L_2 \subseteq L_1 = L_1\epsilon \subseteq L_1L_2$. Two way containment concludes that these two languages are identical. Then $L_1L_2 \notin E_2$. □

4. CONCLUSION AND DISCUSSION

As this was a purely theoretical work there are no direct conclusions aside from the provided proofs of closure properties. In my opinion, there are still a lot of properties of the E_i language families that can be shown with relative ease. The main goal of this work is to provide a stepping stone for future work about a general theory of energy-class subfamilies.

5. FUTURE WORK

On a future work one can aim to find more novel examples of non-E_2 family binary languages. By examining more examples of such languages will reveal if there is an underlying structure of non-E_2 languages that will let us construct such languages from the E_1 or E_2 languages. As this is an ambitious conceptualization, I am not expecting it to be a an easy task to uncover such a structure, assuming it exists. In this work I provided any finding that will help future research into the general theory of energy-subclasses of regular languages.

REFERENCES

1. Öykü Yılmaz, F. Kıyak, M. Üngör and A. C. C. Say, “Energy Complexity of Regular Language Recognition”, *arXiv:2204.06025*, 2022.