Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Supplementary Information and Nondeterministic Finite Automata

Bachelor Thesis

2020
Martin Pašen

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Supplementary Information and Nondeterministic Finite Automata

Bachelor Thesis

Study Programme:	Computer Science
Field of Study:	Computer Science
Department:	Department of Computer Science
Supervisor:	prof. RNDr. Branislav Rovan PhD.

Bratislava, 2020
Martin Pašen

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Martin Pašen
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Supplementary Information and Nondeterministic Finite Automata
Dodatočná informácia a nedeterministické konečné automaty

Anotácia: Práca je ďalším príspevkom k porozumeniu pojmu užitočnosť informácie. Skúma vlastnosti tried jazykov, ktoré môžeme priradiť k danej informácii na základe užitočnosti pre ich akceptovanie nedeterministickými konečnými automatmi.

Cieľ: Nadviazať na doterajší výskum pojmu informácia. Využiť formalizmus nedeterministickych konečných automatov a skúmať vlastnosti tried jazykov, pre akceptovanie ktorých je daná informácia užitočná.

Vedúci: prof. RNDr. Branislav Rovan, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 28.10.2019

Dátum schválenia: 28.10.2019

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname: Martin Pašen
Study programme: Computer Science (Single degree study, bachelor I. deg., full time form)
Field of Study: Computer Science
Type of Thesis: Bachelor´s thesis
Language of Thesis: English
Secondary language: Slovak

Title: Supplementary Information and Nondeterministic Finite Automata

Annotation: The thesis contributes to a further insight into the notion of the usefulness of information. It studies properties of families of languages associated to a given information based on its usefulness for their recognition by nondeterministic finite automata.

Aim: To continue the current investigation of the notion of information. Using the formalism of nondeterministic finite automata investigate properties of the families of languages for which is the given information useful.

Supervisor: prof. RNDr. Branislav Rovan, PhD.
Department: FMFI.KI - Department of Computer Science
Head of department: prof. RNDr. Martin Škoviera, PhD.

Assigned: 28.10.2019

Approved: 28.10.2019 doc. RNDr. Daniel Olejár, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Acknowledgments: I would like to express my gratitude to my advisor, Branislav Rovan. Of course his knowledge of the domain cannot be questioned, but that is not what I am most grateful for. Even though there is a great academic gap between us, I have never felt it. It was as if I worked with a colleague, not a supervisor.

Abstrakt

Hlavným pojmom práce je užitočnosť informácie. Ako formálny základ používame teóriu formálnych jazykov. Práca je zameraná na regulárne jazyky a užitočnosť informácie definujeme ako binárnu reláciu na regulárnych jazykoch. Ako výpočtový model používame nedeterministický konečný automat. Skúmame vlastnosti užitočnosti ako relácie. Prezentujeme uzáverové vlastnosti tried definovaných pomocou dodatočnej informácie (trieda jazykov, ktoré môžu byť zjednodušené pomocou fixnej dodatočnej informácie). Taktiež skúmame pojem informačnej sily dodatočnej informácie a možné vzťahy v ktorých môžu byť dve dodatočné informácie za rôznych predpokladov.

Kľúčové slová: užitočnosť informácie, trieda definovaná dodatočnou informáciou, informačná sila, nedeterministický konečný automat

Abstract

The core concept of this thesis is the usefulness of information. For a formal background of research we use the theory of formal languages. The thesis is focused on regular languages and we define usefulness as a binary relation on regular languages. We choose nondeterministic finite automaton for our computational model. We examine properties of usefulness as a relation. We present closure properties of families defined by advice (family of languages, which can be simplified using fixed advice). We also examine the informational power of supplementary information and possible relations of two languages providing supplementary information.

Keywords: usefulness of supplementary information, family defined by advice, informational power, nondeterministic finite automaton

Contents

List of Figures

List of Tables

Chapter 1

Introduction and definitions

One of the core concepts of computer science is information. An algorithm is "just a modification of information". Computer networking is "just transportation of information". Cryptography is "just safety of information". The database is "just an effective way to store information". In many languages, computer science or science of computation is translated as informatics, the science of information.

Even though the information is such an essential notion, it is only an intuitive concept. Same as an algorithm it has no rigid formal definition. If it is not formalized, you can not use formal techniques in proofs. So on the one side, information and the algorithm are an important part of computer science, on the other side, we can not study them formally. Without formal definition, we can not answer questions such as: is there an algorithm for every problem, or is there a lower bound for the amount of information that we need to obtain about the sequence, to be able to sort it? To get a formally correct answer, we have to find a formal definition for these concepts. That is not an easy task since we can not prove that our definition is correct.

There were many attempts to formalize an algorithm. The one that stuck is a Church-Turing thesis. It says that every algorithm can be simulated by the Turing machine or as Savage said in 1987: "an algorithm is a computational process defined by a Turing machine". With a formalization at disposal, it was possible to prove the properties of an algorithm. For example, it was shown that there are problems, that can not be solved by any algorithm (e. g., a halting problem).

Before such a thesis was accepted by a community, a lot of work had to be done. A lot of different approaches were studied. Many of them were found to be equivalent to Turing's approach, which amplified its chance of success. Surely there were many attempts to show that the Turing machine is too weak. That there exists an algorithm that can not be simulated by a Turing machine. Since no one has come up with a strong argument against the thesis, it got a broad acceptance.

Shannon took a similar approach in his study of information and its size. He

formalized information as a sequence of symbols and its size as the length of the sequence. His approach had great success. Both literally and figuratively it helped the human race to land on the moon.

Even though Shannon has already proposed a formalization of information and it has brought great achievements, we shall take a different approach. The reason is that we shall focus on a different element of information. While Shannon worked mostly with the size of the information, we shall study the usefulness of the information.

Although we shall work with only one formal definition and study usefulness through this definition, we shall always ask whether our reasoning is not based on a detail of definition. Whether a slight change of definition would not break the chain of thoughts. The goal is to present formally and intuitively correct reasoning.

As the background for the formalization of the concept, we shall use the theory of formal languages. Basic knowledge of this field is expected, e.g., definitions and understanding of nondeterministic finite automaton, configuration, acceptance of a word and a language ... For these definitions see [2].

The following definitions shall lead to the key concept of the thesis, the usefulness of information (of advice). Our research concerns regular languages, so both the advice and the problem shall be regular. We shall, therefore, define usefulness as a relation on regular languages. As the title of the thesis suggests, we shall concentrate on nondeterminism and all our definitions shall be in the nondeterministic setting. For analogous definitions and work done in deterministic settings see [5].

Note. We shall work with the nondeterministic automata without epsilon transitions. Most of the time, the only important information about the alphabet shall be its size. Therefore sometimes we might skip the details of the alphabet and only mention its size s. If we do that, we shall assume that the alphabet is the first s letters of the Latin alphabet. For example, if we say that an automaton works with a unary alphabet, we shall assume it is $\{a\}$.

Definition 1.0.1. The state complexity of a nondeterministic finite state automaton $A = (Q, \Sigma, \Delta, q_0, F)$ ($\#_s(A)$) is the number of its states , $\#_s(A) = |Q|$.

Definition 1.0.2. The nondeterministic state complexity of a regular language L ($nsc(L)$) is the minimal number of states required by an NFA to accept L, $nsc(L) = min\{\#_s(A)|L(A) = L\}$.

Example 1.0.1. Languages over the unary alphabet with state complexity 1 are Σ^*, an empty language $\emptyset$, and a language containing only an empty string $\{\epsilon\}$. Languages with state complexity 2 are $\{\epsilon, a\}$, $\{a\}$, Σ^+, $\{a^{2k}|k \in \mathbb{N}\}$ and $\{a^{2k+1}|k \in \mathbb{N}\}$. This can be shown by an exhaustive search. The choice we have when we are constructing an automaton with two states is in a set of accepting states and a transition function. Each

state can be accepting or non-accepting so that is 2^2 options. Each possible transition will be or will not be present, so that is 2^4 options. Together it is $2^6 = 64$ possibilities. But that can be reduced. For example, the second state must be reachable, or there must be at least one accepting state.

Note. While working with languages over the unary alphabet, we can associate words with numbers written in the non-positional system. Word *aaaa* can correspond to number 4, ϵ to 0. An important family of regular languages over the unary alphabet are languages with the structure $\{a^{x*k}|k \in \mathbb{N}\}$ where x is a natural number. If we associate words with its numbers than these languages correspond to divisibility problems. Only and all numbers divisible by x are in $\{a^{x*k}|k \in \mathbb{N}\}$. Since we shall use these problems frequently, we shall abbreviate their notation to $L_{[x]}$.

Definition 1.0.3. The nondeterministic usefulness is a relation $U_n \subseteq \mathcal{R} \times \mathcal{R}$ defined as follows. A regular language L_{adv} is a nondeterministically useful advice for a regular language L_{prob} ($(L_{adv}, L_{prob}) \in U_n$) if:

- $\exists L_{new} \in \mathcal{R} : L_{prob} = L_{adv} \cap L_{new} \wedge nsc(L_{new}) < nsc(L_{prob})$
- $nsc(L_{adv}) < nsc(L_{prob})$

Example 1.0.2. Advice $L_{[2]}$ is useful to a problem $L_{[6]}$. The formal proof shall be presented in the next chapter 2.1.3. But on the intuitive level we know, that if we need to find out whether a number is divisible by 6 and we receive a hint that it is divisible by 2, all we need to do is to check the divisibility by 3.

Note. The first condition of nondeterministic usefulness says that we must be able to construct a simpler solution L_{new} to the old problem L_{prob} after receiving the advice L_{adv} . The second condition says that the advice must be simpler than the problem L_{prob} as well.

The necessity for definition of nondeterministic usefulness comes from the fact, that deterministic and nondeterministic usefulness differ, as shown in [3]. They found a problem L_{prob} that is deterministically decomposable (there exists a useful advice, that allows us to solve the old problem more easily) and nondeterministically indecomposable (there is no advice that could help us simplify the problem).

In this thesis we shall mostly work with a nondeterministic setting, therefore we shall shorten nondeterministically useful to useful, nondeterministic state complexity to state complexity or just complexity and instead of U_n we might use U.

Another important concept of the thesis is the family of languages that can be simplified using the same advice. Later we shall explore its properties (mainly the closure properties).

Definition 1.0.4. The family defined by advice L_{adv} $(\mathcal{L}(L_{adv}))$, is a family of all languages that can be simplified using the advice L_{adv} , $\mathcal{L}(L_{adv}) = \{L|(L_{adv}, L) \in U_n\}$.

Chapter 2

Supplementary information

In this chapter, we shall examine the properties of usefulness U_n and closure properties of families defined by advice. In the first Section 2.1, we shall study the usefulness as a relation. In Subsection 2.1.1 we shall show that U_n is asymmetric and irreflexive. In the following subsection 2.1.2, we shall address the transitivity, prove the divisibility lemma 2.1.3 and present the first schema 2.1. Then we shall direct our attention to the uniqueness of decomposition 2.1.3. In the last Subsection 2.1.4 of the first Section, we shall discuss whether a finite language can be useful.

In the second Section 2.2, we shall examine the closure properties of families defined by advice. In Subsection 2.2.1 we shall present two proofs about closure under intersection as well as the second schema 2.4. The following will be closure under union 2.2.2 and closure under concatenation and iteration 2.2.3.

2.1 Relation U_n

For advice to be useful to a problem, it has to satisfy two conditions. First, it has to simplify the problem (there has to be a new easier solution to the problem), second, it itself has to be easier than the problem. Let us consider only the second condition in the definition of U_n and call the new relation E_n . We only consider the fact that the advice has to be easier than the problem. The relation E_n is a superset of U_n . We shall use this fact since some properties of the relation are inheritable. In other words, if a property holds for E_n it might automatically hold for U_n as well.

2.1.1 U_n is irreflexive and anti-symmetric

A relation R is said to be anti-symmetric if for each pair of distinct elements a and b at most one of $R(a, b)$ and $R(b, a)$ holds. A relation R is said to be irreflexive, if there is no element a such that $R(a, a)$ holds [1]. The relation E_n is based, just on the state complexity of the two languages (requiring $nsc(L_1) < nsc(L_2)$ for $E_n(L_1, L_2)$ to

hold). Therefore it is easy to see that E_n is irreflexive and anti-symmetric, just like the relation $<$ on natural numbers. Since $U_n \subseteq E_n$ it easily follows the following:

Theorem 2.1.1. *The relation U_n is irreflexive and anti-symmetric.*

2.1.2 U_n is not transitive

In the previous subsection we were able to simplify our arguments using the relation E_n and the fact the irreflexivity and anti-symmetry are inheritable properties of relations. This is no longer possible when considering transitivity. While E_n is transitive, we shall prove that U_n is not transitive. We shall proceed as follows.

Note that the advice and the problem are both regular languages, thus we can consider the same language once as the advice and once as the problem. The following question arises, is usefulness transitive? In other words, suppose we have a language L_{adv} , that is useful to an intermediate problem L_{int} and L_{int}, as advice, is useful to a problem L_{prob}. Is L_{adv} automatically useful to L_{prob} ? We shall show that usefulness is not transitive, by presenting a counterexample. First, as a preparation for the transitivity, we shall present the divisibility lemma.

Lemma 2.1.2. *Let n be a non-zero natural number. The nondeterministic state complexity of $L_{[n]}$ is n.*

Proof. The upper bound is trivial (the straightforward construction of an automaton for this language has n states, each corresponding to one possible remainder after division by n). To show that the state complexity is at least n we shall use a proof by contradiction.

A border case is $n = 1$. For this case, we do not have to prove its lower bound, since 1 is a minimal state complexity of all languages. From now on suppose that n is greater than 1.

Suppose there is a non-deterministic automaton A that accepts the language $L_{[n]}$ and has less than n states. The word a^n is in language $L_{[n]}$, therefore there must be an accepting computation on this word. Let $q_0, q_1, \ldots, q_n$ be the sequence of states that A reaches during the computation and let the state q_i to be a state that A is in after reading the prefix a^i. Since A has less than n states, there must be two indices k and l such that $q_k = q_l$, $l > k$, and $l - k < n$. Part of the computation from q_{k+1} to q_l can be either repeated or cut out and the new computation will still be accepting (this idea comes from the pumping lemma). If we cut out this part we get an accepting computation on word $a^{n-(l-k)}$. Since $l - k$ is smaller than n and greater than 1, A accepts a word that is longer than 0 but shorter than n which is a contradiction. □

[1] A relation that is irreflexive and anti-symmetric is sometimes called asymmetric. An example of such relation is, e.g., the relation $<$ on natural numbers

Lemma 2.1.3 (Divisibility lemma)**.** *Let k and l be natural numbers and let k be smaller than l. The advice $L_{[k]}$ is useful to $L_{[l]}$ if and only if there exists a natural number m smaller than l satisfying $scm(k,m) = l$.*

Proof. First implication: there exists a natural number m smaller than l satisfying $scm(k,m) = l \implies L_{[k]}$ is useful to $L_{[l]}$. From the previous lemma 2.1.2 we know that a state complexity of $L_{[k]}$ and $L_{[m]}$ is smaller than the state complexity of $L_{[l]}$. If a number is divisible by two numbers, then it is divisible by their smallest common multiple. Therefore if a number is divisible by k and m it is divisible by l as well. Thus $L_{[l]} = L_{[k]} \cap L_{[m]}$ and $L_{[k]}$ is a useful advice to $L_{[l]}$.

Second implication: $L_{[k]}$ is useful to $L_{[l]} \implies$ there exists a natural number m smaller than l satisfying $scm(k,m) = l$. The advice $L_{[k]}$ is useful to $L_{[l]}$, so there exists a language L_{new} with state complexity smaller than l and the intersection of $L_{[k]}$ and L_{new} is $L_{[l]}$. Let A be the nondeterministic automaton accepting L_{new}. A must accept the word a^l. Similarly to the previous proof, let $q_0, q_1, \ldots, q_l$ be the sequence of states that A is in during the accepting computation on the word a^l. Since A has less than l states, there must be two indices i and j such that $q_i = q_j$, $i > j$, and $j - i < l$. The part of the computation from q_{i+1} to q_j can be either repeated or cut out and the new computation will still be accepting. Let r be the length of that part ($r = j - i$). By repeating the part we can construct accepting computation on a word a^{l+n*r}, where n is any natural number. Let x be the smallest common multiple of r and k. Both $L_{[k]}$ and L_{new} contain the word a^{l+x}, therefore $L_{[l]}$ contains this word as well and x must be divisible by l. Since k divides l (otherwise $L_{[k]}$ would not be a super-set of $L_{[l]}$) there must exist a number m that divides r such that $scm(k,m) = l$. Since m divides r, m is at least as small as r and r is smaller than l so m is smaller than l as well and the proof is complete. □

Now that we have the divisibility lemma prepared, we are ready to prove that usefulness is not transitive.

Theorem 2.1.4. *Usefulness is not transitive.*

Proof. We shall prove the theorem by presenting a counterexample. A language L_{adv} will be useful to an intermediate language L_{int}, L_{int} will be useful to L_{prob} but L_{adv} will not be useful to L_{prob}.

Let k and l be natural numbers greater than 1, with gcd equal to 1. Let $L_{adv} = L_{[k]}$, $L_{int} = L_{[k*l]}$ and $L_{prob} = L_{[k^2*l]}$. The facts that L_{adv} is useful to L_{int}, L_{int} is useful to L_{prob} and that L_{adv} is not useful to L_{prob} are clear consequences of the divisibility lemma and the theorem is proved. □

The Figure 2.1 shows an intuitive schema behind the proof. Every schema of the thesis is very informal. It is not supposed to work as proof. It is supposed to help to

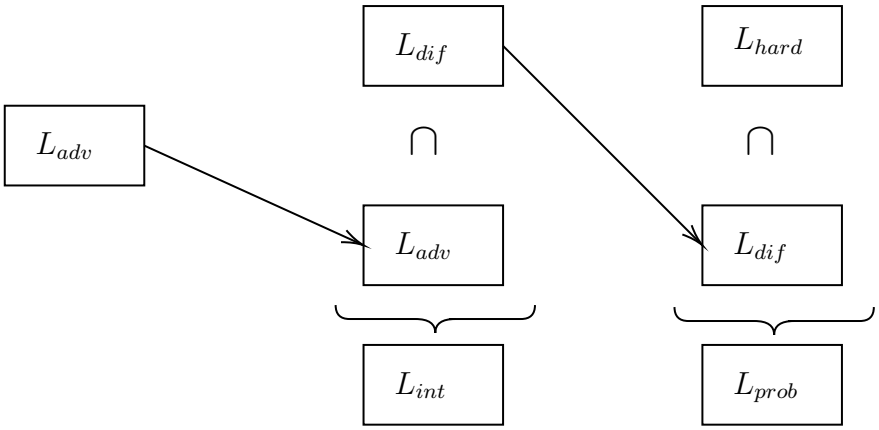


Figure 2.1: Schema for the transitivity.
The figure shows the schema used in the proof of theorem 2.1.4. In the proof we used $L_{[k]}$ as advice, $L_{[l]}$ as a different problem ($gcm(k,l) = 1$) and $L_{[k^2]}$ as a harder version of advice.

build intuition and to see the ideas of the proof in a bigger picture. Languages that are used in the schema are:

- L_{adv} -an infinite language with state complexity greater than 1
- L_{dif} - different language than L_{adv}
- L_{hard} - harder version of L_{adv}

An example could be $L_{adv} = L_{[2]}$, $L_{dif} = L_{[3]}$ and $L_{hard} = L_{[2^2]}$. The language L_{dif} is neither a subset nor a superset of L_{adv} . The state complexity of an intersection of L_{adv} and L_{dif} is a product of the state complexities of L_{adv} and L_{dif} (similar to the fact that random variables are independent if their joint entropy is the sum of individual entropies). The divisibility by 2 can be also seen as keeping only every other number. If we repeat this on the numbers that are left we will get the divisibility by 4. So the divisibility by 4 is in a sense two times repeated divisibility by 2 and therefore we can see it as a harder version of it.

The schema used in the proof is shown in Figure 2.1. In our proof we used $L_{[k]}$ as advice, $L_{[l]}$ as a different problem ($gcm(k,l) = 1$) and $L_{[k^2]}$ as a harder version of the advice. Arrows show how a language is useful to the problem, which sub-problem of the problem it solves.

2.1.3 Uniqueness of decomposition

An alternative view of the advice, the problem, and the new solution is a decomposition. Instead of being in the position, where someone helps us by providing supplementary information and we are trying to simplify the solution to the problem, we can try to

decompose the problem into two simpler subproblems. Imagine we have a job to be done as soon as possible, and we have two workers that can work simultaneously, but they cannot work together on the same assignment. We have to split the problem into two easier assignments, one for each worker. For example, if we need to check whether a number is divisible by 6, one worker can check divisibility by 2, the other divisibility by 3, and combining their results we can get a result for the divisibility by 6.

The undecomposable problem, as the name suggests, is a problem that can not be divided into two simpler problems. If we correlate decomposition of problems with the factorization of numbers, undecomposable problems correspond to prime numbers. While factorization is unique (it does not matter whether you first decompose 30 to $2*15$ or $3*10$, you will always end up with the same primes in the end) decomposition is not.

Note. There is a difference between the decomposition and the decomposition into undecomposable problems. $L_{[2]} \cap L_{[15]}$ is the decomposition of $L_{[30]}$ but it is not the decomposition into undecomposable problems yet.

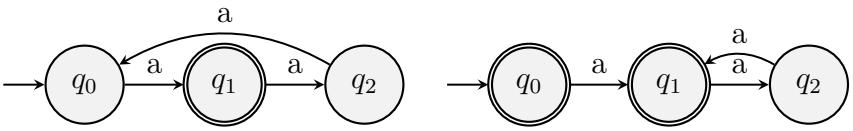
Theorem 2.1.5. *Regular languages do not have a unique decomposition to undecomposable problems.*

Proof. To prove the theorem we shall use the generalization of the divisibility problem. Instead of asking whether a number is divisible by 6, we are going to ask whether the number has remainder 1 after dividing by 6, $L_{prob} = L_{[6]} \cdot a$.

A straightforward decomposition of L_{prob} is $(L_{[3]} \cdot a) \cap (L_{[2]} \cdot a)$. The language $L_{[3]} \cdot a$ does not contain ϵ. Therefore if we add ϵ to the second language it will not affect the result of the intersection. Therefore decomposition $(L_{[3]} \cdot a) \cap (L_{[2]} \cdot a \cup \{a\})$ is also correct.

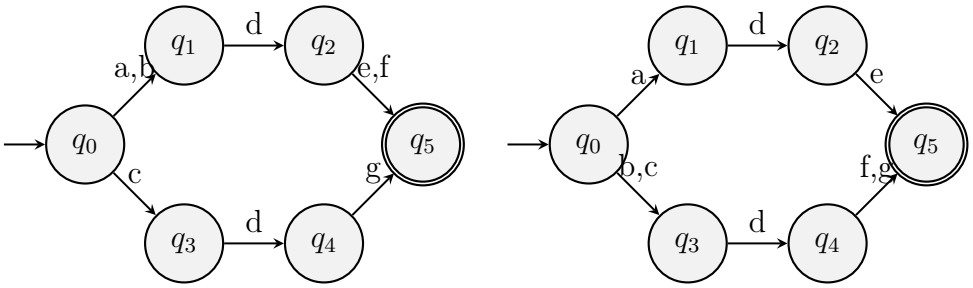
To finish the proof we need to show that both decompositions are decompositions into the undecomposable problems. We need to prove that all three subproblems are undecomposable. Problem $L_{[2]} \cdot a$ is undecomposable, because its complexity is 2 and languages with state complexity 1 are not useful to any problem.

Both $L_{[3]} \cdot a$ and $L_{[2]} \cdot a \cup \{a\}$ are languages with state complexity 3, as shown in Figure 2.2. Languages with state complexity 1 are not useful to any problem, so the complexity of the advice and the new solution (if they exist) must be 2. From Example 1.0.1, we know that there are 5 languages with state complexity 2 : $\{\epsilon, a\}$, $\{a\}$, Σ^+, $\{a^{2k} | k \in \mathbb{N}\}$ and $\{a^{2k+1} | k \in \mathbb{N}\}$. The advice and the new solution have to be a superset of the problem, therefore only Σ^+ is an option. The same language can not be an advice and a new solution simultaneously, so both $L_{[3]} \cdot a$ and $L_{[2]} \cdot a \cup \{a\}$ are undecomposable. □



(a) Automaton for $L_{[3]} \cdot a$ (b) Automaton for $L_{[2]} \cdot a \cup \{\epsilon\}$

Figure 2.2: State complexity of $L_{[3]} \cdot a$ and $L_{[2]} \cdot a \cup \{\epsilon\}$ is lower or equal to 3.



(a) Automaton for $\{ade, adf, bde, bdf, cdg\}$ (b) Automaton for $\{ade, bdf, bdg, cdf, cdg\}$

Figure 2.3: Automata for $\{ade, adf, bde, bdf, cdg\}$ and $\{ade, bdf, bdg, cdf, cdg\}$.

2.1.4 Finite language can be useful

At first sight, it might seem impossible. We know that if advice is useful to a problem, then the problem is a proper subset of the advice. If the finite language is over the unary alphabet, we can get all subsets of the advice by changing some of the accepting states to non-accepting states. We are not adding any states therefore the state complexity of a subset of the language can not have greater complexity.

This does not have to hold for languages with a non-unary alphabet. The reason is that for the unary alphabet, every accepting state corresponds to one word. We can see in the automata in figure 2.3a, that there are states that correspond to more than one word, e. g. , state q_5.

We shall use the automaton 2.3a as an advice and 2.3b as a new solution. Therefore the problem that we shall simplify has to be $L_{prob} = \{ade, bdf, cdg\}$. In the following lemma we shall prove a lower bound for the state complexity of L_{prbo} and then we shall be ready to prove the theorem, that finite language can be useful.

Lemma 2.1.6. *The state complexity of $L_{prob} = \{ade, bdf, cdg\}$ is at least 8.*

Proof. We shall use the proof by contradiction. Suppose that the state complexity of L_{prob} is smaller than 8. That means that there exists an automaton A that accepts L_{prob} and it has less than 8 states. Let F be the following set of pairs $\{(\epsilon, ade), (a, de), (ad, e), (b, df), (bd, f), (c, dg), (cd, g)\}$. This set has a property, that if we take any pair (x_1, x_2) and we construct a word $w = x_1 \cdot x_2$ we get a word from L_{prob}. But if we take any two pairs (x_1, x_2) and (y_1, y_2) and construct a word $w = x_1 \cdot y_2$, we get a word that is not in L_{prob} (it is also called the fooling set).

Fix one accepting computation for every word of L_{prob}. The automaton A has less than 8 states, therefore there must be two pairs (x_1, x_2) and (y_1, y_2) from F, such that A reaches the same state after reading the first parts of those pairs. Therefore switching the prefix x_1 for y_1 does not affect whether A accepts that word or not. Since A accepts $x_1 \cdot x_2$ it also accepts $y_1 \cdot x_2$. That is a contradiction considering the properties of F. □

Theorem 2.1.7. *Finite language can be useful.*

Proof. Let $L_{adv} = \{ade, adf, bde, bdf, cdg\}$, $L_{prob} = \{ade, bdf, cdg\}$ and $L_{new} = \{ade, bdf, bdg, cdf, cdg\}$. Problem L_{prob} is equal to $L_{adv} \cap L_{new}$. Previous lemma and Figure 2.3 show that both L_{adv} and L_{new} are simpler than L_{prob}. Therefore L_{adv} is useful to L_{prob}. □

2.2 Closure properties of families defined by advice

This section is devoted to an examination of families defined by advice, the problems that can be simplified using some fixed advice. We shall focus on closure properties, e. g., let $L_1, L_2 \in \mathcal{L}(L_{adv})$. Does it follow that intersection (union, ...) of L_1 and L_2 is also in $\mathcal{L}(L_{adv})$?

2.2.1 Intersection

We shell present 2 counterexamples. The first counterexample might be seen as "just playing with the alphabet". It is an example of reasoning, that is based on the details of formalization. It would not work if we slightly generalized our definition of $\mathcal{L}(L_{adv})$ by allowing 1 to 1 homomorphisms, it does not offer a lot of insight into the concept of usefulness.

Theorem 2.2.1. *There exists L_{adv} such that $\mathcal{L}(L_{adv})$ is not closed under intersection.*

Proof. The main idea of the proof is that L_{prob1} and L_{prob2} are the same problems, but they use a different alphabet so their intersection will be an empty language. Since the alphabet is important we can not use the simplified notation for divisibility problems.

Let $L_{adv=}\{a^{2k} | k \in \mathbb{N}\} \cup \{b^{2k} | k \in \mathbb{N}\}$, $L_{prob1} = \{a^{6k} | k \in \mathbb{N}\}$ and $L_{prob2} = \{b^{6k} | k \in \mathbb{N}\}$. It is clear that both L_{prob1} and L_{prob2} can be simplified using L_{adv}. Now $L_{prob} = L_{prob1} \cap L_{prob2}$ is the empty language and it can not be simplified by any advice, showing that $\mathcal{L}(L_{adv})$ is not closed under intersection. □

The second proof will be more general and it will not be strictly based on the details of definition. The schema used in this proof is shown in Figure 2.4.

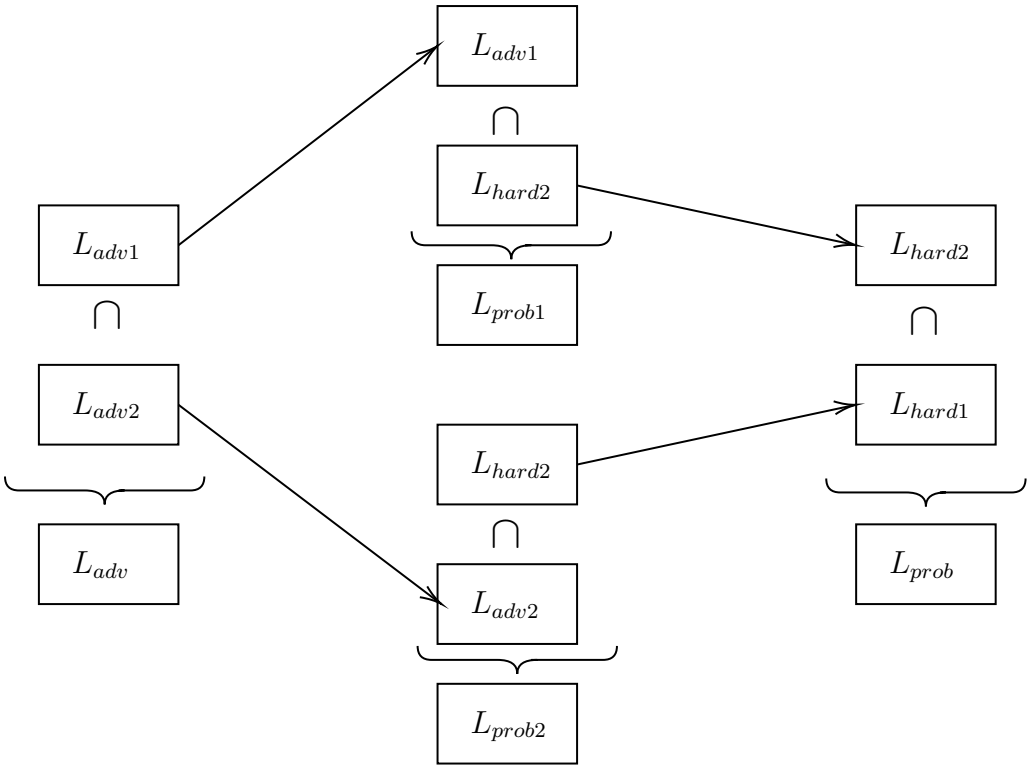


Figure 2.4: Schema for the intersection.
In the proof, we used $L_{[l]}$ as L_{adv1}, $L_{[k]}$ as L_{adv2}, $L_{[l^2]}$ as L_{hard1} and $L_{[k^2]}$ as L_{hard2}.

Proof. Let k and l be natural numbers greater than 1, such that their greatest common divisor is 1. Let $L_{adv} = L_{[k*l]}, L_{prob1} = L_{[k^2*l]}$ and $L_{prob2} = L_{[k*l^2]}$. Using the divisibility lemma we can show that both L_{prob1} and L_{prob2} can be simplified using L_{adv} while $L_{prob} = L_{prob1} \cap L_{prob2} = L_{[k^2*l^2]}$ can not be. □

2.2.2 Union

Theorem 2.2.2. *There exists L_{adv} such that $\mathcal{L}(L_{adv})$ is not closed under union.*

To prove the theorem we shall again present a counterexample. The main idea shall be that by uniting two problems we can obtain an easier problem. For example, let L be any regular language. By uniting it with its complement we obtain Σ^*, whose state complexity is 1.

Proof. Let $L_{adv} = L_{[3]}, L_{prob1} = L_{adv} \cap L_{[2]}$ and $L_{prob2} = L_{adv} \cap L^c_{[2]} = L_{adv} \cap (L_{[2]} \cdot a)$. We can see that both L_{prob1} and L_{prob2} can be simplified by L_{adv}, but the union, $L_{prob} = L_{prob1} \cup L_{prob2}$, is equal to the advice and it can not be simplified by itself. □

2.2.3 Concatenation and iteration

Using the intersection or union we could not obtain "new" words. As long as we made an intersection of two languages that could be helped by some advice (were subsets of that advice), we knew that the intersection was a subset as well. Concatenation and iteration are different. By concatenating two languages or iterating one we can obtain a "new" word and potentially create a language that will no longer be a subset of the advice, therefore the advice will no longer be helpful.

Theorem 2.2.3. *There exists* L_{adv} *such that* $\mathcal{L}(L_{adv})$ *is neither closed under concatenation nor under iteration.*

Proof. Let $L_{adv} = L_{[2]} \cdot a$ and $L_{prob} = L_{[3]} \cap L_{adv}$. L_{prob} can be simplified by L_{adv} . Clearly $L_{prob} \cdot L_{prob}$ and L^*_{prob} contain words of even length, e. g., a^6. Thus none of these languages is a subset of L_{adv} and therefore they can not be in $\mathcal{L}(L_{adv})$. □

Chapter 3

Informational power of supplementary information

This chapter is devoted to the concept of informational power. We shall present our approach to this intuitive (informal) concept and we shall explore questions such as "can some advice have greater informational power than another", or "can some advice be better than another?" Better in the sense that it has greater informational power and smaller or equal price (state complexity).

Our approach is centered around the number of problems that advice can simplify. The more the better. The problem that we have to face is that most languages providing supplementary information are useful to infinitely many problems. For example $L_{[2]}$ is useful to the problem $L_{[2*p]}$, where p can be any prime number besides 2. Since there are infinitely many primes, the cardinality of $\mathcal{L}(L_{[2]})$ is infinite. Because of that, we shall not focus on the informational power of a single language, but we shall compare two languages. We say that L_{adv1} has greater informational power than L_{adv2} when the family defined by L_{adv1} is a strict superset of the family defined by L_{adv2}. We say that L_{adv1} and L_{adv2} have the same informational power when the families defined by those languages are equal and we say L_{adv1} and L_{adv2} have incomparable informational power when their families are incomparable.

Example 3.0.1. The advice $L_{[2]}$ has greater informational power than the empty language. The empty language and a language consisting of only one word have equal informational power. The languages $L_{[2]}$ and $L_{[3]}$ are incomparable.

Note. Instead of greater informational power, we shall sometimes use phrases as stronger advice, greater power, or more useful. We shall use similar phrases for equal informational power and incomparable informational power.

In the example above, we have seen that there are examples for each of those three cases. In what follows we shall explore the possibilities for the information power

comparison of two languages L_{adv1} and L_{adv2} given they are in some relation. In Section 3.1 it is the case when $L_{adv1} \subset L_{adv2}$ and in Section 3.2 it is $(L_{adv1}, L_{adv2}) \in U$. Next we shall explore how the operations $\cup$ and $\cap$ on advice languages translate to relations between the corresponding families. In Section 3.3 we shall explore possible relations between $\mathcal{L}(L_{adv1} \cup L_{adv2})$ and $\mathcal{L}(L_{adv1}) \cup \mathcal{L}(L_{adv2})$. In Section 3.4 we shall address the same question for the intersection.

To present our results in a compact form allowing for an easy overview of the results achieved the theorems shall be presented in the form of a table. Each table shall consist of six columns (one for each possible relation between sets - $\supset, \supseteq, =, \subseteq, \subset, \not\supseteq \wedge \not\subseteq$) and two rows - universal case and existential case. Each theorem thus contains 12 assertions.

The following simplified example of a theorem is provided to illustrate the meaning of the "table form" statement of theorems.

Example 3.0.1. *Theorem: Let L_{adv1} and L_{adv2} be in relation $\square$. Then the following holds for $\mathcal{L}(L_{adv1})$ and $\mathcal{L}(L_{adv2})$:*

	$\supset$	$=$
universal case	no (A)	
existential case		yes (B)

The two assertions shown in the table are to be read as follows:

1. *It is not true that $\mathcal{L}(L_{adv1}) \supset \mathcal{L}(L_{adv2})$ for arbitrary L_{adv1} and L_{adv2} such that $L_{adv1} \square L_{adv2}$. This assertion is proved in Case A part of the proof.*

2. *There exist L_{adv1} and L_{adv2} such that $L_{adv1} \square L_{adv2}$ and $\mathcal{L}(L_{adv1}) = \mathcal{L}(L_{adv2})$ holds. This assertion is proved in Case B part of the proof.*

3.1 Relation between $\mathcal{L}(L_{adv1})$ and $\mathcal{L}(L_{adv2})$ assuming $L_{adv1} \subset L_{adv2}$

Theorem 3.1.1. *Let L_{adv1} and L_{adv2} be such that $L_{adv1} \subset L_{adv2}$. Then following holds for $\mathcal{L}(L_{adv1})$ and $\mathcal{L}(L_{adv2})$:*

	$\supset$	$\supseteq$	$=$	$\subseteq$	$\subset$	$\not\supseteq \wedge \not\subseteq$
universal case	no (A)	no (A)	no (A)	no (A)	no (A)	no (B)
existential case	yes (B)	yes	yes (D)	yes	yes (C)	yes (A)

Proof.

Case (A). Let $L_{adv1} = L_{[2]}$, $L_{adv2} = L_{[4]}$. Using divisibility lemma 2.1.3, we can find a language that can be simplified using L_{adv1} and can not be simplified using L_{adv2}. For example $L_{[12]}$. On the other hand $L_{[6]}$ can be simplified by L_{adv1}, but is not a subset of L_{adv2} and therefore L_{adv2} can not help and we have shown, that L_{adv1} and L_{adv2} are languages with incomparable informational power.

Case (B). Let $L_{adv1} \subseteq \Sigma^*$ be any language useful to at least one problem and $L_{adv2} = \Sigma^*$. Since Σ^* is not useful to any problem, L_{adv1} can not be Σ^* and therefore it is a proper subset of L_{adv2} (it satisfies the assumption). The advice L_{adv1} is helpful to at least one problem while L_{adv2} is not helpful to any problem. Therefore L_{adv1} has greater informational power than L_{adv2}.

Case (C). This case shall be similar to the previous one, but instead of Σ^*, we shall use the other extreme the empty language.

Let L_{adv2} be any language useful to at least one problem and $L_{adv1} = \emptyset$. The advice L_{adv2} can not be the empty language, since it is useful to at least one problem and therefore the subset assumption is satisfied, as well as the fact, that L_{adv2} is stronger than L_{adv1}.

Case B and C together show, that for any advice that is useful to at least one problem there exist subset and superset languages that are weaker.

Case (D). In this case, we shall use both the empty language and Σ^*.

Let Σ be any alphabet, L_{adv1} be the empty language and $L_{adv2} = \Sigma^*$. The subset assumption is satisfied and as we have already seen, neither the empty language nor Σ^* are useful to any problem. Thus their informational power is the same.

□

3.2 Relation between $\mathcal{L}(L_{adv1})$ and $\mathcal{L}(L_{adv2})$ assuming $(L_{adv1}, L_{adv2}) \in U$

Based on the fact usefulness is anti-reflexive ($(L_{adv2}, L_{adv2}) \notin U$) and using our assumption ($(L_{adv1}, L_{adv2}) \in U$) we know that $\mathcal{L}(L_{adv1}) \not\subseteq \mathcal{L}(L_{adv2})$. If usefulness was transitive than L_{adv1} would be always stronger than L_{adv2}. But since we have already shown that usefulness is not transitive, there are more possible relations between these two families.

Theorem 3.2.1. *Let L_{adv1} and L_{adv2} be such that $(L_{adv1}, L_{adv2}) \in U_n$. Then following holds for $\mathcal{L}(L_{adv1})$ and $\mathcal{L}(L_{adv2})$:*

	$\supset$	$\supseteq$	$=$	$\subseteq$	$\subset$	$\not\supseteq \wedge \not\subseteq$
universal case	no (A)	no (A)	no (A)	no (A)	no (A)	no (B)
existential case	yes (B)	yes	no	no	no	yes (A)

Proof.

Case (A). Let $L_{adv1} = L_{[2]}$ and $L_{adv2} = L_{[6]}$. Divisibility lemma 2.1.3 shows that L_{adv1} and L_{adv2} satisfy usefulness assumption. As mentioned earlier L_{adv2} is a problem that can be simplified using L_{adv1} but can not be simplified using L_{adv2} . On the other hand, $L_{[12]}$ can be simplified using L_{adv2} but can not be simplified using L_{adv1} and therefore L_{adv1} and L_{adv2} have an incomparable informational power.

Case (B). The advice has to be a proper superset, so language consisting of 0 words (empty language) automatically can not be useful to any problem. Since empty language can not be simplified (its state complexity is already minimum) languages consisting of 1 word can not be useful to any problem as well. However, as we have seen in Subsection 2.1.4, a finite language can be useful. That means there exists a boundary k, such that languages with less than k words are not useful to any problem and there is a useful advice with exactly k words. Let L_{adv1} be any useful advice with k words and L_{adv2} be any problem that L_{adv1} can simplify. Since L_{adv2} is a proper subset of L_{adv1} , L_{adv2} has less than k words and it is not useful to any problem. Therefore the advice L_{adv1} has greater informational power than L_{adv2} .

In this case, we have answered the question from the beginning of the chapter, whether there exists a better advice. Advice L_{adv1} is strictly better than L_{adv2} . It has greater informational power, as well as it has a lower cost (L_{adv1} is useful to L_{adv2} therefore it has smaller state complexity).

□

3.3 Relation between $\mathcal{L}(L_{adv1} \cup L_{adv2})$ and $\mathcal{L}(L_{adv1}) \cup \mathcal{L}(L_{adv2})$

This and the following section shall differ from the previous ones. We shall explore the union and intersection of languages providing supplementary information compared to the union and intersection of the families defined by those languages. To simplify the notation, we shall use L_{adv} for $L_{adv1} \cup L_{adv2}$ and in the next Section for $L_{adv1} \cap L_{adv2}$.

Theorem 3.3.1. *Let L_{adv1} and L_{adv2} be regular languages. The following holds for $\mathcal{L}(L_{adv1} \cup L_{adv2})$ and $\mathcal{L}(L_{adv1}) \cup \mathcal{L}(L_{adv2})$:*

	$\supset$	$\supseteq$	$=$	$\subseteq$	$\subset$	$\not\supseteq \wedge \not\subseteq$
universal case	no (A)	no (A)	no (A)	no (A)	no (A)	no (B)
existential case	yes (D)	yes	yes	yes (B)	yes (C)	yes (A)

Proof.

Case (A). Let $L_{adv1} = L_{[4]}$ and $L_{adv2} = L_{[4]} \cdot aa$. Concatenation of $L_{[4]}$ and the word aa means that L_{adv2} consists of all words with length, which has remainder 2 after the division by 4. It is the same generalization of the divisibility problem as we have seen in Subsection 2.1.3. We are using a similar approach as we have used in Subsection 2.2.2, where we have noticed that the result of the union of two problems can be a simpler problem. $L_{adv} = L_{adv1} \cup L_{adv2} = L_{[2]}$.

Using the divisibility lemma, we can prove that L_{adv} is useful to $L_{[6]}$. Since $L_{[6]}$ is not a subset of either L_{adv1} or L_{adv2}, they can not simplify it. The problem $L_{[6]}$ shows that $\mathcal{L}(L_{adv1}) \cup \mathcal{L}(L_{adv2})$ is not a superset of $\mathcal{L}(L_{adv})$.

Again by using the divisibility lemma, we can show that $L_{[12]}$ can be simplified using L_{adv1} but can not be simplified using L_{adv}. Therefore $L_{[12]}$ proves the other side of incomparability.

Case (B). In this case, we shall present a sufficient condition, not just an example. If L_{adv1} is a subset or a superset of L_{adv2} then L_{adv1} together with L_{adv2} are at least as strong as L_{adv}.

$L_{adv1} \subseteq L_{adv2} \implies (L_{adv1} \cup L_{adv2}) = L_{adv2} \implies L_{adv} = L_{adv2} \implies \mathcal{L}(L_{adv}) = \mathcal{L}(L_{adv1}) \implies \mathcal{L}(L_{adv}) \subseteq \mathcal{L}(L_{adv1}) \cup \mathcal{L}(L_{adv2})$. So for all cases where L_{adv1} is a subset of L_{adv2}, $\mathcal{L}(L_{adv})$ is a subset of $\mathcal{L}(L_{adv1}) \cup \mathcal{L}(L_{adv2})$. To prove that the superset is also sufficient, it is enough to note that L_{adv1} and L_{adv2} are in the same position and therefore we can swap them. If L_{adv1} is a superset of L_{adv2} than L_{adv2} is a subset of L_{adv1} and we can use the same proof.

An example could be $L_{adv1} = L_{[2]}$ and $L_{adv2} = L_{[4]}$ or $L_{adv1} = L_{adv2}$.

Case (C). We can strengthen the condition from the previous case to get a sufficient condition for this case as well. If L_{adv1} is a subset of L_{adv2} and L_{adv1} is not weaker than or equal to L_{adv2} then L_{adv1} together with L_{adv2} have greater informational power than L_{adv}.

From the previous case we know that $\mathcal{L}(L_{adv})$ is equal to $\mathcal{L}(L_{adv2})$. Considering the fact that L_{adv1} is not weaker than or equal to L_{adv2}, we know that there is a problem that L_{adv1} can simplify but L_{adv2} (and L_{adv}) can not and we get that L_{adv1} together with L_{adv2} have greater informational power than L_{adv}.

For example we could again use $L_{adv1} = L_{[2]}$ and $L_{adv2} = L_{[4]}$, but we can not use $L_{adv1} = L_{adv2}$ since L_{adv1} is equal to L_{adv2}.

Case (D). Let k be the same boundary as in Case B of the previous section (there does not exist useful advice with less than k words but there is useful advice with k words). Let L_{adv} be any useful advice with k words. We know that m must be at least 2 (empty language and languages with one word are not useful to any problem). Therefore we can split L_{adv} into two languages L_{adv1} and L_{adv2} such that $L_{adv} = L_{adv1} \cup L_{adv2}$ and both L_{adv1} and L_{adv2} are proper subsets of L_{adv} . Both L_{adv1} and L_{adv2} have less than k words and therefore they are not useful to any problem. On the other hand, L_{adv} is useful to at least one problem. Thus L_{adv} is stronger advice than L_{adv1} and L_{adv2} together.

□

3.4 Relation between $\mathcal{L}(L_{adv1} \cap L_{adv2})$ and $\mathcal{L}(L_{adv1}) \cap \mathcal{L}(L_{adv2})$

Theorem 3.4.1. *Let L_{adv1} and L_{adv2} be regular languages. The following holds for $\mathcal{L}(L_{adv1} \cap L_{adv2})$ and $\mathcal{L}(L_{adv1}) \cap \mathcal{L}(L_{adv2})$:*

	$\supset$	$\supseteq$	$=$	$\subseteq$	$\subset$	$\not\supseteq \wedge \not\subseteq$
universal case	no (A)	no (A)	no (A)	no (A)	no (A)	no (B)
existential case	yes (c)	yes (B)	yes	yes	yes (D)	yes (A)

Proof.

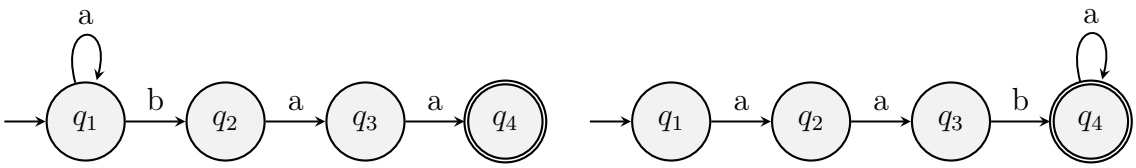
Case (A). Let $L_{adv1} = L_{[2]}$ and $L_{adv2} = L_{[3]}$. Similarly to the previous section, to simplify the notation we shall use L_{adv} for intersection of L_{adv1} and L_{adv2} , $L_{adv} = L_{adv1} \cap L_{adv2} = L_{[6]}$.

The usefulness is anti-reflexive, L_{adv} is not useful to itself. Using the divisibility lemma we can show that both L_{adv1} and L_{adv2} can simplify the problem $L_{[6]}$ and therefore $\mathcal{L}(L_{adv})$ is not a superset of $\mathcal{L}(L_{adv1}) \cap \mathcal{L}(L_{adv2})$.

Again using the divisibility lemma, we can prove that L_{adv} is useful to a problem $L_{[12]}$ but L_{adv1} is not and that proves the second condition of the incomparability.

Case (B). For this case we shall present a sufficient condition (similarly as in the previous section). If L_{adv1} is a subset of L_{adv2} than $\mathcal{L}(L_{adv})$ is a superset of or equal to $\mathcal{L}(L_{adv1}) \cap \mathcal{L}(L_{adv2})$.

$L_{adv1} \subseteq L_{adv2} \implies L_{adv1} \cap L_{adv2} = L_{adv1} \implies L_{adv} = L_{adv1} \implies \mathcal{L}(L_{adv}) = \mathcal{L}(L_{adv1}) \implies \mathcal{L}(L_{adv}) \supseteq \mathcal{L}(L_{adv1}) \cap \mathcal{L}(L_{adv2})$ and that is exactly what we needed to prove.



(a) Automaton for $\{a^k baa | k \in \mathbb{N}\}$ (b) Automaton for $\{aaba^k | k \in \mathbb{N}\}$

Figure 3.1: Automata for $\{a^k baa | k \in \mathbb{N}\}$ and $\{aaba^k | k \in \mathbb{N}\}$.

Case (C). Strengthening the condition from the previous case, by also requiring that L_{adv2} is not equal to or stronger than L_{adv1} we get a sufficient condition for this case.

From the previous case we know that $L_{adv1} \subseteq L_{adv2} \implies \mathcal{L}(L_{adv}) \supseteq \mathcal{L}(L_{adv1}) \cap \mathcal{L}(L_{adv2})$. Since L_{adv2} is not equal to or stronger than L_{adv1}, there is a problem, that can be simplified using L_{adv1} and can not be simplified using L_{adv2}. From the previous case we also know that this problem can be simplified by using L_{adv} and therefore $\mathcal{L}(L_{adv})$ is not equal to $\mathcal{L}(L_{adv1}) \cap \mathcal{L}(L_{adv2})$ but it is a strict superset.

Case (D). Let $L_{adv1} = \{a^k baa | k \in \mathbb{N}\}$ and $L_{adv2} = \{aaba^k | k \in \mathbb{N}\}$. An advice L_{adv} is equal to $\{aabaa\}$ and $\mathcal{L}(L_{adv})$ is an empty set, since L_{adv} is just a one word.

Both L_{adv1} and L_{adv2} can be accepted by NFAs with 4 states (their state complexity is at most 4) as is shown in Figure 3.1, while a NFA for L_{adv} needs at least 5 states (the state complexity of a language that consists of only one word is the length of that word plus one). This together with fact that $L_{adv} = L_{adv1} \cap L_{adv2}$ shows that L_{adv} can be simplified using both L_{adv1} and L_{adv2}, so $L_{adv} \in \mathcal{L}(L_{adv1}) \cap \mathcal{L}(L_{adv2})$, which in combination with the fact that $\mathcal{L}(L_{adv}) = \emptyset$ implies that $\mathcal{L}(L_{adv})$ is a subset of $\mathcal{L}(L_{adv1}) \cap \mathcal{L}(L_{adv2})$.

□

Conclusion

The thesis is a continuation of [1], [4], [3], and [5]. It has the highest resemblance to the thesis [5]. Both theses are focused on regular languages and use a similar definition of usefulness. While [5] works in the deterministic setting we examine nondeterminism. Many of the problems we have used have the same deterministic and nondeterministic complexity, for example, the divisibility problems. Therefore many of the theorems can be translated to the deterministic setting.

We have tried to make the theorems and proofs as independent of details of our definition of usefulness as possible. Therefore they should hold for slight modifications of the notion of usefulness that may be studied in the future. However, substantially different formalization may bring different results.

We can see several possibilities for future research, they are mainly concerned with possible modifications of the notion of usefulness. We expect they may eventually lead to a better understanding of the intuitive notion.

Our definition does not take into account how much does the advice simplify the problem. It only cares whether the advice can or can not help. In Chapter 3 we have discussed a possible problem of formalization that takes into account how useful the advice is. One possible way is to define a simplification as a function of advice and a problem and its result as a state complexity reduction of the problem using the advice. For example, simplification of a problem $L_{[6]}$ by advice $L_{[2]}$ would be equal to 3 and simplification of $L_{[6]}$ by advice $L_{[4]}$ would be equal to 0. Using the simplification, we can define the usefulness of advice as a sum of simplifications of all problems by the advice. Problem with this definition, as we have already mentioned in chapter 3 is that most of the time usefulness would be equal to infinity.

Another possible definition of usefulness, that would consider how much does the advice simplify the problem, is a definition of usefulness as a function. The function would have one parameter. The value of the function in n would be the sum of simplifications of problems with smaller state complexity than n. We could compare two languages providing supplementary information by comparing their functions. There might exist an upper bound for the functions depending on the state complexity of the advice. We could say that the advice is using its full potential if its function would be close to the boundary.

A different possible modification of the definition could allow the advice to be as hard as the problem. On the one hand, it would make the usefulness symmetric and potentially decrease the possibilities in the table in Section 3.4, on the other hand, it would not make much sense considering the view of usefulness as a decomposition of a problem into two simpler problems.

Even though after reading the thesis it might be natural for the usefulness to not be transitive, it was not at first sight. The first example of languages, that do not obey the rules of transitivity, were found by a brute force search done by a computer. A deep examination of the example presented the "hidden scheme" that is shown in 2.1. The concept of the different and the harder problem also helped us with closure properties (with an intersection). We knew that we could construct a counterexample by "playing with alphabets", but we were not sure whether there exists a "real" counterexample or we could slightly generalize our definition to make families defined by advice closed under intersection. These concepts helped us to find the second Schema 2.4. Even though these concepts were just on an intuitive level, they had a great impact and we tried to find fitting formalization, but we have not succeeded. We could not find formalization that would satisfy our assumptions. For example, if a problem L_{prob1} is different from L_{prob2}, we wanted it to be also different from harder versions of problem L_{prob2}. The formalization of different and harder problems is surely an interesting task, that might bring a lot of insight into the domain.

In Section 3.2 we have answered an interesting question, whether there exists strictly better advice. The example we have shown would still work even if we would incorporate how (not only whether) the advice is useful to the definition since the worse advice was not helpful to any problem.

Bibliography

[1] Peter Gaži and Branislav Rovan. Assisted problem solving and decompositions of finite automata. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 292–303. Springer, 2008.

[2] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

[3] Šimon Sádovský. Prídavná informácia a zložitosť nedeterministických konečných automatov. Master Thesis, Comenius University in Bratislava, 2017.

[4] Pavel Labath and Branislav Rovan. Simplifying dpda using supplementary information. In *International Conference on Language and Automata Theory and Applications*, pages 342–353. Springer, 2011.

[5] Jozef Martiš. Dodatočná informácia a triedy jazykov. Bachelor Thesis, Comenius University in Bratislava, 2018.

[6] Branislav Rovan and Šimon Sádovský. On usefulness of information: Framework and nfa case. In *Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday*, pages 85–99. Springer, 2018.