ACTA UNIVERSITATIS WRATISLAVIENSIS No 3397
Kształcenie Językowe 10 (20) Wrocław 2012

Agnieszka Kułacka
King's College London

Natural language versus regular language

1. Introduction

At the beginning of the last century linguists made first attempts to formalise the syntax of a natural language. In 1930s Kazimierz Ajdukiewicz created a sentence calculus called a categorical grammar (see Kułacka 2011a; 2011c). Constructing mathematical models of the syntax of a fragment of a natural language and thus creating artificial languages resembling natural languages became the object of a new discipline called mathematical linguistics, which at first was a subfield of mathematics. Once it became an independent field of research, two branches of mathematical linguistics were developed: synthetic and analytic. The former branch deals with questions regarding formal grammars, their types and the properties of the languages that are generated by them. Synthetic mathematical linguistics also conducts research on automata, which are used to recognise a language generated by a grammar. The analytic branch of mathematical linguistics uses variety of mathematical fields, such as set theory, logic, graph theory, etc., to construct models of the syntax of a fragment of a natural language. It is also likely that the achievements of synthetic branch are used in the research of its analytic counterpart.

One of the formal grammars that were initially used to describe a fragment of the syntax of a natural language was a grammar constructed for regular languages, i.e. the ones recognisable by a finite state automaton. However, Noam Chomsky made attempts (1956, 1957, 1967) to show that English is not a regular language, which was followed by another attempt made by Barbara Partee, Alice TerMeulen and Robert Wall (1990). In this paper I will discuss the important notions and a theorem, which are necessary to understand the line of argumentation as presented in the proof. I will also present the aforementioned proof so it will become more approachable and mathematically sound. Finally, I will demonstrate that a similar proof can be performed for Polish.

2. Automata Theory

Automata Theory is the study of abstract computing devices called automata. It was born in the 1930s with the work of Alan Turing on the capabilities of computing machines. Since that time the research within Automata Theory has been used to model brain function, to design digital circuits, in the development of formal grammars, etc. In this section I will introduce the key notions of the theory, which will be important for understanding the rest of the paper, and describe regular expressions, which are a declarative way of expressing the strings that are recognised by the automaton they are associated with.

2.1. Central notions of Automata Theory

The first important term, an alphabet, Σ, is a finite and non-empty set of symbols. Let us look at some alphabets. There are some common alphabets such as: (a) a binary alphabet, where $\Sigma_1 = \{0,1\}$, (b) an alphabet of capital letters, $\Sigma_2 = \{A, B, ..., Z\}$, etc. We can also define an alphabet that will consist of some words: $\Sigma_3 = \{Maria, Jan, Smiles, Likes\}$.

The next essential notion in Automata Theory is a string, which is also called a word. A string is a finite sequence of symbols chosen from an alphabet. Above I introduced three alphabets. From the binary alphabet, Σ_1, we can construct e.g. *101*, *111*, *10001*, but we cannot construct *102*, because 2 is not a symbol from this alphabet. The following strings *ALPHA*, *OMEGA* are constructed from the elements of the alphabet of capital letters. The last alphabet of symbolic words can give rise to the following strings: *LikesMaria*, *MariaJanSmiles*, etc. At the moment I am not concerned with the grammaticality of the two latter strings, but only whether the symbols constituting a string belong to a given alphabet. There is one distinctive string that can be chosen from symbols of any alphabet. It is called the empty string, denoted by ε, and it consists of zero occurrences of the symbols of an alphabet.

We may split the strings constructed from symbols of alphabets into groups, in terms of their length. The set of all strings of length *n* over the alphabet Σ, denoted by Σ^n, is the *n*th power of the alphabet Σ. Regardless of what symbols constitute an alphabet, $\Sigma^0 = \{\varepsilon\}$. We will now consider some powers of the alphabet.

(a) As mentioned above, $\Sigma^0 = \{\varepsilon\}$.

(b) $\Sigma^1 = \{Maria, Jan, Smiles, Likes\}$. Let us appreciate the difference between Σ and Σ. The former is the set of all strings of length 1 over the alphabet, while the latter is the alphabet.

(c) The second power of Σ is:

$\Sigma^2 = \{$*MariaMaria*;*MariaJan*;*MariaSmiles*;*MariaLikes*;*JanMaria*;*JanJan*;*JanSmiles*; *JanLikes*;

SmilesMaria; *SmilesJan*; *SmilesSmiles*; *SmilesLikes*; *LikesMaria*; *LikesJan*; *LikesSmiles*; *LikesLikes*$\}$.

We can carry on with enumerating the subsequent powers of Σ. In general, the union of all these power sets is denoted by Σ*. If we exclude the empty string, then we will obtain the set of all non-empty strings, Σ*. The relationship between the two sets is so given by $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. We may also concatenate two strings *x* and *y*, which means that we form a new string, *xy*, out of a copy of *x* followed by the copy of *y*. The concatenation of *y* and *x* is different; it is *yx*. The concatenation of the empty string and the string *y* from either side gives the string *y*.

A language, L, is a subset of strings that belong to Σ*, where Σ is an alphabet. We say that L is a language over Σ. One needs to notice that we may not use all the symbols of the alphabet in the strings that belong to a given language and also, that even though an alphabet is finite, the language may be infinite. Let us consider the following examples.

The three most important languages are the languages that are over any alphabet Σ: (1) Σ*, (2) ∅, the empty language, and (3) {ε}, the language that only consists of the empty string, ε. Let us draw a reader's attention to the fact that languages (2) and (3) are not the same. The former has no strings and the latter has one string, the empty string.

Any of the following sets are languages over a chosen alphabet Σ, where Σ = {*Maria, Jan, Smiles, Likes*}

(a) L_1 = {*MariaSmiles; JanSmiles; SmilesJan; LikesSmiles; Jan; ε*};

(b) L_2 = {*JanLikesMaria; JanJan; JanJanJan; Maria*};

(c) L_3 = {*ε; Jan*; *JanJan; JanJanJan; JanJanJanJan; ...*}.

The last of the example languages over Σ is infinite. We can define the elements of the language inductively: (1) $\varepsilon \in L_3$, (2) each of the successive strings of length *n* is formed by a concatenation of the string of length *n-1* and symbol *Jan*, (3) no other strings belong to L_3.

2.2. Regular expressions

Since it has been proven that a language is regular if and only if some regular expression describes it (see Hopcroft, Motwani & Ullman 2001: 90ff, Sipser 1997: 66ff), I will omit defining an automaton as it has no influence on understanding the proof, and I will present a definition of a regular expression instead. In the next section, I will show how one can derive some formal grammar from a regular expression.

A regular expression, *R*, can be constructed inductively. The basis is (1) a for $a \in \Sigma$, where Σ is some alphabet, and ε, ∅ are all regular expressions. The inductive step is as follows: (2) if R_1 and R_2 are regular expressions, then $R_1 + R_2$, R_1R_2 and R_1* are regular expressions. The expressions defined in point (2) describe the following languages: $L(R_1 + R_2) = L(R_1) \cup L(R_2)$, $L(R_1R_2) = L(R_1)L(R_2)$ (concatenation of the strings from R_1 and R_2 in this order), and $L(R_1^*) = (L(R_1))^*$.

Let us consider some examples. We assume that $\Sigma_1 = \{0,1\}$. The symbol *w* represents a string in the language.

(a) $L(\mathbf{01^*}) = \{w|w$ *starts with* 0 *followed by zero or more* 1 $s\}$.
(b) $L(\mathbf{0 + 1}) = \{0,1\}$.
(c) $L(\mathbf{0 + 1^*}) = \{w|w$ *is* 0 *or a sequence of zero or more* 1 $s\}$.
(d) $L((\mathbf{01})^*) = \{w|w\ w|w$ *is a zero or more sequences of* 01$s\}$.

Let us notice that these languages are also the languages being generated by grammars of Type 3, the lowest on Chomsky's Hierarchy, which means that they can be also generated by grammars of higher types (Type 2, Type 1 and Type 0), but the reverse is not necessarily true. In this paper I will only define grammars of Type 3, which generate regular languages.

3. Formal grammar for regular languages

I will start with defining a formal grammar in general and then specify the definition for the ones generating regular languages (cf. Kułacka 2011b). A formal grammar is a quadruple $G = (\Sigma, V, S, P)$, where Σ is a finite set of terminals (symbols used in a string of a language), V is a finite set of variables (auxiliary symbols), $S \in V$ is the start symbol, P is a finite set of production rules, the heads the body of which consists of a sequence of terminals and variables.

If x,y,z are strings of variables and terminals, A is a variable, then $A \rightarrow x$ is a production rule, and we say that yAz yields yxz, and we write $yAz \Rightarrow yxz$. Let x_1, x_2, ..., x_k, $k \geq 0$ be strings of variables and terminals. Then we write $y \overset{*}{\Rightarrow} z$ if $y = z$ (zero steps) or $y \Rightarrow x_1 \Rightarrow x_2 \Rightarrow ... \Rightarrow x_k \Rightarrow z$ (k+1 steps).

The language generated by the formal grammar $G = (\Sigma, V, S, P)$ is $\{w \in \Sigma^* | S \overset{*}{\Rightarrow} w$, where w is a string, Σ is the set of terminals, S is the start variable.

Let us consider some examples of formal grammars and the strings that can be generated. Let $G_1 = (\Sigma, V, S, P)$ be a formal grammar, where $\Sigma = \{$*Maria, Jan, Smiles, Likes*$\}$, $V = S, V, P$), and P is the set consisting of the following rules (the stroke | stands for 'or'):

$S \rightarrow$ *Maria VP* | *Jan VP*;
VP $\rightarrow$ *IV* | *TV Maria* | *TV Jan*;
IV $\rightarrow$ *Smiles*;
TV $\rightarrow$ *Likes*.

We can generate the string: *MariaLikesJan*, which has the following derivation:
$S \Rightarrow$ *Maria VP* $\Rightarrow$ *Maria TV Jan* $\Rightarrow$ *Maria Likes Jan*.

This derivation shows that *Maria Likes Jan* belongs to the language generated by the grammar G_1. Now we will construct grammars for the examples of regular expressions given in the previous section. In the descriptions of a language, the symbol w means a string.

Let $G_2 = (\Sigma, V, S, P)$ be a formal grammar, where $\Sigma = \{0,1\}$, $V = \{S, A\}$ and P is the set consisting of the following rules: $S \rightarrow 0A$, $A \rightarrow \varepsilon|1\ A$.The language generated by G_2 is $\{w|w$ *starts with* 0 *followed by zero or more* 1 $s\}$.

Let $G_3 = (\Sigma, V, S, P)$ be a formal grammar, where $\Sigma = \{0,1\}$, $V = \{S\}$ and P is the set consisting of the following rule: $S \rightarrow 0|1$. The language generated by G_3 is $\{0,1\}$.

Let $G_4 = (\Sigma, V, S, P)$ be a formal grammar, where $\Sigma = \{0,1\}$, $V = \{S, A\}$ and P is the set consisting of the following rules: $S \rightarrow \varepsilon|0|1A$, $A \rightarrow \varepsilon|1\ A$. The language generated by G_4 is $\{w|w$ *is* 0 *or a sequence of zero or more* 1 $s\}$.

Let $G_5 = (\Sigma, V, S, P)$ be a formal grammar, where $\Sigma = \{0,1\}$, $V = \{S, A, B\}$ and P is the set consisting of the following rules: $S \rightarrow \varepsilon|0|A$, $A \rightarrow 1B$, $B \rightarrow \varepsilon|0A$. The language generated by G_5 is $\{w|w$ *is a zero or more sequences of* 01$s\}$.

As we can see all the production rules in the grammars G_1, G_2, G_3, G_4, G_5 are of one of the two forms: $A \rightarrow yB$ or $A \rightarrow x$, where A and B are variables and x is either a terminal or the empty string, ε, y is a terminal. These are the only forms of production rules which occur in grammars of Type 3 that generate regular languages (Partee, TerMeulen & Wall 1990: 451). It is worth noticing that I extended the definition of a formal grammar of Type 3 given by Partee, TerMeulen and Wall by a production rule due to which the empty string, ε, is generated. If it were not included, the languages that are generated by these grammars and described by the equivalent regular expressions would not be the same. As mentioned above, grammars of higher types can also generate regular languages, e.g. $G_6 = (\Sigma, V, S, P)$, where $\Sigma = \{0,1\}$, $V = \{S\}$ and P is the set consisting of the following rule: $S \rightarrow \varepsilon|01S$, generate the same language as G_5, i.e. $\{w|w$ *is a zero or more sequences of* 01$s\}$, where w is a string. This production rule is not of a required form for a production rule of a formal grammar of Type 3.

I will sketch a possible conversion from a regular expression to a formal grammar of Type 3. The alphabet of a regular expression and the set of terminals for a grammar are equal. The basic step is as follows (capital letters represent variables, lower case letters — terminals):

Production rule	Regular expression that describes the same language as generated by the production rule
$A \rightarrow a$	a
$A \rightarrow \varepsilon|aA$	a^*
$A \rightarrow a|b$	$a + b$
$A \rightarrow ab$	ab

For the inductive step, we assume that the regular expressions R_1 and R_2 are associated with a set of production rules of the form on the right, characteristic for grammars of Type 3. The rule for $R_1 + R_2$ is $B \rightarrow R_1|R_2$, where in the body of $B \rightarrow R_1|R_2$ we copy the bodies of R_1 and R_2. For the regular expressions R_1^* and R_1R_2 we will possibly need to introduce more variables to keep the required form, as I did in the construction of G_5. The equivalent skeletons of the rules for these two regular expressions are $B \rightarrow \varepsilon|R_1B$ and $B \rightarrow R_1R_2$.

To prove that a language is not regular, a theorem called a pumping lemma is being applied. This is the theorem that was used in the ideas of Partee, Ter Meulen & Wall's proof that English is not a regular language. It is worth establishing that showing that some languages are not regular does not prove that all natural languages are not regular, but it only means that if one wants to have a general grammar that can be applied to describing all languages, a formal grammar of Type 3 is not the one.

4. Pumping lemma for regular languages

In this section I will only present the theorem without proving it (for proofs one can check Hopcroft, Motwani & Ullman 2001: 126ff; Sipser 1997: 78ff), but providing some of its application in proofs for non-regular languages.

Theorem: Let L be a regular language. Then there exists a natural number n such that for each string w in L such that its length (in terms of the number of terminals) is greater or equal to n (we write $|w| \geq n$), we can split w into three strings x, y, z, so $w = xyz$ such that

(a) $y \neq \varepsilon$;

(b) $|xy| \leq n$;

(c) For all $k \geq 0$, the string xy^kz is also in L.

I need to add that the theorem is applicable only to infinite languages. A regular language described by a regular expression **0 + 1** is {0,1}. If we try to apply this theorem, the number n can only be 1. Both strings are of the length that is greater or equal to 1. If we break either of them into x, y, z, so $w = y$, z, where x and z are the empty strings and y is 0 or 1,then the conditions (a) and (b) are met; $y \neq \varepsilon$ and $|xy| \leq 1$, as it is the length of y. However, the condition (c) is not satisfied for $k \geq 2$, as $y^2 \notin$ L.

Let us now consider an infinite regular language described by a regular expression **01*** .There is such a number $n = 2$, that any string w meeting the condition $|w| \geq 2$ can be broken into x, y, z, so $w = y$, z, where $x = 0$, $y = 1$ and $z = 1^p$, $p \geq 0$, stands for the rest of the string w. The conditions (a) and (b) are met; $y \neq \varepsilon$ and $|xy| \leq 2$. The condition (c) is also satisfied as for all $k \geq 0$, the string 01^k1^p is in L.

I will show the procedure for proving that a language is non-regular and each step of this procedure will be illustrated by an example. Let us prove that $L_1 = \{0^l1^l | l \geq 1\}$ is not regular.

A. We choose a language to be proven non-regular.

AA. We have already chosen a language: $L_1 = \{0^l1^l | l \geq 1\}$.

B. We choose an arbitrary natural number, n.

C. We need to pick a string in L_1, w, such that $|w| \geq n$.

CC. Let the string be $w = 0^n1^n$, $|w| = 2n$.

D. We need to divide w into x, y, z so that the constraints (a) and (b) of pumping lemma are satisfied.

DD. For the condition (b) to be satisfied, the strings x and y contain only 0s. Let $x = 0^{p_1}$, $y = 0^{p_2}$, $z = 0^{p_3}1^n$; $p_2 \geq 1$, so the condition (a) is met, $p_1 + p_2 \leq n$ so the condition (b) is met, and $p_1 + p_2 + p_3 = n$, where $p_3 \geq 0$.

E. We need to pick k, such that xy^kz is not in L.

EE. Let $k = 2$. Then $w_1 = 0^{p_1}0^{2p_2}0^{p_3}1^n$ is not in L_1 since $p_1 + 2p_2 + p_3 = n + p_2 \neq n$ as $p_2 \geq 1$.

There are also some closure properties of regular languages, i.e. operations that applied to regular languages also return a regular language. The proofs are beyond the scope of this paper, but one can check Hopcroft, Motwani & Ullman (2001: 131ff), Sipser (1997: 58ff) for more details concerning them. We will only need the facts that a homomorphism (substitution of strings of one language for symbols of another) of a regular language is a regular language, and that the intersection of two regular languages is regular.

After this presentation of all the necessary knowledge needed to understand the ideas of Chomsky's and Partee, TerMeulen & Wall's proofs, we are now able to appreciate them in a novel version, which I arrived at. For the original proofs one may refer to Noam Chomsky (1956, 1957, 1967) and Barbara Partee, Alice TerMeulen and Robert Wall (1990).

5. Proof

Let us construct a formal grammar for a fragment of English, $G_7 = (\Sigma, V, S, P)$, where $\Sigma = \{if, grass, is, green, then\}$, $V = \{S, A\}$ and P is the set of two production rules: $S \rightarrow if\ S\ then\ A\ |if\ A\ then\ A,\ A \rightarrow grass\ is\ green\}$. Let us notice that for the sake of clarity, I included the space in the set of terminals. We can generate the following strings, which are grammatical sentences of English, but possibly with no interpretation:

(a) *if grass is green then grass is green*;

(b) *if if grass is green then grass is green then grass is green*;

(c) *if if if grass is green then grass is green then grass is green then grass is green*; etc.

Let us define the homomorphism h:{*if*, *grass is green*, *then grass is green*}$\rightarrow$ $\{a, b, c\}$ in the following way:

h(*if*) = a, h(*grass is green*) = b, h(*then grass is green*) = c.

From the previous section we know that the property of regularity is preserved under homomorphism. Therefore, if the language generated by G_7 is regular and the strings after the application of the homomorphism h are of the form a^nbc^n, $n \geq 1$, then the language $\{a^nbc^n | n \geq 1\}$ is also regular. By contraposition, if the language $\{a^nbc^n | n \geq 1\}$ is not regular, neither is $L(G_7)$. Using the procedure

discussed in the previous section, I will prove that the language $\{a^n bc^n | n \geq 1\}$ is not regular. Then one can apply modus ponens, a very common rule of inference, and prove that $L(G_7)$ is not regular.

A. We choose the language: $L_2 = \{a^n bc^n | n \geq 1\}$.

B. We choose an arbitrary natural number, *n*.

C. Let the string be $= a^n bc^n$, $|w| = 2n + 1$.

D. For the condition (b) to be satisfied, the strings *x* and *y* contain only *a*s. Let $x = a^{p_1}$, $y = a^{p_2}$, $z = a^{p_3} bc^n$; $p_2 \geq 1$; so the condition (a) is met, $p_1 + p_2 \leq n$, so the condition (b) is met, and $p_1 + p_2 + p_3 = n$, where $p_3 \geq 0$.

E. Let $k = 2$. Then $w_1 = a^{p_1} a^{2p_2} a^{p_3} bc^n$ is not in L_2 since $p_1 + 2p_2 + p_3 = n + p_2 \neq n$ as $p_2 \geq 1$.

I will show that the following language $\{w | w = if^*$ *grass is green* (*then grass is green*)$^*\}$ is regular. We are not concerned with whether the strings in this language belong to English or any other natural language. The words belonging to this language can be described by a regular expression: *if** *grass is green* (*then grass is green*)* We can also construct a formal grammar of Type 3 that will generate it: $G_8 = (\Sigma, V, S, P)$, where $\Sigma = \{if, grass, is, green, then\}$, $V = \{S, A, B, C\}$, and P is the set of two production rules: $\{S \rightarrow if\ S | grass\ A, A \rightarrow is\ B, B \rightarrow green\ C, C \rightarrow \varepsilon | then\ A\}$.

The last part of the proof is performed with the use of regular language closure property, namely that the intersection of two regular languages is regular. Let us intersect English with $\{if^*$ *grass is green* (*then grass is green*)$^*\}$. As a result, we will get if^n *grass is green* (*then grass is green*)$^n | n \geq 1\}$. Since I proved the latter to be non-regular and $\{if^*$ *grass is green* (*then grass is green*)$^*\}$ is regular, English is non-regular as by contraposition, if the intersection of two languages is not regular, then at least one of them is non-regular. The language $\{if^*$ *grass is green* (*then grass is green*)$^*\}$ is regular, so English is non-regular.

In this proof I used one of Chomsky's ideas of the constructions that may lead to showing that a natural language cannot be generated by a formal grammar of Type 3. One needs to create more powerful tools and possibly that may not be enough due to the variety of properties of a language (see Kułacka 2011c). The other constructions that can serve the same purpose of showing that English is non-regular are "Either S or S", "The man who said that S is arriving today", where *S* stands for a sentence, or sentences involving parenthetical embedding such as the English sentence (the rat(the cat(the dog chased)killed)ate the malt) (Chomsky 1967: 286).

6. Polish

In Polish there are also similar sentence schemata as the ones described in the previous section. In the following schemata *S* stands for embedded sentence. *Jeżeli S, to S*, which translates into *if S then S*. *Albo S albo S* meaning *either S or*

S. Mężczyzna, który powiedział, że S, przyjeżdza dzisiaj is the Polish version of *the man who said that S is arriving today*. Therefore, one can deliver the same proof as for English to show that Polish is not a regular language. The parenthetical embedding construction is not present in Polish. Another sentence schema that can also be the basis of similar proofs is *neither S nor S* and its Polish equivalent *ani S ani S*.

7. Further research

One has to be aware of two facts: (1) proving that some natural languages are not regular is not enough to show that all languages are non-regular, as it is assumed in the literature (see Chomsky 1967: 286; Gazdar & Mellish 1989: 135); (2) it has been proven that hearers process a natural language as if it were a regular language (Gazdar & Mellish 1989: 135). The latter is most likely due to the limitation of human short memory, which cannot store too much information (see Kułacka 2009). These limitations can be avoided to some extent in the case of computers and the possible languages generated by implemented grammars.

It will be interesting to look at other languages possibly not from the Indo-European family to see whether similar conclusions about languages can be drawn. Another line of research will be to establish the characteristics of such constructions in languages, which show that a given language is non-regular, to be able to find them in natural languages.

There is also some confusion between what language a man can produce and comprehend, and a theoretically possible natural language with an abstract language user. Clearly, the former is a regular language as it is finite, while the latter may be infinite and as such possibly non-regular. In this paper I considered a language theoretically possible. Another question is whether we generate linguistic expressions as implementing a mental formal grammar or whether we reproduce clusters of words, while only occasionally producing a novel expression for which generating we use this mental grammar.

References

Chomsky, N. 1956. "Three models for the description of language." *IRE Translations on Information Theory* 2 (3): 113–124.

Chomsky, N. 1957. *Syntactic Structures*. Reprinted in 2002. Berlin: Mouton de Gruyter.

Chomsky, N. 1967. "Introduction to the Formal Analysis of Natural Languages." *Handbook of Mathematical Psychology.* Vol. II. Ed. Luce, R.D., Bush, R.R., Galanter, E. New York: John Wiley and Sons, Inc.

Gazdar, G., Mellish, Ch. 1989. *Natural Languages Processing in PROLOG.* Reading: Addison-Wesley Publishing Company.

Hopcroft, J.E., Motwani, R., Ullman, J.D. 2001. *Introduction to Automata Theory, Languages, and Computation.* Boston: Addison Wesley.

Kułacka, A. 2009. "The Necessity of the Menzerath-Altmann Law." *Anglica Wratislaviensia XLVII*: 55–60.
Kulacka, A. 2011a. "Intensional Logic for a Montague Grammar." *LingVaria* VI, no. 2 (12).
Kułacka, A. 2011b. "Metodologiczne założenia semantyki komputerowej." *Metodologie językoznawstwa*. Łódź: Wydawnictwo Uniwersytetu Łódzkiego.
Kułacka, A. 2011c. "Syntax of a Montague Grammar." *LingVaria* VI, no. 1 (11): 9–23.
Partee, B., Ter Meulen, A., Wall, R.E. 1990. *Mathematical Methods in Linguistics*. London: Kluwer Academic Publishers.
Sipser, M. 1997. *Introduction to the Theory of Computation.* Boston: PWS Publishing Company.