UNRESTRICTED GRAMMARS

Isis Burmeister Pericolo

For one thing, studying language is by itself a part of a study of human intelligence that is, perhaps, the central aspect of human nature.
Noam Chomsky

1 INTRODUCTION

This study intends to discuss unrestricted grammars -- as a computational model, its uses and how it behaves in regards of equivalence to Turing machines. Unrestricted grammars are mainly studied in the field of formal language theory, being the most general class of grammars in the Chomsky hierarchy and allowing greater freedom in its productions than its fellow models.

It being the subject at hand, we become particularly interested in its characteristics as a computational model and what we might be able to do or not using it. As such, studying whether languages created by unrestricted grammars can be described or recognized by machines -- and, specifically here, by a Turing machine --, become important to determine its computational power.

2 HISTORY

Unrestricted grammars were first discussed by Noam Chomsky on two articles written on the decade of 1950. ‘Three Models for the Description of Language’ was published in 1956 by the Department of Modern Languages and Research Laboratory of Electronics, and in it Chomsky starts his investigation into how to formally describe the English language using computational models. Although primarily interested on the study of natural languages, he ended up shifting his focus later on to a more formal approach.

> We find that no finite-state Markov process that produces symbols with transition from state to state can serve as an English grammar. [...] We formalize the notion of “phrase structure” and show that this gives us a method for describing language which is essentially more powerful, though still representable as a rather elementary type of finite-state process. [...] We study the formal properties of a set of grammatical transformations that carry

sentences with phrase structure into new sentences with derived phrase structure, showing that the transformational grammars are processes of the same elementary type of phrase-structure grammars; [...] This paper is concerned with the formal structure of the set of grammatical sentences. We shall limit ourselves to English, and shall assume intuitive knowledge of English sentences and nonsentences.
(Noam Chomsky, 1956)

On his first article tackling the subject, Chomsky uses more natural examples to show how we can construct a derivation tree for an English sentence by using syntactics. However, he does end up with the problem of ambiguity, since a natural language is much more prone to have tonal peculiarities and often will present differences in meaning that, while clear for a native or fluent speaker, are more complicated to explain in a formal setting and might be more of exceptions than a rule - making it not very efficient to provide insight into the use and understanding of that language.

Eventually, with ‘On Certain Formal properties of Grammars’ (1959), Chomsky seems to discover a new relation of structured grammars and how they can be described by other types of models that aren’t as restrictive as finite automata. In this paper, he talks about grammars as a “device that enumerates the sentences of a language” and moves on to discuss a “sequence of restrictions that limit grammars first to Turing machines” and then to other models, until coming back to his previous one, the finite-state Markov sources.

It is then that he formalizes unrestricted grammars for the very first time, too.

3 FORMALIZATION

An unrestricted grammar is a formal grammar $G = (N, \Sigma, P, S)$, where N is a set of nonterminal symbols and Σ is a set of terminal symbols; P is a set of production rules of the kind $\alpha \rightarrow \beta$, where both sides are strings of symbols from $N \cup \Sigma$ and the only restriction is that α cannot be the empty word ε; $S \varepsilon N$ is the starting symbol from which all derivations and productions that result in accepting states start.

Aside from this formal definition, unrestricted grammars are often described as “grammars that can generate recursively enumerable languages” - this means that for every language $L(G)$ an unrestricted grammar G generates, we can use recursion to find and count each accepted word it produces, even though it might be a taxing and slow process, possibly taking an infinite amount of time to perform. It also means that for every G, we can build a Turing machine capable of recognizing $L(G)$, but that will be discussed later on in more detail.

Some authors, including Chomsky himself, may use other names for unrestricted grammar, such as *type-0 grammars*, *phrase structure grammars*, or even *semi-Thue*. The last one of which refers to a similar model, a string rewriting system (SRS), except that this one doesn’t make any distinction between its sets of nonterminal and terminal symbols (not an obligation for unrestricted grammars) and does accept the left-hand side of a production to be the empty string.

4 PROGRAM EXAMPLES

One of the most well-known examples of unrestricted grammars is the triple balancing; a grammar $G = (V,\ \Sigma,\ N,\ S)$ generates the language $\{a^n b^n c^n \mid n \geq 1\}$, where $V = \{S,\ a,\ b,\ c,\ A,\ B,\ C,\ T_a,\ T_b, T_c\}$,

$\Sigma = \{a,\ b,\ c\}$ and

$P\ =\ \{S \rightarrow ABCS, S \rightarrow T_c,$

$CA\ \rightarrow\ AC,\ BA\ \rightarrow\ AB,\ CB\ \rightarrow\ BC,$

$CT_c\ \rightarrow\ T_c c,\ CT_c\ \rightarrow\ T_b c,$

$BT_b\ \rightarrow\ T_b b,\ BT_b\ \rightarrow\ T_a b,$

$AT_a\ \rightarrow\ T_a a,$

$T_a\ \rightarrow \varepsilon\}$

The first two production will generate a string of nonterminals that will look more or less of the form $(ABC)^n T_c$. The three next rules will rearrange the variables A, B and C so that it becomes something like this $A^n B^n C^n T_c$, which will allow the subsequent productions to transform the variables C into the terminals c, the variables B into the terminals b and the variables A into the terminals a, all the while carrying the symbol T_c further to the left, until it reaches the beginning of the string and can be turned into the empty word ε, making it disappear from the end result.

Another example can be the concatenation of two words, making the language $\{ww \mid w \; \varepsilon \; (a,b)*\}$. Having $\Sigma = (a,\ b)$ and a set of production rules

$P = \ \{S \to X\#,\ \ X \to aXA \mid bXb \mid \#,$

$A\# \to A'\#,\ AA' \to A'A,\ BA' \to A'B,\ \#A' \to a\#,$

$B\# \to B'\#,\ AB' \to B'A,\ BB' \to B'B,\ \#B' \to b\#,$

$\#\# \to \varepsilon\}$

A derivation from this grammar into a word from the language would use the nonterminal X to write the first instance of w while leaving variables to the right. It finishes of the first w and puts a symbol $\#$ on its end. Then, the production rules enable the grammar to rearrange the remaining variables so as we go back from the end of the string back to the previous $\#$, we can turn every A and B into their respective terminal symbols a and b. All that's left is a pair of $\#\#$ at the end of the string, which the last rule can make disappear.

5 TURING-COMPLETENESS

We discussed on section 3 about how recursively enumerable languages are the ones that can be recognized by a Turing machine. That infers, by concept, that unrestricted grammars - the ones that generate REL - and Turing machines - the ones that recognize REL -, probably have some kind of relation. In fact, Chomsky, in 1959, not more than thirty years after Alan Turing published his article on the universal machine that takes his name, already discussed that.

We can, with the use of rather simple, generic algorithms, show that they are Turing-equivalent, which, putting it simply, will mean that they will accept the exact same language.

5.1 FROM UNRESTRICTED GRAMMARS TO TURING MACHINES

For this transformation we will have a generic grammar $G = (V,\ \Sigma,\ N,\ S)$ and a Turing machine $M\ =\ (Q,\ T,\ b,\ \Sigma,\ \delta,\ q0,\ F)$ that is non deterministic and has two tapes. Note that, for simplicity reasons, we will skip proving that non deterministic multi-tape Turing machines can simulate and be converted to deterministic TMs with one tape.

The first tape will contain the word w, our input, while the second one will test strings that can be generated from S until they find a match to word w - or not, since for any generic REL we can only guarantee the building of a semidecidable TM, meaning that it may never halt if it doesn't reach an acceptance state. We start, on the second tape, to nondeterministically choose any production rule from our original grammar and try to find matches for its left-hand side string on the tape. When it does, it replaces the string with its right-hand side production (shifting whatever else there is one the tape to make space for it if necessary) and goes back to nondeterministically choose another rule. The last step for any process of this TM is to compare whatever string is on the second tape to the input word w, reaching an acceptance state if so.

By doing that, we can make sure that every single word from language L generated by grammar $G(L)$ can be derived on this second tape, thus ensuring that if w pertains to L, it will be accepted by the Turing machine.

5.2 FROM TURING MACHINES TO UNRESTRICTED GRAMMARS

For the vice-versa of this equivalence, we construct an unrestricted grammar G such that its production rules can turn an acceptance string from a Turing machine M on the form $b\&w_1q_aw_2b$, where b is the blank symbol, $\&$ is a marker symbol for the beginning of the tape, w_1 and w_2 are derived words and q_a is an acceptance state in that TM, into a word $w \,\varepsilon\, L(G)$. For that, we need to have production rules of three different kinds - we will call them initiation, execution and cleaning rules.

On the initiation rules, we have to ensure that that from our variable S we can derive all the blanks necessary on the TM tape and also mark the beginning of it with the marker symbol. Those rules will look something like $S \to bS \mid Sb \mid \&A$ where A is from which the words w_1 and w_2 will be generated.

The next step is to create all the execution rules. These derivations of G wil simulate backwards computations of M, having one rule for each configuration of M that moves the head of the tape, writes, erases, or scans the tape. In example, for each computation of M like $\delta(q, a) = (p, b)$ we will have a rule in G of the form $bp \rightarrow aq$.

By that point, when all of those rules are created, we will reach a state that looks like $b\&wb$, with all those leftover blanks and marker symbols. We move on to the cleaning rules, which will recognize and turn those into the empty word ε, finishing our unrestricted grammar.

The conclusion we can get from those proofs are that unrestricted grammars are Turing-equivalent and this, alone, is not enough to prove Turing-completeness. However, the key detail is that they do the exact same computations, only in reverse order, and they can simulate each other's control system by doing that, which will prove the model's Turing-completeness.

6 REFERENCES

Turing, A.M. (1936). "On Computable Numbers, with an Application to the Entscheidungsproblem". *Proceedings of the London Mathematical Society.* (published 1937).

Chomsky, N. (1956). “Three Models for the Description of Language”. *Department of Modern Language and Research Laboratory of Electronics*. (published 1956).

Chomsky, N. (1959). “On Certain Formal Properties of Grammars”. *Department of Modern Language and Research Laboratory of Electronics*. (published 1959).

PAPADIMITRIOU, Christos H.; LEWIS, Harry R.. **Elements of the theory of computation** 2 ed. Upper Saddle River, New Jersey: Prentice-Hall, 1998. 375 p.

HOPCROFT, John E.; ULLMAN, Jeffrey D.; MOTWANI, Rajeev. **Introdução à teoria de autômatos, linguagens e computação** 2 ed. São Paulo, SP: Elsevier Editora Ltda, 2003. 560 p.

MENEZES, Paulo Blauth. **Linguagens formais e autômatos** 6 ed. Porto Alegre, RS: ARTMED Editora S.A, 2010. 256 p.

DIVERIO, Tiarajú Asmuz; MENEZES, Paulo Blauth. **Teoria da computação:** Máquinas universais e Computabilidade. 3 ed. Porto Alegre, RS: ARTMED Editora S.A, 2011. 288 p.