Reflection in the Chomsky Hierarchy

Henk Barendregt[*] Venanzio Capretta[†] Dexter Kozen[‡]

Abstract

The class of regular languages can be generated from the regular expressions. These regular expressions, however, do not themselves form a regular language, as can be seen using the pumping lemma. On the other hand, the class of enumerable languages can be enumerated by a universal language that is one of its elements. We say that the enumerable languages are *reflexive*.

In this paper we investigate what other classes of the Chomsky Hierarchy are reflexive in this sense. To make this precise we require that the decoding function is itself specified by a member of the same class. Could it be that the regular languages are reflexive, by using a different collection of codes? It turns out that this is impossible: the collection of regular languages is not reflexive. Similarly the collections of the context-free, context-sensitive, and computable languages are not reflexive. Therefore the class of enumerable languages is the only reflexive one in the Chomsky Hierarchy.

In honor of Roel de Vrijer on the occasion of his sixtieth birthday

1 Introduction

Let $\mathcal{T} \subseteq \mathcal{P}(\Sigma^*)$ be a class of languages over an alphabet Σ. Let (C, U) be a couple of languages such that the words of U are pairs $c \bullet w$ with $c \in C$ and w arbitrary. Define, for $c \in C$,

$$L_U^c = \{w \in \Sigma^* \mid c \bullet w \in U\}.$$

The pair (C, U) is called a *universal coding system* for $\mathcal{T}$ if

$$\mathcal{T} = \{L_U^c \mid c \in C\}.$$

$\mathcal{T}$ is called *reflexive* if there exists a universal coding system (C, U) for $\mathcal{T}$, such that $C, U \in \mathcal{T}$.

[*] Institute of Computing and Information Science, Radboud University, Nijmegen, The Netherlands

[†] School of Computer Science, University of Nottingham, UK

[‡] Computer Science Department, Cornell University, Ithaca, New York 14853, USA

We assume that Σ is a fixed alphabet and consider only languages over it, that is, subsets of Σ^*. We assume that Σ has "enough" symbols, in the sense that, given some candidate language of codes C, we freely presume that we can find a symbol $\bullet$ not contained in any word of C.

It turns out that the classes of regular, context-free and context-sensitive languages are all non-reflexive. The proofs of these facts are different for each of the three classes. The class of recursive languages (not officially in the Chomsky hierarchy) is also non-reflexive. It is known that recursively enumerable languages are reflexive, as follows from the existence of a universal Turing machine.

This leaves open the possibility that other classes, smaller than that of recursively enumerable languages, are reflexive. In fact we will show in Proposition 2 that for every effectively countable family of languages, we can construct such a class containing its union. The construction is simple but artificial. The quest is for such a reflexive class that has a uniformly defined family of accepting machines.

1.1 Related Work

A crucial difference between reflexivity and other notions of universality in the literature is that the code of a language is included as part of its representation in the universal language. This imposes a strong uniformity requirement on the coding scheme that is absent with other notions. Greibach [2] constructs a "hardest context-free language" L_0 such that every context-free language is an inverse homomorphic image of L_0 or $L_0 - \{\varepsilon\}$. The language L_0 is constructed from a Dyck language (parenthesis language) on two pairs of matching parentheses. The homomorphism, which is the coding mechanism that witnesses the embedding of a context-free language in the universal language L_0, is not part of L_0, thus imposes no uniformity condition.

A similar result is the Chomsky–Schützenberger theorem [1], which states that every context-free language is the homomorphic image of the intersection of a Dyck language and a regular set. Again, the coding mechanism is not part of the universal language.

Kasai [3] constructs a "universal context-free grammar" for every finite alphabet Σ, a context-free grammar G such that for every context-free language $L \subseteq \Sigma^*$, there is a regular *control set* C that restricts the possible sequences of leftmost derivations of G, such that $L_C(G) = L$. Again, the control set C is independent of $L(G)$. It should be pointed out that the cited results are all positive universality results, whereas reflexivity in our sense does not hold for context-free languages.

2 Standard Encodings

First of all, let us look at the standard encodings for the language classes of the Chomsky hierarchy and ask if they belong to the class that they define.

The standard encoding of regular languages over an alphabet Σ is the language of regular expressions, defined as the set inductively generated by the rules:

- The symbols $\emptyset, \epsilon$ and $\mathbf{a}$ for every $a \in \Sigma$ are regular expressions;
- If $\mathbf{u}$ and $\mathbf{v}$ are regular expressions, then $(\mathbf{u} \cup \mathbf{v})$, $(\mathbf{uv})$ and $(\mathbf{u}^*)$ are regular expressions.

It is a well-known fact that languages that make use of parentheses are not regular, so the language of regular expressions does not belong to the class that it defines. Instead, the following grammar shows that it is context-free:

$$\begin{aligned} \langle \mathsf{reg-exp} \rangle &\to \emptyset \\ &\to \epsilon \\ &\to \mathbf{a} \quad a \in \Sigma \\ &\to (\langle \mathsf{reg-exp} \rangle \cup \langle \mathsf{reg-exp} \rangle) \\ &\to (\langle \mathsf{reg-exp} \rangle \langle \mathsf{reg-exp} \rangle) \\ &\to (\langle \mathsf{reg-exp} \rangle^*) \end{aligned}$$

The standard encoding of context-free languages is by context-free grammars. Such a grammar is a finite sequence of rules, separated by semicolons in our version. Each rule has the form $x \to w$, where x is a variable (non-terminal) from a set V and w is any word over the alphabet $V \cup \Sigma$. Contrary to the previous case, this language is not only context-free, but even regular. In fact, it is defined by the following regular expression:

$$V \to (V \cup \Sigma)^*(; V \to (V \cup \Sigma)^*)^*.$$

A similar consideration can be made for the context-sensitive languages.

So the conclusion is that the language of regular expressions is context-free while the language of context-free grammars is regular! This is for what concerns the traditional representations of languages. But if we start thinking about other possible encodings, the question we are asking becomes muddled. Is there some set of words, other than regular expressions, that encodes regular languages and is itself regular? Since we know that regular languages are effectively countable, we can certainly enumerate them using natural numbers, through some form of Gödel numbering. Therefore the numerals can be used as codes for regular languages, and they themselves form a regular language. The regular expression defining them is simply $\mathbf{S}^*\mathbf{O}$, where $\mathbf{S}$ and $\mathbf{O}$ are symbols denoting successor and zero, respectively.

This, however, is an unnatural answer to the question whether regular languages have an internal representation. The decoding function is going to be quite complex and certainly won't be implementable by a finite automaton, which is the sort of machine associated to a regular language.

3 Reflexivity

A coding system consists of a language C of codes and some decoding mechanism. This can be given in two equivalent ways: either by a universal language U or by a decoding function $d : C \to \mathcal{T}$. We define the notion of reflexivity for a language class $\mathcal{T}$ by requiring that there be a coding language C and a universal class U both within $\mathcal{T}$.

Definition 1 *Let $\mathcal{T}$ be a class of languages over an alphabet Σ, i.e., $\mathcal{T} \subseteq \mathcal{P}(\Sigma^*)$.*

1. *A* coding system *is a pair (C, U) with $C, U \subseteq \Sigma^*$. We call C the* code *language and U the* decoding *language.*
2. *Given a coding system (C, U) and any $c \in C$, define*

   $$L_U^{\mathsf{c}} = \{w \in \Sigma^* \mid c \bullet w \in U\}.$$

   where $\bullet$ is a new symbol not used in C. We call the mapping $\lambda c.L_U^{\mathsf{c}}$ the decoding *function of the system.*
3. *(C, U) is a $\mathcal{T}$-*coding system *iff $C \in \mathcal{T}$ and $U \in \mathcal{T}$.*
4. *(C, U) is* universal *for $\mathcal{T}$ iff $\mathcal{T} = \{L_U^{\mathsf{c}} \mid c \in C\}$.*
5. *$\mathcal{T}$ is* reflexive *if there exists a $\mathcal{T}$-coding system that is universal for $\mathcal{T}$.*

Equivalently, we could have taken the decoding function as a primitive in the definition of a coding system, defining it as a pair (C, d), where C is a language and $d : C \to \mathcal{P}(\Sigma^*)$. The two definitions are related by the transformations:

$$U = \{c \bullet w \mid c \in C \text{ and } w \in d(c)\} \quad \text{and} \quad d(c) = L_U^{\mathsf{c}}.$$

Any system (C, U) codes the class of languages $\mathcal{T}_{(C,U)} = \{L_U^{\mathsf{c}} \mid c \in C\}$. (C, U) is trivially universal for $\mathcal{T}_{(C,U)}$, but it is not necessarily a $\mathcal{T}_{(C,U)}$-coding system, because C and U may not be in $\mathcal{T}_{(C,U)}$. We will call (C, U) itself *reflexive* when this happens, that is, when there are codes $\mathsf{c}, \mathsf{u} \in C$ such that $C = L_U^{\mathsf{c}}$ and $U = L_U^{\mathsf{u}}$. We can always extend a system to a reflexive one minimally.

Proposition 2 *For every coding system (C, U), there exists another coding system (C', U') such that*

$$\{L_{U'}^{\mathsf{c}'} \mid c' \in C'\} = \{L_U^{\mathsf{c}} \mid c \in C'\} \cup \{C', U'\}.$$

Proof. We can safely assume that there are two words $\mathsf{c}, \mathsf{u} \in \Sigma^*$ not containing the separator $\bullet$ and not in C. We are going to use them as the new codes for C' and U', respectively. Define

$$C' = C \cup \{\mathsf{c}, \mathsf{u}\}, \quad U'' = U \cup \{\mathsf{c} \bullet c \mid c \in C'\}, \quad U' = (\mathsf{u}\bullet)^* U''.$$

Then (C', U') is a reflexive coding system extending (C, U). In fact, let $c \in C$, then $L_U^{\mathsf{c}} = L_{U'}^{\mathsf{c}}$; $C' = L_{U'}^{\mathsf{c}}$ and $U' = L_{U'}^{\mathsf{u}}$. □

4 Regular Languages

We show that regular languages are not reflexive according to the previous definition. The proof exploits a well-known characterization of regular languages.

Proposition 3 (Myhill-Nerode) *Given a language L over Σ, define the following equivalence relation $\equiv_L$ on Σ^*:*

$$v \equiv_L w \iff \forall u \in \Sigma^* [vu \in L \Leftrightarrow wu \in L].$$

Then L is regular iff $\Sigma^* / \equiv_L$ is finite.

The standard proof of the Myhill-Nerode result exploits the fact that L is accepted by a finite automaton.

Theorem 4 *The class of regular languages is not reflexive.*

Proof. Assume that regular languages are reflexive through the coding system (C, U). Then U is regular. Hence $\Sigma^* / \equiv_U$ is finite. As

$$c\bullet \equiv_U c'\bullet \iff L^c_U = L^{c'}_U,$$

this implies that there are only a finite number of regular languages, quod non. □

5 Context-Free Languages

Context-free languages (CFLs) can be characterized as those sets of words that are accepted by a push-down automaton, that is, a finite state machine that uses a stack as memory. If there were a universal language U for this class, the automaton corresponding to U would have to use the stack to store both the code c and and the information needed for the computation. But then, how can it "read" the code without destroying the information about a specific word? This informal argument makes it plausible that context-free languages are not reflexive.

A more precise proof follows. This exploits the fact that every CFL has a grammar in *Chomsky normal form*, where the production rules are of the form $A \to BC$ or $A \to a$, where A, B, C are nonterminal symbols and a is a terminal symbol.

Theorem 5 *The class of context-free languages is not reflexive.*

Proof. We use the sets $L_m = \{a^n b^{mn} \mid n \geq 0\}$. Intuitively, the languages L_m are context-free, but not uniformly so; to accept a language with large m requires a large state set or stack alphabet in a pushdown automaton or a large number of nonterminal symbols in a context-free grammar.

Suppose, towards a contradiction, that there were a universal CFL-coding system (C, U). Then with each CFL $L \subseteq \Sigma^*$, there would be associated at least one code $c \in C$ such that for all $w \in \Sigma^*$, $w \in L$ iff $c \bullet w \in U$. U would have a Chomsky normal form grammar, say with start symbol S and d nonterminal symbols. We write $A \xrightarrow{*} x$ if the nonterminal A can derive the string x. Note that the null string ε is not derivable from any nonterminal.

Consider a parse tree for $S \xrightarrow{*} x$. If σ is a path in the parse tree with nodes labelled by nonterminals $A_0, \ldots, A_k$, then for $0 \leq i \leq k-1$, the rule applied to A_i is of the form either (i) $A_i \to BA_{i+1}$ or (ii) $A_i \to A_{i+1}B$ for some B. In case (i), let v_i be the nonnull string generated by B and let $x_i = \varepsilon$. In case (ii), let $v_i = \varepsilon$ and let x_i be the nonnull string generated by B. In either case, $A_i \xrightarrow{*} v_i A_{i+1} x_i$, thus $A_0 \xrightarrow{*} v_\sigma A_k x_\sigma$, where

$$v_\sigma = v_0 v_1 \cdots v_{k-1}, \qquad\qquad x_\sigma = x_{k-1} \cdots x_1 x_0.$$

A *basic pumpable segment* in a parse tree is a path σ whose endpoints are labelled with the same nonterminal but otherwise all other nonterminals along the path are distinct. Basic pumpable segments arise in the proof of the pumping lemma. The length of a basic pumpable segment is at most d. If $A \xrightarrow{*} x$ and the parse tree contains no basic pumpable segment, then $|x| \leq 2^{d-1}$, since the tree is binary branching and of height at most d.

Now let c_m be a code for L_m and choose integers m and n such that

$$m > d2^{d-1}, \qquad\qquad n > |c_m| + 1. \tag{1}$$

Consider a parse tree for $S \xrightarrow{*} c_m \bullet a^n b^{mn}$. Let τ be the path in the parse tree from the start symbol S down to the rightmost a in the string a^n. Then

$$v_\tau = c_m \bullet a^{n-1}, \qquad\qquad x_\tau = b^{mn}.$$

For each $A_i \xrightarrow{*} v_i A_{i+1} x_i$ along the path τ, we must have $|x_i| \leq 2^{d-1}$, otherwise there would be a pumpable segment in the parse tree for x_i and we could generate a string with too many b's. In order to generate all of b^{mn}, we must have at least $mn/2^{d-1}$ such x_i, therefore if k is the length of τ, then

$$k \geq mn/2^{d-1} > d(|c_m| + 1),$$

by the two inequalities (1). This implies that it is possible to find at least $|c_m| + 1$ disjoint basic pumpable segments along the path τ: starting from the bottom and walking upward toward the root, we have a basic pumpable segment as soon as a nonterminal is repeated, then we start again from that nonterminal.

Now for each of these $|c_m|+1$ pumpable segments σ, consider the strings v_σ and x_σ. Each v_σ either contains the separator $\bullet$, or it is completely contained in a^n, or it is completely contained in c_m.

If any v_σ contains $\bullet$, then by pumping σ we can generate a string with the same code c_m but too many $\bullet$'s, which is impossible since L_m doesn't use $\bullet$.

If any v_σ is entirely contained in the a^n, then by pumping σ we can generate a string that violates $a^n b^{mn}$. If v_σ is null, then x_σ is nonnull and we can increase the number of bs without changing the number of as. On the other hand, if v_σ

is nonnull, by pumping σ once we can increase the number of as by at least one. There should be a corresponding increase of at least m bs, but this is impossible since $|x_\sigma| \leq d2^{d-1} < m$.

Finally, if all v_σ are substrings of c_m, then at least one of them must be null, since there are at least $|c_m|+1$ disjoint basic pumpable segments but only $|c_m|$ letters in c_m. By pumping that σ, we can generate a string with the same code c_m but too many b's.

In all cases, we have been able to pump and generate a string that should not be accepted by U. □

6 Context-Sensitive Languages

The class of context-sensitive languages consists of those recognized by a linear-bounded automaton, that is, a Turing machine with linear space complexity. An informal argument suggests that this class cannot be reflexive: for each language there is a linear bound, but certainly there is no universal linear bound for the whole class, which would be necessary if it were reflexive.

The following proof that the class of context-sensitive languages is not reflexive uses a diagonalization argument similar to Russell's paradox.

Definition 6 (i) *Given a language L over Σ, define the* root *of L by*

$$\sqrt{L} = \{w \in \Sigma^* \mid w \bullet w \in L\}.$$

(ii) *A class of languages $\mathcal{T}$ is called* Russellian *if $\mathcal{T}$ is closed under taking complements and roots.*

Theorem 7 *If $\mathcal{T}$ is Russellian, then $\mathcal{T}$ is not reflexive.*

Proof. Suppose towards a contradiction that $\mathcal{T}$ is Russellian and reflexive. Let (C, U) be a $\mathcal{T}$-coding system universal for $\mathcal{T}$. Then $U \in \mathcal{T}$. By closure under complement and root it follows that

$$\sqrt{\overline{U}} = \{w \in \Sigma^* \mid w \bullet w \notin U\} \in \mathcal{T}.$$

Hence $\sqrt{\overline{U}} = L_U^r$ for some $r \in C$. Then we obtain a contradiction:

$$\begin{array}{rcll} r \bullet r \in U & \Leftrightarrow & r \in L_U^r & \text{as } L_U^r = \{w \mid r \bullet w \in U\} \\ & \Leftrightarrow & r \in \sqrt{\overline{U}} & \text{as } L_U^r = \sqrt{\overline{U}} \\ & \Leftrightarrow & r \bullet r \in \overline{U} & \text{by definition of } \sqrt{} \\ & \Leftrightarrow & r \bullet r \notin U. & \end{array}$$

□

Theorem 8 *The class of context-sensitive languages is not reflexive.*

Proof. We show that the class of context-sensitive languages is Russellian. It is well known that the family of context-sensitive languages is closed under complement.

As to closure under root, let L be a context-sensitive language. Then there exists a Turing machine M with a linear space bound $ax + b$ on the size x of the input that accepts L. Let us define a new machine M' that operates in the following way: given the input w, it first replaces it with $w \bullet w$ and then runs M. Clearly M' is also linearly bounded, with the bound $a(2x+1)+b = 2ax+a+b$. Hence $\sqrt{L}$ is context-sensitive. □

Remark 9 *The class of computable languages is clearly closed under complements and roots, hence Russellian. Theorem 7 shows (the well-known fact) that also this class is not reflexive.*

7 Turing Recursively Enumerable Languages

The following result is essentially due to Turing.

Theorem 10 *The class of recursively enumerable languages is reflexive.*

Proof. Turing proved the existence of a universal machine M_u that can simulate every Turing machine T using a *program*: there exists a code t such that, for every input w, the result of the computation of M_u on $t \bullet w$ is the same as that of T on w.

The set of codes t is itself recursively enumerable (and can in fact be made context-free). This, together with the characterization of recursively enumerable languages as those accepted by a Turing machine, gives the result. □

Postscript

We leave it to the linguists to discuss whether the fact that the class of context-sensitive languages is not reflexive has implications for the place of say English in the Chomsky hierarchy.

Chomsky's nativist theory posits that the human mind has an innate module for language that is "tuned" to the specific native language in the first years of life. The class of "natural languages" consists of those that can be tuned in this way; nobody knows precisely where it lies, probably somewhere between context-free and context-sensitive. But this is a statement about the syntax of the language, not about its semantics.

In our approach we also talk about the semantics: the issue of reflection involves the meaning of the language. In our presentation the syntax is that of the language C and the semantics is given by U. If C is English, it can well be context-sensitive without U being context-sensitive. After all, we can describe Turing machines and r.e. languages in English.

References

[1] Noam Chomsky and Marcel-Paul Schützenberger. The algebraic theory of context free languages. In P. Braffort and D. Hirschberg, editors, *Computer Programming and Formal Systems*, pages 118–161. North-Holland, 1963.

[2] Sheila A. Greibach. The hardest context-free language. *SIAM J. Comput.*, 2(4):304–310, December 1973.

[3] Takumi Kasai. A universal context-free grammar. *Information and Control*, 28:30–34, 1975.