Reflection in the Chomsky hierarchy

Henk Barendregt[1]and Venanzio Capretta[2]

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

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

1 Introduction

A class $\mathcal{C}$ of formal languages is called *reflexive* if it contains a *universal language* U, describing all members of $\mathcal{C}$ in a well-defined way.

We investigate which classes of formal languages in the Chomsky hierarchy are reflexive. The Chomsky hierarchy consists of four classes: the regular ($\mathcal{R}$), context-free ($\mathcal{CF}$), context-sensitive ($\mathcal{CS}$) and Turing enumerable ($\mathcal{TE}$) languages. Each class is associated with a kind of computing machine that accepts exactly the languages in it: finite automata, push-down automata, linearly bounded Turing machines and unlimited Turing machines. This can be found in any standard text on formal languages and automata. Not officially in the Chomsky hierarchy is the class of Turing computable languages ($\mathcal{TC}$), accepted by always halting Turing machines. One has $\mathcal{R} \subseteq \mathcal{CF} \subseteq \mathcal{CS} \subseteq \mathcal{TC} \subseteq \mathcal{TE}$.

Speaking of natural languages, it is often said informally that, for example, English is rich enough to describe itself. This means that we can give an accurate description of the grammar of the English language in English. More generally, English can be used to describe the grammar of any natural language.

We want to make this intuition mathematically precise. Let $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$ be a class of languages over an alphabet Σ. Let (C, U) be a couple of languages such that the words of U are in the form of pairs $c \bullet w$ where $c \in C$, $\bullet$ is a distinguished symbol of Σ, and w can be an arbitrary word. We assume that C is a language over the alphabet $\Sigma - \{\bullet\}$, so there is no ambiguity in the demarcation between c and w. Defining, for $c \in C$,

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

we say that $c \in C$ *codes* L_U^c. The pair (C, U) is a *universal coding* for $\mathcal{C}$ if

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

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

It turns out that the classes of regular, context-free, context-sensitive, and computable languages are all non-reflexive. The proof of these facts are different for the first three classes, while for the class of computable languages the proof is the same as for that of the context-sensitive ones. It is known that Turing enumerable languages are reflexive, as follows from the existence of a universal Turing machine.

2 Standard encodings

First of all, let us look at the standard encodings for the language classes of the Chomsky hierarchy and ask whether 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 following 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 so are $(\mathbf{u} \cup \mathbf{v})$, $(\mathbf{uv})$ and $(\mathbf{u}^*)$.

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:

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

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 on 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 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 that of 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 $(\mathbf{S}^*)\mathbf{O}$. 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. Therefore, it is natural to impose restrictions on a coding. It should belong to the class that it is coding in a sense defined below.

3 Reflexivity

A coding 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 $d : C \to \mathcal{C}$. We define the notion of reflexivity for a language class $\mathcal{C}$ by requiring that there is a coding language C and a universal language U both inside $\mathcal{C}$.

The classes of languages that we consider are parameterized over the alphabet. To emphasize the alphabet, we write $\mathcal{C}_\Sigma$. For example, we may write $\mathcal{R}_{\{a,b\}}$ for the class of regular languages over the two-symbol alphabet $\{a, b\}$. In our definition we need a special *separator* symbol $\bullet$. We also need that the language C of codes does not use this symbol. This is no real restriction: In any alphabet with at least three symbols, we can always *reserve* one of the symbols for a special purpose and re-encode all the symbols as strings of the remaining ones. So if there is a language of codes that uses $\bullet$, we can always translate it into one that does not.

Definition 1 *Let Σ be an alphabet containing a distinguished symbol $\bullet$. Let $\mathcal{C}$ be a class of languages over Σ, that is $\mathcal{C} \subseteq \mathcal{P}(\Sigma^*)$.*

1. *A* coding *over Σ is a pair (C, U) with $C \subseteq (\Sigma - \{\bullet\})^*$ and $U \subseteq \Sigma^*$. We call C the* code *language and U the* decoding *language.*

2. *Given a coding (C, U) and any $c \in C$, define*

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

   The map $\lambda\!\!\lambda c.L_U^c$ is the decoding *function corresponding to the coding.*

3. *(C, U) is a $\mathcal{C}$-*coding *iff $C \in \mathcal{C}$ and $U \in \mathcal{C}$.*

4. *(C, U) is* universal *for $\mathcal{C}$ iff $\mathcal{C} = \{L_U^c \mid c \in C\}$.*

5. *$\mathcal{C}$ is* reflexive *if there exists a $\mathcal{C}$-coding that is universal for $\mathcal{C}$.*

Equivalently, we could have taken the decoding function as a primitive in the definition of a coding, 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

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

4 Regular languages

We show that regular languages are not reflexive according to the previous definition. The proof exploits the following characterization of regular languages.

Proposition 2 (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 3 *The class $\mathcal{R}$ of regular languages is not reflexive.*

Proof. Assume that regular languages are reflexive through the coding (C, U). Then U is regular. Hence $\Sigma^*/\equiv_U$ is finite. For every two codes $c, c' \in C$,

$$c\bullet \equiv_U c'\bullet \iff L_U^c = L_U^{c'}.$$

Then there would be only a finite number of regular languages, quod non. □

5 Context-free languages

Context-free languages can be characterized as those sets of words that are accepted by by push-down automata, that is, a finite state machine that uses a stack as memory. For 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 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. As a stepping stone, we show that it is impossible to generate all context-free languages from a finite collection of them, using only concatenation and union.

Lemma 4 *For any natural number $n > 0$, the context-free language*

$$B_n = \{a^k b^n a^k \mid k \in \mathbb{N}\}$$

cannot be decomposed as $B_n = L_1 L_2$ with L_1 and L_2 non trivial (that is, not the singleton empty word).

Proof. Suppose $B_n = L_1 L_2$. Since $b^n \in B_n$, we must have that either in L_1 or in L_2 there is at least one word made only of bs.

Suppose $b^k \in L_1$ for some $k > 0$. Then it is impossible for a word in L_2 to have a trailing a, since there are no words with the form $b^k wa$ in B_n. But then it must be that $L_1 = B_n$ and $L_2 = \{\epsilon\}$.

Similarly, if $b^k \in L_2$ for some $k > 0$, then $L_1 = \{\epsilon\}$ and $L_2 = B_n$. □

We need a stronger result, with a proof exploiting the same idea.

Lemma 5 *For any natural numbers $n > 0$ and m_1, m_2, define*

$$B_{n,m_1,m_2} = \{a^{k+m_1} b^n a^{k+m_2} \mid k \in \mathbb{N}\}.$$

Then B_{n,m_1,m_2} is context-free. A language L is called n-special if L is infinite and $L \subseteq B_{n,m_1,m_2}$ for some $m_1, m_2 \in \mathbb{N}$. Then we have

$$L = L_1 L_2 \textit{ is } n\textit{-special} \Rightarrow L_1 \textit{ or } L_2 \textit{ is } n\textit{-special.}$$

Proof. Assume $L = L_1 L_2$ is n-special, and let $w = a^{k+m_1} b^n a^{k+m_2} \in L$, as L is not empty. Then w can be decomposed as $w = w_1 w_2$ with $w_1 \in L_1$ and $w_2 \in L_2$. We proceed by cases on where the split occurs.

Case 1. The split occurs withing the b^n part, we come to the conclusion that L_1 and L_2 must both be singletons, contradicting the infinity of L. Indeed, if $w_1 = a^{k+m_1}b^{n_1}$ and $w_2 = b^{n_2}a^{k+m_2}$ with $n_1, n_2 > 0$ such that $n_1 + n_2 = n$, then the fact that $w_1 \in L_1$ forces L_2 to be the singleton $\{w_2\}$, since for any $w_2' \neq w_2$ one has $w_1 w_2' \notin B_{n,m_1,m_2}$. Similarly L_1 has to be a singleton. This contradicts that L_1L_2 is infinite and hence this case cannot occur.

Case 2. $w_1 = a^h$ and $w_2 = a^{h'}b^n a^{k+m_2}$ with $h + h' = k + m_1$. The fact that $w_2 \in L_2$ forces L_1 to be the singleton $\{a^h\}$. Then L_2 must be

$$L_2 = \{a^{k+m_1-h}b^n a^{k+m_2} \mid a^{k+m_1}b^n a^{k+m_2} \in L\} \subseteq B_{n,m_1',m_2'},$$

with $m_1' = m_1 - h$ and $m_2' = m_2$, which is infinite.

Case 3. $w_2 = a^h$ is treated similarly. □

Corollary 6 *The class $\mathcal{CF}$ of context-free languages cannot be generated by a finite number of them, using concatenation and finite union.*

Proof. For a class of languages $\mathcal{F}$, write $\mathcal{F}^+$ for the least class of languages containing all elements of $\mathcal{F}$ that is closed under union and concatenation.

Assume towards a contradiction that $\mathcal{F}$ is a finite set of context-free languages such that $\mathcal{F}^+$ is the class of all context-free languages. W.l.o.g. one has $\emptyset \notin \mathcal{F}$, as one has $(\mathcal{F} - \{\emptyset\})^+ = \mathcal{F}^+$, by taking the empty union. Therefore, since B_{n_1,m_1,m_2} and $B_{n_2,m_1',m_2'}$ are disjoint, for $n_1 \neq n_2$, there is an $n_0 > 0$ such that for all $L \in \mathcal{F}$ and all $m_1, m_2 \geq 0$ one has $L \not\subseteq B_{n_0,m_1,m_2}$. By 'induction on the generation' of $\mathcal{F}^+$ we show for this n_0 that

$$\forall L \in \mathcal{F}^+. L \text{ is } \textit{not } n_0\text{-special.} \tag{1}$$

For the base case $L \in \mathcal{F}$ this follows by the choice of n_0. Now assume $L = L_1 \cup L_2$ is n_0-special. Then by the pigeonhole principle, say, $L_1 \subseteq B_{n_0,m_1,m_2}$ is infinite, so n_0-special, contradicting the induction hypothesis. Finally assume $L = L_1L_2$ is n_0-special. Then by Lemma 5, say, L_2 is n_0-special, also contradicting the induction hypothesis. This establishes (1). Since $B_{n_0} = B_{n_0,0,0}$ is context-free, hence in $\mathcal{F}^+$, and moreover infinite, (1) yields a contradiction. □

Let us recall how a push-down automaton M works with an input from Σ^*. Such an M has a finite set of states S, with a unique start state q_s and a subset $F \subseteq S$ of accepting states. Also there is an alphabet Σ_M (possibly $\neq \Sigma$). At any stage in the computation M is in a 'configuration' of the form $[q, w, \alpha]$, where q is a state, $w \in \Sigma^*$ a word to be read, and $\alpha \in \Sigma_M^*$ is a stack content. A finite set of transitions of M is given, each of the form $(q, a, b) \to (q', b')$, qualifying

$$[q, aw, b\alpha] \to [q', w, b'\alpha]$$

as an admissible computation step for any w and α. The effect is that the machine reads and takes a from the input, reads and pops b from the stack and finally pushes b' onto the stack. Any of a, b and b' can be the *null* symbol, meaning that the computation step will neither read a symbol from the input, nor pop or push a symbol from or onto the stack, respectively.

A computation trace is of the form $c_1, \ldots, c_k$, where the c_is are configurations and $c_i \to c_{i+1}$ is an admissible computation step for all $1 \leq i < k$. If there exists such a trace we write $c_1 \vdash c_k$ (the automaton M is non-deterministic). Machine M *accepts* a word $w \in \Sigma^*$ if $[q_s, w, \epsilon] \vdash [q, \epsilon, \epsilon]$, for some accepting state $q \in F$.

Definition 7 *Given M a push-down automaton, define two binary relations $\vdash_i$ and $\vdash_s$ on its configurations.*

Write $[q, w, \alpha] \vdash_i [q', v, \beta]$ if the end configuration is the first with input v in a trace of M starting in $[q, w, \alpha]$. Since input can only be read, v is necessarily a postfix of w, that is, $w = w_0 v$ for some w_0. The relation $\vdash_i$ is used to specify the portion of the computation that consumes a given prefix w_0 of the input.

Write $[q, w, \alpha] \vdash_s [q', v, \beta]$ if the end configuration is the first with stack β in a trace of M starting in $[q, w, \alpha]$. Contrary to $\vdash_i$, in this case many symbols can be pushed onto and popped out of the stack before reaching β. However, we will use it only in the case that $\alpha = a\beta$ for some symbol a.

For every symbol $a \in \Sigma$ and states q and q', we define the language

$$L_{a,q,q'} = \{w \mid \forall v, \alpha.[q, wv, a\alpha] \vdash_s [q', v, \alpha]\}.$$

Theorem 8 *The class $\mathcal{CF}$ of context-free languages is not reflexive.*

Proof. Suppose towards a contradiction that the class of context-free languages is reflexive via (C, U). Then there exists a push-down automaton M that accepts the universal language U.

That is, a word of the form $c \bullet w$ belongs to U iff there exists a computation trace for M starting in configuration $[q_s, c \bullet w, \epsilon]$ and ending in configuration $[q, \epsilon, \epsilon]$ with q and accepting state. Let c' be the content of the stack after consuming the code from the input, that is: $[q_s, c \bullet w, \epsilon] \vdash_i [q_0, w, c']$. The state q_0 and stack content c' depend only on c and not on w. Consider c' as an internalization of the language code c.

Let $c' = a_0 \cdots a_n$. The trace of the processing of the input w can be divided in $n + 1$ steps, each terminating with the popping of one of the a_is. Hence w can be decomposed as $w = w_0 \cdots w_n$, such that the trace of the computation looks as follows:

$$\begin{aligned}[q_0, w_0 \cdots w_n, a_0 \cdots a_n] &\vdash_s [q_1, w_1 \cdots w_n, a_1 \cdots a_n] \\ &\;\vdots \\ &\vdash_s [q_n, w_n, a_n] \\ &\vdash_s [q_{n+1}, \epsilon, \epsilon].\end{aligned}$$

The first part of the computation tells us that $w_0 \in L_{a_0,q_0,q_1}$ and similarly for the other parts. Putting them all together we obtain that:

$$w \in L_{a_0,q_0,q_1} L_{a_1,q_1,q_2} \cdots L_{a_n,q_n,q_{n+1}}.$$

This must be true for every word in L^c_U, therefore we have:

$$L^c_U = \cup\{L_{a_0,q_0,q_1} L_{a_1,q_1,q_2} \cdots L_{a_n,q_n,q_{n+1}} \mid \begin{array}{l} q_1, \ldots, q_{n+1} \text{ states of } M, \\ q_{n+1} \text{ accepting}\}. \end{array}$$

Therefore every context-free language is obtained by concatenation and union from the finite set of languages in the form $L_{a,q,q'}$. This is impossible by Lemma 6. □

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. The following 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 9 (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 $\mathcal{C}$ is called* Russellian *if $\mathcal{C}$ is closed under taking complements and roots.*

Theorem 10 *If $\mathcal{C}$ is Russellian, then $\mathcal{C}$ is not reflexive.*

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

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

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

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

□

Theorem 11 *The class $\mathcal{CS}$ of context-sensitive languages is not reflexive.*

Proof. By Theorem 10 it suffices to show that $\mathcal{CS}$ is Russellian. It is well-known that $\mathcal{CS}$ 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. □

7 Turing computable and enumerable languages

Also the class of computable (recursive) languages is not reflexive.

Theorem 12 *The class $\mathcal{TC}$ of computable languages is not reflexive.*

Proof. $\mathcal{R}$ is closed under complements. It is also closed under roots: if L is computable, then $f : \sqrt{L} \to L$ with $f(w) = w \bullet w$ is a computable reduction of $\sqrt{L}$ to L, making $\sqrt{L}$ computable. It follows that $\mathcal{R}$ is Russellian and therefore not reflexive. □

The following result is essentially due to Turing.

Theorem 13 *The class $\mathcal{TE}$ of Turing enumerable languages is reflexive.*

Proof. Turing proved the existence of a universal machine M_u such that every Turing machine T can be simulated by it 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

Chomsky's nativist theory states 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 lays, probably somewhere between context-free and context-sensitive. 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 of a language is coded by C and its semantics is given by U. If C is English, it can well be that it is context-sensitive without U being context-sensitive. After all, we can describe Turing machines and recursively enumerable languages in English.

Our results leave open the possibility that some other yet unknown class, smaller than that of the Turing enumerable languages, is reflexive. Therefore English may belong to such a class.

We leave it to the linguists to discuss whether the fact that all classes that we discussed, except that of the Turing enumerable languages, are not reflexive has implications for the place of English in the Chomsky hierarchy.