POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Characterizing Non Counting Operator Precedence Languages in a Locally Testable manner

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Giorgio Corbetta

Advisor: Prof. Matteo Pradella

Academic year: 2022-2023

1. Introduction

The goal of this thesis is to expand the characterization of Operator Precedence Non-Counting Languages, by considering local testability.
Operator Precedence Languages, introduced by Robert W. Floyd in [1], are Structured Context-Free Languages. This family, because of having the local parsability property, may become more important in a world in which the growth of computing power is directed forward parallelization. In the literature, there have been proposed different characterizations for the class of Operator Precedence Languages (OPL). In particular, in [3], there can be found different characterizations inspired by the ones used for Regular Languages: Operator Precedence Grammars (OPG), Monadic Second Order Logic (MSO); also, Operator Precedence Expressions (OPE) can be found in [3]. Moreover, for each one of those characterizations, there has been found a way to restrict it to represent only Operator Precedence Non-Counting languages (NC_{OP}), and it turned out that those restrictions are similar to the ones on Regular Languages and relative representations.
Among all those characterizations for the OPL and NC_{OP} families, inspired by the ones on Regular Languages, there is not present any representation for NC_{OP} corresponding to the Locally Testable one. The family of Locally Testable Languages, as presented in [4], has been proven to coincide with the family of Non-Counting Regular Languages. For this reason, this work aims to provide a characterization, if exists, of Non-Counting Operator Precedence Languages inspired by the Locally Testable one.

2. Background

In this section are reported the main definitions and concepts present in the literature that are necessary for this work.
The first concept to be introduced is the concept of Non-Countingness. In particular, this concept suits well Regular Languages, however, it has been mutated also to fit other families of languages, like Context-Free or Operator Precedence Languages too. The family of Regular Non-Counting Languages is important to be presented because it constitutes one of the pillars on which the final result is built, and it is from its characterizations that the ones of NC_{OP} are inspired.

Definition 2.1 (Non Counting). *Given a language $\eta \subseteq \Sigma^*$, η is Non Counting if there exists a number n such that for all $x, y, z \in \Sigma^*$, $m \in \mathbb{N}$ it holds $x \cdot y^n \cdot z \in \eta \Leftrightarrow x \cdot y^{n+m} \cdot z \in \eta$.*

If η is not Non-Counting, then it is Counting.

2.1. Locally Testable

The class of Locally Testable Languages is a subclass of Regular Languages and provides a characterization based on sets of substrings, differently from grammars, regular expressions, automata, or logic. Those languages, described for example in [4], are defined starting from the concepts of *k-testable* and languages *Locally Testable in the strict sense*:

Definition 2.2 (*k-testable*). *Given an alphabet Σ and a character $\# \notin \Sigma$, a language $\eta \subseteq \Sigma^*$ is said to be k-testable if there exist a number k and a set $Y \subseteq ((\bigcup_{i \in \mathbb{N},\, i<k-1} \# \cdot \Sigma^i \cdot \#) \cup (\# \cdot \Sigma^{k-1} \cup \Sigma^{k-1} \cdot \#) \cup \Sigma^k)$*
A language $\eta \subseteq \Sigma^*$ is said to be k-testable if there exist a number k and a set $Y \subseteq ((\bigcup_{i \in \mathbb{N},\, i<k-1} \# \cdot \Sigma^i \cdot \#) \cup (\# \cdot \Sigma^{k-1} \cup \Sigma^{k-1} \cdot \#) \cup \Sigma^k)$ such that the language η recognises the strings $s \in \Sigma^*$ iff:

- *$|s| \leq k-2 \rightarrow \# \cdot s \cdot \# \in Y$;*
- *$|s| > k-2 \rightarrow$ each substring of length k of $\# \cdot s \cdot \#$ is in Y.*

If the language η is k-testable, then it is said to be Locally Testable in the strict sense, or LT.

Starting from those concepts, there can be defined the family of Locally Testable Languages using the closure under boolean operations and concatenation:

Definition 2.3 (Locally Testable Languages). *The class of Locally Testable Languages, or LTO, is defined as the smallest set of languages containing all the LT languages and such that it is closed under the boolean operations and the concatenation.*

As previously stated, the main result that makes this family important for our goal is:

Theorem 2.1 (*LTO = NC*). *The class of Locally Testable Languages, LTO, coincides with the class of Regular Non-Counting Languages.*

Thus Locally Tesable languages provide another different characterization for Non-Counting Regular Languages with respect to Star Free Regular Expressions, Aperiodic Grammars, First Order Logic, and Aperiodic Finite State Automata.

2.2. Operator Precedence Languages

Another crucial concept for this work is the one of Operational Precedence Languages, or OPLs. Those languages, presented by Robert Floyd in [1], can be defined starting from the concept of Operator Precedence Grammars, or OPGs. This approach makes it easier to introduce the main idea underlying OPLs: the Operator Precedence Relations, or OPRs.

Definition 2.4 (Operator Precedence Grammar, Relations and Language). *Let $G = (\Sigma, V_N, P, S)$ be a Context-Free grammar such that, being $V = \Sigma \cup V_N$, for each rhs part α of the rules in P it is valid that $\alpha \notin V^* V_N V_N V^*$, i.e., at least one terminal is interposed between any two nonterminals occurring in α.*
Let $\Sigma_\# = \Sigma \cup \{\#\}$, $a, b \in \Sigma$, $A, B \in V_N$, $C \in V_N \cup \{\varepsilon\}$, $\alpha, \beta \in V^*$, then:
$\mathcal{L}_G(A) = \{a \in \Sigma \mid \exists C : A \Rightarrow_G^* Ca\alpha\}$ and $\mathcal{R}_G(A) = \{a \in \Sigma \mid \exists C : A \Rightarrow_G^* \alpha aC\}$.
The Operator Precedence Relations (OPRs) are defined over $\Sigma_\# \times \Sigma_\#$ as:

- *equal in precedence:*
  $a \doteq b \Leftrightarrow \exists A \rightarrow \alpha aCb\beta \in P$;
- *takes precedence:*
  $a \gtrdot b \Leftrightarrow \exists A \rightarrow \alpha Bb\beta \in P, a \in \mathcal{R}_G(B)$
  $a \gtrdot \# \Leftrightarrow a \in \mathcal{R}_G(B), B \in S$;
- *yields precedence:*
  $a \lessdot b \Leftrightarrow \exists A \rightarrow \alpha aB\beta \in P, b \in \mathcal{L}_G(B)$
  $\# \lessdot b \Leftrightarrow b \in \mathcal{L}_G(B), B \in S$.

The Operator Precedence Relations can be collected into a $|\Sigma_\#| \times |\Sigma_\#|$ array, called Operator Precedence Matrix of the grammar, OPM(G), containing, for each ordered pair $(a, b) \in \Sigma_\# \times \Sigma_\#$ the sets of relations holding between the two characters a and b.
If for each pair $(a, b) \in \Sigma_\# \times \Sigma_\#$, $|OPM(G)_{(a,b)}| \leq 1$, OPM(G) is said to be conflict-free and the grammar G is an Operator Precedence Grammar (OPG).
An Operator Precedence Language (OPL) is a language expressible via an Operator Precedence Grammar.

It is known, for instance from [3], that an Operator Precedence Language is a Structured Context-Free Language so that to each string belonging to to an OPL, there can be associated a unique structure. For the sake of conciseness, the concept of structure is described only for Operator Precedence Languages, and not for general Context-Free Languages.

For OPLs, the structure of strings is associated with the concept of chain, that can be defined as:

Definition 2.5 (chain). *Let M be an OPM and Σ an alphabet such that M is defined over $\Sigma_\# \times \Sigma_\#$, then the couple (Σ, M) is called an OP-alphabet.*

- *A simple chain is a word $a_0a_1a_2\cdots a_na_{n+1}$, written as ${}^{a_0}\lfloor a_1a_2\cdots a_n\rfloor^{a_{n+1}}$, such that: $a_0, a_{n+1} \in \Sigma_\#$, $a_i \in \Sigma$ for every $i : 1 \le i \le n$, $M_{a_0a_{n+1}} \neq 0$, and $a_0 \lessdot a_1 \doteq a_2 \doteq \cdots \doteq a_n \gtrdot a_{n+1}$.*
- *A composed chain is a word $a_0x_0a_1x_1a_2\cdots a_nx_na_{n+1}$, with $x_i \in \Sigma^*$, where ${}^{a_0}\lfloor a_1a_2\cdots a_n\rfloor^{a_{n+1}}$ is a simple chain, and either $x_i = \varepsilon$ or ${}^{a_i}\lfloor x_i\rfloor^{a_{i+1}}$ is a chain (either simple or composed), for every $i : 0 \le i \le n$. Such a composed chain will be written as ${}^{a_0}\lfloor x_0a_1x_1a_2\cdots a_nx_n\rfloor^{a_{n+1}}$.*
- *The body of a chain ${}^{a}\lfloor x\rfloor^{b}$, simple or composed, is the word x.*

The set of all the strings $s \in \Sigma^*$ that are bodies of chains over the OP-alphabet (Σ, M) is defined to be the Max Language of M.

Notice that the structure of a string belonging to an OPL, relies only on the MaxLanguage, thus on the OPM, and not directly on the grammar itself.

Given an OPL η, a string $s \in \eta$, and being w a substring of s such that it is preceded by the character a and followed by the character b inside $\#.s.\#$, then w is said to be *well chained* on s iff ${}^{a}\lfloor x\rfloor^{b}$ is a chain. Trivially, a string is well-chained iff it belongs to the MaxLanguage.

There have been presented all the concepts needed to introduce how the Non-Counitngness can be defined for OPLs.

Definition 2.6 (NC_{OP}). *Given a OP-alphabet(Σ, M) and an OP language $\eta \subseteq \Sigma^*$, η is Non-Counting Operator Precedence Languages iff $\exists n > 1$ such that, for all strings $x, u, w, v, y \in \Sigma^*_{any}$, $m \in \mathbb{N}$ where w and uwv are well chained on xu^nwv^ny w.r.t M then $xu^nwv^ny \in \eta \Leftrightarrow xu^{n+m}wv^{n+m}y \in \eta$.*

2.2.1 Operator Precedence Expressions and Fence

In the literature, as in [3], different characterizations for OPLs have been proposed and discussed. In particular, there are different representations of OPLs inspired by the notorious ones used for Regular Languages: Operator Precedence Grammars (OPG), Operator Precedence Expressions (OPE), and Monadic Second Order Logic (MSO). From those characterizations, the OPE introduces the concept of *fence* that reveals to be useful for the purposes of this work.

The *fence* operation, in a few words, requests that a certain substring is well chained on the final string.

In particular, OPEs are defined as Regular Expressions expanded with the *fence* operation defined as:

Definition 2.7 (fence over OPE). *Given an OP alphabet (Σ, M), where M is complete, an OPE E and its language $L_M(E) \subseteq \Sigma^*$ are defined as follows.*

The meta-alphabet of OPE uses the same symbols as REs, together with the two symbols '[' and ']'. Let E_1 and E_2 be two OPEs:

$a[E_1]b$, called the fence operation, i.e., we say E_1 in the fence a, b, is an OPE with:

if $a, b \in \Sigma$: $L_M(a[E_1]b) = a \cdot \{x \in L_M(E_1) \mid M(a \cdot x \cdot b) = \lfloor a \cdot M(x) \cdot b\rfloor\} \cdot b$;

if $a = \#, b \in \Sigma$: : $L_M(\#[E_1]b) = \{x \in L_M(E_1) \mid M(x \cdot b) = \lfloor M(x) \cdot b\} \cdot b$;

if $a \in \Sigma, b = \#$: $L_M(a[E_1]\#) = a \cdot \{x \in L_M(E_1) \mid M(a \cdot x) = \lfloor a \cdot M(x)\rfloor\}$.

Where E_1 must not contain $\#$.

It has been proven that OPL family coincides with the family of languages definable via OPEs, OPGs, or MSOs, and that the NC_{OP}, instead, coincides with the Aperiodic OPGs, Star Free OPEs, and First Order Logic.

3. Extending Locally Testable Languages

A first attempt to define a subclass of NC_{OP} using an LTO-like manner consists of finding the intersection of a Locally Testable language and a MaxLangage. By noticing that MaxLanguages are Operator Precedence Non-Counting Languages, and leveraging theorem 2.1, the language defined in this way is trivially NC_{OP} just because the intersection of an NC_{OP} language and an NC language, is itself NC_{OP}. Thus this definition does not add any new characteristic for the NC_{OP} languages.

Thus, how can Locally Testable Languages be

expanded?
In order to extend the expressiveness of LTO, there will be introduced the *fence-subs*, an operation whose intent is to bring the *fence* operation in an LTO-like format. In order to introduce the *fence-subs*, it is convenient to present before the *stringSub* operation with the languages definable with it, and then add the fence concept to it.

Definition 3.1 (*stringSub*). *Being Σ, Φ two alphabets such that $\Phi \cap \Sigma = \emptyset$, $\eta \subseteq (\Sigma \cup \Phi)^*$ a language, and $map : \Phi \to \wp(\Sigma^*)$ a function mapping characters of Φ into languages over Σ^*, then the function $stringSub(\eta, map)$ returns the language composed by all the possible substitutions in the strings of η of the characters $- \in \Phi$ with any string in $map(-)$.*

By leveraging the *stringSub* concept, there can be defined the class of Locally Testable Extended Languages, or LTEO as the smallest set of languages, closed under boolean operations and concatenation, that contains both LT languages and all the languages definable as $stringSub(\eta, map)$, where $\eta \subseteq (\Sigma \cup \Phi)^*$ is Locally Testable, and the codomain of *map* is contained in LTEO itself.
Notice that an LTEO language η is not NC_{OP} nor NC, and neither the intersection of η with a Max Language results to be NC_{OP}. However, the LTEO family showed to be a subfamily of Regular Languages, which was quite unexpected.

3.1. Relation with Regular Languages

Given any alphabet $\Sigma = \{a, b, \ldots\}$ and any element i, it is trivial to build a new *melted* alphabet $\Sigma_i = \{a_i, b_i, \ldots\}$ such that $\Sigma \cap \Sigma_i = \emptyset$. This same process can also be applied on a set I of elements, resulting in $\Sigma_I = \bigcup_{i \in I} \Sigma_i$. Moreover, it is trivial to see that it is possible to map characters, strings, and languages, from being defined over Σ to being defined over some Σ_I and vice versa.
In particular, two languages $\eta_1 \subseteq \Sigma^*$ and $\eta_2 \subseteq \Sigma_I^*$ for some alphabet Σ and set I, they are said to be *f-equivlent* if η_1 corresponds to the mapping of η_2 over Σ.
Relying on the *f-equivalence* concept, there have been studied three operations such that, being $\eta, \xi \subseteq \Sigma^*$ and $\eta_m, \xi_m \subseteq \Sigma_I^*$ such that η is *f-equivalent* to η_m and ξ is*f-equivalent* to ξ_m, then:

- $\eta.\xi$ is *f-equivalent* to $\eta_m \cdot_m \xi_m$;
- $\eta \cap \xi$ is *f-equivalent* to $\eta_m \cap_m \xi_m$;
- $\neg \xi$ is *f-equivalent* to $\neg_m \xi_m$;

Leveraging those newly defined operations, there is defined the family of LTR languages as the smallest set of languages defined over Σ_I that contains the strictly Locally Testable languages and that is closed under $\cdot_m$, $\neg_m$, and $\cap_m$. Two important results of LTR are:

1. Any LTR language is a Regular Language;
2. Any language that is *f-equivalent* to an LTR language, is regular too.

is that any LTR language is a Regular Language. Another essential result is that for any language $\xi = stringSub(\eta, map)$, with η that is LT, and with *map* that has as codomain a set of languages which are *f-equivalent* to LTR ones, then there exists an LTR language ξ_m that is *f-equivalent* to ξ. This implies that any LTOE language is *f-equivalent* to an LTR language, thus LTOE is a subset of Regular Languages.

4. Locally Testable over Operator Precedence Languages

There is now presented the *fence-subs* operation, which is similar to *stringSub*, but it also checks to substitute substrings that are well chained on the final string. This check is the same that the *fence* operation does, making *fence-subs* a mix between *fence* and *stringSub*.

Definition 4.1 (*fence-subs*). *Given:*
an OP-alphabet (Σ, M);
two alphabets Σ, Φ such that $\Sigma \cap \Phi = \emptyset$ and $\Sigma_\Phi = \Sigma \cup \Phi$;
a language $\eta \subseteq \Sigma_\Phi^*$;
a function $map : \Phi \to \wp(\Sigma^*)$ that maps each character $-_i \in \Phi$ to one language $map(-_i) = \eta_i \subseteq \Sigma^*$;
then $fence\text{-}subs(\eta, map)$ defines the language containing the strings s such that either $s \in \eta$ or there exists $s' \in \eta$ such that $s \in stringSub(\{s'\}, map)$ and, being the character $-_j \in \Phi$ occurring in $\# \cdot s' \cdot \#$ preceeded by the character a and followed by the character b and being substituted in s by the string $x_j \in \eta_j = map(-_j)$, then:

- *if $a, b \in \Sigma$: $M(a \cdot x_j \cdot b) = \lfloor a \cdot M(x_j) \cdot b \rfloor$;*
- *if $a = \#, b \in \Sigma$: $M(x_j \cdot b) = \lfloor M(x_j) \cdot b \rfloor$;*
- *if $a \in \Sigma, b = \#$: $M(a \cdot x_j) = \lfloor a \cdot M(x_j) \rfloor$.*

However, even the family of languages defined as LTOE using *fence-subs* instead of *stringSub* is not a subset of NC_{OP} family, and this is because also this definition is too expressive. To reduce the expressiveness of this class, the following restriction is introduced:

Definition 4.2 (SubString Language). *Given a language $\eta \in \wp(\Sigma^*)$, the SubString Language of η is the language defined as:*
$SSL(\eta) = \{x \in \Sigma^* \mid \exists w \in \eta$ *such that x is a substring of w*$\}$.

Definition 4.3 (Fence Substitution Restriction). *Given an OP-alphabet (Σ, M) and $\tau =$ fence-subs(ξ, map), it is said that τ is Fence Sub Restricted (FSR) iff, for each element $s \in SSL(\tau) \cap (\Sigma^3_{\#} \cdot \Sigma^*)$ there exists at most one string $o \in SSL(\xi) \cap \Sigma^3_{\#,\Phi} \cdot \Sigma^*_{\#,\Phi}$ such that $s =$ fence-subs(o, map).*[1]

Considering this restriction, the following family of languages can be defined:

Definition 4.4 (LTOP). *Given an OP-alphabet (Σ, M) it is defined as LTOP family of languages the smallest set of languages that contains:*

1. *the empty language: $\emptyset$*
2. *any language $\tau =$ fence-subs(ξ, map) where ξ is LT_{k,Σ_Φ} and $map : \Phi \to LTOP$ and where there subsists the FSR on τ*

And that it is closed under Boolean operations and concatenation.

It is proven that the LTOP class of languages is contained in the NC_{OP} family:

Theorem 4.1 ($LTOP \subseteq NC_{OP}$). *The family of languages defined by LTOP is a subset of the NC_{OP} family.*

Thus LTOP definition is the Locally Testable characterization of an NC_{OP} family of languages, which was the aim of this work.
Moreover, comparing the newly defined LTOP languages with the known characterizations, it turns out that the family of languages expressible via Star Free OPE is contained in the family of LTOP languages. This result, together with those regarding Star Free OPE, and theorem 4.1, make it trivial to state the following:

Theorem 4.2 ($LTOP = NC_{OP}$). *The class of LTOP languages coincides with the class of Non-Counting languages over OP.*

[1]In this definition, as before, $\Sigma_\Phi = \Sigma \cup \Phi$, $\Sigma_{\#} = \Sigma \cup \{\#\}$, and $\Sigma_{\#,\Phi} = \Sigma_\Phi \cup \{\#\}$.

This result outperforms the original goal for this work.
As the last step, it remains to prove that it is decidable whether a language defined in an LTOP manner satisfies the FSR property or not. In order to answer this question, there are several considerations, starting from a reformulation of FSR property. Being η an LT language, it is proven that also $SSL(\eta)$ is an LT language. It is also proven that the FSR relies only upon the intersection of the *fence-subs* languages of the substrings of length 3 of ξ with the *fence-subs* language of $SSL(\xi)$. Thus, being the substrings of length 3 of the language ξ a finite number, the FSR property relies on a finite number of comparisons of languages defined via *fence-subs* over LT languages.
Any language $\xi =$ *fence-subs*(η, map), where η is LT and each element of the codomain of *map* is *f-equivalent* to the intersection of an LTR language and a suitable Max Language, is such that there exists a language ξ' that is *f-equivalent* to ξ and that is the intersection of an LTR language and a suitable Max Language. Being the Max Language a Context-Free Language, and being a LTR Language a Regular Language, by leveraging the known fact that the intersection of a Regular Language and a Context Free one results in a Context Free Language, there can be applied the Pumping Lemma for Context-Free Languages[2] to prove that it is decidable if the *f-equivalent* language is empty or not. By noticing that a language η is empty iff it is empty also any other language η_1 that is *f-equivalent* to it, it is trivial to see that the FSR property relies upon the emptiness of a finite set of Context-Free Languages, thus FSR property is decidable.

5. Conclusions

In conclusion, the initial goal of the thesis has been reached as it has been found an LTO-like characterization of an NC_{OP} family. The family of languages definable via LTOP turned out to be not only a subfamily of NC_{OP}, but to coincide with NC_{OP} itself. This, together with the fact that it is decidable whether or not a language is LTOP, enriches the characterizations of OPLs and in particular of NC_{OP}.
Moreover, LTOP enriches the correlation between NC_{OP} and NC families, bringing another

[2]The Bar-Hillel Lemma

representation from the word of Regular Languages to the Operator Precedence one, and expands the variety of known equivalent characterizations, presented in [2], for the NC_{OP} family.

References

[1] Robert W. Floyd. Syntactic Analysis and Operator Precedence. *J. ACM*, 10(3):316–333, 1963.

[2] Dino Mandrioli and Matteo Pradella. Generalizing input-driven languages: Theoretical and practical benefits. *Computer Science Review*, 27:61–87, 2018.

[3] Dino Mandrioli, Matteo Pradella, and Stefano Crespi-Reghizzi. Aperiodicity, star-freeness, and first-order definability of structured context-free languages. *CoRR*, abs/2006.01236, 2020.

[4] Seymour Papert Robert McNaughton. *Counter-Free Automata*. The MIT Press, 1971.