Strict Local Testability: a Linguistic Model for Syntactical Analysis

6 L − 2

Antonio Magnaghi and Hidehiko Tanaka
The University of Tokyo

1 Introduction

Local Testability (LT) is an active area of research in the field of formal languages. It is a fruitful concept in describing different phenomena that extend to various contexts and situations. On an abstract level, LT expresses the possibility to ascertain the value of an attribute of a complex instance from a class of objects through the analysis of simple constituents of the instance itself, without requiring to consider the whole object at once. The interest in LT goes beyond formal languages: it can be applied, for instance, to pattern recognition, parallel parsing, error recovery, neural networks, symbolic logic and algebraic structures [2, 4]. In the following sections we will pinpoint some considerations about LT in a strict sense ($LTs.s.$): it identifies a subclass of the class LT. Firstly, the paper will outline some results we could achieve on $LTs.s.$ regular languages. Then, preliminary considerations will be examined about our intent to generalize the mathematical framework and decidibility results to tree languages.

2 String $LTs.s.$ and language parsing

This section focuses on k-local testability in a strict sense ($LT_k s.s.$) for string languages. Moreover it underlines the links of this class of languages to the linguistic property of Aperiodicity, directly providing motivations to a further investigation about the more general case of tree languages.

Based on [4], for every integer k a language L on the alphabet Σ is $LT_k s.s.$ if sets $\alpha_k, \beta_k, \gamma_k \subseteq \Sigma^k$ exist such that:

$$\forall x \in \Sigma^*, x \in L \iff$$

$$L_k(x) \in \alpha_k \wedge I_k(x) \subseteq \beta_k \wedge R_k(x) \in \gamma_k \qquad (1)$$

In (1) $L_k(x)$ is the k-prefix of x, R_k is the k-suffix and $I_k(x)$ is the set of proper substrings of x.

For instance, let us consider the regular language L over the alphabet $\Sigma = \{a, b\}$, whose strings are such that the occurrence of symbol a implies the occurrence of the substring abb: $L = \{x : x = uav \Rightarrow v = bbw, \forall a \in x, w \in \Sigma^*\}$. $L \in LT_3 s.s.$ because, simply by a decomposition of an arbitrary string y in its 3-substrings, it is possible to locally analyze the string and evaluate whether it meets the syntactical constraint defining L. Let $\alpha_3 = \{abb, bab, bba, bbb\}$, $\beta_3 = \alpha_3$, $\gamma_3 = \{abb, bbb\}$ denote respectively the sets of all 3-prefixes, 3-internal-substrings, 3-suffixes of sentences belonging to the language. Such string collections represent the repository of recognition patterns to utilize when y is parsed. The correctness of y can be evaluated moving a 3-letter-wide one-dimension window from the left to the right end of the string. Every time the window is slid right-ward by one letter, y is incorrect if the visible substring is not contained either in α_3 (if it is the prefix) or in β_3 (if it is a proper substring) or in γ_3 (if it is the suffix). On the other hand, when such a test produces a positive result on all 3-substrings and the right end of y is reached, then the sentence is recognized as an element of L.

The example above clarifies the basic ideas underlying the concept of $LTs.s.$ Our interest in the property of local testability bases on its link to the linguistic universal of Aperiodicity. Formally, the class of Aperiodic languages is generated by the closure of $LTs.s.$ class w.r.t. boolean operators and concatenation. Aperiodicity expresses the property that the language definition is free from constraints counting modulo an integer over any sentence constituents. Such a characteristic is empirically met by all natural languages as well as artificial ones. It is known that, instead, aperiodic languages can easily be algorithmically inferred [3]. Hence $LTs.s.$ constitutes a relevant linguistic model for syntactical analysis. A systematic characterization of locally testable languages was presented in the past [2]. However, the adopted techniques relied on the study of the algebraic structure of the syntactic monoid. Therefore the conceptual framework results elaborate and computational problems were left unresolved.

3 New Results on Strings

We tried to frame the concept of $LTs.s.$ in an original combinatoric perspective. Our approach aimed at capturing strict local testability in a direct manner, without employing any algebraic property of the syntactic monoid. Instead, a set-theoretically based analysis is carried out in order to link local testability to topological properties of the automaton of language L. The specific points we addressed can be summarized as follows:

1. *Characterization of $LT_k s.s.$ property*: for a specific integer value of k, the analyzed decision problem is: "$L \in LT_k s.s.$?"

   A sufficient and necessary condition is formulated. It involves topological properties of paths in the accepting automaton. Such a characterization has the advantage to impose determinism as a unique constraint, without requiring the automaton to be reduced. Nonetheless the minimal automaton case is studied and then conveniently employed in subsequent considerations.

2. *Development of an algorithm to decide $LTs.s.$ property (existential problem)*: given language L,

does a value of k exist such that $L \in LT_k s.s.$?

Our approach consists, first, in defining the Prefix-Path-Intersection Graph (PPIG). For its construction a fixed-point algorithm is formulated. Then its complexity is shown to be $o(|\Sigma|^2 mn)$, where Σ, m and n are, respectively, the alphabet of L, the number of edges and the number of states of the finite state automaton accepting L.

3. *Development of an algorithm for the optimization problem of order evaluation*: for a language $L \in LTs.s.$, which is the minimum value k_{min} such that $k_{min} = \min_k\{k : L \in LT_k s.s.\}$?

   The study of the PPIG properties relates the length of the longest path in the PPIG to the order of language L. Then, finally, we obtained the major result: the order of L can be evaluated in $o(|\Sigma|^2 mn)$.

In addition to previous results, the introduced approach seems to have a worthwhile characteristic: the syntactic monoid and its algebraic structure are not involved. This leads us to think that such an approach might effectively give insights on how to extend our considerations to different contexts, where linear sequences of symbols are replace by more complex linguistic structures. Therefore we are now investigating on tree languages.

4 Preliminary results on Tree Languages

The representative power of regular languages does not allow to describe important linguistic aspects. Hence programming languages are always defined through context-free grammars (type 2 in Chomsky's classification [1]) and during parsing activity the input program is represented, internally to the compiler, by an abstract syntax tree. However context-free grammars are capable to describe also counting phenomena, which are never present in linguistic communication processes. Therefore this kind of formalism appears to be excessively general. We aim at investigating the adequacy of $LTs.s.$ on trees as a linguistic model to describe syntactical aspects. Let us consider the grammar G and the sentence $x = a + a + a * a$.

```
G:      E -> E + T | T
        T -> T * F | F
        F -> a
```

G defines arithmetic expressions in two operators of different precedence. The skeleton language $S(G)$ of G is the set of all derivation trees $T(G)$ of sentences of G where non-terminals have been deleted from internal nodes. Intuitively the tree language $S(G)$ is 2-strictly locally testable in the sense that every skeleton can be parsed by elements belonging to convenient sets α_2, β_2. α_2 contains all possible depth-2 prefix recognition skeletons generated from the axiom of G, and β_2 consists of depth-2 internal/frontier recognition skeletons that can be generated by non-terminals reachable from the axiom: in this case E and T. Let s be the skeleton associated to sentence x. s can be decomposed in depth-2 trees belonging to α_2 and β_2 ensuring that it is syntactically correct.

Let L be a tree language, α_k a set of k-prefix recognition trees, and β_k a set of k-internal/k-frontier recognition trees. We propose the following definition. L is in $LT_k s.s.$ w.r.t. α_k, β_k iff:

$$\forall\ tree\ \ t : t \in L \iff pre_k(t) \in \alpha_k \wedge int_k(t) \subseteq \beta_k \quad (2)$$

In (2) the operator $pre_k(t)$ produces the k-depth subtree rooted in the root of t and the operator $int_k(t)$ produces the set of k-depth subtrees of t whose root is at depth level one at least. The introduced definition tries to extend $LTs.s.$ property from linear sequences of symbols to more articulated structures with a topological asset. The developed model shows a remarkable property concerning grammatical inference. Through a convenient labeling procedure of the nodes of elements in α_k, β_k, it is possible to infer a generative context-free grammar whose skeleton language coincides with the language defined by (2). In addition, the analysis of such and algorithm reveals that $LTs.s.$ tree languages are accepted by deterministic top-down tree automata. On the other hand it is known that a non-deterministic top-down tree automaton is required for the general case of an arbitrary context-free grammar. This shows that our definition is a refinement compared to type 2 grammars. Hence, such a characteristic constitutes a preliminary supporting result to the aim of capturing peculiar linguistic aspect of LT.

5 Future Research Activity

Future research activity will elaborate a proper mathematical framework for tree languages in order to address problems of $LTs.s.$ decidibility and order evaluation. In particular, as an applied aspect, software tools will be implemented for testing the soundness of theoretical considerations w.r.t. the syntax of programming languages.

References

[1] A. Aho, J. Ullman. *The Theory of Parsing, Translation, and Compiling.* Volume 1, Addison-Wesley, 1974

[2] J. A. Brzozowski. *Hierarchies of aperiodic languages.* R.A.I.R.O. Information Théorique (vol. 10, No. 8, août 1976, pp. 33-49)

[3] S. Crespi-Reghizzi, M. A. Melkanoff, L. Lichten. *The use of grammatical inference for designing programming languages.* Comm. ACM 16, 2 (Feb. 1973), pp. 83-90

[4] R. McNaughton, S. Papert. *Counter-free automata.* M.I.T. Press, Cambridge, MA, 1971

[5] M. Steinby. *A theory of language varieties.* in Tree Automata and Languages, M. Nivat and A. Podelski (editors), Elsevier Publ., 1992, pp. 57-81