Learning Node Label Controlled Graph Grammars (Extended Abstract)

Christophe Costa Florêncio

Department of Computer Science, K.U. Leuven, Leuven, Belgium
Tel.: +32.(0)16327863; Fax: +32.(0)16327996
Chris.CostaFlorencio@cs.kuleuven.be

1 Introduction

Within the data mining community there has been a lot of interest in mining and learning from graphs (see [1] for a recent overview). Most work in this area has has focussed on finding algorithms that help solve real-world problems. Although useful and interesting results have been obtained, more fundamental issues like learnability properties have hardly been adressed yet. This kind of work also tends not to be grounded in graph grammar theory, even though some approaches aim at inducing grammars from collections of graphs.

This paper is intended as a step towards an approach that is more theoretically sound. We present results concerning learnable classes of graph grammars.

2 Graph Grammars

Many approaches to representing graph languages exist, the present paper is restricted to the popular node label controlled (NLC) grammars. These consist of production rules with a non-terminal label at the left-hand side (lhs) of a rule, and on the right-hand side (rhs) a graph called the *daughter graph*, and an *embedding relation*. The daughter graph has its nodes labelled with both terminal and non-terminal labels.

Generation of a graph starts with the *axiom*, a node labelled with the start non-terminal symbol. Rules from the grammar are applied such that non-terminal nodes are replaced by daughter graphs, which are connected to the *host graph* according to the embedding relations. The graph language generated by a grammar consists of all graphs thus obtained that have terminal labels exclusively.

The embedding relation specifies how the daughter graph is connected by considering just the *neighbourhood* of the replaced node. For each vertex in the daughter graph, the embedding relation specifies either 'empty' or a node label. All nodes in the neighbourhood with the label will be connected to that vertex.

We assume that all rules in all grammars are productive, i.e., do not contain useless symbols, and that unit- and ϵ-productions are absent. We also assume that every rule contains at least one terminal (cf lexicalized TAG, for example).

A. Clark, F. Coste, and L. Miclet (Eds.): ICGI 2008, LNAI 5278, pp. 286–288, 2008.

Note that for many classes of graph grammar the generating grammar does not necessarily function as a parser as well. The reason is that, as part of a derivation step, the edges incident on the node to be replaced are removed, and the daughter graph that is inserted is connected in a pre-determined way. In this setting, there is no way to recover the removed edges. This may however be required for deciding membership of some given graph.

A number of restricted subclasses of *NLC* grammars can be found in the literature, we will focus on Boundary *NLC* (*B-NLC*), which disallows edges between non-terminal vertices in the rhs. This is the most expressive class of NLC grammars known for which parsers can effectively be obtained.

The *graph language generated by* grammar G will be denoted $GL(G)$, the *derivation language generated by* grammar G will be denoted $DL(G)$.

Informally speaking, a *derivation tree* for a graph and graph grammar is a tree that reflects the generation process from grammar to derived graph, i.e., the nodes correspond to the applications of rewrite rules. We define them so that the nodes are labelled with the daughter graphs of their corresponding rules. The daughters of any node in the tree correspond to the rewritings applied to the non-terminals in the rhs. The number of daughters is exactly the number of non-terminals, that is, these rules are considered to all be applied in one step.

In the case of a rule that has no non-terminals in the rhs (a *terminal* rule), the corresponding node is a leaf. In the present context, the embedding relations can be safely ignored, we thus leave these out of the derivation tree representation.

3 Learnability

We are interested in learnability in the technical sense of identification in the limit ([2]). In this paradigm a class of languages is considered learnable just if there exists an algorithm over sequences of input data that converges on a correct hypothesis after a finite number of presentations. It is assumed that all data is presented eventually. A sufficient condition for a class to be identifiable in the limit ([3]) is being r.e., consisting of just recursive languages and having the property of *infinite elasticity*:

Definition 1. ***Infinite elasticity****[3, 4]*
A class $\mathcal{L}$ of languages is said to have infinite elasticity *if there exists an infinite sequence $\langle s_n \rangle_{n\in\mathbb{N}}$ of sentences and an infinite sequence $\langle L_n \rangle_{n\in\mathbb{N}}$ of languages in $\mathcal{L}$ such that for all $n \in \mathbb{N}$, $s_n \notin L_n$, and $\{s_0, \ldots, s_n\} \subseteq L_{n+1}$.*

A class $\mathcal{L}$ of languages is said to have finite elasticity *if it does not have infinite elasticity.*

So, one way of proving learnability of a class is demonstrating it has finite elasticity and to extend the class by exploiting a closure property. The following theorem, from [5], is useful when the relation between language element and possible derivation is finite-valued. It is generally easier to prove finite elasticity of a class of derivation languages than of a class of string languages.

Let Σ and Υ be two alphabets, a relation $R \subseteq \Sigma^* \times \Upsilon^*$ is said to be *finite-valued* just if for every $s \in \Sigma^*$, there are at most finitely many $u \in \Upsilon^*$ such that Rsu. If M is a language over Υ, define a language $R^{-1}[M]$ over Σ by $R^{-1}[M] = \{s \mid \exists u(Rsu \wedge u \in M)\}$.

Theorem 2. *Let $\mathcal{M}$ be a class of languages over Υ that has finite elasticity, and let $R \subseteq \Sigma^* \times \Upsilon^*$ be a finite-valued relation. Then $\mathcal{L} = \{R^{-1}[M] \mid M \in \mathcal{M}\}$ also has finite elasticity.*

In order to obtain learnable subclasses of *B-NLC*, additional restrictions need to be imposed. Let k be an upper bound on the number of occurences of any terminal, and let *k-B-NLC* denote the class of all *B-NLC* grammars with k as such a bound. This bound implies a bound on the number of occurences of distinct non-terminal parts of daughter graphs. Since we assume a fixed alphabet and terminals in all rules, a bound on the number of rules in the grammar is implied, which implies a bound on the number of non-terminals.

Proposition 3. *For $k = 1$, $DL(\mathcal{G}_{k\text{-}B\text{-}NLC})$ has finite elasticity.*

Proof. Sketch: Assume that this class has infinite elasticity with trees $t_1, \ldots$ and derivation languages $D_1, \ldots$, with corresponding grammars $G_1, \ldots$. For any i, the set G_i'' of grammars in the class that generate a minimal derivation language and are consistent with $t_1 \ldots t_{i-1}$ is of finite cardinality. The grammar G_i must be such that it is a superset of some such grammar, with a substitution applied to it. There are just a finite number of such substitutions for each G', so after p there can only occur a finite number of different grammars. Since $t_i \notin DL(G_i)$ and $\{t_1, \ldots t_{i-1}\} \subseteq DL(G_i)$, each of these grammars can only occur a finite number of times in the sequence. Thus, the whole sequence $G_1, \ldots$, and thus the whole sequence $D_1, \ldots$, must be of finite length. □

Applying Theorem 2 twice, this result can be generalized to $k > 1$, and then from derivation- to graph language. It then follows that For any k, $GL(\mathcal{G}_{k\text{-}B\text{-}NLC})$ is learnable from positive data (graphs) by a consistent and conservative learner.

References

[1] Cook, D.J., Holder, L.B. (eds.): Mining Graph Data. John Wiley & Sons, Chichester (2006)
[2] Gold, E.M.: Language identification in the limit. Information and Control 10, 447–474 (1967)
[3] Wright, K.: Identification of unions of languages drawn from an identifiable class. In: The 1989 Workshop on Computational Learning Theory, pp. 328–333. Morgan Kaufmann, San Mateo (1989)
[4] Motoki, T., Shinohara, T., Wright, K.: The correct definition of finite elasticity: Corrigendum to identification of unions. In: The Fourth Workshop on Computational Learning Theory. Morgan Kaufmann, San Mateo (1991)
[5] Kanazawa, M.: A note on language classes with finite elasticity. Technical Report CS-R9471, CWI, Amsterdam (1994)