A Pumping Lemma for Output Languages of Attributed Tree Transducers

Armin Kühnemann and Heiko Vogler

Grundlagen der Programmierung, Institut für Softwaretechnik I,
Fakultät Informatik, Technische Universität Dresden, D–01062 Dresden, Germany,
e–mail: {ak15,vogler} @ inf.tu–dresden.de

Abstract. An attributed tree transducer is a formal model for studying properties of attribute grammars. We introduce a pumping lemma for output languages of noncircular, producing, and visiting attributed tree transducers. The pumping lemma can be applied to gain two results: (1) there is no noncircular, producing, and visiting attributed tree transducer which computes the set of all monadic trees with exponential height as output and (2) there is a hierarchy of noncircular, producing, and visiting attributed tree transducers with respect to their number of attributes.

Keywords: Attribute Grammars, Tree Languages, Pumping Lemma

1 Introduction

First Aho and Ullman have inspected pumping lemmata for output languages of translation schemes in [1], namely for generalized syntax directed translations. Perrault and Ésik have introduced in [8] and [4], respectively, pumping lemmata for (nondeterministic) top–down tree transducers (cf. [9, 10, 2]). Engelfriet, Rozenberg, and Slutzki have presented in [3] a pumping lemma for output languages of deterministic top–down tree–to–string transducers which has a structure that is closely related to the pumping lemma for context–free string languages. The proof of this lemma had a big influence on the development of the pumping lemma for attributed tree transducers, which was introduced in [7].

Attributed tree transducers have been investigated by Fülöp in [5]. They are abstractions of attribute grammars (cf. [6]) in the sense that they take trees over an arbitrary ranked alphabet of input symbols rather than derivation trees as argument, and that the values of the attributes are also trees over a ranked alphabet of output symbols.

Like in attribute grammars, the set of attributes is partitioned into the set of synthesized and inherited attributes which are associated to the input symbols and which compute their values in a bottom–up manner and in a top–down manner, respectively. In contrast to attribute grammars, to every input symbol the whole set of attributes is associated. Roughly speaking, computing the value of a synthesized attribute occurrence of a node x of an input tree, the values of the inherited attribute occurrences of x and of the synthesized attribute occurrences of its sons (if they exist) may be used and, computing the value of an inherited attribute occurrence of x, the values of the inherited attribute occurrences of its father (if it exists) and of the synthesized attribute occurrences of x and of its brothers may be used.

We consider only total deterministic attributed tree transducers: For every node x of an input tree which is labeled by a particular input symbol and for every synthesized attribute s, the computation of the attribute occurrence of s at x is fixed by exactly one rule. Similarly, for every node x which is labeled by a particular input symbol and for every inherited attribute i, the computation of the attribute occurrence of i at the j–th son of x is fixed by exactly one rule.

As in attribute grammars, these dependencies can induce circularities among the attribute occurrences. We restrict the attributed tree transducers to be non-circular and we designate a synthesized attribute s_{in} as initial attribute. Thus we designate an initial attribute occurrence at the root of every input tree of which the value will be the output tree. Then every attributed tree transducer M computes a total function from input trees to output trees. The output language of an attributed tree transducer M is defined as the range of this function.

Additionally, we restrict our pumping lemma to producing and visiting attributed tree transducers. An attributed tree transducer is producing, if every rule application delivers at least one new output symbol. An attributed tree transducer is visiting, if for every input tree and for every node x of it, the value of at least one attribute occurrence of x is needed to compute the value of s_{in} at the root.

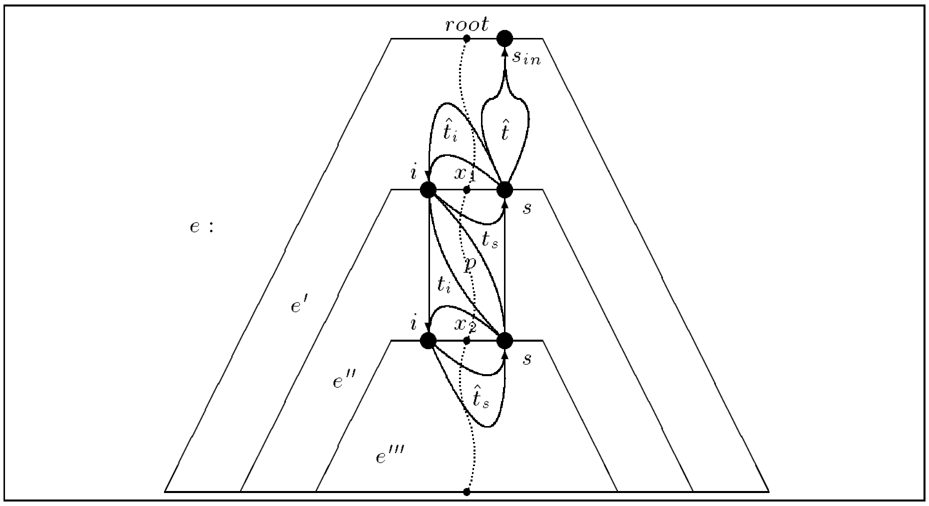
The main idea of our pumping lemma is adopted from the pumping lemma for context–free string languages. We consider input trees belonging to a sufficiently large output tree to obtain new pumped output trees: For every producing and visiting attributed tree transducer M, a natural number n_M, called the pumping index of M, can be constructed. If we choose an output tree t from the output language of M which has at least n_M nodes, then every input tree e which can be transformed into t has the following property: e is high enough, such that it has a path p, on which two different nodes x_1 and x_2 can be found, which have the same attribute–set, i.e. the same set of attributes, for which the attribute occurrence of x_1 (and of x_2, respectively) is needed to calculate s_{in} at the root of e. Assuming that x_1 is closer to the root than x_2, we can define the following decomposition of e into input patterns. Roughly speaking,

- e' is the tree e without the subtree which has x_1 as root.
- e'' is the tree which has x_1 as root without the subtree which has x_2 as root.
- e''' is the tree which has x_2 as root.

This decomposition of e induces a decomposition of the output tree t into output patterns. Roughly speaking,

- The pattern $\hat{t}$ corresponds to the normal form of the attribute occurrence s_{in} at the root of e, which is calculated only on the nodes of e'.
- For every synthesized attribute s in the attribute–set of the two nodes x_1 and x_2, the pattern t_s (and $\hat{t}_s$) corresponds to the normal form of the attribute occurrence s at x_1 (and at x_2, respectively), which is calculated only on the nodes of e'' (and e''', respectively).

– For every inherited attribute i in the attribute–set of the two nodes x_2 and x_1, the pattern t_i (and $\hat{t}_i$) corresponds to the normal form of the attribute occurrence i at x_2 (and at x_1, respectively), which is calculated only on the nodes of e'' (and e', respectively).



Since the two nodes x_1 and x_2 are compatible with respect to their attribute–sets, we can construct new input trees by repeating e'' arbitrarily many times. The output patterns t_s and t_i must be used for every repetition of e'' to obtain the corresponding new output trees. Thus the pumping process itself can be described by using only the output patterns.

References

1. A.V. Aho and J.D. Ullman. Translations on a context free grammar. *Inform. and Control*, 19:439–475, 1971.
2. J. Engelfriet. Bottom–up and top–down tree transformations — a comparison. *Math. Syst. Theory*, 9:198–231, 1975.
3. J. Engelfriet, G. Rozenberg, and G. Slutzki. Tree transducers, L systems, and two–way machines. *J. Comput. Syst. Sci.*, 20:150–202, 1980.
4. Z. Ésik. Decidability results concerning tree transducers. *Acta Cybernetica*, 5:1–20, 1980.
5. Z. Fülöp. On attributed tree transducers. *Acta Cybernetica*, 5:261–279, 1981.
6. D.E. Knuth. Semantics of context–free languages. *Math. Syst. Theory*, 2:127–145, 1968.
7. A. Kühnemann and H. Vogler. A pumping lemma for output languages of attributed tree transducers. *Acta Cybernetica*, 11:261–305, 1994.
8. C.R. Perrault. Intercalation lemmas for tree transducer languages. *J. Comput. Syst. Sci.*, 13:246–277, 1976.
9. W.C. Rounds. Mappings and grammars on trees. *Math. Syst. Theory*, 4:257–287, 1970.
10. J.W. Thatcher. Generalized^2 sequential machine maps. *J. Comput. Syst. Sci.*, 4:339–367, 1970.