Journal of Computer and System Sciences 82 (2016) 1201–1222

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Linking theorems for tree transducers

Zoltán Fülöp [a,1,2], Andreas Maletti [b,*,2]

[a] Department of Foundations of Computer Science, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary
[b] Institute of Computer Science, Universität Leipzig, Augustusplatz 10-11, 04109 Leipzig, Germany

ARTICLE INFO

Article history:
Received 17 May 2014
Received in revised form 16 March 2015
Accepted 4 May 2016
Available online 2 June 2016

Keywords:
Synchronous grammar
Tree transducer
Linking theorem
Compositions
Expressive power

ABSTRACT

Linear extended multi bottom-up tree transducers are presented in the framework of synchronous grammars, in which the input and the output tree develop in parallel by rewriting linked nonterminals (or states). These links are typically transient and disappear once the linked nonterminals are rewritten. They are promoted to primary objects here, preserved in the semantics, and carefully studied. It is demonstrated that the links computed during the derivation of an input and output tree pair are hierarchically organized and that the distance between (input and output) link targets is bounded. Based on these properties, two linking theorems are developed that postulate the existence of certain natural links in each derivation for a given input and output tree pair. These linking theorems allow easy, high-level proofs that certain tree translations cannot be implemented by (compositions of) linear extended multi bottom-up tree transducers.

1. Introduction

The notion of a multi bottom-up tree transducer was originally introduced and studied in [4,26], albeit under different names. The deterministic variant was rediscovered in [13,14], where the name "multi bottom-up tree transducer" was coined. Quite recently, it was established [11,28] that the (weighted) linear extended variant has very nice algorithmic properties. It was thus further developed into a formal model for tree-to-tree translation [29,31], which is a sub-discipline in syntax-based statistical machine translation [23]. An open-source implementation of a statistical machine translation system based on shallow linear extended multi bottom-up tree transducers [6] inside the Moses framework [24] is available and has been evaluated on an English-to-German translation task.

Here we consider linear extended multi bottom-up tree transducers (for short: MBOT) and present them in the form of synchronous grammars [7]. In such grammars, the nonterminals (or states) occurring in sentential forms are linked, and the linked nonterminals are replaced at the same time. Consequently, productions (or rules) of synchronous grammars often have at least two components: the input side and the output side. An MBOT rule might have even more than two components because it contains a vector of output trees. More formally, an MBOT is a finite-state tree transducer, in which the rules are of the form $\langle \ell, q, \vec{r} \rangle$, where the left-hand side ℓ is a tree that is linear in the states (i.e., each state can occur at most once), q is a state, and the right-hand side $\vec{r}$ is a vector of trees, in which states can also occur. It is required that all

* Corresponding author.
E-mail addresses: fulop@inf.u-szeged.hu (Z. Fülöp), maletti@informatik.uni-leipzig.de (A. Maletti).

[1] Supported by the program TÁMOP-424B/2-11/1-2012-0001, grant B2/2R/3350 and by the NKFI grant K 108448.
[2] Supported by the German Research Foundation (DFG) grant MA/4959/1-1 and the German Academic Exchange Service (DAAD) and Hungarian Scholarship Board Office (MÖB) exchange project "Theory and Applications of Automata" (grant 5567).

http://dx.doi.org/10.1016/j.jcss.2016.05.003
0022-0000/

states that occur in $\vec{r}$ also appear in ℓ. In contrast to other presentations [11] we do not use ranked alphabets — neither for the input and output symbols nor for the states. It is easy to see that our (rank-free) formalization (syntactically) includes all ranked versions including those that permit different ranked alphabets for the input and output symbols. Our model thus becomes slightly more powerful than traditional MBOT since it allows the same symbol to occur with different ranks in the input or output trees.

The semantics of our MBOT is defined by means of synchronous rewriting or, more generally, with the help of a derivation relation over sentential forms. In the synchronous rewriting approach, several parts of the sentential form develop (via the rules) at the same time. Typically, the left-hand side of the rule contributes to the input tree of the sentential form and the right-hand side contributes at the same time to the output tree of the sentential form. For MBOT, the right-hand side consists of a vector of trees, so it can act simultaneously at several positions in the output tree. The input and output positions that are supposed to develop in parallel are recorded by links (v, w), which (in our case) relate a position v in the input tree to a position w in the output tree. Consequently, a form of an MBOT is a tuple $\langle \xi, A, I, \zeta \rangle$ consisting of a (partial) input tree ξ, two sets $A, I \subseteq \mathrm{pos}(\xi) \times \mathrm{pos}(\zeta)$ of links, and a (partial) output tree ζ. The elements of A and I are called active and inactive links, respectively. The active links fulfill the already mentioned purpose of recording which parts are supposed to develop in parallel, whereas inactive links simply record all links that have been active at some point during the derivation. In this way, we preserve all links and can later argue about their structure, which will allow us to prove properties about MBOT.

On these forms we now define our derivation steps. A form $\langle \xi, A, I, \zeta \rangle$ derives a form $\langle \xi', A', I', \zeta' \rangle$, written as $\langle \xi, A, I, \zeta \rangle \Rightarrow \langle \xi', A', I', \zeta' \rangle$, if we can select a rule $\langle \ell, q, \vec{r} \rangle$ and an occurrence $v \in \mathrm{pos}(\xi)$ of q in the input tree ξ such that $\vec{r}$ has as many components as there are output positions $A(v) = \{w \mid (v, w) \in A\}$ that are actively linked to v and

- ξ' is obtained from ξ by replacing the occurrence v of q by the tree ℓ,
- A' is obtained from A by removing the used links $\{(v, w) \mid w \in A(v)\}$ and adding the links induced by the rule $\langle \ell, q, \vec{r} \rangle$,
- I' is obtained from I by simply adding the used links $\{(v, w) \mid w \in A(v)\}$, and
- ζ' is obtained from ζ by replacing the linked subtrees $A(v)$ in ζ by $\vec{r}$ (in lexicographic order).

As usual, we apply derivation steps until no active links and no occurrences of states remain. The initial sentential form consists simply of two actively linked initial states (and no disabled links). Since we are interested in the links encountered during the derivation, the set of computed dependencies consists of all $\langle t, I, u \rangle$ such that $\langle q_0, \{(\varepsilon, \varepsilon)\}, \emptyset, q_0 \rangle \Rightarrow^* \langle t, \emptyset, I, u \rangle$, where q_0 is an initial state and t and u are trees, in which no state occurs.

Making additional information from the derivation process (like the links) visible in the computed output has been explored already. In particular, the origin [34,35], which associates to output positions the input position from which this output fragment was created, has been intensively studied. For example, in [12] origin information is used to characterize those macro tree transducers that are MSO definable, and in [25] it is used to get a Myhill–Nerode characterization of deterministic top-down tree transducers. Information on the type of transition used (in a push-down automaton) yields the visibly push-down languages [2], and in the recent work [5] origin information is built into the semantics of a string transducer. However, for us the links are purely a tool, so the computed tree relation is obtained from the computed dependencies simply by projecting onto the input and output trees (i.e., removing the links) as usual. This computed tree relation coincides with the tree relation computed by means of other semantics [4,11,13].

Our goal is to provide generic linking theorems (see Theorems 5 and 6), which given a tree relation with particular properties (essentially it must contain a specific tree relation) predict certain natural links that must be present in a dependency containing a specific tree pair. Roughly speaking, the linking theorems will establish that whenever we preserve an input subtree in the output (i.e., we copy this part of the input tree verbatim to the output), then these occurrences must be linked. This conventional and intuitive wisdom had to be essentially reproved for each particular tree relation under investigation because the typical arguments used (e.g., the fooling technique) require negative information (i.e., information about tree pairs that are *not* in the desired tree relation), which often means that the proof cannot be reused in similar scenarios. Our linking theorems only use positive information (i.e., only the knowledge that certain tree pairs are in the tree relation), so they readily transfer to similar scenarios. We can then use these established links and general properties of dependencies computed by MBOT to show very easily that certain (classic) tree relations cannot be computed by (compositions of) certain subclasses of MBOT.

Before we start with the investigation of the properties of dependencies computed by MBOT, we show an example of a proof utilizing the classic fooling technique. This example proof from [33] shall demonstrate how the traditional proof technique works, so that it can be compared to our later solution using the linking theorems. However, for the linking theorems to be useful, we first need to establish some basic properties that we can use to reason with links. It turns out that the links in each dependency are organized hierarchically [25,30]. More precisely, a set L of links is input hierarchical if for all $(v_1, w_1), (v_2, w_2) \in L$ the condition $v_1 \prec v_2$ implies that $w_2 \not\preceq w_1$ and that there exists a $(v_1, w_1') \in L$ such that $w_1' \preceq w_2$, where $\preceq$ is the prefix order. Moreover, it is strictly input hierarchical if $v_1 \prec v_2$ implies $w_1 \preceq w_2$, and $v_1 = v_2$ implies $w_1 \preceq w_2$ or $w_2 \preceq w_1$. Trivially, if L is strictly input hierarchical, then it is also input hierarchical. Finally, L is (strictly) output hierarchical if L^{-1} is (strictly) input hierarchical. We prove that the links in the dependencies computed by an MBOT are always input hierarchical and strictly output hierarchical (see Theorem 2). If the vector $\vec{r}$ has at most one component for every rule $\langle \ell, q, \vec{r} \rangle$ of an MBOT M, then M is actually a (linear) extended top-down tree transducer with regular look-ahead [33] (for short: XTOP$^{\mathrm{R}}$). The links in the dependencies computed by an XTOP$^{\mathrm{R}}$ are even strictly input