ON TREE TRANSDUCERS FOR PARTIAL FUNCTIONS

Joost ENGELFRIET

Twente University of Technology, Dept. of Applied Mathematics, P.O. Box 217, Enschede, Netherlands

Received June 1977

Tree transducer, regular look-ahead, syntax-directed translation

In [4, Theorem XI.7.1] it is shown that if a partial function can be realized by a (nondeterministic) finite state a-transducer, then it can also be realized by a (deterministic) generalized bimachine (introduced in [11]). Thus the generalized bimachines compute precisely all partial functions which are realizable by a-transducers. It can easily be shown that instead of a generalized bimachine (which roughly consists of two deterministic sequential machines working in opposite directions on the input string) one can also take a deterministic gsm with regular look-ahead, i.e. a deterministic generalized sequential machine whose moves are determined also in the basis of whether or not the rest of the input string belongs to certain regular languages (cf. [3]). In this note we generalize this result to tree transducers [12,9,8,2,5,6] and show that for tree transducers the role of the generalized bimachine is played by the deterministic top-down tree transducer with regular look-ahead (abbreviated to dt'-fst), introduced in [6]. In particular we prove that if a partial function of trees can be realized by any finite sequence of (bottom-up or top-down) nondeterministic tree transducers, then it can also be realized by a dt'-fst. This implies that every string function obtained from a sequence of tree transducers by considering yields, is a generalized syntax-directed translation, as defined in [1].

It seems to be open whether the composition of $n + 1$ tree transducers is more powerful than that of $n$ tree transducers (for ranges this is conjectured in [8]). The above result shows that it is not true for partial functions. Thus, examples to prove the hierarchy proper are necessarily relations which are not functions.

We adopt the notation and terminology of [5] and [6]. We recall that both finite state tree transducers and the tree transformations they realize, are called fst. The composition of relations $R_1$ and $R_2$ is

$$R_1 \circ R_2 = \{(a, c) | (a, b) \in R_1$$

and $(b, c) \in R_2$ for some $b\}.$

and the domain of $R_1$ is

$$\text{dom}(R_1) = \{a | (a, b) \in R_1 \text{ for some } b\}.$$

In the following lemma the basic construction is given.

Lemma. For each top-down fst $T$ there exists a deterministic top-down fst $T'$ with regular look-ahead such that $T' \subseteq T$ and $\text{dom}(T') = \text{dom}(T)$.

Proof. We have to find a dt'-fst $T'$ which for each input tree $t$ computes one of the possible output trees that $T$ produces from $t$. The idea of the proof is simply to order rules of $T$ with the same left-hand side; whenever several of these rules are applicable by $T$ (and this can be checked by regular look-ahead), $T'$ will apply the first of the applicable rules in the given order. The formal construction is as follows.

Let $T = \langle \Sigma, \Delta, Q, Q_d, R \rangle$. We may assume that $Q_d = \{q_0\}$ for some $q_0 \in Q$. For each $p \in Q$, let $T'(p)$ denote the t-fst $\langle \Sigma, \Delta, Q, \{p\}, R \rangle$; thus $T = T(q_0)$. Note that $\text{dom}(T'(p)) \in \text{RECOG}$ [9]. We construct
Let $T' = \langle \Sigma, \Delta, Q, \{q_0\}, R' \rangle$ as follows.

Consider $q \in Q$ and $\sigma \in \Sigma_n$ ($n > 0$), and let

$$q(\sigma(x_1 \ldots x_n)) \rightarrow r_1 \ldots \rightarrow q(\sigma(x_1 \ldots x_n)) \rightarrow r_k$$

with $r_i \in T_\Delta[Q(X_n)]$ be all rules in $R$ with left-hand side $q(\sigma(x_1 \ldots x_n))$ ordered from 1 to $k$. These rules will be changed into rules of $R'$ (with left-hand side $q(\sigma(x_1 \ldots x_n))$) as follows. First we define for each $j$, $1 \leq j \leq k$, a mapping $D_j : X_n \rightarrow \text{RECOG}$ by

$$D_j(x_i) = \cap \{ \text{dom}(T(\sigma)) : \text{occurs in } r_i \}$$

for $1 \leq i < n$ (in particular, if no $p(x_i)$ occurs in $r_i$, then $D_j(x_i) = T_\Sigma$). Thus, if $t_1 \in D_j(x_i)$ for all $1 \leq i < n$, then we know that the computation

$$q(\sigma(t_1 \ldots t_n)) \Rightarrow r_j[t_1, \ldots, t_n]$$

can be finished successfully by $T$ (and vice versa).

Next we define a mapping $D_{\lambda} : X_n \rightarrow \text{RECOG}$ for each boolean $n \times k$ matrix $\lambda$ (i.e. $\lambda_{i,j}$ is either 0 or 1 for all $i$ and $1 \leq j \leq k$). Intuitively $\lambda_{i,j}$ indicates whether or not a tree is in $D_j(x_i)$. Formally, for $1 \leq i < n$, $D_\lambda(x_i)$ is the intersection of

$$\cap \{ D_j(x_i) \mid \lambda_{i,j} = 1, 1 \leq j \leq k \}$$

and

$$\cap \{ D_j(x_i) \mid \lambda_{i,j} = 0, 1 \leq j \leq k \},$$

where $c$ denotes complement w.r.t. $T_\Sigma$. By definition, for each $q \in Q$ and $\sigma \in \Sigma_n$, $R'$ contains all rules (with regular look-ahead) $q(\sigma(x_1 \ldots x_n)) \rightarrow j(\sigma, D_\lambda)$ where $j(\sigma)$ is the first number $j$ ($1 \leq j \leq k$) such that $\lambda_{i,j} = 1$ for all $1 \leq i < n$; if no such number exists, then there is no rule corresponding to $s$. This means that, for given look-ahead $D_\lambda$, $T'$ picks the first rule of $T$ for which it knows that the computation can be finished successfully. Note that for all $t_1, \ldots, t_n \in T_\Sigma$, there exists $s$ such that $t_i \in D_\lambda(x_i)$ for all $i$, $1 \leq i < n$; hence all possibilities are taken into account. Note also that if $s_1 \neq s_2$ then $D_{s_1}$ and $D_{s_2}$ are non-overlapping in the sense that $D_{s_1}(x_i) \cap D_{s_2}(x_i) = \emptyset$ for some $i$, $1 \leq i \leq n$; hence $T'$ is deterministic (see Definition 2.5 of [6]). It should now be clear that $T'$ satisfies the requirements of the lemma. \hfill \Box

Theorem. Let $T_1, \ldots, T_n$ be top-down or bottom-up fst, such that $T_1$ has input alphabet $\Sigma$ and $T_n$ output alphabet $\Delta$. Let $f$ be a mapping from $T_\Delta$ into some set $S$. If $T_1 \circ T_2 \circ \ldots \circ T_n \circ f$ is a partial function (from $T_\Sigma$ into $S$), then there exists a deterministic top-down fst $T_0$ with regular look-ahead such that $T_1 \circ T_2 \circ \ldots \circ T_n \circ f = T_0 \circ f$.

Proof. By the decomposition results of [5] it may be assumed that all $T_i$ are top-down fst. It may also be assumed that each $T_i$ only produces trees that are in the domain of $T_{i+1} \circ \ldots \circ T_n \circ f$. In fact, this domain can be recognized by a finite tree automaton (Lemma 1.2 of [6]) and $T$-FST $\cap T_i$-FST $\subseteq T$-FST (see for instance the proof of Lemma 2.10(1) of [6] for the case of 'trivial' look-ahead).

Let, for $1 \leq i \leq n$, $T'_i$ denote the dt'-fst corresponding to $T_i$ as in the previous lemma. It should be clear from the above assumption and the fact that $T_1 \circ \ldots \circ T_n \circ f$ is a partial function, that

$$T_1 \circ \ldots \circ T_n \circ f = T'_1 \circ \ldots \circ T'_n \circ f.$$

Since $T'_n \subseteq T_{i} \circ T_{i+1} \circ \ldots \circ T_n \circ f$, and hence $T_1 \circ \ldots \circ T_n \circ f = T'_n \circ f$. \hfill \Box

We now discuss some applications of this theorem.

(1) Taking $f$ to be the identity on $T_\Delta$, the theorem shows that $T_{D_\Delta} \cap T_{D_T}$ equals the class of partial functions that can be realized by computations of (bottom-up or top-down) finite state tree transformations. Note that each dt'-fst is the composition of a dtb-fst with a dt-fst (Theorem 2.6 of [6]).

(2) Taking $f$ to be the yield function $T_\Delta \rightarrow \Delta$, the theorem concerns partial tree to string functions.

Consider a partial string to string function $g : \Sigma_\Delta \rightarrow \Delta_\Sigma$ such that

$$g = \{ (\text{yield}(t_1), \text{yield}(t_2)) \mid (t_1, t_2) \in T_1 \circ \ldots \circ T_n \}.$$

for certain fst $T_1, \ldots, T_n$. It clearly follows from the theorem that there exists a dt'-fst $T_0$ such that

$$g = \{ (\text{yield}(t_1), \text{yield}(t_2)) \mid (t_1, t_2) \in T_0 \}.$$

Using a slight variant of Theorem 2.6 of [6] it is easy to show that there exist a recognizable tree language $R$ and a dt-fst $T_0$ such that

$$g = \{ (\text{yield}(t_1), \text{yield}(t_2)) \mid (t_1, t_2) \in R \}.$$

Since each recognizable tree language is the projection of the set of derivation trees of a context-free gram-
mar [12] and since dt-fst are strongly related to the 
generalized syntax-directed translations (GSDT) of
[1], it can now easily be shown that g is a GSDT (cf.
also the remarks on page 442 of [1]). Since the
reverse also holds, this shows that GSDT equals the
class of all partial functions of the form \( \{(\text{yield}(t_1),
\text{yield}(t_2)) \mid (t_1, t_2) \in T_1 \circ \cdots \circ T_n \} \) for 
\( T_1, \ldots, T_n \).

(3) Suppose now that the input trees to \( T_1 \circ \cdots \circ T_n \)
are derivation trees of programs in a programming
language with context-free syntax, and that the output
trees are (structured) programs in some object
code. For each \( t \in T_\Delta \), let \( f(t) \) denote the input/output
behaviour of the object code \( t \) (or any other
desired semantics). Usually one requires that each
input derivation tree has unique semantics, i.e. that
\( T_1 \circ \cdots \circ T_n \circ f \) is a partial function. The theorem
shows the existence of one dt\*fst \( T_0 \) which defines
the same semantics as \( T_1 \circ \cdots \circ T_n \). Thus, as long as
the semantics remains unique, the theorem allows us
to use nondeterminism and an arbitrary number of
passes in the generation of the object code.

(4) E(D)\*OL system [10] may be viewed as
(d)\*fst with monadic input trees (see [7]). Taking \( \Sigma \)
monadic, \( n = 1 \) and \( f = \text{yield} \), the theorem shows that
if an ETOL system has the property that each se-
quence of tables can only generate one string, then it
has an equivalent EDTOL system.

(5) We finally note that if both \( \Sigma \) and \( \Delta \) are mona-
dic ranked alphabets and \( f \) is the identity, then the
theorem shows the fact mentioned at the beginning of
this note: each non-deterministic \( a \)-transducer that
computes a partial function, can be simulated by a
deterministic gsm with regular look-ahead.

Acknowledgement

I thank Giora Slutzki for suggesting the topic of
this paper.

References

[1] A.V. Aho and J.D. Ullman, Translations on a context-
[2] B.S. Baker, Tree transductions and families of tree
extension of LR(k) grammars, JCSS 7 (1973) 66–96.
[4] S. Eilenberg, Automata, Languages and Machines,
[5] J. Engelfriet, Bottom-up and top-down tree transforma-
tions – a comparison, Math. Syst. Theory 9 (1975)
198–231.
[7] J. Engelfriet, Surface tree languages and parallel deriv-
[8] W.F. Ogden and W.C. Rounds, Composition of n trans-
ducers, 4th Symp. on Theory of Computing (1972)
198–206.
[10] G. Rozenberg, Extension of tabled OL-systems and
[12] J.W. Thatcher, Generalized \( a \)-sequential machine maps,