Chapter 1

Tree Transducers

1.1 Deterministic Bottom-up Finite-state Tree Transducers

1.1.1 Orientation

This section is about deterministic bottom-up finite-state tree transducers. The term *finite-state* means that the amount of memory needed in the course of computation is independent of the size of the input. The term *deterministic* means there is single course of action the machine follows to compute its output. The term *transducer* means this machine solves *transformation problem*: given an input object x, what object y is x transformed into? The term *tree* means we are considering the transformation problem from trees to trees. The term *bottom-up* means that for each node a in a tree, the computation transforms the children of a node before transforming the node. This contrasts with *top-down* transducers which transform the children after transforming their parent. Visually, these terms make sense provided the root of the tree is at the top and branches of the tree move downward.

A definitive reference for finite-state automata for trees is freely available online. It is "Tree Automata Techniques and Applications" (TATA) (Comon *et al.*, 2007). The presentation here differs from the one there, as mentionepd below.

1.1.2 Definitions

Recall the definition of trees with a finite alphabet Σ and the symbols [ ] not in Σ. All such trees belonged to the treeset Σ^T. In addition to this, we will need to define a new kind of tree which has *variables* in the leaves. I will call these trees *Variably-Leafed.* We assume a countable set of variables X containing variables $x_1, x_2, \ldots$.

Definition 1 (Variably-Leafed Trees)**.**

Base Cases (Σ): *For each $a \in \Sigma$, $a[\,]$ is a tree.*
Base Cases (X): *For each $x \in X$, $x[\,]$ is a tree.*

Inductive Case: *If $a \in \Sigma$ and $t_1 t_2 \dots t_n$ is a string of trees of length n then $a[t_1 t_2 \cdot \dots t_n]$ is a tree.*

Let $\Sigma^T[X]$ denote the set of all variably-leafed trees of finite size using Σ and X.

Note that $\Sigma^T \subsetneq \Sigma^T[X]$. In the tree transducers we write below, the variably-leafed trees will play a role in the omega function as well as the the intermediate stages of the transformation.

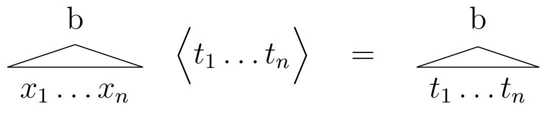
With this definition in place, we can define our first tree transducer.

Definition 2 (DBFTA)**.** *A* Deterministic Bottom-up Finite-state Acceptor (DBFTT) *is a tuple (Q, Σ, F, δ) where*

- *Q is a finite set of states;*
- *Σ is a finite alphabet;*
- *$F \subseteq Q$ is a set of* accepting *(*final*) states; and*
- *$\delta : Q^* \times \Sigma \to Q \times \Sigma^T[X]$ is* the transition function. The pre-image of δ must be finite. *This means we can write it down—for example, as a list.*

  If transition $(\vec{q}, a, r, t) \in \delta$ it means that if the children of node a have been assigned states $\vec{q}$ then the state r will be assigned to a the tree t will be the output employed at this stage in the derivation. It will be helpful to refer to the "first" and "second" outputs of delta with δ_1 and δ_2 respectively. So for all $(\vec{q}, a, r, v) \in \delta$, we have $\delta_1(\vec{q}, a) = r$ and $\delta_2(\vec{q}, a) = t$. These are are the "state" transition and the "output" transitions, respectively.

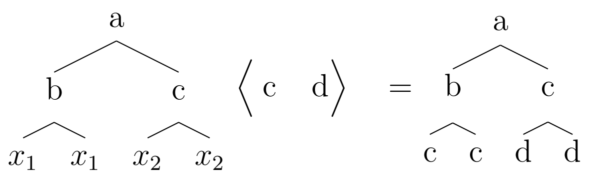
The key to understanding how a tree-to-tree transducer is defined is to understand how variables are used to rewrite and expand output trees. If $t_1 \dots t_n$ is a list of trees and $t_x \in \Sigma^T[X]$ is a variably leafed tree with variables $x_1, \dots x_n$ then let $t_x\langle t_1 \dots t_n \rangle = t \in \Sigma^T$ obtained by replacing each variable x_i in t_x with t_i. Here is a visualization of this notation.



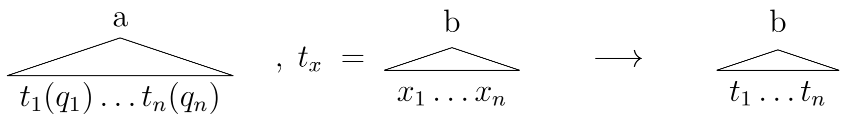
Example 1.

$$[\ b[x_1\ x_1]\ \ c[x_2\ x_2]\ ]\Big\langle t_1 = c[\,], t_2 = d[\,]\Big\rangle = [\ b[c[\,]\ c[\,]]\ \ c[d[\,]\ d[\,]]\ ]$$

Visually, we are taking the tree shown below and substituting t_1 for x_1 and t_2 for x_2.



Now we explain how substitution is used in the process of transducing a tree into another. Given $\delta_2(q_1q_2\dots q_n, a) = t_x$ and a tree $a[t_1t_2\dots t_n]$ with states $q_1q_2\dots q_n$, respectively, then the output tree will be the one obtained by replacing each x_i with t_i in t_x. A schematic of this is shown below.



Then we can define a function "process" $\pi : \Sigma^T \to Q \times \Sigma^T$ which will process the tree and produce its output. It is defined recursively as follows.

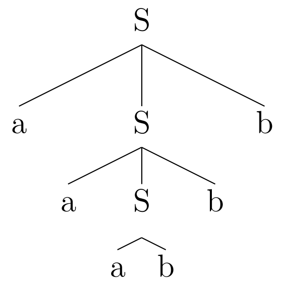
$$\begin{aligned}
\pi(a[\,]) &= \big(\delta_1(\lambda, a), \delta_2(\lambda, a)\big) \\
\pi(a[t_1\cdots t_n]) &= (q, t) \text{ where} \\
&\qquad q = \delta_1\big(q_1\cdots q_n, a\big) \text{ and} \\
&\qquad t = \delta_2\big(q_1\cdots q_n, a\big)\langle s_1\cdots s_n\rangle \text{ and} \\
&\qquad (q_1, s_1)\cdots(q_n, s_n) = \pi(t_1)\cdots\pi(t_n)
\end{aligned} \tag{1.1}$$

Definition 3 (Tree-to-tree function of a DBFTT). *The function defined by the transducer T is $\big\{(t, s) \mid t, s \in \Sigma^T, \pi(t) = (q, s), q \in F\big\}$. If (t, s) belongs to this set, we say T* transduces *t to s and write $T(t) = s$.*

Example 2. Consider the transducer T constructed as follows.

- $Q = \{q_a, q_b, q_S\}$
- $\Sigma = \{a, b, S\}$
- $F = \{q_S\}$
- $\delta_1(\lambda, a) = q_a$
- $\delta_1(\lambda, b) = q_b$
- $\delta_1(q_aq_b, S) = q_S$
- $\delta_1(q_aq_Sq_b, S) = q_S$
- $\delta_2(\lambda, a) = a[\,]$
- $\delta_2(\lambda, b) = b[\,]$
- $\delta_2(q_aq_b, S) = S[x_2x_1]$
- $\delta_2(q_aq_Sq_b, S) = S[x_3x_2x_1]$

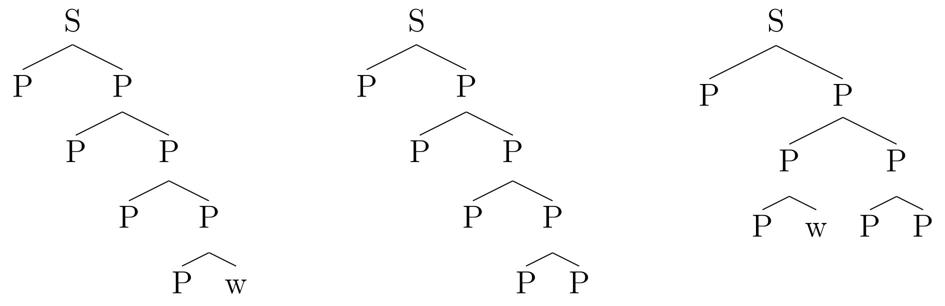
Let us work out what T transforms the tree below into.



Example 3. Consider the transducer T constructed as follows. We let $Q = \{q_w, q_p, q_S\}$, $\Sigma = \{w, P, S\}$, and $F = \{q_S\}$. The transition and ouput functions are given below.

- $\delta(\lambda, P) = q_p, P[\ ]$
- $\delta(\lambda, w) = q_w, w[\ ]$
- $\delta(q_pq_p, P) = q_p, P[x_1x_2]$
- $\delta(q_wq_p, P) = q_w, P[x_1x_2]$
- $\delta(q_pq_w, P) = q_w, P[x_1x_2]$
- $\delta(q_pq_w, S) = q_S, S[\ w[\ ]\ S[x_1x_2]\ ]$
- $\delta(q_wq_p, S) = q_S, S[\ w[\ ]\ S[x_1x_2]\ ]$
- $\delta(q_pq_p, S) = q_S, S[x_1x_2]$

Let us work out how T transforms the trees below.



1.2 Deterministic Top-down Finite-state Tree Transducers

1.2.1 Orientation

This section is about deterministic bottom-up finite-state tree transducers. The term *finite-state* means that the amount of memory needed in the course of computation is independent of the size of the input. The term *deterministic* means there is single course of action the machine follows to compute its output. The term *transducer* means this machine solves *transformation problem*: given an input object x, what object y is x transformed into? The term *tree* means we are considering the transformation problem from trees to trees. The term *top-down* means that for each node a in a tree, the computation transforms the node before transforming its children. This contrasts with *bottom-up* transducers which transform the children before transforming their parent. Visually, these terms make sense provided the root of the tree is at the top and branches of the tree move downward.

A definitive reference for finite-state automata for trees is freely available online. It is "Tree Automata Techniques and Applications" (TATA) (Comon *et al.*, 2007). The presentation here differs from the one there, as mentioned below.

1.2.2 Definitions

As before, we use variably leafed trees $\Sigma^T[X]$.

Definition 4 (DTFTT)**.** *A* Deterministic Top-down Finite-state Acceptor (DTFTT) *is a tuple* $(Q, \Sigma_r, q_0, \delta)$ *where*

- Q *is a finite set of states;*
- Σ *is a finite alphabet;*
- $q_0 \in Q$ *is the initial state; and*
- $\delta : Q \times \Sigma \times \mathbb{N} \to Q^* \times \Sigma^T[X]$ *is* the transition function. *As before, we will derive "state" and "output" transitions, and notate them with* δ_1 *and* δ_2*, respectively. So for all* $(q, a, \vec{r}, t) \in \delta$*, we have* $\delta_1(q, a) = \vec{r}$ *and* $\delta_2(q, a) = t$*.*

We also define the "process" function $\pi : Q \times \Sigma^T \to \Sigma^T$ which will process the tree and produce its output. It is defined as follows.

$$\begin{aligned} \pi(q, a[\,]) &= \delta_2(q, a, 0) \\ \pi(q, a[t_1 \cdots t_n]) &= \delta_2(q, a, n)\big\langle \pi(q_1, t_1) \cdots \pi(q_n, t_n)\big\rangle \\ &\quad \text{where } q_1 \cdots q_n = \delta_1(q, a, n) \end{aligned} \tag{1.2}$$

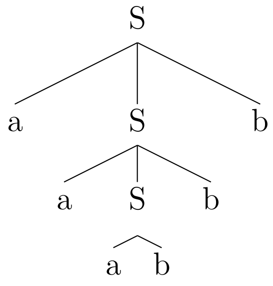
Intuitively, δ_2 transforms the root node into a variably leafed tree. The variables are replaced with the children of the root node. These children are also trees with states assigned by δ_1. Then π transforms each tree-child as well.

Definition 5 (Tree-to-tree function of a DTFTT)**.** *The function defined by the transducer* T *is* $\big\{(t, s) \mid t, s \in \Sigma^T, \pi(q_0, t) = s\big\}$*. If* (t, s) *belongs to this set, we say* M transduces t *to* s *and write* $T(t) = s$*.*

Example 4. Consider the transducer T constructed as follows.

- $Q = \{q, q_S\}$
- $\Sigma = \{a, b, S\}$
- $q_0 = q_S$
- $\delta_1(q, a, 0) = \lambda$
- $\delta_1(q, b, 0) = \lambda$
- $\delta_1(q_S, S, 3) = q q_S q$
- $\delta_1(q_S, S, 2) = qq$
- $\delta_2(q, a, 0) = a[\,]$
- $\delta_2(q, b, 0) = b[\,]$
- $\delta_2(q, S, 3) = S[x_3 x_2 x_1]$
- $\delta_2(q, S, 2) = S[x_2 x_1]$

Let see how T transforms the tree below.



Exercise 1. Recall the "wh-movement" example from before. Explain why this transformation *cannot* be computed by a deterministic top-down tree transducer.

1.3 Theorems about Deterministic Tree Transducers

Theorem 1 (composition closure)**.** *The class of deterministic bottom-up transductions is closed under composition, but the class of top-down deterministic transductions is not.*

Theorem 2 (Incomparable)**.** *The class of deterministic bottom-up transductions is incomparable with the the class of top-down deterministic transductions.*

This theorem is based on the same kind of examples which separated the left and right sequential functions. Let relations $U = \{(f^n a, f^n a) \mid n \in \mathbb{N}\} \cup \{(f^n b, g^n b) \mid n \in \mathbb{N}\}$ and $D = \{(ff^n a, ff^n a) \mid n \in \mathbb{N}\} \cup \{(gf^n a, gf^n b) \mid n \in \mathbb{N}\}$. U is recognized by a DBFTT but not any DTFTT and D is recognized by a DTFTT but not any DBFTT.

Bibliography

Comon, H., M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2007. Tree automata techniques and applications. Available on: `http://tata.gforge.inria.fr/`. Release October, 12th 2007.