Applications of Tree Automata Theory
Lecture V: Theory of Tree Transducers

Andreas Maletti

Institute of Computer Science
Universität Leipzig, Germany

on leave from: Institute for Natural Language Processing
Universität Stuttgart, Germany

maletti@ims.uni-stuttgarg.de

Yekaterinburg — August 25, 2014
Roadmap

1. Theory of Tree Automata
2. Parsing — Basics and Evaluation
3. Parsing — Advanced Topics
5. Theory of Tree Transducers
6. Machine Translation — Advanced Topics

Always ask questions right away!
Motivation

From Extracted Rules to Tree Transducers
Syntax-based Machine Translation

Extracted rules

\[
\begin{align*}
\text{Yugoslav} & \rightarrow_{q_Y} \text{AlywgwsI Afy} \\
\text{Voislav} & \rightarrow_{q_V} \text{fwysI Af} \\
\text{Serbia} & \rightarrow_{q_S} \text{SrbyA} \\
\text{NML}(q_Y, q_P) & \rightarrow_{q_{\text{NML}}} \text{NP}(q_P, q_Y) \\
\text{PP}(q_f, q_{\text{NP}}) & \rightarrow_{q_{\text{PP}}} \text{PP}(q_f, q_{\text{NP}}) \\
\text{NP-SBJ}(q_{\text{NML}}, q_V) & \rightarrow_{q_{\text{NP-SBJ}}} \text{NP-SBJ}(q_{\text{NML}}, \text{NP}(q_V))
\end{align*}
\]

\[
\begin{align*}
\text{President} & \rightarrow_{q_P} \text{Alr}yys \\
\text{for} & \rightarrow_{q_f} \text{En} \\
\text{NP}(q_S) & \rightarrow_{q_{\text{NP}}} \text{NP}(q_S)
\end{align*}
\]
corresponds to two productions

\[ q_{NP-SBJ} \rightarrow NP-SBJ(q_{NML}, q_V) \]

\[ q_{NP-SBJ} \rightarrow NP-SBJ(q_{NML}, NP(q_V)) \]
General idea

Synchronous grammars are essentially two grammars over the same nonterminals whose productions are paired

Convention

same nonterminals are synchronized (or linked) and develop at the same time
Approach

- join two productions $q_1 \rightarrow r_1$ and $q_2 \rightarrow r_2$ to 
  $(q_1, q_2) \rightarrow (r_1, r_2)$
Approach

- join two productions $q_1 \rightarrow r_1$ and $q_2 \rightarrow r_2$ to $(q_1, q_2) \rightarrow (r_1, r_2)$
- demand $q_1 = q = q_2$ for simplicity and write $r_1 \xrightarrow{q} r_2$
**Approach**

- Join two productions $q_1 \rightarrow r_1$ and $q_2 \rightarrow r_2$ to $(q_1, q_2) \rightarrow (r_1, r_2)$
- Demand $q_1 = q = q_2$ for simplicity and write $r_1 \xrightarrow{q} r_2$
- Paired productions develop input and output tree at the same time
From Automata to Transducers

Used rule: \[ q \rightarrow q \]

Next rule: \[
\begin{array}{c}
S \\
\text{CONJ} \\
wq \\
q
\end{array}
\]
From Automata to Transducers

Used rule:

```
q
S
CONJ
wa
```

Next rule:

```
q
S
VP
q1
p
q2
```
From Automata to Transducers

Used rule:

Next rule:

Lecture V: Tree Transducers
From Automata to Transducers

Used rule:

```
S
  q1
  VP
    q2
    V
      saw
```

Next rule:

```
S
  q1
  CONJ
    wa
    V
      q1
      ra’aa
```

```
NP
  DT
    the
  NP
    r
    q1
    r
```
From Automata to Transducers

Used rule:

Next rule:

Lecture V: Tree Transducers
From Automata to Transducers

Used rule:

```
N  r  N
boy  atefl
```

Next rule:

```
NP  r  q_2
DT
the
```

```
S
CONJ
wa
V
ra’aa
NP
atefl
```

```
S
NP
q_2
```
From Automata to Transducers

Used rule:

```
NP  DT  N  V  NP
  the  boy  saw  DT  r
```

Next rule:

```
N  r  N
  door  albab
```
From Automata to Transducers

Used rule:

```
N  r  N
door  r  albab
```

Next rule:

```
S
  CONJ
    wa
    V
      NP
        ra’aa
        N
          atefl
          albab
```
Remarks

- synchronization breaks almost all existing constructions (e.g., the normalization construction)

→ the basic grammar model very important
Remarks

- synchronization breaks almost all existing constructions (e.g., the normalization construction)
  → the basic grammar model very important

Major models

1. linear top-down tree transducer
   - input-side grammar: TA
   - output-side grammar: RTG

2. linear extended top-down tree transducer
   - input-side grammar: RTG
   - output-side grammar: RTG
Top-down Tree Transducers
Synchronous grammar rule:

Top-down tree transducer rule:
variables $X = \{x_0, x_1, \ldots, \} = \{x_i \mid i \in \mathbb{N}\}$
Top-down Tree Transducer

Notation

- variables \( X = \{ x_0, x_1, \ldots, \} = \{ x_i \mid i \in \mathbb{N} \} \)
- unary top-concatenation set \( Q; T \subseteq T_\Sigma(V) \)

\[
Q(T) = \{ q(t) \mid q \in Q, t \in T \}
\]
variables $X = \{x_0, x_1, \ldots, \} = \{x_i \mid i \in \mathbb{N}\}$

unary top-concatenation set $Q; T \subseteq T_\Sigma(V)$

$Q(T) = \{q(t) \mid q \in Q, t \in T\}$

$\text{var}(t) = \{x \in X \mid x \text{ occurs in } t\}$

$t \in T_\Sigma(X)$
Top-down Tree Transducer

Notation

- variables $X = \{x_0, x_1, \ldots, \} = \{x_i \mid i \in \mathbb{N}\}$
- unary top-concatenation

$$Q(T) = \{q(t) \mid q \in Q, t \in T\}$$

- $\text{var}(t) = \{x \in X \mid x \text{ occurs in } t\}$

- $t \in T_\Sigma(X)$ linear if each $x \in \text{var}(t)$ occurs at most once
Top-down Tree Transducer

Definition (THATCHER, 1970)

A top-down tree transducer is a tuple \((Q, \Sigma, \Delta, I, R)\)

- alphabet \(Q\) states
- alphabets \(\Sigma\) and \(\Delta\) input/output symbols
- \(I \subseteq Q\) initial states
- finite set \(R \subseteq Q(\Sigma(X)) \times T_\Delta(Q(X))\) rules
  - \(\text{var}(r) \subseteq \text{var}(\ell)\)
  - \(\ell\) is linear
  - for all \((\ell, r) \in R\)
  - for all \((\ell, r) \in R\)
Top-down Tree Transducer

Example

Mirror-image top-down tree transducer \((Q, \Sigma, \Sigma, Q, R)\) with

- \(Q = \{q\}\)
- \(\Sigma = \{\sigma, \gamma, \alpha\}\)
- the following rules in \(R\)

\[
\begin{align*}
q & \xrightarrow{\gamma} q \\
\gamma & \xrightarrow{x_1} q \\
\sigma & \xrightarrow{x_1 x_2} q q \\
\alpha & \xrightarrow{x_2 x_1} \alpha
\end{align*}
\]
Definition (Derivation)

Sentential forms $\xi, \zeta \in T_\Delta(Q(T_\Sigma))$

$\xi \Rightarrow_M \zeta$

if there exist $\ell \rightarrow r \in R$, position $w \in \text{pos}(\xi)$, substitution $\theta : X \rightarrow T_\Sigma$

- $\xi = \xi[\ell\theta]_w$
- $\zeta = \xi[r\theta]_w$
Top-down Tree Transducer

Example

Lecture V: Tree Transducers
Definition

\[ M = \{\langle t, u \rangle \in T_\Sigma \times T_\Delta \mid \exists q \in I: q(t) \Rightarrow_M^* u\} \]
Top-down Tree Transducer

Definition

\[ M = \{ \langle t, u \rangle \in T_\Sigma \times T_\Delta \mid \exists q \in I: q(t) \Rightarrow^*_M u \} \]

Example

Top-down tree transducer \( N \) with

\[ \{ \langle \sigma(t, u), \sigma(u, t) \rangle \mid t, u \in T_{\{\gamma, \alpha\}} \} \subseteq N \]

\[ q \]
\[ \gamma \]
\[ x_1 \]
\[ \rightarrow \]
\[ q \]
\[ \gamma \]
\[ x_1 \]

\[ q \]
\[ \sigma \]
\[ x_1 \]
\[ \rightarrow \]
\[ q \]
\[ \sigma \]
\[ x_2 \]
\[ x_1 \]

\[ q \]
\[ \alpha \]
\[ \rightarrow \]
\[ \alpha \]
### Definition (Syntactic restrictions)

A transducer \((Q, \Sigma, \Delta, I, R)\) is

- **linear** if \(r\) is linear for every \(\ell \to r \in R\)
- **nondeleting** if \(\text{var}(r) = \text{var}(\ell)\) for every \(\ell \to r \in R\)
- **strict** if \(r \notin Q(X)\) for every \(\ell \to r \in R\)

**Example**

Mirror-image transducer is linear, nondeleting, and strict (lns-TOP)
Top-down Tree Transducer

**Definition (Syntactic restrictions)**

Transducer \((Q, \Sigma, \Delta, I, R)\) is

- **linear** if \(r\) is linear
- **nondeleting** if \(\text{var}(r) = \text{var}(\ell)\)
- **strict** if \(r \not\in Q(X)\)

for every \(\ell \rightarrow r \in R\)

**Example**

Mirror-image transducer is **linear**, **nondeleting**, and **strict**

(Ins-TOP)
### Properties [Engelfriet, 1975]

<table>
<thead>
<tr>
<th>T1</th>
<th>“Copying of an input tree and and processing the copies differently”</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2</td>
<td>“Cannot inspect deleted input tree”</td>
</tr>
</tbody>
</table>
Top-down Tree Transducer

Properties [ENGELFRIET, 1975]

T1 “Copying of an input tree and and processing the copies differently”
T2 “Cannot inspect deleted input tree”

Remark

T2 has been addressed
⇝ top-down tree transducers with regular look-ahead
[ENGELFRIET, 1977]
Regular look-ahead

Can be simulated by allowing un-linked nonterminals on the input side

- these develop without effect on the output
- can generate any regular tree language
Top-down Tree Transducer

Definition (COMP)

\[ \tau \subseteq T_\Sigma \times T_\Delta \text{ and } \tau' \subseteq T_\Delta \times T_\Gamma \]

\[
\tau ; \tau' = \{(s, u) \mid \exists t \in T_\Delta : (s, t) \in \tau, (t, u) \in \tau'\}
\]

Example (Double mirror-image)

\[ N ; N = \text{id} \]

\[
\begin{array}{c}
q \\
\downarrow \\
\gamma \\
\downarrow \\
x_1 \\
\gamma \rightarrow \gamma \\
\downarrow \\
\gamma \\
\downarrow \\
x_1 \\
\end{array} \quad \begin{array}{c}
\gamma \\
\downarrow \\
x_1 \\
\end{array} \quad \begin{array}{c}
q \\
\downarrow \\
\sigma \\
\downarrow \\
x_1 \\
\downarrow \\
x_2 \\
\end{array} \quad \begin{array}{c}
q \\
\downarrow \\
\sigma \\
\downarrow \\
q \\
\downarrow \\
x_2 \\
\end{array} \quad \begin{array}{c}
q \\
\downarrow \\
\alpha \\
\end{array} \quad \begin{array}{c}
q \\
\downarrow \\
\alpha \\
\end{array}
\]
### Notation

- **TOP** = class of tree transformations computable by top-down tree transducers
- **TOP\(^R\)** = class of ... transducers with regular look-ahead
- **x-TOP\(^{(R)}\)** = class of ... transducers with properties \(x\)

### Example

**In-TOP** = class of tree transformations computable by linear and nondeleting top-down tree transducers
Top-down Tree Transducer

composition closure indicated in subscript
Rotations

\[ \text{ROT} = \{ \langle \sigma(\sigma(t_1, t_2), t_3), \sigma(t_1, \sigma(t_2, t_3)) \rangle \mid t_1, t_2, t_3 \in T_{\Sigma} \} \]
Top-down Tree Transducer

Rotations

\[ \text{ROT} = \{ \langle \sigma(\sigma(t_1, t_2), t_3), \sigma(t_1, \sigma(t_2, t_3)) \rangle \mid t_1, t_2, t_3 \in T_\Sigma \} \]

Preservation of regularity (PRES)

Given \( \tau \subseteq T_\Sigma \times T_\Delta \) and \( L \subseteq T_\Sigma \) regular, is \( \tau(L) \) regular?

\[ \tau(L) = \{ u \mid \exists t \in L : (t, u) \in \tau \} \]
## Top-down Tree Transducer

<table>
<thead>
<tr>
<th>Model \ Criterion</th>
<th>ROT</th>
<th>SYM</th>
<th>PRES</th>
<th>PRES⁻¹</th>
<th>COMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ins-TOP</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>In-TOP</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Is-TOP</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X₂</td>
</tr>
<tr>
<td>I-TOP</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X₂</td>
</tr>
<tr>
<td>Is-TOP^R</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>I-TOP^R</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TOP</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>X∞</td>
</tr>
<tr>
<td>TOP^R</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>X∞</td>
</tr>
</tbody>
</table>

(SYM = symmetric)
Tree Transducers

Extended Top-down Tree Transducers
A top-down tree transducer is a tuple $(Q, \Sigma, \Delta, I, R)$

- finite set $Q$ states
- alphabets $\Sigma$ and $\Delta$ input/output symbols
- $I \subseteq Q$ initial states
- finite set $R \subseteq Q(\Sigma(X)) \times T_\Delta(Q(X))$ rules
  - $\text{var}(r) \subseteq \text{var}(\ell)$
  - $\ell$ is linear

for all $(\ell, r) \in R$ for all $(\ell, r) \in R$
An extended top-down tree transducer is a tuple $(Q, \Sigma, \Delta, I, R)$

- finite set $Q$ states
- alphabets $\Sigma$ and $\Delta$ input/output symbols
- $I \subseteq Q$ initial states
- finite set $R \subseteq Q(T_{\Sigma}(X)) \times T_{\Delta}(Q(X))$ rules
  - $\text{var}(r) \subseteq \text{var}(\ell)$ for all $(\ell, r) \in R$
  - $\ell$ is linear for all $(\ell, r) \in R$
Extended Top-down Tree Transducer

Example

\[
\begin{align*}
q &\xrightarrow{x_1} q_{S} \\
S &\xrightarrow{\text{CONJ}} q_{S} \\
&\quad \xrightarrow{x_1} \text{wa-} \\
S' &\xrightarrow{\text{VP}} q_{V} \quad q_{NP} \quad q_{NP} \\
&\quad \xrightarrow{x_1} \text{NP} \quad \text{NP} \\
q_{NP} &\xrightarrow{\text{NP}} \text{NP} \\
&\quad \xrightarrow{x_2} \text{NP} \quad \text{NP} \\
&\quad \xrightarrow{x_3} \text{NP} \quad \text{NP} \\
&\xrightarrow{\text{VT}} \text{NP} \\
&\quad \xrightarrow{\text{NT}} \text{NP} \\
q_{NP} &\xrightarrow{\text{NP}} \text{NP} \\
&\quad \xrightarrow{x_2} \text{NP} \quad \text{NP} \\
&\quad \xrightarrow{x_3} \text{NP} \quad \text{NP} \\
&\xrightarrow{\text{VT}} \text{NP} \\
&\quad \xrightarrow{\text{NT}} \text{NP} \\
q_{V} &\xrightarrow{\text{VT}} \text{VT} \\
&\quad \xrightarrow{\text{NT}} \text{VT} \\
&\xrightarrow{\text{VT}} \text{VT} \\
&\quad \xrightarrow{\text{NT}} \text{VT} \\
\end{align*}
\]
Extended Top-down Tree Transducer

**Definition (same as before)**

Sentential forms \( \xi, \zeta \in T_\Delta(Q(T_\Sigma)) \)

if there exist \( \ell \rightarrow r \in R \), position \( w \in pos(\xi) \), substitution \( \theta : X \rightarrow T_\Sigma \)

- \( \xi = \xi[\ell\theta]_w \)
- \( \zeta = \xi[r\theta]_w \)
Extended Top-down Tree Transducer

Definition (same as before)

Sentential forms $\xi, \zeta \in T_\Delta(Q(T_\Sigma))$

$\xi \Rightarrow_M \zeta$

if there exist $\ell \rightarrow r \in R$, position $w \in \text{pos}(\xi)$, substitution $\theta : X \rightarrow T_\Sigma$

- $\xi = \xi[\ell \theta]_w$
- $\zeta = \xi[r \theta]_w$

$M = \{(t, u) \in T_\Sigma \times T_\Delta \mid \exists q \in I : q(t) \Rightarrow^*_M u\}$
Extended Top-down Tree Transducer

Application of the rule

Lecture V: Tree Transducers
Extended Top-down Tree Transducer

Simulation by top-down tree transducer
Extended top-down tree transducer $(Q, \Sigma, \Delta, I, R)$ is

- **linear**, **nondeleting**, **strict** as before
Extended top-down tree transducer $(Q, \Sigma, \Delta, I, R)$ is
- linear, nondeleting, strict as before
- $\varepsilon$-free if $\ell \notin Q(X)$ for every $\ell \rightarrow r \in R$
Extended Top-down Tree Transducer

Definition (Syntactic restrictions)

Extended top-down tree transducer \((Q, \Sigma, \Delta, I, R)\) is

- **linear**, **nondeleting**, **strict** as before
- **\(\epsilon\)-free** if \(\ell \notin Q(X)\) for every \(\ell \rightarrow r \in R\)

Example

Our example transducer is linear, nondeleting, strict, and \(\epsilon\)-free

\[
\begin{align*}
q \xrightarrow{x_1} q_S \quad q \xrightarrow{x_1} \text{CONJ} & \quad q_S \\
S \xrightarrow{x_1} \text{wa-} & \quad q_S \\
S \xrightarrow{x_1} \text{VP} & \quad q_V \\
V \xrightarrow{x_2} \text{saw} & \quad q_V \\
V \xrightarrow{x_3} \text{ra'aa} & \quad q_V \\
S' \xrightarrow{x_1} & \quad q_{NP} \\
\text{NP} \xrightarrow{x_3} \text{the} & \quad q_{NP} \\
\text{NP} \xrightarrow{x_2} \text{boy} & \quad q_{NP} \\
\text{NP} \xrightarrow{x_1} \text{atefl} & \quad q_{NP} \\
\text{NP} \xrightarrow{x_3} \text{albab} & \quad q_{NP}
\end{align*}
\]
## Extended Top-down Tree Transducer

### Properties [~ et al., 2009]

<table>
<thead>
<tr>
<th>X1</th>
<th>Finite look-ahead</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>Deep attachment of variables</td>
</tr>
<tr>
<td>X3</td>
<td>Infinitely many outputs for one input</td>
</tr>
</tbody>
</table>

Remarks:
- T1 and T2 still apply
- $\mathrm{XTOP} = \text{class notation}$
Extended Top-down Tree Transducer

Properties [∼ et al., 2009]

X1 Finite look-ahead
X2 Deep attachment of variables
X3 Infinitely many outputs for one input

Remarks

- T1 and T2 still apply
- XTOP = class notation
Extended Top-down Tree Transducer

composition closure indicated in subscript
### Extended Top-down Tree Transducer

#### Table: Model vs Criterion

<table>
<thead>
<tr>
<th>Model \ Criterion</th>
<th>ROT</th>
<th>SYM</th>
<th>PRES</th>
<th>PRES&lt;sup&gt;-1&lt;/sup&gt;</th>
<th>COMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ln-TOP</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>l-TOP</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td>l-TOP&lt;sup&gt;R&lt;/sup&gt;</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TOP&lt;sup&gt;R&lt;/sup&gt;</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X&lt;sub&gt;∞&lt;/sub&gt;</td>
</tr>
<tr>
<td>ln&lt;sub&gt;ε&lt;/sub&gt;-XTOP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td>Ins-XTOP</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X&lt;sub&gt;∞&lt;/sub&gt;</td>
</tr>
<tr>
<td>ls&lt;sub&gt;ε&lt;/sub&gt;-XTOP&lt;sup&gt;(R)&lt;/sup&gt;</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td>l&lt;sub&gt;ε&lt;/sub&gt;-XTOP</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X&lt;sub&gt;4&lt;/sub&gt;</td>
</tr>
<tr>
<td>l&lt;sub&gt;ε&lt;/sub&gt;-XTOP&lt;sup&gt;R&lt;/sup&gt;</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X&lt;sub&gt;3&lt;/sub&gt;</td>
</tr>
<tr>
<td>(s)l&lt;sub&gt;ε&lt;/sub&gt;-XTOP&lt;sup&gt;(R)&lt;/sup&gt;</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X&lt;sub&gt;∞&lt;/sub&gt;</td>
</tr>
<tr>
<td>XTOP</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>X&lt;sub&gt;∞&lt;/sub&gt;</td>
</tr>
<tr>
<td>XTOP&lt;sup&gt;R&lt;/sup&gt;</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>X&lt;sub&gt;∞&lt;/sub&gt;</td>
</tr>
</tbody>
</table>
Extended Multi Bottom-up Tree Transducers
Extended Multi Bottom-up Tree Transducer

**Definition (ENGELFRIET et al., 2009)**

An extended multi bottom-up tree transducer $(Q, \Sigma, \Delta, F, R)$

- alphabet $Q$ states
- alphabets $\Sigma$ and $\Delta$ input/output symbols
- $F \subseteq Q$ final states
- finite set $R \subseteq T_{\Sigma \cup Q}(X) \times T_{\Delta \cup Q}(X)$ rules
  - $\text{var}(r) \subseteq \text{var}(\ell)$ for all $(\ell, r) \in R$
  - $\ell$ is linear for all $(\ell, r) \in R$
  - $q \in Q$ occur in $\ell$ only directly above elements of $X$
  - $q \in Q$ occurs in $r$ only at the root
Extended Multi Bottom-up Tree Transducer

Definition (Engelfriet et al., 2009)

An extended multi bottom-up tree transducer \((Q, \Sigma, \Delta, F, R)\)

- alphabet \(Q\) states
- alphabets \(\Sigma\) and \(\Delta\) input/output symbols
- \(F \subseteq Q\) final states
- finite set \(R \subseteq T_{\Sigma \cup Q}(X) \times T_{\Delta \cup Q}(X)\) rules
  - \(\text{var}(r) \subseteq \text{var}(\ell)\)
  - \(\ell\) is linear
  - \(q \in Q\) occur in \(\ell\) only directly above elements of \(X\)
  - \(q \in Q\) occurs in \(r\) only at the root

Properties

- linear, nondeleting, strict, \(\varepsilon\)-free as before
Extended Multi Bottom-up Tree Transducer

Example (Duplication)

Extended multi bottom-up tree transducer \((Q, \Sigma, \Sigma, \{f\}, R)\)

- \(Q = \{q, f\}\) and \(\Sigma = \{\sigma, a, b, e\}\)
- \(R\) contains exactly:

\[
\begin{align*}
e & \rightarrow q \\
e & \rightarrow a \quad a \\
x_1 x_2 & \rightarrow q \quad q \\
x_1 x_2 & \rightarrow b \\
x_1 x_2 & \rightarrow f
\end{align*}
\]
Extended Multi Bottom-up Tree Transducer

Example (Duplication)

Extended multi bottom-up tree transducer \((Q, \Sigma, \Sigma, \{f\}, R)\)

- \(Q = \{q, f\} \) and \(\Sigma = \{\sigma, a, b, e\}\)
- \(R\) contains exactly:

\[
\begin{align*}
Q & \rightarrow a \\
q & \rightarrow a \\
b & \rightarrow b \\
\sigma & \rightarrow f
\end{align*}
\]

Properties

linear, nondeleting, strict, and \(\varepsilon\)-free
Extended Multi Bottom-up Tree Transducer

Rule:

\[
\begin{array}{c}
\sigma \\
q \\
x_1 \\
\sigma \\
p \\
x_2 \\
q \\
x_3 \ x_4 \\
\rightarrow \\
q \\
\delta \\
x_2 \\
x_4 \\
x_3 \\
\end{array}
\]

Derivation:

\[
\begin{array}{c}
t \\
\sigma \\
q \\
\delta \\
p \\
x_1 \\
q \\
x_3 \ x_4 \\
\Rightarrow M \\
\end{array}
\]

\[
\begin{array}{c}
t \\
q \\
u_1 \\
q \\
u_2 \\
\Rightarrow M \\
q \\
u_3 \ u_4 \\
u_2 \\
u_4 \ u_3 \\
u_1 \\
\end{array}
\]
Extended Multi Bottom-up Tree Transducer

Derivation
Extended Multi Bottom-up Tree Transducer

Derivation

Lecture V: Tree Transducers  A. Maletti  ·  40
Extended Multi Bottom-up Tree Transducer

Derivation
Extended Multi Bottom-up Tree Transducer

\[ e \rightarrow q \]
\[ a \]
\[ x_1 \]
\[ x_2 \]
\[ \rightarrow q \]
\[ a \]
\[ x_1 \]
\[ x_2 \]
\[ \rightarrow b \]
\[ b \]
\[ x_1 \]
\[ x_2 \]
\[ \rightarrow f \]
\[ q \]
\[ x_1 \]
\[ x_2 \]

Derivation

\[ a \]
\[ b \]
\[ \Rightarrow M \]
\[ b \]
\[ \Rightarrow M \]
\[ q \]
\[ \Rightarrow M \]
\[ b \]
\[ b \]

Lecture V: Tree Transducers
Extended Multi Bottom-up Tree Transducer

\[
\begin{align*}
  e & \rightarrow q \\
  a & \rightarrow q \\
  b & \rightarrow q \\
  x_1 x_2 & \rightarrow x_1 x_2
\end{align*}
\]

**Derivation**

\[
\begin{align*}
  a \quad b & \Rightarrow M \\
  b \quad b & \Rightarrow M \\
  b \quad q & \Rightarrow M \\
  q \quad b \quad b & \Rightarrow M \\
  a \quad a & \Rightarrow M
\end{align*}
\]
Extended Multi Bottom-up Tree Transducer

\[ e \rightarrow q \]
\[ a \]
\[ b \]
\[ x_1 \]
\[ x_2 \]

\[ q \rightarrow a \]
\[ a \]
\[ x_1 \]
\[ x_2 \]

\[ q \rightarrow b \]
\[ b \]
\[ x_1 \]
\[ x_2 \]

\[ q \rightarrow \sigma \]
\[ f \]
\[ x_1 \]
\[ x_2 \]

Derivation

Lecture V: Tree Transducers
Extended Multi Bottom-up Tree Transducer

Definition

\[ \tau_M = \{(t, u) \in T_\Sigma \times T_\Delta \mid \exists q \in F : t \Rightarrow^*_M q(u)\} \]
Extended Multi Bottom-up Tree Transducer

**Definition**

\[ \tau_M = \{(t, u) \in T_\Sigma \times T_\Delta \mid \exists q \in F: t \Rightarrow^*_M q(u)\} \]

**Example (Duplication)**

It computes \( \{(t, \sigma t t) \mid t \in T_\Sigma\} \)

Its image is not a regular tree language
Extended Multi Bottom-up Tree Transducer

Definition

Extended multi bottom-up tree transducer \((Q, \Sigma, \Delta, F, R)\) is

- **extended bottom-up tree transducer (XBOT)** if \(R \subseteq T_\Sigma(Q(X)) \times Q(T_\Delta(X))\)
- **multi bottom-up tree transducer (MBOT)** if \(\ell \in \Sigma(T_Q(X))\) for all \(\ell \rightarrow r \in R\)
- **bottom-up tree transducer (BOT)** if both conditions hold

Example (Duplication)

\[
\begin{align*}
e & \rightarrow q e e \\
a x_1 x_2 & \rightarrow q a x_1 a x_2 \\
b x_1 x_2 & \rightarrow q b x_1 b x_2 \\
q x_1 x_2 & \rightarrow f \sigma x_1 x_2
\end{align*}
\]
**Extended Multi Bottom-up Tree Transducer**

**Definition**

Extended multi bottom-up tree transducer \((Q, \Sigma, \Delta, F, R)\) is

- **extended bottom-up tree transducer** (XBOT) if
  \[ R \subseteq T_\Sigma(Q(X)) \times Q(T_\Delta(X)) \]

- **multi bottom-up tree transducer** (MBOT) if
  \[ \ell \in \Sigma(T_Q(X)) \text{ for all } \ell \rightarrow r \in R \]

- **bottom-up tree transducer** (BOT) if both conditions hold

**Example (Duplication)**

\[
\begin{align*}
e \rightarrow & \quad q \\
e \rightarrow & \quad a \\
x_1 \quad x_2 & \rightarrow a \quad a \\
x_1 \quad x_2 & \rightarrow b \quad b \\
& \rightarrow f
\end{align*}
\]
Theorem [ENGELFRIET et al., 2009]

\[ I-XTOP^R = I-XBOT \]

Proof.

Standard construction trading input-deletion for output-deletion
see \( I-TOP \subseteq I-BOT \) by [ENGELFRIET ’75]
Theorem [ENGELFRIET et al., 2009]

\[ \text{l-XTOP}^R = \text{l-XBOT} \]

Proof.

Standard construction trading input-deletion for output-deletion see \( \text{l-TOP} \subseteq \text{l-BOT} \) by [ENGELFRIET ’75]
Theorem [ENGELFRIET et al., 2009]

\[ \text{XMBOT} = \text{n-XMBOT} \]
Theorem [ENGELFRIET et al., 2009]

\[ \text{XMBOT} = \text{n-XMBOT} \]

Proof.

- guess subtrees that will be deleted
- process them using look-ahead
Extended Multi Bottom-up Tree Transducer

Theorem [ENGELFRIET et al., 2009]

\[ \text{XMBOT} = \text{n-XMBOT} \]

Proof.

- guess subtrees that will be deleted
- process them using look-ahead

\[ \text{In-XMBOT} \]

\[ \begin{align*}
\text{In-XBOT} & \quad \text{I-XBOT} \\
\text{In-XTOP} & \quad \text{I-XTOP}
\end{align*} \]
Extended Multi Bottom-up Tree Transducer

Theorem [ENGELFRIET et al., 2009]

\[ \text{XMBOT} = \text{n-XMBOT} \]

Proof.

- guess subtrees that will be deleted
- process them using look-ahead

Diagram:

```
  In-XMBOT
   /    \\   
  In-XBOT I-XBOT In-MBOT
 / \   \ /    /     \
In-XTOP I-XTOP I-MBOT
```
Extended Multi Bottom-up Tree Transducer

Theorem [ENGELFRIET et al., 2009]

\[ \varepsilon\text{-XMBOT} = \text{MBOT} \]
Theorem [ENGELFRIET et al., 2009]

$$\varepsilon\text{-XMBOT} = \text{MBOT}$$

Proof.

- decompose large left-hand sides using "multi"-states
- attach finite effect of $$\varepsilon$$-rules
Theorem [ENGELFRIET et al., 2009]

\[ \varepsilon\text{-XMBOT} = \text{MBOT} \]

Proof.

- decompose large left-hand sides using “multi”-states
- attach finite effect of \( \varepsilon \)-rules

```
In-XMBOT
  /\             /\            /\        
In-XBOT  I-XBOT In-MBOT   I-XBOT
  /\   /\        /\  /\    
In-XTOP I-XTOP I-MBOT
```
Extended Multi Bottom-up Tree Transducer

Theorem [ENGELFRIET et al., 2009]

\[ \varepsilon\text{-XMBOT} = \text{MBOT} \]

Proof.
- decompose large left-hand sides using “multi”-states
- attach finite effect of \( \varepsilon \)-rules
### Definition

A tree transformation \( \tau \) is **sensible** if 
\[
|\text{pos}(u)| \in \mathcal{O}(|\text{pos}(t)|) \quad \text{for all} \quad (t, u) \in \tau \quad \text{(linear i-o size relation)}
\]
## Extended Multi Bottom-up Tree Transducer

### Definition

A tree transformation \( \tau \) is **sensible** if \( |\text{pos}(u)| \in O(|\text{pos}(t)|) \) for all \((t, u) \in \tau\) (linear i-o size relation).

### Theorem \([\sim, 2012]\)

**sensible XTOP \( \subseteq \text{In-MBOT} \)**
Definition

tree transformation $\tau$ sensible if $|\text{pos}(u)| \in O(|\text{pos}(t)|)$ for all $(t, u) \in \tau$ (linear i-o size relation)

Theorem [~, 2012]

sensible XTOP $\subseteq$ ln-MBOT

Proof.

- use construction of [ENGELFRIET, MANETH, 2003]
- obtain finitely copying $\varepsilon$-XTOP
- apply [ENGELFRIET et al., 2009] to obtain $l\varepsilon$-XMBOT
- previous theorems yield ln-MBOT
Corollary

All SMT-relevant extended top-down tree transducers can be simulated by linear and nondeleting extended multi bottom-up tree transducers
Theorem

\[ \text{In-MBOT} \not\subseteq \text{XTOP}^R \]
Extended Multi Bottom-up Tree Transducer

Theorem

\[ \text{In-MBOT} \not\subseteq \text{XTOP}^R \]

Theorem [GILDEA, 2012]

\[ \text{yd}_{\text{out}}(\text{In-MBOT}) = \text{LCFRS} \]
## Summary

<table>
<thead>
<tr>
<th>Model \ Criterion</th>
<th>ROT</th>
<th>SYM</th>
<th>PRES</th>
<th>PRES$^{-1}$</th>
<th>COMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ln-TOP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>I-TOP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X_2</td>
</tr>
<tr>
<td>I-TOP^R</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TOP^R</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X_∞</td>
</tr>
<tr>
<td>Ins_ε-XTOP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X_2</td>
</tr>
<tr>
<td>Ins-XTOP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X_∞</td>
</tr>
<tr>
<td>Is_ε-XTOP^R</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓_2</td>
</tr>
<tr>
<td>l_ε-XTOP</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X_4</td>
</tr>
<tr>
<td>l_ε-XTOP^R</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X_3</td>
</tr>
<tr>
<td>(s)l-XTOP^R</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X_∞</td>
</tr>
<tr>
<td>XTOP^R</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X_∞</td>
</tr>
<tr>
<td>l(n)-XMBOT</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XMBOT</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X_∞</td>
</tr>
<tr>
<td>reg.-preserving l-XMBOT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>invertible l-XMBOT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Selected references

**Arnold, Dauchet**: *Morphismes et Bimorphismes d’Arbres*

**Engelfriet**: *Bottom-up and Top-down Tree Transformations — A Comparison.*
Math. Systems Theory 9, 1975

**Engelfriet, Maneth**: *Macro Tree Translations of Linear Size Increase are MSO Definable.*

**Engelfriet, Lilin, ∼**: *Extended Multi Bottom-up Tree Transducers — Composition and Decomposition.*
Acta Inf. 46, 2009

**Rounds**: *Mappings and Grammars on Trees*
Math. Systems Theory 4, 1970

**Thatcher**: *Generalized\(^2\) Sequential Machine Maps*