A Note on Sequential Rule-Based POS Tagging
Sylvain Schmitz

To cite this version:
Sylvain Schmitz. A Note on Sequential Rule-Based POS Tagging. FSMNLP, 2011, Blois, France. pp.83–87. hal-00600260v2

HAL Id: hal-00600260
https://hal.science/hal-00600260v2
Submitted on 21 Nov 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
A Note on Sequential Rule-Based POS Tagging

Sylvain Schmitz

LSV, ENS Cachan & CNRS, Cachan, France
schmitz@lsv.ens-cachan.fr

Abstract

Brill’s part-of-speech tagger is defined through a cascade of leftmost rewrite rules. We revisit the compilation of such rules into a single sequential transducer given by Roche and Schabes (Comput. Ling. 1995) and provide a direct construction of the minimal sequential transducer for each individual rule.

Keywords. Brill Tagger; Sequential Transducer; POS Tagging

1 Introduction

Part-of-speech (POS) tagging consists in assigning the appropriate POS tag to a word in the context of its sentence. The program that performs this task, the POS tagger, can be learned from an annotated corpus in case of supervised learning, typically using hidden Markov model-based or rule-based techniques. The most famous rule-based POS tagging technique is due to Brill (1992). He introduced a three-parts technique comprising:

1. a lexical tagger, which associates a unique POS tag to each word from an annotated training corpus. This lexical tagger simply associates to each known word its most probable tag according to the training corpus annotation, i.e. a unigram maximum likelihood estimation;

2. an unknown word tagger, which attempts to tag unknown words based on suffix or capitalization features. It works like the contextual tagger, using the presence of a capital letter and bounded sized suffixes in its rules: for instance in English, a -able suffix usually denotes an adjective;

3. a contextual tagger, on which we focus in this paper. It consists of a cascade of string rewrite rules, called contextual rules, which correct tag assignments based on some surrounding contexts.

In this note, we revisit the proof that contextual rules can be translated into sequential transducers proposed by Roche and Schabes (1995): whereas Roche and Schabes give a separate proof of sequentiality and exercise it to show that their constructed non-sequential transducer can be determinized (at the

Historically, what we call here “sequential” used to be called “subsequential” (Schützenberger 1977), but we follow the more recent practice initiated by Sakarovitch (2009).
expense of a worst-case exponential blow-up), we give a direct translation of a
costual rule into the minimal normalized sequential transducer, by adapt-
ing Simon (1994)'s string matching automaton to the transducer case. Our
resulting sequential transducers are of linear size (before their composition). A
similar construction can be found in Mihov and Schultz (2007), but no claim
of minimality is made there.

2 Contextual Rules

2.1 Example

We start with an example by Roche and Schabes (1995): Let us suppose the
following sentences were tagged by the lexical tagger (using the Penn Treebank
tagset):

* Chapman/NNP killed/VBN John/NNP Lennon/NNP
* John/NNP Lennon/NNP was/VBD shot/VBD by/IN Chapman/NNP
  He/PRP witnessed/VBD Lennon/NNP killed/VBN by/IN Chapman/NNP

There are mistakes in the first two sentences: killed should be tagged as a past
tense form “VBD”, and shot as a past participle form “VBN”.

The contextual tagger learns contextual rules over some tagset Σ of form
uav → ubv (or a → b | u | v using phonological rule notations Kaplan and Kay,
1994), meaning that the tag a rewrites to b in the context of u | v, where the
context is of length |uv| bounded by some fixed k + 1; in practice, k = 2 or k = 3
(Brill 1992 and Roche and Schabes 1995) use slightly different templates than
the one parametrized by k we present here). For instance, a first contextual rule
could be

  np vbn → np vbd

resulting in a new tagging

  Chapman/NNP killed/VBD John/NNP Lennon/NNP
  * John/NNP Lennon/NNP was/VBD shot/VBD by/IN Chapman/NNP
  * He/PRP witnessed/VBD Lennon/NNP killed/VBN by/IN Chapman/NNP

A second contextual rule could be

  vbd in → vbn in

resulting in the correct tagging

  Chapman/NNP killed/VBD John/NNP Lennon/NNP
  John/NNP Lennon/NNP was/VBD shot/VBN by/IN Chapman/NNP
  He/PRP witnessed/VBD Lennon/NNP killed/VBN by/IN Chapman/NNP

As stated before, our goal is to compile the entire sequence of contextual rules
learned from a corpus into a single sequential function.

2.2 Cascade of Contextual Rules

Let us first formalize the semantics we will employ in this note for Brill's contex-
tual rules. Let C = r1r2 · · · rn be a finite sequence of string rewrite rules in

2This is not exactly the semantics assumed by either Brill nor Roche and Schabes who
used iterated-application semantics, resp. contextual and non contextual, instead of the single-
application semantics we use here. This has little practical consequence.

2
\[ \Sigma^* \times \Sigma^* \] with \( \Sigma \) a POS tagset of fixed size. In practice the rules constructed in Brill’s contextual tagger are length-preserving and 1-change-bounded, i.e. they modify a single letter, but this is not a useful consideration for our transducer construction. Each rule \( r_i = u_i \rightarrow v_i \) defines a leftmost rewrite relation \( \overset{r_i}{\Rightarrow} \) defined by

\[ w \overset{r_i}{\Rightarrow} w' \text{ iff } \exists x, y \in \Sigma^*, w = xu_iy \land w' = xv_iy \]

\[ \land \forall z, z' \in \Sigma^*, w \neq zu_i'z' \land x \leq_{\text{pref}} z \]

where \( x \leq_{\text{pref}} z \) denotes that \( x \) is a prefix of \( z \). Note that the domain of \( \overset{r_i}{\Rightarrow} \) is \( \Sigma^* \cdot u_i \cdot \Sigma^* \). The behavior of a single rule is then the relation \( [r_i] \) included in \( \Sigma^* \times \Sigma^* \) defined by

\[ [r_i] \overset{\text{def}}{=} \overset{r_i}{\Rightarrow} \cup \text{Id}_{\Sigma^* \setminus (\Sigma^* \cdot u_i \cdot \Sigma^*)}, \]

i.e. it applies \( \overset{r_i}{\Rightarrow} \) on \( \Sigma^* \cdot u_i \cdot \Sigma^* \) and the identity on its complement \( \Sigma^* \setminus (\Sigma^* \cdot u_i \cdot \Sigma^*) \).

The behavior of \( C \) is then the composition

\[ [C] \overset{\text{def}}{=} [r_1] \circ [r_2] \circ \cdots \circ [r_n]. \]

Note that this behavior does not employ the transitive closure of the rewriting rules.

A naive implementation of \( C \) would try to match each \( u_i \) at every position of the input string \( w \) in \( \Sigma^* \), resulting in an overall complexity of \( O(|w| \cdot \sum_i |u_i|) \).

One often faces the problem of tagging a set of sentences \( \{u_1, \ldots, u_m\} \), which yields \( O((\sum_i |u_i|) \cdot (\sum_j |w_j|)) \). As shown in Roche and Schabes’ experiments, compiling \( C \) into a single sequential transducer \( T \) results in huge savings, with overall complexities in \( O(|w| + |T|) \) and \( O(|T| + \sum_j |w_j|) \) respectively.

Each \( [r_i] \) is a rational function, being the union of two rational functions over disjoint domains. Let \( |r_i| \) be the length \( |u_i|+|v_i| \leq k \). Roche and Schabes (1995, Section 8.2) provide a construction of an exponential-sized transducer \( T_{r_i} \) for each \( [r_i] \), and compute their composition \( T_C \) of size \( |T_C| = O(\prod_{i=1}^n 2^{|r_i|}) \). As they show that each \( [r_i] \) is actually a sequential function, their composition \([C]\) is also sequential, and \( T_C \) can be determinized to yield a sequential transducer \( T \) of size doubly exponential in \( \sum_{i=1}^n |r_i| \leq nk \) (see Roche and Schabes 1995, Section 9.3). By contrast, our construction directly yields linear-sized minimal sequential transducers for each \( [r_i] \), resulting in a final sequential transducer of size \( O(\prod_{i=1}^n |r_i|) = O(2^n \log k) \).

### 3 Sequential Transducer of a Rule

Intuitively, the sequential transducer for \([r_i]\) is related to the string matching automaton (Simon 1994; Crochemore and Hancart 1997) for \( u_i \), i.e. the automaton for the language \( \Sigma^* u_i \). This insight yields a direct construction of the minimal sequential transducer of a contextual rule, with at most \( |u_i| + 1 \) states. Let us recall a few definitions:
3.1 Preliminaries

Overlaps, Borders (see e.g. Crochemore and Hancart 1997, Section 6.2). The overlap \( ov(u, v) \) of two words \( u \) and \( v \) is the longest suffix of \( u \) which is simultaneously a prefix of \( v \).

\[
\begin{align*}
\text{ov}(u,v) & \quad \text{u} & \quad \text{v} \\
\end{align*}
\]

A word \( u \) is a border of a word \( v \) if it is both a prefix and a suffix of \( v \), i.e. if there exist \( v_1, v_2 \) in \( \Sigma^* \) such that \( v = uv_1 = v_2u \). For \( v \neq \varepsilon \), the longest border of \( v \) different from \( v \) itself is denoted \( \text{bord}(v) \).

\[
\begin{align*}
\text{bord}(v) & \quad v & \quad v_1 & \quad v_2 \\
\end{align*}
\]

**Fact 3.1.** For all \( u, v \) in \( \Sigma^* \) and \( a \) in \( \Sigma \),

\[
\text{ov}(ua, v) = \begin{cases} 
\text{ov}(u, v) \cdot a & \text{if } \text{ov}(u, v) \cdot a \leq \text{pref } v \\
\text{bord}(\text{ov}(u, v) \cdot a) & \text{otherwise}.
\end{cases}
\]

**Sequential Transducers** (see e.g. Sakarovitch 2009, Section V.1.2). Formally, a sequential transducer from \( \Sigma \) to \( \Delta \) is a tuple \( T = \langle Q, \Sigma, \Delta, q_0, \delta, \eta, \iota, \rho \rangle \) where \( \delta : Q \times \Sigma \to Q \) is a partial transition function, \( \eta : Q \times \Sigma \to \Delta^* \) a partial transition output function with the same domain as \( \delta \), i.e. \( \text{dom}(\delta) = \text{dom}(\eta) \), \( \iota \in \Delta^* \) is an initial output, and \( \rho : Q \to \Delta^* \) is a partial final output function. \( T \) defines a partial sequential function \( \mathcal{J}_T : \Sigma^* \to \Delta^* \) with

\[
\mathcal{J}_T(w) \overset{\text{def}}{=} \iota \cdot \eta(q_0, w) \cdot \rho(\delta(q_0, w))
\]

for all \( w \) in \( \Sigma^* \) for which \( \delta(q_0, w) \) and \( \rho(\delta(q_0, w)) \) are defined, where \( \eta(q, \varepsilon) = \varepsilon \) and \( \eta(q, wa) = \eta(q, w) \cdot \eta(\delta(q, w), a) \) for all \( w \) in \( \Sigma^* \) and \( a \) in \( \Sigma \).

Let us note \( T(q) \) for the sequential transducer with \( q \) for initial state. We write \( u \wedge v \) for the longest common prefix of strings \( u \) and \( v \); the longest common prefix of all the outputs from state \( q \) can be written formally as \( \bigwedge_{v \in \Sigma^*} [T(q)](v) \).

A sequential transducer is normalized if this value is \( \varepsilon \) for all \( q \in Q \) such that \( \text{dom}([T(q)]) \neq \emptyset \), i.e. if the transducer outputs symbols as soon as possible; any sequential transducer can be normalized.

The translation of a sequential function \( f \) by a word \( w \) in \( \Sigma^* \) is the sequential function \( w^{-1}f \) with

\[
\text{dom}(w^{-1}f) \overset{\text{def}}{=} w^{-1}\text{dom}(f) \quad w^{-1}f(u) \overset{\text{def}}{=} \left( \bigwedge_{v \in \Sigma^*} f(wv) \right)^{-1} \cdot f(wu)
\]

for all \( u \) in \( \text{dom}(w^{-1}f) \). As in the finite automata case where minimal automata are isomorphic with residual automata, the minimal sequential transducer for a sequential function \( f \) is defined as the translation transducer \( \langle Q, \Sigma, \Delta, q_0, \delta, \eta, \iota, \rho \rangle \), where
• \( Q \stackrel{\text{def}}{=} \{ w^{-1}f \mid w \in \Sigma^* \} \) (which is finite),
• \( q_0 \stackrel{\text{def}}{=} \varepsilon^{-1}f \),
• \( \iota \stackrel{\text{def}}{=} \bigwedge_{w \in \Sigma^*} f(v) \) if \( \text{dom}(f) \neq \emptyset \) and \( \iota = \varepsilon \) otherwise,
• \( \delta(w^{-1}f,a) \stackrel{\text{def}}{=} (wa)^{-1}f \),
• \( \eta(w^{-1}f,a) \stackrel{\text{def}}{=} \bigwedge_{v \in \Sigma^*} (w^{-1}f)(av) \) if \( \text{dom}((wa)^{-1}f) \neq \emptyset \) and \( \eta(w^{-1}f,a) = \varepsilon \) otherwise, and
• \( \rho(w^{-1}f) \stackrel{\text{def}}{=} (w^{-1}f)(\varepsilon) \) if \( \varepsilon \in \text{dom}(w^{-1}f) \), and is otherwise undefined.

### 3.2 Main Construction

Here is the definition of our transducer for a contextual rule:

**Definition 3.2 (Transducer of a Contextual Rule).** The sequential transducer \( T_r \) associated with a contextual rule \( r = u \rightarrow v \) with \( u \neq \varepsilon \) is defined as

\[
T_r \stackrel{\text{def}}{=} \langle \text{pref}(u), \Sigma, \Sigma, \varepsilon, \delta, \eta, \varepsilon, \rho \rangle
\]

with the set of prefixes of \( u \) as state set, \( \varepsilon \) as initial state and initial output, and for all \( a \in \Sigma \) and \( w \in \text{pref}(u) \),

\[
\delta(w,a) \stackrel{\text{def}}{=} \begin{cases} wa & \text{if } wa \leq_{\text{pref}} u \\
w & \text{if } w = u \\
\text{bord}(wa) & \text{otherwise} \end{cases}
\]

\[
\rho(w) \stackrel{\text{def}}{=} \begin{cases} \varepsilon & \text{if } w \leq_{\text{pref}} (u \land v) \\
(u \land v)^{-1}w & \text{if } (u \land v) <_{\text{pref}} w <_{\text{pref}} u \\
\varepsilon & \text{otherwise, i.e. if } w = u \\
a & \text{if } wa \leq_{\text{pref}} (u \land v) \\
\varepsilon & \text{if } (u \land v) <_{\text{pref}} wa <_{\text{pref}} u \\
(u \land v)^{-1}v & \text{if } wa = u \\
a & \text{if } w = u \\
\rho(w)a \cdot \rho(\text{bord}(wa))^{-1} & \text{otherwise.} \end{cases}
\]

For instance, the sequential transducer for the rule \( abab \rightarrow abbbb \) is shown in **Figure 1** (one can check that \( abab \land abbbb = ab \), \( \text{bord}(b) = \varepsilon \), \( \text{bord}(aa) = a \), \( \text{bord}(abbb) = \varepsilon \), \( \text{bord}(abaa) = a \), and \( \text{bord}(ababa) = aba \)). The intuition behind the definition of \( \eta(w,a) \) is to decompose the rewriting according to \( u \rightarrow v \) into four phases:

1. while in the common prefix \( u \land v \) of \( u \) and \( v \), implement the identity function (states \( \varepsilon \), \( a \), and \( ab \) in **Figure 1**),
2. as soon as we start reading a symbol of \( u \) that does not match that of \( v \) (upon reading \( a \) in state \( ab \) in **Figure 1**), we stop outputting symbols and wait for the whole of \( u \) to be read,
3. if \( u \) has been read, we output the remaining rewritten string \( (u \land v)^{-1} \cdot v \) we had been saving (upon reading \( b \) in state \( abab \) in **Figure 1**).
Figure 1: The sequential transducer constructed for $ababb \rightarrow abbbb$.

4. after having read the first occurrence of $u$ in full, we merely implement the identity again (state $ababb$ in Figure 1).

5. If on the other hand we realize that $u$ cannot be read after all in some state $w$ upon reading some $a$ (e.g. transition on $a$ in state $aba$ in Figure 1), we need to flush the missing output $(u \land v)^{-1} \cdot w = \rho(w)$, minus the saved output if the state we reach is itself in phase 2.

It remains to show that this sequential transducer is indeed the minimal normalized sequential transducer for $[r]$.

**Proposition 3.3** (Correctness). Let $r = u \rightarrow v$ with $u \neq \varepsilon$. Then $[\mathcal{T}_r] = [r]$.

**Proof.** Let us first consider the case of input words in $\Sigma^* \setminus (\Sigma^* \cdot u \cdot \Sigma^*)$:

**Claim 3.3.1.** For all $w$ in $\Sigma^* \setminus (\Sigma^* \cdot u \cdot \Sigma^*)$,

$$
\delta(\varepsilon, w) = ov(w, u) \quad \eta(\varepsilon, w) = w \cdot \rho(ov(w, u))^{-1}.
$$

**Proof of the claim.** By induction on $w$: since $u \neq \varepsilon$, the base case is $w = \varepsilon$ with

$$
\delta(\varepsilon, \varepsilon) = \varepsilon = ov(\varepsilon, u) \quad \eta(\varepsilon, \varepsilon) = \varepsilon = \varepsilon \cdot \varepsilon^{-1} = \varepsilon \cdot \rho(\varepsilon)^{-1}.
$$

For the induction step, we consider $wa$ in $\Sigma^* \setminus (\Sigma^* \cdot u \cdot \Sigma^*)$ for some $w$ in $\Sigma^*$ and $a$ in $\Sigma$:

$$
\delta(\varepsilon, wa) = \delta(\delta(\varepsilon, w), a) \quad \delta(\varepsilon, wa, u) = \delta(\delta(w, u), a) \quad (\text{by def.})
$$

where by definition of $\delta$, we have $\delta(ov(w, u), a) = ov(w, u) \cdot a$ if $ov(w, u) \cdot a \preceq \text{pref } u$ and $\delta(ov(w, u), a) = \text{bord}(ov(w, u) \cdot a)$ otherwise (the case $ov(w, u) = u$ is impossible since $w$ in $\Sigma^* \setminus (\Sigma^* \cdot u \cdot \Sigma^*)$). In all cases:

$$
\delta(\varepsilon, wa) = ov(wa, u) \quad (\text{by Fact 3.1})
$$

$$
\eta(\varepsilon, wa) = \eta(\varepsilon, w) \cdot \eta(\delta(\varepsilon, w), a) \quad (\text{by def.})
$$

$$
= w \cdot \rho(\delta(\varepsilon, w))^1 \cdot \eta(\delta(\varepsilon, w), a) \quad (\text{by ind. hyp.})
$$

$$
= w \cdot \rho(w')^{-1} \cdot \eta(w', a) ; \quad (\text{by setting } w' = \delta(\varepsilon, w))
$$

we need to do a case analysis for this last equation:

**Case** $w' a \not\preceq \text{pref } u$. Then $\eta(w', a) = \rho(w') \cdot a \cdot \rho(\text{border}(w'a))^{-1}$, which yields

$$
\eta(\varepsilon, wa) = w \cdot \rho(w')^{-1} \cdot \rho(w') \cdot a \cdot \rho(\delta(\varepsilon, wa))^{-1} \quad (\text{by Fact 3.1})
$$

$$
= wa \cdot \rho(\delta(\varepsilon, wa))^{-1}.
$$
Case $w'a <_{\text{pref}} u$. Then $\delta(\varepsilon, wa) = w'a$, and we need to further distinguish between several cases:

- $w'a <_{\text{pref}} (u \land v)$ then $\rho(w') = \varepsilon$, $\eta(w', a) = a$, and $\rho(w'a) = \varepsilon$;
- $w' = (u \land v)$ then $\rho(w') = \varepsilon$, $\eta(w', a) = \varepsilon$, and $\rho(w'a) = (u \land v)^{-1} \cdot w'a$;
- $(u \land v) <_{\text{pref}} w'$ then $\rho(w') = (u \land v)^{-1} \cdot w'$, $\eta(w', a) = \varepsilon$, and $\rho(w'a) = (u \land v)^{-1} \cdot w'a$.

Thus, by definition of $\delta$, the claim implies that

$$\delta(\varepsilon, w) = u' \quad \eta(\varepsilon, w) = w \cdot \rho(u')^{-1}.$$  

Thus, if $(u \land v) <_{\text{pref}} u'$

$$\eta(\varepsilon, wa) = w \cdot ((u \land v)^{-1} \cdot u')^{-1} \cdot (u \land v)^{-1} \cdot v = w \cdot u'^{-1} \cdot v = wa \cdot u^{-1} \cdot v.$$  

Otherwise i.e. if $u' = (u \land v)$:

$$\eta(\varepsilon, wa) = w \cdot u'^{-1} \cdot v = wa \cdot u^{-1} \cdot v.$$  

Thus in all cases $\llbracket T_r \rrbracket (wa) = \llbracket r \rrbracket (wa)$, and since $T_r$ starting in state $u$ (i.e. $T_r(u)$) implements the identity over $\Sigma^*$, we have more generally $\llbracket T_r \rrbracket = \llbracket r \rrbracket$.

**Lemma 3.4** (Normality). Let $r = u \rightarrow v$. Then $T_r$ is normalized.

**Proof.** Let $w \in \text{Prefix}(u)$ be a state of $T_r$; let us show that $\mathcal{A}[\llbracket T_r(u) \rrbracket (\Sigma^*)] = \varepsilon$.

If $(u \land v) <_{\text{pref}} w <_{\text{pref}} u$ let $u' = w^{-1} \cdot u \in \Sigma^+$, and consider the two outputs

$$\llbracket T_r(u) \rrbracket (u') = \eta(w, u') \rho(u) = (u \land v)^{-1} \cdot v$$

$$\llbracket T_r(u) \rrbracket (\varepsilon) = \rho(w) = (u \land v)^{-1} \cdot w.$$  

Since $(u \land v) <_{\text{pref}} u$ we can write $u$ as $(u \land v)au''a'$, and either $v = (u \land v)be'$ or $v = u \land v$, for some $a \neq b$ in $\Sigma$ and $u''$, $v'$ in $\Sigma^*$; this yields $w = (u \land v)au''$ and thus $\llbracket T_r(u) \rrbracket (u') \land \llbracket T_r(u) \rrbracket (\varepsilon) = \varepsilon.$
otherwise $\rho(w) = \varepsilon$, which yields the lemma. \hfill \Box

**Proposition 3.5** (Minimality). Let $r = u \rightarrow v$ with $u \neq \varepsilon$ and $u \neq v$. Then $T_r$ is the minimal sequential transducer for $[r]$.

**Proof.** Let $w <_{\text{pref}} w'$ be two different states in $\text{Prefix}(u)$; we proceed to prove that $[w^{-1}T_r] \neq [w'^{-1}T_r]$, hence that no two states of $T_r$ can be merged. By Lemma 3.3 it suffices to prove that $[T_r(w)] \neq [T_r(w')]$, thus to exhibit some $x \in \Sigma^*$ such that $[T_r(w)](x) \neq [T_r(w')](x)$. We perform a case analysis:

if $w \leq_{\text{pref}} (u \land v)$ then $w <_{\text{pref}} (u \land v)$ thus $[T_r(w)](x) = x$ for all $x \notin w^{-1}$. $\Sigma^* \cdot u \cdot \Sigma^*$, consider

$$[T_r(w)](w'^{-1}u) = w'^{-1}u \neq w'^{-1}v = [T_r(w')](w'^{-1}u);$$

if $w \leq_{\text{pref}} (u \land v)$ and $w' = u$ then $[T_r(w')](x) = x$ for all $x$ and we consider

$$[T_r(w)](w^{-1}u) = w^{-1}v \neq w^{-1}v = [T_r(w')](w^{-1}u);$$

otherwise that is if $w \leq_{\text{pref}} (u \land v)$ and $(u \land v) <_{\text{pref}} w' \leq_{\text{pref}} u$, or $(u \land v) <_{\text{pref}} w \leq_{\text{pref}} w' \leq_{\text{pref}} u$, we have $\rho(w) \neq \rho(w')$ thus

$$[T_r(w)](\varepsilon) \neq [T_r(w')](\varepsilon).$$ \hfill \Box

**4 Conclusion**

The results of the previous section yield (the cases $u = \varepsilon$ and $u = v$ are trivial):

**Theorem 4.1.** Given a contextual rule $r = u \rightarrow v$, one can construct directly the minimal normalized sequential transducer $T_r$ of size $O(|r|)$ for $[r]$.

The remaining question is whether we can obtain better upper bounds on the size of the sequential transducer $T$ for a cascade $C = r_1 \cdots r_n$ than $O(2^{n \log k})$. It turns out that there are cascades of length $n$ for which no sequential transducer with a subexponential (in $n$) number of states can exist, thus our construction is close to optimal.

**References**


