Errata to Grammatical Inference: Learning Automata and Grammars

Colin de la Higuera
LINA, University of Nantes

April 26, 2011

Preamble

These errors have been found through the efforts of a number of people. Even if I don't welcome bad news, I do appreciate feedback. Many thanks to Hasan Ibne Akram, Domagoj Babic, Joshua Jones, Jean-Gert Nesselbosch and Menno van Zaanen, for pointing out these items.

I have tried to follow the following rule in this text: whatever is coloured in red is an error in the book and whatever is in green is the proposed correction.

The version is still provisional.

Chapter 1: Introduction

Table 2, page 8 has two cells inverted: values 0 and 3 should be interchanged, resulting in new Table 2 below.

	s	a
s	1	5
a	3	0

Table 2: The game matrix.

Page 14, the automaton represented in Figure 1.9(b) is kept: wrong. Should be the automaton represented in Figure 1.9(a) is kept.

Chapter 3: Basic Stringology

Page 55, Example 3.4.2. The correct vector is (1, 4, 2, 2, 2, 0,...), because the first value corresponds to $|x|_\lambda$.

Page 61, inside *Definition 3.5.3* of the spectrum kernel, we should read The associated kernel is defined as

$$\kappa^{S,k}(x,y) = \langle \phi^{S,k}(x), \phi^{S,k}(y) \rangle = \sum_{u \in \Sigma^k} \phi_u^{S,k}(x) \phi_u^{S,k}(y).$$

Furthermore in the following example, $\kappa^{S,2}(\texttt{aabaaac}, \texttt{baaa}) = 6 + 1$ (for $k = 2$).

Chapter 11: Window Languages

Concerning **Definition 11.1.1**, page 218 and also the bottom of previous page 217. The following formula is wrong (twice):

$$\forall l, l' \in \Sigma^\star, \forall w \in \Sigma^{=k}, lw \in L \iff l'w \in L.$$

What should be is:

$$\forall l, l' \in \Sigma^\star, \forall w \in \Sigma^{\geq k}, \exists u \in \Sigma^\star, lwu \in L \wedge l'wu \in L$$
$$\Longrightarrow$$
$$\forall v \in \Sigma^\star, lwv \in L \Leftrightarrow l'wv \in L.$$

Furthermore, also on page 218, **Example 11.1.1** is misleading, since bb can be recognised by a window of size 2. The definitions remain correct, it just is a bad example. Take this new Figure 11.1 (which is an alternative for Figure 11.1, page 219, in the book). The automaton from Figure 11.1 recognises the language $\texttt{a}+\texttt{aa}^*\texttt{b}((\texttt{ba}+\texttt{a})\texttt{b})^*$. The language is clearly 3-$\boldsymbol{\mathcal{TSS}}$ but is not 2-$\boldsymbol{\mathcal{TSS}}$: with a window of size only two, we would not be able to avoid substring bbb.

Further elements can be given concerning the roles of the different sets involved in **Definition 11.1.2**:

Definition 1 *Given $k > 0$, a* k-testable machine *($k - \boldsymbol{\mathcal{TSS}}$) is a 5-tuple $Z_k = \langle \Sigma, I, F, T, C \rangle$ with:*

- Σ *is a finite alphabet;*

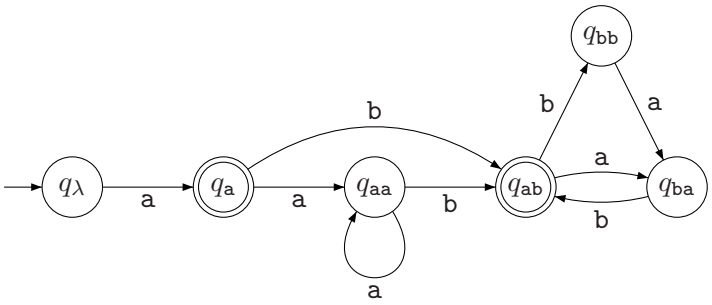

Figure 11.1: A DFA corresponding to the $3-\boldsymbol{\mathcal{TSS}}$ machine $Z_3 = \langle\{\mathtt{a},\mathtt{b}\}, I = \{\mathtt{aa},\mathtt{ab}\}, F = \{\mathtt{ab}\}, C = \{\mathtt{a}\}, T = \{\mathtt{aaa},\mathtt{aab},\mathtt{aba},\mathtt{abb},\mathtt{bab},\mathtt{bba}\}\rangle$.

- $I, F \subseteq \Sigma^{k-1}$ *(prefixes of length k-1 and suffixes (or finals) of length k-1);*
- $C \subseteq \Sigma^{<k}$ *(short strings);*
- $T \subseteq \Sigma^{k}$ *(allowed segments).*

It is important that sets I, F and C are related. Indeed $C \cap \Sigma^{k-1} \subset F$ and $I \cap F \subset C$ for reasons of consistency.

Chapter 12: Informed learners

The algorithm 12.14 (RPNI-CONSTRUCTS) that is supposed to build a characteristic sample forgets to add strings that reach the final states (each final state should be reached at least once by a string in S_+).

One possibility is to add to S_+, for each $q_u \in \mathbb{F}_{\mathbb{A}}$ the string $\mathrm{SP}(q_u)$.

Chapter 15: Learning context-free grammars

Page 308, in *Example 15.2.1*,

- 2-ancestors(N_1) = $\{N_1\}$, 2-ancestors(N_2) = $\{N_1, N_2\}$,
- 1-ancestors(N_1) = $\{N_1\}$, 1-ancestors(N_2) = $\{N_1, N_2\}$.

Chapter 18: Learning transducers

Small errata

page	line	is	should be
376	beginning of Section 18.2	The first algorithm to learn transducers,	As a first algorithm to learn transducers,
376	3 of Section 18.2	ALERGIA	and ALERGIA
376	3 of Subsection 18.2.1	that the information is still unknown	this absence of information
376	end of Subsection 18.2.1	$\forall u \in \widehat{\Gamma^\star} \perp \cdot u = u \cdot \perp = \perp$	$\forall u \in \widehat{\Gamma^\star}, \perp \cdot u = u \cdot \perp = \perp$
376	end of Subsection 18.2.1	$\forall A \in 2^{\widehat{\Gamma^\star}} \text{lcp}(A \cup \{\perp\}) = \text{lcp}(A)$	$\forall A \in 2^{\widehat{\Gamma^\star}}, \text{lcp}(A \cup \{\perp\}) = \text{lcp}(A)$
377	last line of def 18.2.1	it	this transducer
377	end of Subsection 18.2.3	*add sentence*	An example of this strange looking transducer can be found in Figure 18.14, page 385.
380	b1	$\tau_2(q_2, a)$	$\tau_1(q_2, a)$
381	b4	exampe	example
385	b8	the finals λ and 1 are	the finals λ and 1 are

The numbering in each page is from the top (x) or from the bottom (bx). The figures, captions, tables are not counted.

About Ostia

About $\perp$

In Section 18.2.1 (page 376) when introducing $\perp$ we should add, as properties:

- $\text{lcp}(\{\perp\}) = \perp$
- $\forall u \in \Gamma^\star$, $u^{-1}\perp = \perp$.

About the definition of onward transducers

Definition 18.2.1, page 377 is wrong and should be:

Definition 2 (18.2.1 Onward transducer) *A transducer is* ***onward*** *if*$_{def}$ $\forall q \in Q$, $\mathrm{lcp}\big(\{u : \exists a \in \Sigma, \exists q' \in Q\ (q, a, u, q') \in E\} \cup \{\sigma(q)\}\big) = \lambda$.

About Algorithm Onward

Page 378:
Algorithm ONWARD-PTT takes two arguments: the first is a PTT $\mathcal{T}$, the second is a state q. When first called in order to make the PTT onward, q should be set to q_λ. The value returned is a pair $(\mathcal{T}, f)$. $\mathcal{T}$ is the corresponding onward PTT rooted in u, f is the prefix that has been forwarded.

Algorithm 18.2: ONWARD-PTT.

Data: a PTT : $\mathcal{T} = \langle Q, \Sigma, \Gamma, q_\lambda, E, \sigma \rangle$, $q \in Q$
Result: an equivalent onward PTT : $\mathcal{T} = \langle Q, \Sigma, \Gamma, q_\lambda, E, \sigma \rangle$, a string $f = \mathrm{lcs}\{q\}$
for $a \in \Sigma$ **do**
  if $\tau_2(q, a) \in Q$ **then** $(\mathcal{T}, f_a) \leftarrow$ONWARD-PTT$(\mathcal{T}, \tau_2(q, a))$;
  $\tau_1(q, a) \leftarrow \tau_1(q, a) \cdot f_a$
end
$f \leftarrow \mathrm{lcp}\big(\{\tau_1(q, a) : a \in \Sigma\} \cup \{\sigma(q)\}\big)$;
if $f \neq \lambda$ **then**
  for $a \in \Sigma$ **do** $\tau_1(q, a) \leftarrow f^{-1}\tau_1(q, a)$;
  $\sigma(q) \leftarrow f^{-1}\sigma(q)$
end
return $(\mathcal{T}, f)$

The folding algorithm

The folding algorithm (18.6, page 382) needs to be rewritten as here, Algorithm 18.6.

Note that Algorithm Merge has also to be modified to take into account the RED states.

The example 18.3.1

In **Example 18.3.1**, pages 381–383, the starting point should be a transducer in which everything is forwarded, in order to have no doubt about the result. Yet in Figure 18.8 the **1** in the transition from $q_{\mathtt{aab}}$ to $q_{\mathtt{aabb}}$ should not be there and can be advanced. The correct starting point is therefore as represented in Figure 18.8.

Algorithm 18.6: OSTIA-FOLD.

Input: a transducer $\mathcal{T}$, a set of RED states, two states q and q'
Output: $\mathcal{T}$ updated, where subtree in q' is folded into q

```
w ←OSTIA-OUTPUTS(σ(q), σ(q'));
if w = fail then
    return fail
else
    σ(q) ← w;
    for a ∈ Σ do
        if τ(q', a) is defined then
            if τ(q, a) is defined then
                if τ2(q, a) ∈ RED then
                    return fail
                else
                    T ←OSTIA-PUSHBACK(T, q, q', a);
                    T ←OSTIA-FOLD(T, τ2(q, a), τ2(q', a))
                end
            else
                τ(q, a) ← τ(q', a)
            end
        end
    end
    return T
end
```

Example 1 *Let us run this merge-and-fold procedure on a simple example. Consider the transducer represented in Figure 18.8. Suppose we want to merge states* $q_{\mathtt{aa}}$ *with* q_λ*. Notice that* $q_{\mathtt{aa}}$ *is the root of a tree.*

We first redirect the edge $(q_{\mathtt{a}}, \mathtt{a}, \mathbf{1}, q_{\mathtt{aa}})$*, which becomes* $(q_{\mathtt{a}}, \mathtt{a}, \mathbf{1}, q_\lambda)$ *(Figure 18.9).*

We now can fold $q_{\mathtt{aa}}$ *into* q_λ*. This leads to pushing back the second* $\mathbf{1}$ *on the edge* $(q_{\mathtt{aa}}, \mathtt{a}, \mathbf{1}, q_{\mathtt{aaa}})$ *and also on the edge* $(q_{\mathtt{aa}}, \mathtt{a}, \mathbf{1}, q_{\mathtt{aab}})$*. The resulting situation is represented in Figure 18.10.*

Then (Figure 18.11) $q_{\mathtt{aaa}}$ *is folded into* $q_{\mathtt{a}}$*, and finally* $q_{\mathtt{aab}}$ *into* $q_{\mathtt{b}}$*. The result is represented in Figure 18.12.*

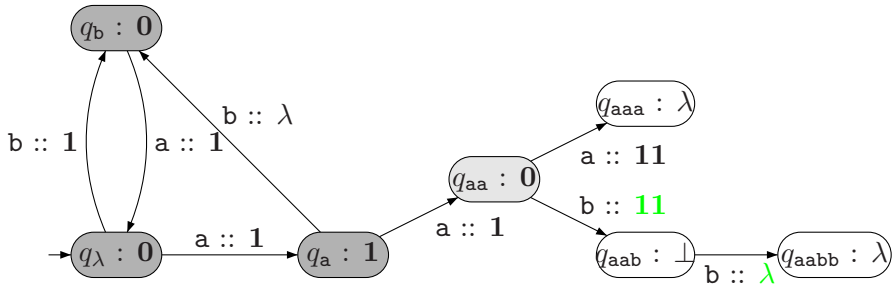

Figure 18.8: Before merging $q_{\mathtt{aa}}$ with q_λ and redirecting transition $\tau_2(q_{\mathtt{a}}, \mathtt{a})$ to q_λ.

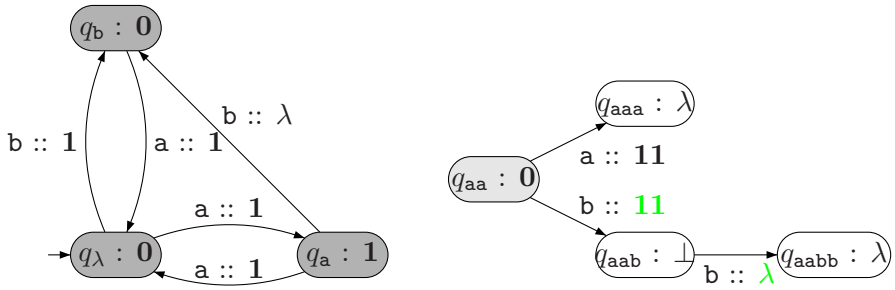

Figure 18.9: Before folding $q_{\mathtt{aa}}$ into q_λ.

A run of algorithm Ostia

Suppose we want to identify the transducer represented in Figure 18.13 using Ostia. The target transducer takes as input any sequence of symbols from $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, and replaces each $\mathtt{a}$ which is **not** followed by another $\mathtt{a}$, by a $\mathbf{0}$. If not $\mathtt{a}$ becomes $\mathbf{0}$ and $\mathtt{b}$ becomes $\mathbf{1}$.

Typical examples (which make the learning sample), are $(\mathtt{a}, \mathbf{1})$, $(\mathtt{b}, \mathbf{1})$, $(\mathtt{aa}, \mathbf{01})$, $(\mathtt{ab}, \mathbf{01})$, $(\mathtt{aaa}, \mathbf{001})$, $(\mathtt{abab}, \mathbf{0101})$.

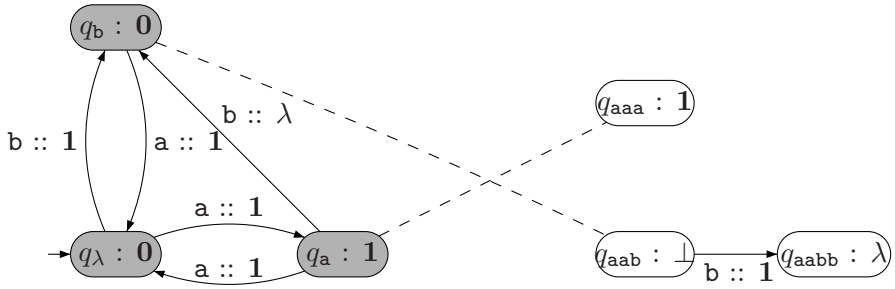

Figure 18.10: $q_{\mathtt{aa}}$ is folded into q_λ .

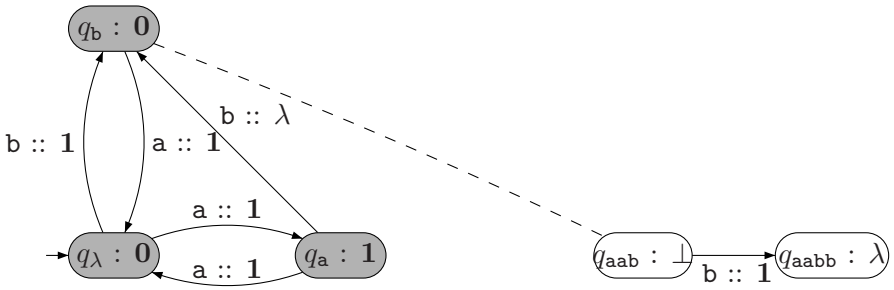

Figure 18.11: $q_{\mathtt{aaa}}$ is folded into $q_{\mathtt{a}}$.

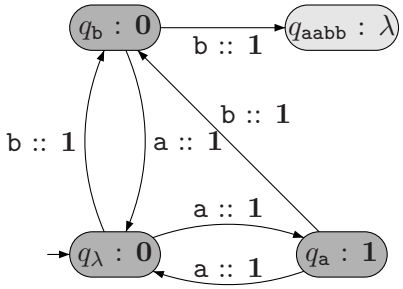

Figure 18.12: $q_{\mathtt{aab}}$ is folded into $q_{\mathtt{b}}$.

We first use Algorithm Build-Ptt (18.1) and build the corresponding Ptt represented in Figure 18.14. Algorithm Onward-Ptt (18.2) is then called and we obtain the onward Ptt (Figure 18.15).

The first merge that we test is between states $q_{\mathtt{a}}$ (which is Blue) and the unique Red state q_λ; the merge is rejected because $\sigma(q_{\mathtt{a}}) = \mathbf{1}$ and $\sigma(q_{\mathtt{aa}}) = \mathbf{01}$ are different, so the states cannot be merged together. So $q_{\mathtt{a}}$ is promoted.

Ostia then tries to merge $q_{\mathtt{b}}$ with q_λ; this merge is accepted, and the

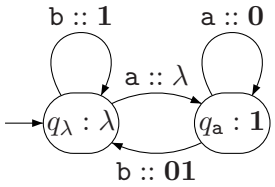

Figure 18.13: The target.

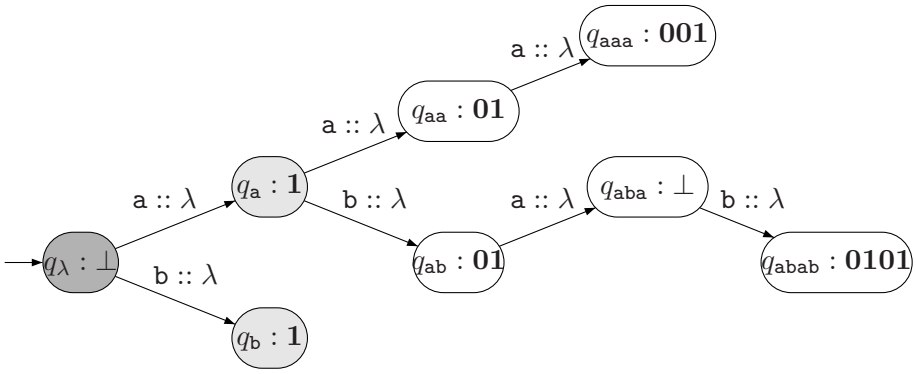

Figure 18.14: The PTT constructed from $S = \{(\mathtt{a}, \mathbf{1}), (\mathtt{b}, \mathbf{1}), (\mathtt{aa}, \mathbf{01}), (\mathtt{ab}, \mathbf{01}), (\mathtt{aaa}, \mathbf{001}), (\mathtt{abab}, \mathbf{0101})\}$.

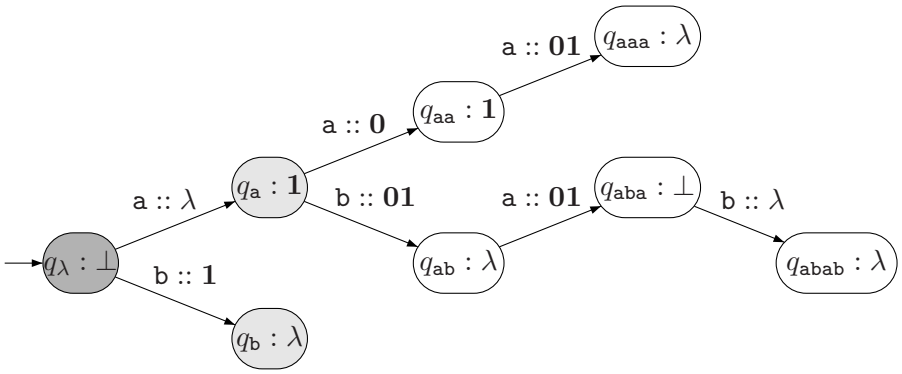

Figure 18.15: Making the PTT onward.

new transducer is depicted in Figure 18.16.

The next attempt is to merge $q_{\mathtt{aa}}$ with q_λ; this merge is rejected again because $\sigma(q_\lambda) = \lambda$ and $\sigma(q_{\mathtt{aa}}) = \mathbf{01}$ which are incompatible. Note that you there is no pushing back that can be done on the σ values.

So in the transducer represented in Figure 18.16, OSTIA merges $q_{\mathtt{aa}}$ with $q_{\mathtt{a}}$. It is accepted and the automaton represented in Figure 18.17 is built.

The next (and last) BLUE state is $q_{\mathtt{ab}}$ which OSTIA tries to merge with q_λ. The merge is accepted. There are some pushing back operations that

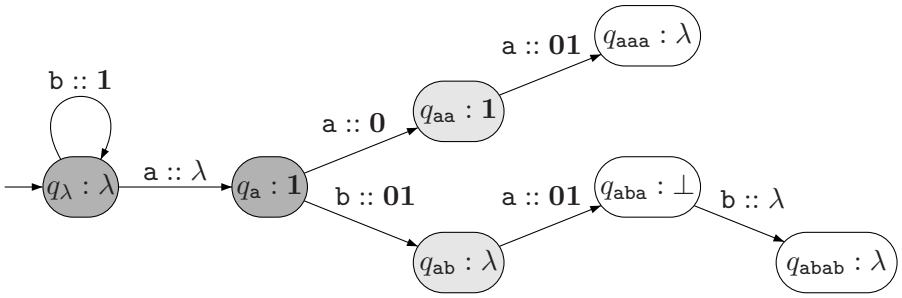

Figure 18.16: After merging $q_\mathtt{b}$ and q_λ.

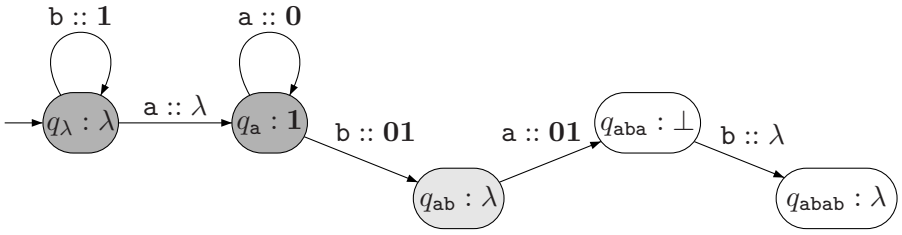

Figure 18.17: After merging $q_\mathtt{aa}$ and $q_\mathtt{a}$.

get done (and that we do not detail here).

The algorithm halts since there are no Blue states left. The result is the target.