Incremental construction and maintenance of morphological analysers based on augmented letter transducers

Alicia Garrido-Alenda, Mikel L. Forcada and Rafael C. Carrasco
www.*interNOSTRUM*.com
Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant, E-03071 Alacant, Spain.
{alicia,mlf,carrasco}@dlsi.ua.es

Abstract

We define deterministic augmented letter transducers (DALTs), a class of finite-state transducers which provide an efficient way of implementing morphological analysers which tokenize their input (i.e., divide texts in tokens or words) as they analyse it, and show how these morphological analysers may be maintained (i.e., how surface form–lexical form transductions may be added or removed from them) while keeping them minimal; efficient algorithms for both operations are given in detail. The algorithms may also be applied to the incremental construction and maintentance of other lexical modules in a machine translation system such as the lexical transfer module or the morphological generator.

1 Introduction

interNOSTRUM (Canals-Marote et al. 2001) is an online Spanish–Catalan machine translation system (`http://www.internostrum.com`) that attains great speed through the use of finite-state transducers (FSTs) in its lexical modules, for instance, in the source-language morphological analyser. The final FSTs in interNOSTRUM are a special class of letter transducers (Roche & Schabes 1997) that are obtained by compiling linguistic specifications (such as *morphological* and *bilingual dictionaries*) given in a linguist-readable language.

During the maintenance of the machine translation (MT) system, two operations are very common: the addition of a set of transductions (corresponding, for example, to the (surface form, lexical form) pairs[1] corresponding to a new word in the vocabulary), and the removal of a set of transductions (for instance, when an error is detected in the morphological analyser). Until now, these operations were performed by adding or removing the entries from the dictionaries and recompiling them into minimal FSTs, which was too slow to be convenient. More precisely, the complete dictionary was not compiled; instead, dictionaries were divided in sections, compiled separately, and the corresponding FSTs were merged into a single FST; addition or removal of an entry involved only the recompilation of a particular dictionary section and the merging of all FSTs; this was better but still too slow. In some situations, real-time addition

[1] The *surface form* is the (possibly inflected) form of the word as it appears in the text; the *lexical form* gives the base form of the word, its part of speech, and grammatical information about its inflection.

and removal of transductions may be crucial: for example, when debugging the lexical modules of the MT system or when a user wants to add new words for immediate use.

In this paper, we show a simple and efficient method to modify a minimal FST so that a single transduction is added to or removed from the language accepted by it. The algorithms presented here are derived from those presented by Carrasco & Forcada (2002) for finite-state acceptors (FSAs), which in turn extend the range of those by Daciuk et al. (2000) to cyclic FSAs and to the removal of entries. The algorithms in this paper are applicable to a particular class of FSTs called *deterministic augmented letter transducers* (deterministic ALTs or DALTs). Any FST may always be turned into an equivalent letter transducer (Roche & Schabes 1997); as will be shown, augmented letter transducers are a convenient way of implementing morphological analysers: (1) they analyse text simultaneously to segmenting (tokenizing) it into contextual, morphologically-motivated units (which may be composed of more than one word) by using one-character lookahead; (2) they may be easily manipulated as customary finite-state machines over an alphabet of input–output symbol pairs, and (3) these operations do not alter the existing input–output alignments, which may capture the regularities detected and explicitly coded in the dictionaries by a linguist as partial transductions for morphemes or as paradigms containing sets of alternative partial transductions.

This paper has the following parts: section 2 defines the deterministic augmented letter transducers that will be used in this paper; section 3 describes how to use them as morphological analysers; single-transduction DALTs are defined in section 4; the minimal DALTs resulting from adding or removing a transduction are described in detail in section 5; section 6 describes the algorithms, and, finally, some closing remarks are given in section 7.

2 Deterministic augmented letter transducers (DALT)

Any FST may always be turned into an equivalent letter transducer (Roche & Schabes 1997). The *deterministic augmented letter transducers* (DALT) used in our analysers are $T = (Q, L, \delta, q_I, \xi_s, \xi_w)$, where Q is a finite set of states; L a set of transition labels $L = ((\Sigma \cup \{\epsilon\}) \times \Gamma) \cup (\Sigma \times (\Gamma \cup \{\epsilon\}))$ where Σ is the input alphabet, Γ the output alphabet, and ϵ the *empty symbol* (see below); $\delta : Q \times L \to Q$ the transition function; $q_I \in Q$ the initial state; $\xi_s : Q \to 2^{\Sigma \cup \{\$\}}$ a function that assigns a *strong validation* (or *strong lookahead*) *set* $\xi_s(q)$ to each state, $\xi_w : Q \to 2^{\Sigma \cup \{\$\}}$ a function that assigns a *weak validation* (or *weak lookahead*) *set* $\xi_w(q)$ to each state, with \$ the end-of-input (end-of-file) marker, and with $\xi_s(q) \cap \xi_w(q) = \emptyset$ for all $q \in Q$ (the use of validation sets will become clearer in section 3). States such that $\xi_s(q) \cup \xi_w(q) \neq \emptyset$ will be called *acceptance* states. According to the definition of L, state transition labels may therefore be of three kinds: $(\sigma : \gamma)$, meaning that symbol $\sigma \in \Sigma$ is read and symbol $\gamma \in \Gamma$ is written; $(\sigma : \epsilon)$, meaning that a symbol is read but nothing is written; and $(\epsilon : \gamma)$, meaning that nothing is read but a symbol is written. In morphological analysis, the symbols in Σ are those found in texts, and the symbols in Γ are those necessary to form the lemmas and those representing morphological information. Lookahead is only necessary for morphological analysis; the lexical transfer and the morphological generators, which operate on delimited lexical forms, do not need lookahead conditions and may be seen as DALTs having either $\xi_s(q) = \Sigma \cup \{\$\}$ or $\xi_s(q) \cup \xi_w(q) = \emptyset$, that is, as letter transducers having only the classical acceptance and nonacceptance states (Roche & Schabes 1997).

Even if DALTs are *deterministic* with respect to the alphabet L, they are in general non-deterministic with respect to Σ. Input nondeterminism is substantial to the coding of many of the regularities found by linguists and represents ambiguity in a straightforward way, whereas FSTs such as Mohri's (1997) p-subsequential transducers, which are made deterministic by realigning the transductions and therefore destroying the linguistically motivated alignments, are forced to model ambiguity by producing different output suffixes after reaching a final state. On the other hand, experiments with real dictionaries and real corpora, show that the input nondeterminism (i.e., the average number of alive transductions during the process) is very low and ranges around 2.

In this paper, we will define δ to be a *total* mapping; the corresponding DALTs will be called *complete*. This involves no loss of generality, as any DALT may be made complete by adding a new *absorption* state $\perp$ to Q, so that all undefined transitions point to it and such that $\delta(\perp, l) = \perp$ for all $l \in L$, and $\xi_s(\perp) = \xi_w(\perp) = \emptyset$. Using complete DALT is convenient for the theoretical discussion in this paper; real implementations of automata and the corresponding algorithms need not contain an explicit representation of the absorption state and its incoming and outgoing transitions.

For complete DALTs, the extended mapping $\delta^* : Q \times L^* \to Q$ (the extension of δ to transductions in L^*) is defined simply such that $\delta^*(q, \lambda) = q$, with λ the empty string in L^*, and $\delta^*(q, xl) = \delta(\delta^*(q, x), l)$ for all $l \in L$ and $x \in L^*$.

It is convenient to define the following languages for each transducer T and each lookahead $\sigma \in \Sigma \cup \{\$\}$: the language of transductions strongly accepted by T, $\mathcal{L}^s_\sigma(T) = \{t : \sigma \in \xi_s(\delta^*(q_I, t))\}$, and the language of transductions weakly accepted by T, $\mathcal{L}^w_\sigma(T) = \{t : \sigma \in \xi_w(\delta^*(q_I, t))\}$ (note that, for each lookahead $\sigma \in \Sigma \cup \{\$\}$, $\mathcal{L}^s_\sigma(T) \cap \mathcal{L}^w_\sigma(T) = \emptyset$). The corresponding families of right languages for each state q, $\mathcal{R}^s_\sigma(T, q)$, and $\mathcal{R}^w_\sigma(T, q)$ are defined analogously but substituting q for q_I.

It is very easy to build a non-deterministic ALT from a morphological dictionary by grafting the individual transductions corresponding to each (surface form, lexical form) entry in the morphological dictionary, such as `(tails,tailNP)`, which a linguist may have chosen to align, e.g., as `(t,t)(a,a)(i,i)(l,l)(`ϵ`,N)(s,P)`, into a single initial state (for details, see Garrido et al. 1999). Since ALTs are isomorph to finite-state automata, they may therefore be determinized with respect to the alphabet L and minimized using adapted versions of existing algorithms for finite automata (Hopcroft & Ullman 1979).

A minimal DALT is one in which no two states are equivalent. Analogously to deterministic finite automata, equivalent states are those having the same sets of right transduction languages. Note that DALT minimization does not change the alignments originally given between surface forms and lexical forms; this is because, unlike in minimization algorithms for regular FSTs (Mohri 1997), minimization operates at the level of the pair alphabet L.

3 Using DALTs as morphological analysers

A string $w' \in \Gamma^*$ is considered to be a *strong* (resp. *weak*) *transduction* of an input string $w \in \Sigma^*$ if there is at least one path from the initial state q_I to a state q such that the input symbol following w is in the validation set $\xi_s(q)$ (resp. $\xi_w(q)$). In general, there may be more than a valid transduction for a string w (in analysis, this would correspond to the *lexical ambiguity* shown by *homographs*, surface forms having two or more lexical forms, i.e., two or more morphological analyses).

"Tokenize as you analyse": The validation sets defined in DALTs through functions ξ_s and ξ_w enable morphological analysers to both tokenize the input (segment it into surface forms suitable for analysis) and analyse it at the same time (much in a similar manner as the lexical scanners generated by the Unix utility `lex`, Lesk 1975). Input is buffered for convenience. The DALT analyses the input by maintaining the following sets:

- An SPO (*set of partial outputs*). The SPO initially contains the pair (ϵ, q_I); a fresh SPO is built from the previous SPO after each input symbol and contains all the (output string, state) pairs (z, q) formed by the last states q reached and their associated partial output strings $z \in \Gamma^*$.
- Two SVO (sets of *validated outputs*), the *strong* SVO and the *weak* SVO, which are initially empty and contain (output string, state) pairs formed by the most recently strongly and weakly validated acceptance states and their associated output strings.

The input position p of the symbol with which the sets of acceptance states in each SVO were reached is also stored. After reading each input symbol σ,

- a new SPO is built from the previous SPO (the new SPO contains $(z\gamma, q')$ if (γ, q) was in the previous SPO and there is a $\gamma \in \Gamma \cup \{\epsilon\}$ such that $\delta^*(q, (\sigma, \gamma) = q')$; and
- pairs (z, q) in the previous SPO such that $\sigma \in \xi_s(q)$ (resp. $\sigma \in \xi_w(q)$) are used to overwrite the strong (resp. weak) SVO and its position p (if no such pair occurs, the SVO and p are left intact).

Input is read until all elements of the current SPO contain the absorption state $\perp$; then, output strings corresponding to the strong SVO are written, and, if empty, output strings corresponding to the weak SVO are written; finally, the DALT restarts at the initial state and the character immediately after the position p of the last SVO. If the SVOs are empty at that point, the DALT writes the symbol read immediately after the initial state as an "unanalysed symbol", and restarts at the next symbol and at the initial state.

Strongly validated acceptance states $(q : \xi_s(q) \neq \emptyset)$ are used, for example, to identify regular words using a validation set with all non-word symbols (so that the analyser does not stop at `bar` when input is `barber`); in particular, unconditional strong acceptance states $(q : \xi_s(q) = \Sigma \cup \{\$\})$ are used to identify tokens such as punctuation marks (`:`,

-), whitespace, or apostrophated forms (`d'`, `l'`). On the other hand, weakly validated acceptance states ($q : \xi_w(q) \neq \emptyset$) are used to clip "unknown words" (such as `barnwazz`) and produce some kind of *guessed* (weak) transductions. As has been said, if a strong transduction is present, weak transductions are ignored.

This left-to-right, longest-match way of functioning makes it very easy to treat (variable or invariable) multi-word units (MWUs), for input: if a MWU is not complete, the acceptance state reached will correspond to a smaller unit, which will be clipped and whose transduction will be output (for example, if the dictionary contains "`George`" and the MWU "`George Washington`", when reading "`George W. Bush`" the MWU "`George Washington`" will abort at the "`.`", the transduction of "`George`" will be output and the analyser will be ready to process the remaining text, "` W. Bush`").

4 Single-transduction DALT

In most cases, the morphological dictionary will grow through the addition of a transduction $t \in L^*$ —corresponding to a new (surface form, lexical form) pair— which has to be strongly accepted, but conditionally to a lookahead set $S^t \subseteq \Sigma \cup \{\$\}$. We find it therefore convenient to define the (complete) *single-transduction DALT* for transduction t strongly accepted with lookahead set S^t, denoted $T^t = (Q^t, L, \delta^t, q_I^t, \xi_s^t, \xi_w^t)$, such that $\mathcal{L}_\sigma^s(T^t) = \{t\}$ for $\sigma \in S^t$.[2] This DALT has $Q^t = \mathrm{Pr}(t) \cup \{\perp^t\}$, where $\mathrm{Pr}(t)$ is the set of all prefixes of transduction t and $\perp^t$ denotes the absorption state, $q_I^t = \lambda$, $\xi_w^t(x) = \emptyset$ for all $x \in Q^t$, and ξ_s^t such that $\xi_s^t(t) = S^t$ and $\xi_s^t(x) = \emptyset$ for $x \neq t$. The next-state function is defined as follows: if $xl \in \mathrm{Pr}(t)$ then $\delta(x, l) = xl$; else $\delta(x, l) = \perp^t$. Note that the single-transduction DALT for a transduction t has $|Q^t| = |t| + 2$ states.

5 Adding and removing a transduction

5.1 Adding a transduction

Given a DALT T, it is easy to build a new complete DALT T' such that it accepts all the strong and weak transductions accepted by T plus a new transduction t strongly accepted with a validation set S^t; that is:

$$\begin{aligned}
\forall \sigma \in \Sigma \cup \{\$\},\ \mathcal{L}_\sigma^s(T') &= \mathcal{L}_\sigma^s(T) \cup \begin{cases} \{t\} & \text{if } \sigma \in S^t \\ \emptyset & \text{otherwise} \end{cases} \\
\forall \sigma \in \Sigma \cup \{\$\},\ \mathcal{L}_\sigma^w(T') &= \mathcal{L}_\sigma^w(T) - \begin{cases} \{t\} & \text{if } \sigma \in S^t \\ \emptyset & \text{otherwise} \end{cases}
\end{aligned} \tag{1}$$

Full details of how T' is derived are not given here; however, the construct described here (based on Carrasco & Forcada's (2002) construct for finite-state automata, and described there in more detail) builds upon the classical Cartesian-product construction of a deterministic finite automaton accepting the intersection of two regular languages found in formal language theory textbooks (Hopcroft & Ullman 1979:p. 59), applied to T and the single-transduction automaton T^t but with a special assignment of acceptance states to ensure the above conditions.

[2] Single-transduction DALTs may analogously be defined for weak acceptance.

The new DALT, $T' = (Q', L, \delta', q'_I, \xi'_s, \xi'_w)$ has $Q' = Q \times Q^t$, δ' such that for all $(q, q^t) \in Q'$ and for all $l \in L$, $\delta'((q, q^t), l) = (\delta(q, l), \delta^t(q^t, l))$ and $q'_I = (q_I, q^t_I)$. Before discussing functions ξ'_w and ξ'_s, it is convenient to realize that states may be seen as belonging to four groups (the nomenclature is inspired in that used by Daciuk et al. 2000).

- States of the form $(q, \perp^t)$, with $q \in Q - \{\perp\}$, equivalent to those non-absorption states of T which are not reached by any prefix of t; they will be called *intact* states because they have the same transition structure as their counterparts in T; that is, if $\delta(q, l) = r$, then $\delta'((q, \perp^t), l) = (r, \perp^t)$. For large DALTs (dictionaries) T, these are the great majority of states (the number of intact states ranges between $|Q| - |t| - 1$ and $|Q|$); therefore, it will be convenient in practice to consider T' as a modified version of T and will be treated as such in the algorithms presented in this paper.

- States of the form (q, x) with $q \in Q - \{\perp\}$ and $x \in \Pr(t)$, and such that $\delta^*(q_I, x) = q$; they will be called *cloned* states; in particular, the new start state, $q'_I = (q_I, \lambda)$ is also a cloned state. The remaining states in $(Q - \{\perp\}) \times \Pr(t)$ —most of the states in $Q \times Q^t$— may be discarded because they are unreachable from the new start state q'_I. Cloned states are modified versions of the original states $q \in Q - \{\perp\}$: all of their outgoing transitions point to the corresponding intact states in Q', $(\delta(q, l), \perp^t)$, except for the transition with symbol $l : xl \in \Pr(t)$, which now points to the corresponding cloned state $(\delta(q, l), xl)$. There are at most $|t| + 1$ cloned states.

- States of the form $(\perp, x)$, with $x \in \Pr(t)$. These states will be called *queue* states; states of this form appear only if in the original automaton $\delta^*(q_I, x) = \perp$ for some $x \in \Pr(t)$. There are at most $|t|$ queue states.

- An absorption state $\perp' = (\perp, \perp^t)$, with $\xi'_s(\perp') = \xi'_w(\perp') = \emptyset$.

The new DALT T', which for most large dictionaries is only slightly larger than T has the following lookahead sets:

q'	INTACT $((q, \perp^t))$	CLONED $((q, x))$	QUEUE $((\perp, x))$
$\xi'_s(q') =$	$\xi_s(q)$	$\xi_s(q) \cup S^t$ if $x = t$ $\xi_s(q)$ otherwise	S^t if $x = t$ $\emptyset$ otherwise
$\xi'_w(q') =$	$\xi_w(q)$	$\xi_w(q) - S^t$ if $x = t$ $\xi_w(q)$ otherwise	$\emptyset$

It is straightforward to show that the right languages for intact states in T' are the same as their counterparts in T: $\mathcal{R}^s_\sigma(T', (q, \perp^t)) = \mathcal{R}^s_\sigma(T, q)$, and $\mathcal{R}^w_\sigma(T', (q, \perp^t)) = \mathcal{R}^w_\sigma(T, q)$, for all $\sigma \in \Sigma \cup \{\$\}$; since the original DALT was minimal, this means that minimization will not affect intact states, which are usually most of the states in T'. This is the main reason for the efficiency of the algorithms presented in this paper: minimization may be accomplished in a small number of operations. It is not difficult to show that minimization may be performed by initializing a register R with all of the intact states and then testing, one by one, queue and cloned states against states in R (starting with the last queue state $(\perp, t)$ or, if it does not exist, the last cloned state (q, t), and descending in $\Pr(t)$) and adding them to the register if they are not found to be equivalent to a state in R (performing this check backwards avoids having to test the equivalence of states by visiting their descendants recursively). This is the most costly part of the algorithms and has an asymptotic complexity of $O(|Q||t|)$, since $|R|$ is of the order of $|Q|$. Minimization (including the elimination of unreachable states in T') appears in section 6 as part of the transduction addition and removal algorithms.

5.2 Removing a transduction

Again, given a DALT T, it is easy to build a new complete DALT T' such that it accepts all the strong and weak transductions accepted by T minus transduction t strongly accepted with validation set S^t; that is:

$$\begin{aligned}
\forall \sigma \in \Sigma \cup \{\$\},\ \mathcal{L}^s_\sigma(T') &= \mathcal{L}^s_\sigma(T) - \begin{cases} \{t\} & \text{if } \sigma \in S^t \\ \emptyset & \text{otherwise} \end{cases} \\
\forall \sigma \in \Sigma \cup \{\$\},\ \mathcal{L}^w_\sigma(T') &= \mathcal{L}^w_\sigma(T)
\end{aligned} \qquad (2)$$

The resulting DALT has the same set of reachable states in Q' as the one in section 5.1, and therefore the same close-to-minimality properties; however, since t is supposed to be in $\mathcal{L}^s_\sigma(T)$ for all $\sigma \in S^t$, no queue states will be formed (in fact, if $t \notin \mathcal{L}^s_\sigma(T)$, a nonaccepting queue with all states eventually equivalent to $(\perp, \perp^t)$ may be formed). The lookahead sets are:

q'	INTACT $((q, \perp^t))$	CLONED $((q, x))$	QUEUE $((\perp, x))$
$\xi'_s(q') =$	$\xi_s(q)$	$\xi_s(q) - S^t$ if $x = t$; $\xi_s(q)$ otherwise	$\emptyset$
$\xi'_w(q') =$	$\xi_w(q)$	$\xi_w(q)$	$\emptyset$

Note that if transduction t with validation set S^t is removed from the DALT, it may still be possible that t is still strongly accepted with a smaller validation set (which is unlikely in usual dictionary maintenance). Minimization may be performed analogously to the transduction addition case.

6 Algorithms

6.1 Adding a transduction

Figure 1 shows the algorithm that may be used to add a transduction to an existing DALT, which follows the construction in section 5.1. The resulting DALT is viewed as a modification of the original one: therefore, intact states are not created; instead, unreachable intact states are eliminated later. The register R of states not needing minimization is initialized with Q. The algorithm has three parts:

- First, the cloned and queue states are built and added to Q by using function clone() for all prefixes of t. The function returns a cloned state (with all transitions created) if the argument is a nonabsorption state in $Q - \{\perp\}$ or a queue state if it operates on the absorption state $\perp \in Q$; in both cases, it updates lookahead sets according to the table in section 5.1.

- Second, those intact states which have become unreachable as a result of designating the cloned state q'_I as the new start state are removed from Q and R and the start state is replaced by its clone (unreachable states are simply those states having no incoming transitions as constructed by the algorithm as a consequence of the removal of other unreachable intact states). Note that only intact states in $\delta(q_I, x)$ for some $x \in \Pr(t)$ may become unreachable as a result of having been cloned; therefore, if states are visited in ascending length of x, function unreachable() simply has to check for the absence of incoming transitions.

- And third, the queue and cloned states are checked (starting with the last state) against the register R using function replace_or_register(), which is an adaptation of the non-recursive version found in algorithm 2 of (Daciuk et al. 2000): basically, if the argument state is found to be equivalent to a state in R, it is replaced by its equivalent in R; if not, it is added to R.

Finally, the new (minimal) DALT is returned. In real implementations, absorption states are not explicitly stored; this results in small differences in the implementations of the functions clone() and replace_or_register().

6.2 Removing a transduction

The algorithm for removing a strong transduction from the languages accepted by a DALT T only differs from the previous algorithm in that the clone() function uses the table in section 5.2 instead of that in section 5.1. Since the string removal algorithm will usually be asked to remove a string which was in $\mathcal{L}^s_\sigma(T)$ for some σ, function clone() will usually generate only cloned states and no queue states.

7 Concluding remarks

Deterministic augmented letter transducers (DALTs), a class of letter transducers (Roche & Schabes 1997) defined in this paper, are an efficient way of implementing morphological analysers which tokenize their input (i.e., divide texts in tokens or words) as they analyse it, and have been used in a machine translation system (Garrido et al. 1999; Canals-Marote et al. 2001). They may also be used, without the lookahead mechanism to build lexical transfer and morphological generation modules. This paper shows how DALTs may be maintained (i.e., how input-output pairs or transductions may be added or removed from them) while keeping them minimal; algorithms for both operations are described. The algorithms here bear some resemblance to those described in (Daciuk et al. 2000) for acyclic deterministic finite automata; however, the algorithms here apply to transducers, allow for the removal of transductions, and are not restricted to acyclic transducers. The time complexity of the algorithms is in $O(|Q||t|)$ (as in Daciuk et al. 2000).

Acknowledgements: Supported by the Caja de Ahorros del Mediterráneo, by the Vice-rectorate for Information Technologies of the Universitat d'Alacant and by the Spanish *Comisión Interministerial de Ciencia y Tecnología* through grant TIC2000-1599-C02-02.

algorithm **addtrans**

Input: $T = (Q, L, \delta, q_I, \xi_s, \xi_w)$ (minimal, complete),
$t \in L^*$, with strong lookahead set S^t

Output: $T' = (Q', L, \delta', q'_I, \xi'_s, \xi'_w)$ minimal, complete,
and such that conditions (1) hold

$R \leftarrow Q$ [initialize register]
$q'_I \leftarrow \text{clone}(q_I)$ [clone initial state; update validation sets]
$q_{\text{last}} \leftarrow q'_I$
for $i = 1$ to $|t|$
  $q \leftarrow \text{clone}(\delta^*(q_I, t_1 \cdots t_i))$ [create cloned and queue states; update validation sets]
  $\delta(q_{\text{last}}, t_i) \leftarrow q$
  $q_{\text{last}} \leftarrow q$
end_for
$i \leftarrow 1$
$q_{\text{current}} \leftarrow q_I$
while($i \leq |t|$ and unreachable(q_{current}))
  $q_{\text{next}} \leftarrow \delta(q_{\text{current}}, t_i)$
  $Q \leftarrow Q - \{q_{\text{current}}\}$ [remove unreachable state from Q
  and update transitions in δ]
  $R \leftarrow R - \{q_{\text{current}}\}$ [remove also from register]
  $q_{\text{current}} \leftarrow q_{\text{next}}$
  $i \leftarrow i + 1$
end_while
if unreachable(q_{current})
  $Q \leftarrow Q - \{q_{\text{current}}\}$
  $R \leftarrow R - \{q_{\text{current}}\}$
end_if
$q_I \leftarrow q'_I$ [replace start state]
for $i = |t|$ downto 1
  replace_or_register($\delta^*(q_I, t_1 \cdots t_i)$) [check queue and cloned states one by one]
end_for
return $T = (Q, L, \delta, q_I, \xi_s, \xi_w)$
end_algorithm

Figure 1: Algorithm to add a transduction t with strong validation set S^t to a DALT while keeping it minimal.

References

Canals-Marote, Raül, Anna Esteve-Guillén, Alicia Garrido-Alenda, Maria I. Guardiola-Savall, Amaia Iturraspe-Bellver, Sandra Montserrat-Buendia, Sergio Ortiz-Rojas, Hermínia Pastor-Pina, Pedro M. Pérez-Antón & Mikel L. Forcada: 2001, 'The Spanish–Catalan machine translation system interNOSTRUM', in *Proceedings of MT Summit VIII*, Santiago de Compostela, Spain.

Carrasco, Rafael C. & Mikel L. Forcada: 2002, 'Incremental construction and maintenance of minimal finite-state automata', *Computational Linguistics*, in press.

Daciuk, Jan, Stoyan Mihov, Bruce W. Watson & Richard E. Watson: 2000, 'Incremental construction of minimal acyclic finite-state automata', *Computational Linguistics*, **26**(1): 3–16.

Garrido, A., A. Iturraspe, S. Montserrat, H. Pastor & M.L. Forcada: 1999, 'A compiler for morphological analysers and generators based on finite-state transducers', *Procesamiento del Lenguaje Natural*, (25): 93–98.

Hopcroft, J. E. & J. D. Ullman: 1979, *Introduction to automata theory, languages, and computation*, Reading, MA: Addison–Wesley.

Lesk, M.E.: 1975, 'Lex — a lexical analyzer generator', Tech. Rep. Technical Report 39, AT&T Bell Laboratories, Murray Hill, N.J.

Mohri, Mehryar: 1997, 'Finite-state transducers in language and speech processing', *Computational Linguistics*, **23**(2): 269–311.

Roche, E. & Y. Schabes: 1997, 'Introduction', in E. Roche & Y. Schabes, eds., *Finite-State Language Processing*, Cambridge, Mass.: MIT Press, pp. 1–65.