Finite State Automata and Arabic Writing

Michel Fanton
CERTAL-INALCO[1]
73 rue Broca
F75013 Paris France
email : certal2@ext.jussieu.fr

Abstract

Arabic writing has specific features, which imply computational overload for any arabicized software. Finite state automata are well known to give efficient solutions for translation problems which can be formalized as regular languages. These automata are as more easily built that their alphabet have been reduced through a careful linguistic analysis. This reduction makes it possible to write directly an automaton without going through the intermediate stage of contextual rules, which have to be translated into an automaton for the sake of efficiency. This paper presents two Moore automata, the first one, taken as an *example*, gives a solution to the choice of right shape for a letter to be printed or displayed (usually known as *contextual analysis*), the second one studies the more complex problem of determining the right carrying letter for hamza. Every arabicized software has to face these questions and finite state automata are certainly a good answer to them.

INTRODUCTION

Arabic writing has specific features, which imply computational overload for any arabicized software. The first one, well known now for many years, is the fact that Arabic printing tries to imitate handwriting. Because of this, consonants and long vowels can have four or only two shapes depending of their ability to be bound to the following letter and of where they appear in the word.

These shapes can be very different : for example letter ه[2] (h)

isolated	final	medial	initial
ه	ـه	ـهـ	هـ

or present only small variations : for example letter س (s)

isolated	final	medial	initial
س	ـس	ـسـ	سـ

Letters which cannot be bound to the next one have only two shapes, for example letters د (d) and و (w and ū)

isolated	final	isolated	final
د	ـد	و	ـو

During the seventies and the beginning of the eighties, hard controversies took place within the Arabs concerned with these questions, linguists and computer scientists. Finally in 1983 the ASMO (Arab Society for Normalization which unfortunately does not exist any more), influenced by Pr. Lakhdar-Ghazal from IERA (Rabat Morocco) chose to give a unique code to all shapes of one particular letter. This is certainly a good choice from a linguistic point of view, but even so, compromises had to be made to take into account writing habits that conflicted with it. Letter hamza is the most noticeable example of such a compromise for reasons we shall explain later.

1 CONTEXTUAL ANALYSIS

Whatever be the choice made for coding, from a typesetting or a computational point of view, there must be different codes for the different shapes of a letter. So every arabicized software has to use two systems for coding : the reduced code we have just introduced and the extended code in which the different shapes have different

[1]CERTAL : Centre d'Études et de Recherche en Traitement Automatique des Langues, INALCO : Institut National des Langues et Civilisations Orientales

[2]the Arabic parts of this paper have been typeset using Klaus Lagally's ArabTEX

codes. Up to UNICODE, no normalization exists for the second one. So every arabicized software has to solve the problem of choosing the right shape of every printed or displayed letter.

1.1 Rules for letter shape determination

This determination, frequently known as *contextual analysis* can be summarized into the following set of unformal rules:

1. *At the beginning of a word:*
   - If the letter is a binding letter it takes the INITIAL shape.
   - If it is a non binding one it takes the ISOLATED shape.
2. *In the middle of a word* (there is at least one letter following the current one):
   (a) If the letter is a binding letter then
   - If it follows a binding letter it takes the MEDIAL shape.
   - If it follows a non binding letter it takes the INITIAL shape.

   (b) If the letter is a non binding letter
   - If it follows a binding letter it takes the FINAL shape.
   - If it follows a non binding letter it takes the ISOLATED shape.
3. *At the end of a word* (for both types of letters)
   - If it follows a binding letter it takes the FINAL shape.
   - If it follows a non binding letter it takes the ISOLATED shape.

1.2 Moore and Mealy automata

Moore automata are *state assigned output machines* : the output function assigns output symbols to each state. They differ from Mealy automata, *transition assigned finite state machines*, where output symbols are associated with transitions between states. Mealy automata are sometimes called *finite transducers*. The two machine types have been demonstrated to produce the same input-output mappings[3].

[3] see (Aho and Ullman, 1972) and (Hopcroft and Ullman, 1979) for a full account of these matters

Mealy automata are certainly a better choice when bidirectional applications are considered. As the question is to identify succession of symbols of a certain type we found it clearer to use a Moore automaton.

1.3 A Moore automaton for contextual analysis

1.3.1 Source language of the automaton

It follows from the determination rules that we only need to know what particular letter we are dealing with only at the output stage. All we have to know is wether it is a binding or a non binding letter[4]. The alphabet of the automaton should be $A = \{\#\} \cup L$ where L is the set of arabic letters present in the reduced code and # the word boundaries. The set of letters will then be partitioned into three sets :

$$A \rightarrow A' = \{\{\#\}, N, B\}$$

N being the set of non binding letters and B the set of binding letters. If we denote respectively n and b an arbitrary element of each of these sets, the source language of the automaton can be reduced to:

$$A_1 = \{\#, n, b\}$$

$$L_1 = \{\#(n \vee b)^*\#\}$$

where $\vee$ denotes disjunction and $*$ is the Kleene star

1.3.2 Grammar and automaton for L_1

Language L_1 can be generated by the simple grammar :

$$\begin{array}{lcl} m & \rightarrow & \#A\# \\ A & \rightarrow & A(n \mid b) \end{array}$$

or the as simple automaton :
initial states = {1}
final states: = {5}
transitions

[4] As this question has only been taken as an example, the alphabet has been oversimplified. A full working automaton should cope, as far as arabic is concerned, with two additional problems : hamza on the line to which no preceding letter can be bound to and lām alif ligature. It should also give a proper treatment of non arabic letters and symbols. But this would not affect the here described method.

	b	n	#	output
1	$\emptyset$	$\emptyset$	2	$\emptyset$
2	3	4	$\emptyset$	#
3	3	4	5	b
4	3	4	5	n
5	$\emptyset$	$\emptyset$	$\emptyset$	#

1.3.3 Target language of the automaton

The alphabet for the target language L_2, given what has been said before and using the same method of partioning and then reducing the alphabet could be at first sight:

$$A_2 = \{\#, I, i, m, f\}$$

where I denotes a letter in *isolated* shape, i, m and f stand for *initial*, *medial* and *final* shape. But letters from N have only two shapes *final* and *isolated*. Moreover *isolated* and *final* shapes of letters from B can only appear at the end of a word, which is not the case for the corresponding shapes of letters from N. So, the following modified version of A_2 will be prefered :

$$A_2 = \{\#, I_n, I_b, i, m, f_n, f_b\}$$

where I_n stands for *isolated shape of a letter from N* and so on. With these symbols the target language L_2 can be described by the regular expression :

$$L_2 = \{\#(I_n^* i m^* f_n I_n^*)^*(I_b \vee f_b \vee \epsilon)\#\}$$

where ϵ denotes as usual the empty string.

1.3.4 Translation automaton

The translation process of a sequence of L_1 into a legal sequence of L_2 can be operated through the following automaton :
initial states = {1}
final states = {8}
transitions :

	n	b	#	output
1	2	{3,7}	1	#
2	2	{3,7}	8	I_n
3	6	{4,5}	$\emptyset$	i
4	6	{4,5}	$\emptyset$	m
5	$\emptyset$	$\emptyset$	8	f_b
6	2	{3,7}	8	f_n
7	$\emptyset$	$\emptyset$	8	I_b
8	$\emptyset$	$\emptyset$	$\emptyset$	#

This automaton is clearly nondeterministic. This is due to the fact that a letter from B can appear in final or isolated shape when situated at the end of a word, in initial or medial shape when another letter follows it. Because of this nondeterministic feature, every transition should appear as a set. When this set is a singleton, the "only" state has been put without braces for an easier reading.

It can be easily augmented to take account of occasional short vowels or shadda[5] (˜) that could occur : the transitions to add would force the automaton to loop onto the same state, whatever be it since vowels or shadda can only appear after a consonant and do not influence its shape.

1.3.5 PROLOG test program

This program is a straightforward translation of the above described grammar and automaton. The predicate *test* allows to limit the generation of inputs to a given length. In the results we chose to limit the length of the input to 6 included word boundaries.

```
%
% generation of elements of L1
%
m --> [#],a,[#].
a --> ([n];[b]).
a -->  a,([n];[b]).
%
% translation automaton
%
initial_state(1).
final_state(8).

tr(1,#,1).  tr(4,b,5).
tr(1,n,2).  tr(4,b,4).
tr(1,b,3).  tr(4,n,6).
tr(1,b,7).  tr(5,#,8).
tr(2,#,8).  tr(6,#,8).
tr(2,n,2).  tr(6,n,2).
tr(2,b,3).  tr(6,b,3).
tr(2,b,7).  tr(6,b,7).
tr(3,n,6).  tr(7,#,8).
tr(3,b,5).
tr(3,b,4).

output(1,#).      output(5,fb).
output(2,'In').   output(6,fn).
```

[5] sign denoting a double letter

```
output(3,i).      output(7,'Ib').
output(4,m).      output(8,#).

forme(Input,Output):-
initial_state(Is),
path(Is,Fs,Input,Output),
final_state(Fs).

path(S,S,[],[]).
path(S1,S2,[X|Xs],[Y|Ys]):-
tr(S1,X,S),
output(S,Y),
path(S,S2,Xs,Ys).

test(L):-
m(M,[]),
length(M,L1),
((L1 > L,!,nl,fail);true),
printing_form(M,F),
nl,write(M),tab(1),write(F),fail.
test(_).
```

1.3.6 Program results

input	output
[#,n,#]	[#,In,#]
[#,b,#]	[#,Ib,#]
[#,n,n,#]	[#,In,In,#]
[#,n,b,#]	[#,In,Ib,#]
[#,b,n,#]	[#,i,fn,#]
[#,b,b,#]	[#,i,fb,#]
[#,n,n,n,#]	[#,In,In,In,#]
[#,n,n,b,#]	[#,In,In,Ib,#]
[#,n,b,n,#]	[#,In,i,fn,#]
[#,n,b,b,#]	[#,In,i,fb,#]
[#,b,n,n,#]	[#,i,fn,In,#]
[#,b,n,b,#]	[#,i,fn,Ib,#]
[#,b,b,n,#]	[#,i,m,fn,#]
[#,b,b,b,#]	[#,i,m,fb,#]
[#,n,n,n,n,#]	[#,In,In,In,In,#]
[#,n,n,n,b,#]	[#,In,In,In,Ib,#]
[#,n,n,b,n,#]	[#,In,In,i,fn,#]
[#,n,n,b,b,#]	[#,In,In,i,fb,#]
[#,n,b,n,n,#]	[#,In,i,fn,In,#]
[#,n,b,n,b,#]	[#,In,i,fn,Ib,#]
[#,n,b,b,n,#]	[#,In,i,m,fn,#]
[#,n,b,b,b,#]	[#,In,i,m,fb,#]
[#,b,n,n,n,#]	[#,i,fn,In,In,#]
[#,b,n,n,b,#]	[#,i,fn,In,Ib,#]
[#,b,n,b,n,#]	[#,i,fn,i,fn,#]
[#,b,n,b,b,#]	[#,i,fn,i,fb,#]
[#,b,b,n,n,#]	[#,i,m,fn,In,#]
[#,b,b,n,b,#]	[#,i,m,fn,Ib,#]
[#,b,b,b,n,#]	[#,i,m,m,fn,#]
[#,b,b,b,b,#]	[#,i,m,m,fb,#]

2 WRITING OF LETTER HAMZA

The hamza can be written in five different manners (أ, إ, ؤ, ئ, ء) depending mainly upon:

- its position within the word
- the preceding and the following vowel

As the choice made for coding, was to adhere to a linguistic point of view, there should have been only one code for all these shapes and carrying consonants. But, as it has just been said, to determine the correct writing of hamza, one has to know the surrounding vowels, and it is of common knowledge that the Arabs do not usually write short vowels. These essential data being missing, no algorithm can take place to fulfil this task for a common usage such as display a text on a screen. Thus, the ASMO decided to have distinct codes for the different carriers of hamza, but not of course for their different shapes which can be determined as seen before.

So why is this question of any interest ? If we consider NLP applications for Arabic, it could worth considering this problem at generation stage. For instance many vowel alternations occur in the conjugation of verbs, and when a hamza is present in the verb root, the hamza writing will vary accordingly.

For example the verb قَرَأَ *qara'a* - he (has) read - changes to يَقْرَؤُونَ *yaqra'ūna* - they read (present) - and to قُرِئَ *quri'a* - it (has) been read. And at the generation stage vowels are known even if we decided not to write them. The only alternative would be to put all the forms in a dictionary. At CERTAL, our philosophy is to use all the possible means to reduce the size of dictionaries. Hence this question appeared to us worth studying.

2.1 Rules of hamza writing

1. When a hamza is at the beginning of a word it is written

- over an alif (أ) if the next vowel is an "a" (ـَ) as in أَقْرَأُ *'aqra'u* - I read (present)- or an "u" (ـُ) as in أُكْتُبْ *'uktub* - write ! -
- under an alif (إ) if the next vowel is an "i" (ـِ) as in إِعْلَامٌ *'i'lām* - information

2. When a hamza is within a word (i.e. preceded and followed by some consonant) it is written
   - over an alif (أ) when
     - preceded by a sukūn (ـْ) and followed by an "a" as in يَسْأَلُ *yas'alu* - he asks -
     - preceded by an "a" and followed by a sukūn as in يَأْكُلُ *ya'kulu* - he eats -
     - preceded by an "a" and followed by an "a" as in سَأَلَ *sa'ala* - he (has) asked -
   - over a waw (ؤ) when
     - preceded by a sukūn and followed by an "u" as in يَبْؤُسُ *yab'usu* - he is strong, brave -
     - preceded by an "a" and followed by an "u" or an "ū" as in يَؤُوبُ *ya'ūbu* - to return or to suffer -
     - preceded by a "u" and followed by a sukūn as in يُؤْثِرُ *yu'thiru* - he prefers -
     - preceded by an "u" and followed by an "a" as in يُؤَثِّرُ *yu'aththiru* - he influences -
     - preceded by an "u" and followed by an "u" or an "ū" as in بُؤُوسٌ *bu'ūsun* - distresses -
     - precede by an "ū" and followed by an "u"
   - over a ya (ئ) when
     - preceded by an "i" whatever be the following vowel as in بِئْرٌ *bi'run* - well - بِئَارٌ *bi'ārun* plural of the same word
     - followed by an "i" whatever be the preceding vowel as in قَائِدٌ *qā'idun* - leader, director, commandant,... -
   - without any carrying letter when
     - preceded by an "ā" and followed by an "a" as بَدَاءَةٌ *badā'atun* - beginning -
     - preceded by an "ū" and followed by an "a" as in يَسُوءَانِ *yasū'āni* - they (both) become bad -

3. When a hamza is at the end of a word it is written
   - without any carrier when
     - the preceding vowel is a sukūn[6] as in جُزْءٌ *juz'un* - a part -
     - the preceding vowel is an "ā" as in أَجْزَاءٌ *ajzā'un*, plural of the same word
     - the preceding vowel is an "ū" as in يَسُوءُ *yasū'u* - it becomes bad -
     - the preceding vowel is an "ī" as in يَجِيءُ *yajī'u* - he arrives -
   - over alif when the preceding vowel is an "a" and the following is one of "a", "an", "u", "un" as in مُبْتَدَأٌ *mubtada'un* اَلْمُبْتَدَأُ *al-mubtada'u* مُبْتَدَأً *mubtada'an* اَلْمُبْتَدَأَ *al-mubtada'a*, different forms of the word meaning (grammatical) subject
   - under alif when the preceding vowel is an "a" and the following is "i" or "in" مُبْتَدَإٍ *mubtada'in* اَلْمُبْتَدَإِ *al-mubtada'i*, indirect case of the same word

[6] there are some exception when the preceding consonant is "y" as in شَيْئًا *shay'an* undetermined direct case - a thing -

- over waw when the preceding vowel is "u" as in جَرُؤَ *jaru'a* - he (has) risked - يَجْرُؤُ *yajru'u* - he riskes -
- over ya when the preceding vowel is "i" as in خَاطِئٌ *khati'un* ٱلْخَاطِئَ *al-khati'a* ٱلْخَاطِئِ *al-khati'i* - wrong -

A full account of the rules governing hamza writing have just been given. Usual presentations of hamza writing add to these rules, the rules of madda (آ) writing. Madda is a contraction used for a hamza followed by an ā or a hamza followed itself by a sukūn. This happens in some derivations or conjugations, thus we considerer it as pertaining to the whole set of transformations which occur in those cases.

آكُلُ *ākulu* ← أَأْكُلُ *'a'kulu* -I eat -

آخَذَ *ākhaḍa* ← أَأْخَذَ *'aākhaḍa* - he blamed -

Besides, except for elementary schools and Coranic Recitation, noboby cares about ending short vowels. So, if the last vowel is not long it is treated as it were a sukūn, i.e. no vowel. This is always true of modern arabic and this reduces the number of rules involved at the end of a word.

2.2 A Moore automaton for hamza writing

With the aforementioned restrictions these rules can also be implemented as a Moore automaton.

2.2.1 Source language of the automaton

It follows from the determination rules that we have to know

- if the consonant to be processed is a hamza (whatever its carrier has to be) or not,
- wether a vowel is present before or after the hamza,
- and if so, what are the surrounding vowels (short or long).

Again the presence of a shadda is non pertinent and can be treated as mentioned for the contextual analysis. The alphabet for the source language L_3 can be, using the same method as before :

$$A_3 = \{\#, l, hz, su, a, u, i, \bar{a}, \bar{u}, \bar{i}, \}$$

where hz is a hamza with any carrier, l any consonant other than hamza and su stands for sukūn. The only other constraints for this language are :

1. a sukūn cannot
   - neither follow the first consonant
   - nor follow a consonant already preceded by a sukūn
2. a hamza cannot follow another hamza[7]

The regular expression corresponding to L_3 would be too complicated to be really clarifying so we shall go directly to the definition of a generating automaton for this language.
initial states = {1}
final states: = {21}
transitions

Because of the narrowness of this style columns, the transition tables have been devided in two parts. The last column of the second table gives the output corresponding to every state.

	hz	*l*	*a*	*u*	*i*
1	∅	∅	∅	∅	∅
2	3	4	∅	∅	∅
3	∅	∅	5	6	7
4	∅	∅	8	9	10
5	∅	{17,4}	∅	∅	∅
6	∅	{17,4}	∅	∅	∅
7	∅	{17,4}	∅	∅	∅
8	{18,3}	{17,4}	∅	∅	∅
9	{18,3}	{17,4}	∅	∅	∅
10	{18,3}	{17,4}	∅	∅	∅
11	∅	4	∅	∅	∅
12	∅	4	∅	∅	∅
13	∅	4	∅	∅	∅
14	3	4	∅	∅	∅
15	3	4	∅	∅	∅
16	3	4	∅	∅	∅
17	∅	∅	∅	∅	∅

[7] this is true since we are at writing stage, not derivation or inflection stage

	hz	l	a	u	i
18	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
19	3	4	$\emptyset$	$\emptyset$	$\emptyset$
20	$\emptyset$	4	$\emptyset$	$\emptyset$	$\emptyset$
21	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

	$\bar{a}$	$\bar{u}$	$\bar{\imath}$	su	#	O
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	2	$\emptyset$
2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	2	#
3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	2	hz
4	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	2	l
5	11	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	a
6	$\emptyset$	12	$\emptyset$	$\emptyset$	$\emptyset$	u
7	$\emptyset$	$\emptyset$	13	$\emptyset$	$\emptyset$	i
8	14	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	a
9	$\emptyset$	15	$\emptyset$	$\emptyset$	$\emptyset$	u
10	$\emptyset$	$\emptyset$	16	$\emptyset$	$\emptyset$	i
11	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	21	$\bar{a}$
12	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	21	$\bar{u}$
13	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	21	$\bar{\imath}$
14	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	21	$\bar{a}$
15	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	21	$\bar{u}$
16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	21	$\bar{\imath}$
17	$\emptyset$	$\emptyset$	$\emptyset$	19	$\emptyset$	l
18	$\emptyset$	$\emptyset$	$\emptyset$	20	$\emptyset$	hz
19	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	21	su
20	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	21	su
21	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	#

2.2.2 Target language of the automaton

The only differences with the source language lie in the distinct carriers for the letter hamza:

$$A_4 = \{\#, l, hwc, hoa, hua, how, hoy, su, a, u, i, \bar{a}, \bar{u}, \bar{\imath},\}$$

where *hwc* stands for hamza without a carrier, *hoa* for hamza on alif, *hua* for hamza under alif, *how* for hamza on waw and *hoy* for hamza on ya.

2.2.3 Translation automaton

initial states = {1}
final states: = {36}
transitions

	l	hz	su	a	u	i
1	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
2	$\emptyset$	$\emptyset$	$\emptyset$	4	5	6
3	$\emptyset$	$\emptyset$	$\emptyset$	7	8	9
4	16	{10, 32}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
5	16	{12, 33}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

	l	hz	su	a	u	i
6	16	{14, 34}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
7	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
8	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
9	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
10	$\emptyset$	$\emptyset$	21	22	26	30
11	16	{19, 31}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
12	$\emptyset$	$\emptyset$	24	25	26	30
13	16	{20, 31}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
14	$\emptyset$	$\emptyset$	27	28	29	30
15	16	{21, 31}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
16	$\emptyset$	$\emptyset$	17	4	5	6
17	2	{18, 31}	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
18	$\emptyset$	$\emptyset$	$\emptyset$	22	26	30
19	$\emptyset$	$\emptyset$	$\emptyset$	23	26	30
20	$\emptyset$	$\emptyset$	$\emptyset$	28	29	30
21	2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
22	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
23	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
24	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
25	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
26	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
27	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
28	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
29	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
30	16	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
31	$\emptyset$	$\emptyset$	35	$\emptyset$	$\emptyset$	$\emptyset$
32	$\emptyset$	$\emptyset$	35	$\emptyset$	$\emptyset$	$\emptyset$
33	$\emptyset$	$\emptyset$	35	$\emptyset$	$\emptyset$	$\emptyset$
34	$\emptyset$	$\emptyset$	35	$\emptyset$	$\emptyset$	$\emptyset$
35	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
36	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

	$\bar{a}$	$\bar{u}$	$\bar{\imath}$	#	output
1	$\emptyset$	$\emptyset$	$\emptyset$	1	#
2	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	l
3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
4	11	$\emptyset$	$\emptyset$	$\emptyset$	a
5	$\emptyset$	13	$\emptyset$	$\emptyset$	u
6	$\emptyset$	$\emptyset$	15	$\emptyset$	i
7	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$[hoa, a]$
8	$\emptyset$	13	$\emptyset$	$\emptyset$	$[hoa, u]$
9	$\emptyset$	$\emptyset$	15	$\emptyset$	$[hua, i]$
10	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
11	$\emptyset$	$\emptyset$	$\emptyset$	36	$\bar{a}$
12	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
13	$\emptyset$	$\emptyset$	$\emptyset$	36	$\bar{u}$
14	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

	$\bar{a}$	$\bar{u}$	$\bar{\imath}$	#	output
15	∅	∅	∅	36	$\bar{\imath}$
16	∅	∅	∅	∅	l
17	∅	∅	∅	∅	su
18	∅	∅	∅	∅	∅
19	∅	∅	∅	∅	∅
20	∅	∅	∅	∅	∅
21	∅	∅	∅	∅	$[hoa, su]$
22	∅	∅	∅	∅	$[hoa, a]$
23	∅	∅	∅	∅	$[hwc, a]$
24	∅	∅	∅	∅	$[how, su]$
25	∅	∅	∅	∅	$[how, a]$
26	∅	∅	∅	∅	$[how, u]$
27	∅	∅	∅	∅	$[hoy, su]$
28	∅	∅	∅	∅	$[hoy, a]$
29	∅	∅	∅	∅	$[hoy, u]$
30	∅	∅	∅	∅	$[hoy, i]$
31	∅	∅	∅	∅	hwc
32	∅	∅	∅	∅	hoa
33	∅	∅	∅	∅	how
34	∅	∅	∅	∅	hoy
35	∅	∅	∅	36	su
36	∅	∅	∅	∅	#

2.2.4 Test program results

A PROLOG program similar to the one used for contextual analysis gives the following results:

input	output
[#,hz,a,l,a,â,l,a,hz,su,#]	[#,hoa,a,l,a,â,l,a,hoa,su,#]
[#,hz,a,l,a,â,l,u,hz,su,#]	[#,hoa,a,l,a,â,l,u,how,su,#]
[#,hz,a,l,a,â,l,i,hz,su,#]	[#,hoa,a,l,a,â,l,i,hoy,su,#]
[#,hz,a,l,a,hz,a,l,a,â,#]	[#,hoa,a,l,a,hoa,a,l,a,â,#]
[#,hz,a,l,a,hz,u,l,a,â,#]	[#,hoa,a,l,a,how,u,l,a,â,#]
[#,hz,a,l,a,hz,i,l,a,â,#]	[#,hoa,a,l,a,hoy,i,l,a,â,#]
[#,hz,a,l,u,hz,a,l,a,â,#]	[#,hoa,a,l,u,how,a,l,a,â,#]
[#,hz,a,l,u,hz,u,l,a,â,#]	[#,hoa,a,l,u,how,u,l,a,â,#]
[#,hz,a,l,u,hz,i,l,a,â,#]	[#,hoa,a,l,u,hoy,i,l,a,â,#]
[#,hz,a,l,i,hz,a,l,a,â,#]	[#,hoa,a,l,i,hoy,a,l,a,â,#]
[#,hz,a,l,i,hz,u,l,a,â,#]	[#,hoa,a,l,i,hoy,u,l,a,â,#]
[#,hz,a,l,i,hz,i,l,a,â,#]	[#,hoa,a,l,i,hoy,i,l,a,â,#]
[#,hz,a,l,i,hz,i,l,a,â,#]	[#,hoa,a,l,i,hoy,i,l,a,â,#]
[#,hz,u,l,a,â,l,a,hz,su,#]	[#,hoa,u,l,a,â,l,a,hoa,su,#]
[#,hz,i,l,a,â,l,a,hz,su,#]	[#,hua,i,l,a,â,l,a,hoa,su,#]
[#,l,a,l,a,hz,a,l,a,â,#]	[#,l,a,l,a,hoa,a,l,a,â,#]
[#,l,a,l,a,hz,u,l,a,â,#]	[#,l,a,l,a,how,u,l,a,â,#]
[#,l,a,l,a,hz,i,l,a,â,#]	[#,l,a,l,a,hoy,i,l,a,â,#]
[#,l,a,l,u,hz,a,l,a,â,#]	[#,l,a,l,u,how,a,l,a,â,#]
[#,l,a,l,u,hz,u,l,a,â,#]	[#,l,a,l,u,how,u,l,a,â,#]
[#,l,a,l,u,hz,i,l,a,â,#]	[#,l,a,l,u,hoy,i,l,a,â,#]
[#,l,a,l,i,hz,a,l,a,â,#]	[#,l,a,l,i,hoy,a,l,a,â,#]
[#,l,a,l,i,hz,u,l,a,â,#]	[#,l,a,l,i,hoy,u,l,a,â,#]
[#,l,a,l,i,hz,i,l,a,â,#]	[#,l,a,l,i,hoy,i,l,a,â,#]
[#,l,a,â,hz,a,l,a,hz,su,#]	[#,l,a,â,hss,a,l,a,hoa,su,#]

input	output
[#,l,a,â,hz,u,l,a,hz,su,#]	[#,l,a,â,how,u,l,a,hoa,su,#]
[#,l,a,â,hz,i,l,a,hz,su,#]	[#,l,a,â,hoy,i,l,a,hoa,su,#]
[#,l,u,l,u,û,hz,a,l,su,#]	[#,l,u,l,u,û,hss,a,l,su,#]
[#,l,u,l,u,û,hz,u,l,su,#]	[#,l,u,l,u,û,how,u,l,su,#]
[#,l,u,l,u,û,hz,i,l,su,#]	[#,l,u,l,u,û,hoy,i,l,su,#]
[#,l,u,l,i,î,hz,a,l,su,#]	[#,l,u,l,i,î,hoy,a,l,su,#]
[#,l,u,l,i,î,hz,u,l,su,#]	[#,l,u,l,i,î,hoy,u,l,su,#]
[#,l,u,l,i,î,hz,i,l,su,#]	[#,l,u,l,i,î,hoy,i,l,su,#]
[#,l,u,l,su,hz,a,l,su,#]	[#,l,u,l,su,hoa,a,l,su,#]
[#,l,u,l,su,hz,i,l,su,#]	[#,l,u,l,su,hoy,i,l,su,#]
[#,l,u,l,su,hz,u,l,su,#]	[#,l,u,l,su,how,u,l,su,#]
[#,l,u,l,a,hz,su,l,a,â,#]	[#,l,u,l,a,hoa,su,l,a,â,#]
[#,l,u,l,u,hz,su,l,a,â,#]	[#,l,u,l,u,how,su,l,a,â,#]
[#,l,u,l,î,hz,su,l,a,â,#]	[#,l,u,l,i,hoy,su,l,a,â,#]

CONCLUSION

As a matter of conclusion we hope to have shown that, through a careful choice of a formal language, linguistic rules can be specified as tractable automata.

References

A. V. Aho and J. D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1: Parsing. Prentice-Hall.

Arab League Arab Organization for Standardization and Metrology (ASMO), 1982. *Data processing 7 bit Coded Arabic Character Set for Information Interchange.*

Arab School on Science and Technology 1st Fall Session Rabat Morocco. 1983. *Applied Arabic Linguistics and Signal & Information Processing*, P.O. Box 7028 Damascus Syria.

Arab School of Science and Technology 7th Summer Session, Zabadani Valley - Syria. 1985. *Informatics and Applied Arabic Linguistics*, P.O. Box 7028 Damascus Syria.

R. Blachère and M. Gaudefroy-Demombynes. 1952. *Grammaire de l'arabe classique.* G.P. Maisonneuve & Larose, 3^e edition.

1985. *Computer Processing of the Arabic Language.* April 14-16, 1985 Kuwait.

M. Fanton. 1997. L'écriture arabe : du manuscrit à l'ordinateur. *La Tribune Internationale des Langues vivantes*, (21), mai.

J. E. Hopcroft and J. D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley.

K. Lagally. 1992. ArabTEX a system for typesetting arabic user manual version 3.00. Technical Report 1993/11, Universität Stuttgart, Fakultät Informatik, Breitwiesenstraße 20-22, 70565 Stuttgart, Germany. Document électronique fourni avec le logiciel.

A. Lakhdar Ghazal. 1983. L'alphabet arabe et les machines. In *Applied Arabic Linguistics and Signal & Information Processing* (Ara, 1983), pages 233–258.

W. Wright. 1859. *A Grammar of the Arabic Language.* Cambridge University Press, 3^e edition.