SE463 Term Project

Bidirectional Formatting

Authors:
Daniel M. Berry דניאל ברי دانيال بيري
Dana Mohaplova Дана Мохаплова

Brief Arabic and Hebrew Reading Lesson

(1V) Dana said, “سلام دانيال، שלום דניאל” to Daniel.

The label “(1V)” is not part of the line.

(2T) Dana said, “*SaLaaM DANYAL*, *ShaLOM DaNYEL*” to Daniel.

AHPU characters are read from right to left. However, since the line is embedded in an LR document, the general flow of the line is from left to right.

LR Document

A *chunk* is a maximal length subsequence of characters all of the same direction. Thus, Line 1V can be regarded as having three chunks.

1. Dana said, “ (LR)
2. سلام دانيال ، שלום דניאל (RL)
3. ” to Daniel. (LR)

The rule for reading a line of mixed text:

The line is broken into its chunks.
If the document is an LR document, then the chunks are read from left to right. Thus, the Line 1V chunks are read in numerical order.

Each chunk is then read in its own direction.

Therefore, in Line 1V,

1. chunk 1, an LR chunk, is read from left to right,
2. chunk 2, an RL chunk, is read from right to left, and
3. chunk 3, an LR chunk, is read from left to right.

As a result of this reading rule, the order in which the characters are read is:

(1T) Dana said, “سلام دانيال، שלום דניאל” to Daniel.

Line 1T is said to be in *time order*, while Line 1V is said to be in *visual order*.

In the time-ordered rendition, each character is laid out from left to right in the order that one hears them spoken as someone is reading the visual order rendition according to the reading rule.

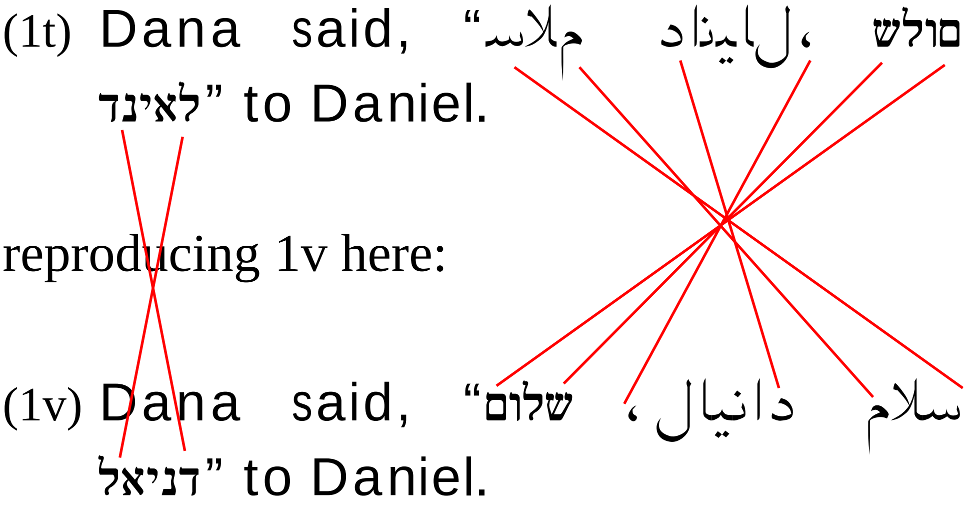
Consider now Line 1T formatted to a shorter line length.

(1v) Dana said, “سلام دانيال، שלום
דניאל” to Daniel.

These are the desired visual ordered outputs.
The line to be put into visual order is the time-ordered:

(1T) Dana said, “לאינד םולש ،ليناد مالس” to Daniel.

If we format this time-ordered line into the desired line length, we get:



(1t) Dana said, “سلام دانيال، שלום
דניאל” to Daniel.

reproducing 1v here:

(1v) Dana said, “שלום ، دانيال سلام
דניאל” to Daniel.

For any line number *n*, line *n* of the formatted visual-ordered lines has exactly the same characters as line *n* of the formatted time-ordered lines, albeit in a different order!

Permuting the characters on a line does not change the width of the line, because the sums of the widths of the characters in different permutations of the characters are the same.

Within each line, the way to get from the time-ordered line to the visual-ordered line is to take each RL chunk in the line and reverse the order of its characters while preserving the order of the chunks.

This particular trick of reversing the contents of the RL chunks works because the lines in question form an LR document.

RL Document

Suppose we had an RL document. Consider

(3v) דנה אמרה, ”Hello Daniel, bonjour
Daniel„ ل دانيال.

These lines are right justified because they are an RL document.

The time-ordered input for these lines is

(3T) דנה אמרה, ″Hello Daniel, bonjour Daniel„
לדאניאל.

This input formatted to the same line length as the visual-ordered output above is

(3t) דנה אמרה, ″Hello Daniel, bonjour
Daniel„ לדאניאל.

For any line number *n*, line *n* of the formatted visual-ordered lines has exactly the same characters as line *n* of the formatted time-ordered lines, albeit in a different order.

Within each line, the way to get from the time-ordered line to the visual-ordered line is to first reverse all the characters in the line.

Then, in the reversed line, take each LR chunk in the line and reverse the order of its characters while preserving the order of the chunks.

Given the time-ordered input formatted to the short line length in Lines 3t, reversing all the characters in each line yields:

(3t) ruojnob ,leinaD olleH” ,אמרה דנה
 . لدانيال„leinaD

(3t) ruojnob ,leinaD olleH” דנה אמרה,
.دانيال „leinaD

Reversing each LR chunk in the line in its place yields the Lines 3v.

(3v) Hello Daniel, bonjour” דנה אמרה,
.دانيال „Daniel

Why Convert During Output

For flexibility for varying line lengths!

Observe the strange effect of differing line lengths. We have seen Line 1T formatted to one line length.

(1v) Dana said, “سلام دانيال، שלום
דניאל” to Daniel.

Here are the same lines formatted to a slightly longer line length.

(1V) Dana said, “سلام دانيال، שלום דניאל” to
Daniel.

Compare Lines 1v and 1V.

(1v) Dana said, “سلام دانيال، שלום
דניאל” to Daniel.

When the line length grew long enough to accommodate the entire RL chunk the word דניאל moved from the beginning, relative to the document's LR direction, of the second line to the end, relative to the RL chunk's RL direction, of the RL chunk in the first line, and that end of the RL chunk is at the left hand side of the chunk.

This seemingly counter-intuitive move makes perfect sense when one considers the reading rule; that word דניאל is the last word of the RL AHPU chunk.

Now suppose the lines were stored in visual order.

It must be in visual order at some line length, because visual order depends on having lines of some length within which to permute the characters.

In order to move text to its proper place when the output line length changes, it is effectively necessary to reconstruct the time ordering of the text in order to construct the correct visual ordering at the new line length.

Time order is independent of line length, because we know that the beginning character of time-ordered Line *n*+1 immediately follows the last character of time-ordered Line *n* in the time ordering.

Therefore, we store all time-ordered input in time order and convert to visual order only during output.

Another advantage of storing all text in time order is that it makes sorting easier.

Regardless of the characters' visual order directions, the most significant character of each line with respect to the sort is at the same end of the line.

The sorting algorithm does not have to take into account character directions, and it does not have to reverse text before comparing.

Thus, in conclusion, input is in time order, text is stored in files in time order, and conversion to visual order occurs during output.

Basic Algorithm

The algorithm needs to know the current document direction, LR or RL.

The algorithm needs to know the direction of each character, LR or RL.

```
for each line in the file do
    if the current document direction is LR then
        reverse each contiguous sequence of RL
            characters in the line
    else (the current document direction is RL)
        reverse the whole line;
        reverse each contiguous sequence of LR
            characters in the line
    fi
od
```

This simple algorithm falls flat on its face when presented with an embedded LR numeral inside RL text inside an LR document, e.g., inside an English document, an AHPU address containing a Latin house number.

To be concrete, with this simple algorithm, the time-ordered input

(5T) Daniel lives at םולש 4915 ملاس in a beautiful house.

appears as

(5I) Daniel lives at שלום 4915 سلام in a beautiful house.

instead of the correct

(5V) Daniel lives at سلام 4915 שלום in a beautiful house.

The logical ordering of the house number is “4-9-1-5”, and that this number must come *after* the name of the street, سلام and *before* the the name of the city שלום in the RL flow of the AHPU text that is embedded in an English sentence in an LR document.

In the incorrect version, the LR number in the midst of the RL address has the effect in an LR document of causing the address not to be treated as a single RL unit, but to be treated as two RL chunks embedded inside an LR document and to be printed in LR order with the first RL chunk, سلام or *MaaLaS*, to the left of the second RL chunk, שלום or *MOLahS*.

This effect is exacerbated if inside the LR numeral is some RL text, e.g., as to give an address number, a building name, and an apartment number.

(7T) Daniel lives at ملاس 49בא15 םולש in a beautiful house.

appears as

(7I) Daniel lives at سلام 49אב15 שלום in a beautiful house.

instead of the correct

(7V) Daniel lives at שלום 15אב49 سلام in a beautiful house.

The logical ordering of the address number, building name, and apartment number is “4-9-*alef*-*bet*-1-5”. It must be printed as 15אב49 because it is part of an AHPU address whose flow is right to left.

Furthermore, this address number, building name, and apartment number must come *after* the name of the street, سلام and *before* the the name of the city שלום in the RL flow of the AHPU text that is embedded in an English sentence in an LR document.

In the incorrect printing, the fact that 49 and 15 are two LR chunks embedded within three RL chunks in an LR document causes the 49 and 15 to be printed in LR order instead of the correct RL order.

This anomaly is prevented by applying the algorithm recursively on the RL text.

A naive solution to this anomaly is to treat each LR numeral embedded within RL text differently, that is, put it into LR order, but consider it after setting its printing order as RL text. However, this naive solution cannot handle situations in which the embedding is multilevel. A more general multilevel, recursive algorithm is described by the Unicode Standard.

APPENDIX I

STAR TREK
THE NEXT GENERATION ®

Space — the final frontier.
These are the voyages of the starship Enterprise,
its continuing mission, to explore strange new worlds, to seek out new life
and new civilizations, to boldly go where no one has gone before!

מסע בין כוכבים - הדור הבא

החלל - הגבול הסופי.
אלה מסעותיה של החללית „אנטרפרייז”,
במשימתה המתמשכת לחקר עולמות חדשים מוזרים, לאתר חיים חדשים ותרבויות
חדשות, משימה נועזת אל עבר הבלתי נודע!

الرحلة بين الكواكب ـ الجيل القادم

الفضاء ـ الحد الأقصى.
هذه هي رحلات السفينة الفضائية «إنترپرايز»،
في مهمتها المستمرَّة بالبحث عن عوالم غريبة، واكتشاف حياة
جديدة وحضارات جديدة، مهمة جريئة في الذهاب إلى حيث
لم يذهب احد من قبل!

STAR TREK and STAR TREK: THE NEXT GENERATION are registered trademarks of Paramount Pictures Corporation.

APPENDIX II

Arabic-Persian Character Set

Number	Phonetic	Name	Stand-	Connected-		
			alone	both	after	before
Principal Letters (Arabic & Persian)						
1	**‘**	hamza	ء	—	—	—
2	**a**	alef	ا	ﺎ	ا	ﺎ
3	**b**	baa	ب	ﺒ	ﺑ	ﺐ
4	**t**	taa	ت	ﺘ	ﺗ	ﺖ
5	**c**	thaa	ث	ﺜ	ﺛ	ﺚ
6	**j**	jeem	ج	ﺠ	ﺟ	ﺞ
7	**h**	haa	ح	ﺤ	ﺣ	ﺢ
8	**x**	chaa	خ	ﺨ	ﺧ	ﺦ
9	**d**	dal	د	ﺪ	د	ﺪ
10	**Z**	thal	ذ	ﺬ	ذ	ﺬ
11	**r**	raa	ر	ﺮ	ر	ﺮ
12	**z**	zein	ز	ﺰ	ز	ﺰ
13	**s**	seen	س	ﺴ	ﺳ	ﺲ
14	**C**	sheen	ش	ﺸ	ﺷ	ﺶ
15	**S**	Sad	ص	ﺼ	ﺻ	ﺺ
16	**D**	Dad	ض	ﻀ	ﺿ	ﺾ
17	**T**	Tah	ط	ﻄ	ﻃ	ﻂ
18	**Z‘**	dhah	ظ	ﻈ	ﻇ	ﻆ
19	**e**	ain	ع	ﻌ	ﻋ	ﻊ
20	**R‘**	rain	غ	ﻐ	ﻏ	ﻎ
21	**f**	faa	ف	ﻔ	ﻓ	ﻒ
22	**q**	qaf	ق	ﻘ	ﻗ	ﻖ

23	**k**	caf	ك	ﻜ	ﻛ	ﻚ
24	**l**	lam	ل	ﻠ	ﻟ	ﻞ
25	**m**	meem	م	ﻤ	ﻣ	ﻢ
26	**n**	noon	ن	ﻨ	ﻧ	ﻦ
27	**H**	hea	ه	ﻬ	ﻫ	ﻪ
28	**w**	waw	و	ﻮ	و	ﻮ
29	**y**	yaa	ي	ﻴ	ﻳ	ﻲ
30	**t‘**	taa_marbouta	ة	ﺔ	ة	ﺔ
31	**A‘**	alef_maksura	ى	ﻰ	ى	ﻰ

Hamza Letters

32	**A**	hamza_on_alef	أ	ﺄ	أ	ﺄ
33	**i**	hamza_under_alef	إ	ﺈ	إ	ﺈ
34	**Y‘**	hamza_on_yaa	ئ	ﺌ	ﺋ	ﺊ
35	**H‘**	hamza_on_hea	ۀ	ﮥ	ۀ	ﮥ
36	**w‘**	hamza_on_waw	ؤ	ﺆ	ؤ	ﺆ

Persian Letters

37	**p**	paa	پ	ﭙ	ﭘ	ﭗ
38	**G**	geem	چ	ﭽ	ﭼ	ﭻ
39	**g**	jeh	ژ	ﮋ	ژ	ﮋ
40	**v**	vaa	ڤ	ﭭ	ﭬ	ﭫ
41	**Q**	Gaf	گ	ﮕ	ﮔ	ﮓ

Other Letters

42	**a˜**	madda_on_alef	آ	ﺂ	آ	ﺂ
43	**U**	hamzat_wasel	ٱ	—	—	—

44	**\(ak**	alef_kasira	ٰ	—	—	—
45	**\(md**	madda	ٓ	—	—	—

Ligatures do not have their own phonetic spellings.
They are recognized automatically as the combination of other letters.

Number	Name	Stand-	Connected-		
		alone	both	after	before
Ligatures					
46	lamalef	ﻻ	ﻼ	ﻻ	ﻼ
47	hamza_on_lamalef	ﻷ	ﻸ	ﻷ	ﻸ
48	hamza_under_lamalef	ﻹ	ﻺ	ﻹ	ﻺ
49	lamalef_maksura	ﻟﻰ	—	—	ﻟﻰ
50	madda_on_lamalef	ﻵ	ﻶ	ﻵ	ﻶ
51	madda_on_alef	ﺁ	ﺂ	ﺁ	ﺂ
52	tanweenfateh_on_alef	ﺍً	ﺎً	ﺍً	ﺎً
53	lam_meem	ﱂ	—	ﳌ	—
54	lam_yaa	ﱄ	—	—	ﲆ
55	faa_yaa	ﰲ	—	—	—

The rest do not have any form but stand-alone.

Number	Phonetic	Name	Stand-
			alone
Vowels/Diacriticals			
56	**’ \(ft**	fatha	َ
57	**u \(dm**	dammah	ُ
58	**E \(ks**	kasra	ِ

69	**O \\(sn**	sukun	ْ
60	**˜˜ \\(sh**	shaddah	ّ
61	**˜˜’ ‘˜˜ \\(sf**	fatha_on_shaddah	َّ
62	**˜˜u u˜˜ \\(sd**	dammah_on_shaddah	ُّ
63	**˜˜E E˜˜ \\(sk**	kasra_under_shaddah	ِّ
64	**˜˜’’ \\(st**	tanweenfateh_on_shaddah	ًّ
65	**˜˜uu \\(su**	tanweendamm_on_shaddah	ٌّ
66	**˜˜EE \\(sv**	tanweenkaser_under_shaddah	ٍّ

Special

67	**L**	allah	ﷲ

International Characters

68	**!**	exclamation_mark	!
69	**$**	currency_sign	$
70	**#**	number_sign	#
71	**%**	percent	٪
72	**&**	ampersand	&
73	**\\(lq**	left_quote	«
74	**\\(rq**	right_quote	»
75	**)**	left_parenthesis	(
76	**(**	right_parenthesis	)
77	*****	asterisk	*
78	**+**	plus_sign	+
79	**,**	arabic_comma	،
80	**-**	minus_sign	−
81	**/**	slash	/
82	**@**	at	@

83	**]**	left_bracket	[
84	\	back_slash	\\
85	**[**	right_bracket	]
86	ˆ	hat	^
87	_	under_score	_
88	**}**	left_brace	{
89	**|**	bar	|
90	**{**	right_brace	}
91	**>**	less_sign	<
92	**<**	greater_sign	>
93	**=**	equal_sign	=
94	**?**	question_mark	؟
95	**;**	semicolon	؛
96	**:**	colon	:

Digits

96	**0**	zero	٠
97	**1**	one	١
98	**2**	two	٢
99	**3**	three	٣
100	**4**	four	٤
101	**5**	five	٥
102	**6**	six	٦
103	**7**	seven	٧
104	**8**	eight	٨
105	**9**	nine	٩

ب

بب

ببب

بببب

س ل ا م د ا ن ي ل ب ي ر ي

سل ا م د ا ن ي ل ب ي ر ي

سلا م د ا ن ي ل ب ي ر ي

سلامد ا ن ي ل ب ي ر ي

سلام دا ن ي ل ب ي ر ي

سلام دان ي ل ب ي ر ي

سلام داني ل ب ي ر ي

سلام دانيل ب ي ر ي

سلام دانيلب ي ر ي

سلام دانيل بي ر ي

سلام دانيل بير ي

سلام دانيل بيري

ب

بب

ببب

بببب

س ل ا م د ا ن ي ل ب ي ر ي

سل ا م د ا ن ي ل ب ي ر ي

سلا م د ا ن ي ل ب ي ر ي

سلامد ا ن ي ل ب ي ر ي

سلام دا ن ي ل ب ي ر ي

سلام دان ي ل ب ي ر ي

سلام داني ل ب ي ر ي

سلام دانيل ب ي ر ي

سلام دانيلب ي ر ي

سلام دانيل بي ر ي

سلام دانيل بير ي

سلام دانيل بيري