SE463 Term Project

Bidirectional Formatting

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

Background

The property that the Arabic, Hebrew, Persian, and Urdu languages have in common is that they are written from right to left. Nearly all* of the languages of Europe are written from left to right. It is not too much of a simplification to consider each language to have its own alphabet. Even when two languages, e.g., English and French, seem to share the same alphabet, i.e., Latin, one may not use all the letters, and they may have differences in the use of accents and diacriticals. For example, French does not use "w" except for foreign words. French uses the accents "´", "`", "^", and English does not, except when it has borrowed words from other languages. One can regard the French alphabet as being the set of all possibly accented letters, i.e., including "a", "á", "à", "o", and "ô". Therefore, instead of talking about languages, we shall talk about alphabets. The Arabic, Hebrew, Persian, and Urdu (AHPU) alphabets are called *right-to-left* (RL) alphabets and their characters are called RL characters. The European alphabets are called *left-to-right* (LR) alphabets and their characters are called LR characters.

Mixed text is text consisting of both LR and RL characters from both LR and RL alphabets. The treatment of mixed text is interesting. Consider the mixed text line,

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

The label "(1v)" is not part of the line. For those of you that cannot read AHPU, Line 1v with the AHPU characters rendered phonetically in script-like Latin characters is

(2t) Dana said, "*SaLaaM DANYAL, ShaLOM DaNYEL*" to Daniel.

Each AHPU character is represented by potentially several consecutive Latin letters, the first of which is upper case and the rest of which are lower case. Each AHPU letter is a consonant, a vowel, or a consonant *with* a vowel. Thus "س" is rendered as "*Sa*".

What is interesting about the mixed text is that the 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. 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, "
2. שלום דניאל ، دانيال سلام
3. " to Daniel.

Observe that the first chunk comma, the two quotation marks, and the period are LR characters. The second chunk comma, that appears to be backwards and upsidedown with respect to the normal Latin comma, is an RL character. Note also that by the maximal length property of the chunks, consecutive chunks of a single line of mixed text are in

* While we know of no exceptions, we are not 100% certain that there are no exceptions, so we are being safe and saying "Nearly all"

alternating directions. For the example line, the directions of the chunks are LR, RL, and LR, in that order.

The rule for reading a line of mixed text is that 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 captured by the following rendition of Line 1v.

(1t) Dana said, "سلام دانیال، 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.

Thus, if we use the phonetic representation of AHPU characters, the time-ordered Line 1t corresponds to

(2t) Dana said, "*SaLaaM DANYAL, ShaLOM DaNYEL*" to Daniel.

while the visual-ordered Line 1v corresponds to

(2v) Dana said, "*LEYNaD MOLahS ,LAYNAD MaaLaS*" to Daniel.

From now on, whenever we show RL characters in the script-like Latin characters, we will print them in visual order from right to left, unless we are trying to show the time ordering of the characters.

Processing Mixed Text

Have you ever wondered how web sites with AHPU text work? Clearly, the input to create a file should come in time order, because time order is the order in which one hears or thinks the characters, and if one is typing as one hears or thinks of the text, it is helpful if he or she can type the characters as he or she hears or thinks them. Equally clearly, when the contents of a file are displayed or printed, the characters should be displayed or printed in visual order, because if the characters are in visual order, then a reader following the reading rule will reconstruct the time ordering of the characters as he or she pronounces or thinks what he or she has read. The question is, "When is the best time to convert from time order to visual order?" The choices are "during input" or "during output". After years of trying different options, there is near universal agreement that the conversion should be done during output. After we have seen the algorithm, we shall explain why conversion should be done during output.

Consider now Lines 1t and 2t formatted to a shorter line length.

(1vs) Dana said, "سلام دانیال ،
שלום דניאל" to Daniel.

(2vs) Dana said, "*,LAYNAD MaaLaS*
LEYNaD MOLahS" to Daniel.

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

(1t) Dana said, "سلام دانیال، שלום דניאל" to Daniel.

(2t) Dana said, "*SaLaaM DANYAL, ShaLOM DaNYEL*" to Daniel.

If we format these time-ordered lines into the desired line length, we get

(1ts) Dana said, "سلام دانیال، שלום
דניאל" to Daniel.

(2ts) Dana said, "*SaLaaM DANYAL,*
ShaLOM DaNYEL" to Daniel.

If you compare the formatted visual-ordered lines with the formatted time-ordered lines, you will see that 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. (Please verify from the two lists of formatted lines that this reversal of RL chunks has the desired effect.)

This particular trick of reversing the contents of the RL chunks works because the lines in question form an LR document. That is, the direction of the whole text is from left to right.

Suppose we had an RL document. Consider

(3vs) Hello Daniel, bonjour" ,דנה אמרה
Daniel„ لدانیال.

(4vs) Hello Daniel," ,*HaRMA HaNaD*
.*LAYNADL* " bonjour Daniel

These lines are right justified because they constitute an RL document. The time-ordered input for these lines is

(3t) הרמא הנד, "Hello Daniel, bonjour Daniel„ لدانیال.

(4t) *DaNaH AMRaH,* "Hello Daniel, bonjour Daniel" *LDANYAL.*

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

(3ts) הרמא הנד, "Hello Daniel, bonjour
Daniel„ لدانیال.

(4ts) *DaNaH AMRaH,* "Hello Daniel,
bonjour Daniel" *LDANYAL.*

Here again, 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 3ts and 4ts, reversing all the characters in each line yields:

(3tsr) ruojnob ,leinaD olleH" ,דנה אמרה
."leinaD„ל دانيال

(4tsr) ,leinaD olleH" ,*HaRMA HaNaD*
LAYNADL "leinaD ruojnob

Reversing each LR chunk in the line in its place yields the Lines 3vs and 4vs. Please verify from the two lists of formatted lines that this reversal of whole lines and then reversal of LR chunks has the desired effect.

Why Convert During Output

The main reason that we have found it best to convert from time order to visual order during output is flexibility for varying line lengths. Obesrver the strange effect of differing line lengths. We have seen Lines 1t and 2t formatted to one line length.

(1vs) Dana said, "שלום ، دانيال سلام
דניאל" to Daniel.

(2vs) Dana said, ",*LAYNAD MaaLaS*
LEYNaD MOLahS" to Daniel.

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

(1vm) Dana said, "שלום דניאל ، دانيال سلام" to
Daniel.

(2vm) Dana said, "*MOLahS ,LAYNAD MaaLaS*
LEYNaD" to Daniel.

Compare Lines 1vs and 1vm. 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. The same observation can be made about the word *MOLahS* in Lines 2vs and 2vm.

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

Based on these examples, it is possible to write a general algorithm that works on each line of a time-ordered file to convert that line into visual order. The algorithm needs to know the current document direction, which can be LR or RL. It needs to know also the direction of each character it finds so that it can break the line into single-directional chunks, each consisting of maximal length sequences of characters all of the same direction. At the top level of abstraction, the algorithm reads the characters of each line, and permutes these characters so that the single-directional chunks of a line flow in the current document's direction and the characters in each chunk flow in the chunk's single direction.

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 algorithm assumes that each line is in only one document direction. Thus, it will be necessary to assure either that document direction changes occur only at line boundaries or to decide that the document direction that applies for a line is that in effect at one well-defined point in the line, e.g., at the end, as measured by time order.

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; the LR numeral splits the RL text into two pieces and the two pieces end up being in the document's LR order relative to each other rather than the required RL order relative to each other. This anomaly is prevented by applying the algorithm recursively on the RL text. To be concrete, without this provision, the time-ordered input

(5t) Daniel lives at سלام 4915 שולש in a beautiful house.

(6t) Daniel lives at *SaLaaM* 4915 *ShaLOM* in a beautiful house.

would appear as

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

(6i) Daniel lives at *MaaLaS* 4915 *MOLahS* in a beautiful house.

instead of the correct

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

(6v) Daniel lives at *MOLahS* 4915 *MaaLaS* in a beautiful house.

Note that 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 سلام 49בא15 שולש in a beautiful house.

(8t) Daniel lives at *SaLaaM* 49*AB*15 *ShaLOM* in a beautiful house.

would appear as

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

(8i) Daniel lives at *MaaLaS* 49*BA*15 *MOLahS* in a beautiful house.

instead of the correct

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

(8v) Daniel lives at *MOLahS* 15*BA*49 *MaaLaS* 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.

An analogous problem would happend with an embedded RL numeral inside LR text inside an RL document. However, in practice, this problem does not arise because in AHPU, with the exception of an ancient form of numerals in Hebrew, numerals are written with the most significant digit to the left, as in English (and other LR languages).

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, in what is item 5 in the list of sources of information in the next section.

To process the time-order input,

(7t) Daniel lives at سلام 49בא15 שולש in a beautiful house.

the Unicode algorithm would first break the text, with each space shown as an undescore ("_") of the current font, into runs (= chunks) and assign levels (even level = LR, odd level = RL):

Level	Run
0	Daniel_lives_at_
1	ـمﻻسـ
2	49
1	בא
2	15
1	__סולש
0	_in_a_beautiful_house.

Then it would begin a recursive reversal, level by level from the maximum down to 1; at each Level *i* less than the maximum, each run of text at Levels *i* through the maximum gets reversed in place.

Reverse Level 2:

(7tr2) Daniel_lives_at_ـمﻻسـ94בא51__סולש_in_a_beautiful_house.

Reverse Levels 1–2:

(7tr1) Daniel_lives_at_שלום__15אב49ـسلامـ_in_a_beautiful_house.

This is the visual-order output.

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

=

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

Sources of information:

1. A website with Arabic text: http://www.aljazeera.net/

2. A website with Hebrew text: http://www.haaretz.co.il/

3. How to view an Arabic Web site: http://www.arabion.net/arbtxt.html and http://arabic.tripod.com/ViewingArabic.htm

4. How to view a Hebrew Web site: http://www.hebrewblackboard.com/i-how-to-read-hebrew.php

5. A website about the UNICODE standard for bidirectional text: http://www.unicode.org/reports/tr9/

6. Papers by Berry *et al* on bi-directional formatting:
http://se.uwaterloo.ca/~dberry/FTP_SITE/reprints.journals.conferences/BuchmanBerryGonczarowski1985BidiTroff.pdf
http://se.uwaterloo.ca/~dberry/FTP_SITE/reprints.journals.conferences/arabic.journal.paper.pdf
http://se.uwaterloo.ca/~dberry/FTP_SITE/reprints.journals.conferences/keshide.journal.paper.pdf

7. Paper by Berry *et al* on bidirectional editing with vi.iv:
http://se.uwaterloo.ca/~dberry/FTP_SITE/reprints.journals.conferences/vi.iv.journal.paper.pdf

8. Site about a bidirectional editor, Yudit: http://www.yudit.org/bidi/userguide.html

9. Paper by Berry *et al* on tri-directional formatting:
http://se.uwaterloo.ca/~dberry/FTP_SITE/reprints.journals.conferences/full.tri.dir.journal.paper.pdf

Examples

Below are the three examples presented in a variety of line widths so that you can see the effects of line breaks at different points on the reversing algorithm. In each case, the example is presented formatted to the given line length in time order and then it is presented in the same line length in visual order. First, we give the LR documents.

Dana said, "سלام لاينداد، שולש
לאינד" to Daniel.

Dana said, "سلام دانيال، שלום
דניאל" to Daniel.

Dana said, "*SaLaaM DANYAL,*
ShaLOM DaNYEL" to Daniel.

Dana said, "*,LAYNAD MaaLaS*
LEYNaD MOLahS" to Daniel.

Daniel lives at سلام 4915 שולש
in a beautiful house.

Daniel lives at שלום 4915 سلام
in a beautiful house.

Daniel lives at *SaLaaM* 4915
ShaLOM in a beautiful house.

Daniel lives at 4915 *MaaLaS*
MOLahS in a beautiful house.

Daniel lives at سلام 49בא15
שולש in a beautiful house.

Daniel lives at 15אב49 سلام
שלום in a beautiful house.

Daniel lives at *SaLaaM* 49*AB*15
ShaLOM in a beautiful house.

Daniel lives at 15*BA*49 *MaaLaS*
MOLahS in a beautiful house.

Dana said, "לאינד סולש ،لايناد ملاسم" to Daniel.

Dana said, "שלום דניאל ، دانيال سلام" to Daniel.

Dana said, "*SaLaaM DANYAL, ShaLOM DaNYEL*" to Daniel.

Dana said, "*MOLahS ,LAYNAD MaaLaS LEYNaD*" to Daniel.

Daniel lives at مسلا 4915 סולש in a beautiful house.

Daniel lives at שלום 4915 سلام in a beautiful house.

Daniel lives at *SaLaaM* 4915 *ShaLOM* in a beautiful house.

Daniel lives at *MOLahS* 4915 *MaaLaS* in a beautiful house.

Daniel lives at مسلا 49בא15 סולש in a beautiful house.

Daniel lives at שלום 15אב49 سلام in a beautiful house.

Daniel lives at *SaLaaM* 49*AB*15 *ShaLOM* in a beautiful house.

Daniel lives at *MOLahS* 15*BA*49 *MaaLaS* in a beautiful house.

Dana said, "לאינד סולש ،لايناد ملاسم" to Daniel.

Dana said, "שלום דניאל ، دانيال سلام" to Daniel.

Dana said, "*SaLaaM DANYAL, ShaLOM DaNYEL*" to Daniel.

Dana said, "*LEYNaD MOLahS ,LAYNAD MaaLaS*" to Daniel.

Daniel lives at سلام 4915 שולש in a beautiful house.

Daniel lives at שלום 4915 سلام in a beautiful house.

Daniel lives at *SaLaaM* 4915 *ShaLOM* in a beautiful house.

Daniel lives at *MOLahS* 4915 *MaaLaS* in a beautiful house.

Daniel lives at سلام 49בא15 שולש in a beautiful house.

Daniel lives at שלום 15אב49 سلام in a beautiful house.

Daniel lives at *SaLaaM* 49*AB*15 *ShaLOM* in a beautiful house.

Daniel lives at *MOLahS* 15*BA*49 *MaaLaS* in a beautiful house.

Dana said, "سلام لايناد، לאינד שולש" to Daniel.

Dana said, "سلام دانيال ، שלום דניאל" to Daniel.

Dana said, "*SaLaaM DANYAL, ShaLOM DaNYEL*" to Daniel.

Dana said, "*LEYNaD MOLahS ,LAYNAD MaaLaS*" to Daniel.

Daniel lives at سلام 4915 שולש in a beautiful house.

Daniel lives at שלום 4915 سلام in a beautiful house.

Daniel lives at *SaLaaM* 4915 *ShaLOM* in a beautiful house.

Daniel lives at *MOLahS* 4915 *MaaLaS* in a beautiful house.

Daniel lives at سلام 49בא15 שולש in a beautiful house.

Daniel lives at שלום 15אב49 سلام in a beautiful house.

Daniel lives at *SaLaaM* 49*AB*15 *ShaLOM* in a beautiful house.

Daniel lives at *MOLahS* 15*BA*49 *MaaLaS* in a beautiful house.

Then, we give the RL documents. Note that each time-ordered input is really an LR document. Thus the time-ordered inputs are left justified, but the visual-ordered outputs are right justified.

הנד הרמא, "Hello Daniel, bonjour
Daniel„ لايئادل.

דנה אמרה, "Hello Daniel, bonjour
Daniel„ لدانيال.

DaNaH AMRaH, "Hello Daniel,
bonjour Daniel" *LDANYAL.*

Hello Daniel," *,HaRMA HaNaD*
.LAYNADL "bonjour Daniel

הנד הרמא, "Hello Daniel, bonjour Daniel„
لادئايل.

דנה אמרה, "Hello Daniel, bonjour Daniel„
لدانيال.

DaNaH AMRaH, "Hello Daniel, bonjour
Daniel" *LDANYAL.*

Hello Daniel, bonjour" *,HaRMA HaNaD*
.LAYNADL "Daniel

הנד הרמא, "Hello Daniel, bonjour Daniel„
لادئايل.

דנה אמרה, "Hello Daniel, bonjour Daniel„
لدانيال.

DaNaH AMRaH, "Hello Daniel, bonjour Daniel"
LDANYAL.

"Hello Daniel, bonjour Daniel" *,HaRMA HaNaD*

.LAYNADL

הרמא הנד, "Hello Daniel, bonjour Daniel„ لايناد ل.

دنه امره، "Hello Daniel, bonjour Daniel„ لدانيال.

DaNaH AMRaH, "Hello Daniel, bonjour Daniel"
LDANYAL.

"Hello Daniel, bonjour Daniel" *,HaRMA HaNaD*
.LAYNADL