ROBUST VERTICAL TEXT LAYOUT

USING THE UNICODE BIDI ALGORITHM TO HANDLE COMPLEXITIES IN TYPESETTING MULTI-SCRIPT VERTICAL TEXT

by Elika J. Etemad (fantasai)

Abstract

As cross-cultural written communication increases, the technology underlying that communication needs to handle the intersection of their layout conventions. Vertical text is the traditional mode of text layout for many East Asian writing systems. It is also used for effects such as vertical headers in horizontal layout. However, few formatting systems today can do true vertical text layout, and most of those can only handle common scripts in right-to-left columns. Methods for typesetting left-to-right columns or uncommon script combinations such as Mongolian and Arabic thus often involve unwieldy BIDI overrides and delicate glyph rendering tweaks. These workarounds are awkward and can break the portability of the underlying text. The model outlined in this paper uses the intrinsic properties of the characters and an expansion of Unicode's logic to lay out the text without these hacks. Such a system can scale to gracefully handle any combination of scripts, can correctly lay out text with any combination of styling properties, and can integrate well with the layered Unicode + Markup + Styling design of semantically-tagged documents on the Web. This model, developed for the next revision of the CSS3 Text Module, is described here as a CSS system, but the concepts can apply to non-CSS layout systems as well.

This paper focuses on methods for automatically handling character ordering, shaping, and glyph orientation switches when typesetting lesser-known scripts and unusual script combinations in vertical layout, without reworking existent horizontal layout algorithms or adding script-specific new modules. Assumes some familiarity with Unicode BIDI.

Introduction

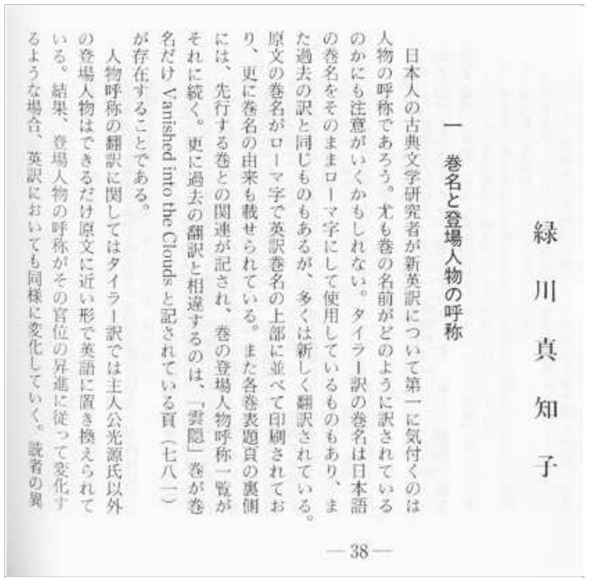
There is a multitude of different writings sytems in use in the world today. They operate on different linguistic principles and different graphic principles, and they all present their own challenges to a multi-script typesetting system: combining characters, contextual shaping, diacritics, bidirectionality, ligatures, etc. This document focuses on one of these challenges: Given a sequence of characters in logical order (the order you read them in, not the order they appear on the page), and given that you already know how to compose the necessary graphemes, **how do you correctly and automatically arrange them into lines**, especially for arbitrarily mixed horizontal and vertical scripts?

Introduction to Vertical Text

To understand how to do vertical text layout, one must first understand what vertical text *is*.

Vertical Text: A Definition

Vertical text is text flowed into vertical lines instead of horizontal ones.



緑川真知子

一　巻名と登場人物の呼称

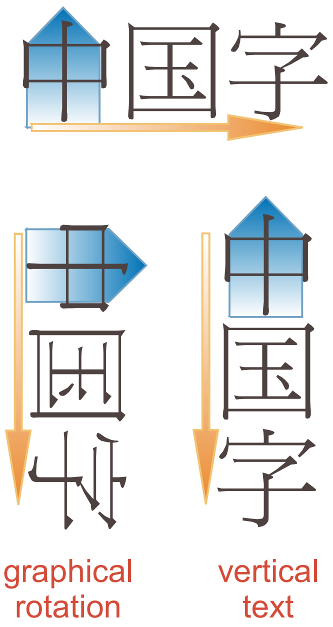
日本人の古典文学研究者が新英訳について第一に気付くのは人物の呼称であろう。尤も巻の名前がどのように訳されているのかにも注意がいくかもしれない。タイラー訳の巻名は日本語の巻名をそのままローマ字にして使用しているものもあり、また過去の訳と同じものもあるが、多くは新しく翻訳されている。原文の巻名がローマ字で英訳巻名の上部に並べて印刷されており、更に巻名の由来も載せられている。また各巻表題頁の裏側には、先行する巻との関連が記され、巻の登場人物呼称一覧がそれに続く。更に過去の翻訳と相違するのは、「雲隠」巻が巻名だけ Vanished into the Clouds と記されている頁（七八一）が存在することである。

人物呼称の翻訳に関してはタイラー訳では主人公光源氏以外の登場人物はできるだけ原文に近い形で英語に置き換えられている。結果、登場人物の呼称がその官位の昇進に従って変化するような場合、英訳においても同様に変化していく。読者の翼

—38—

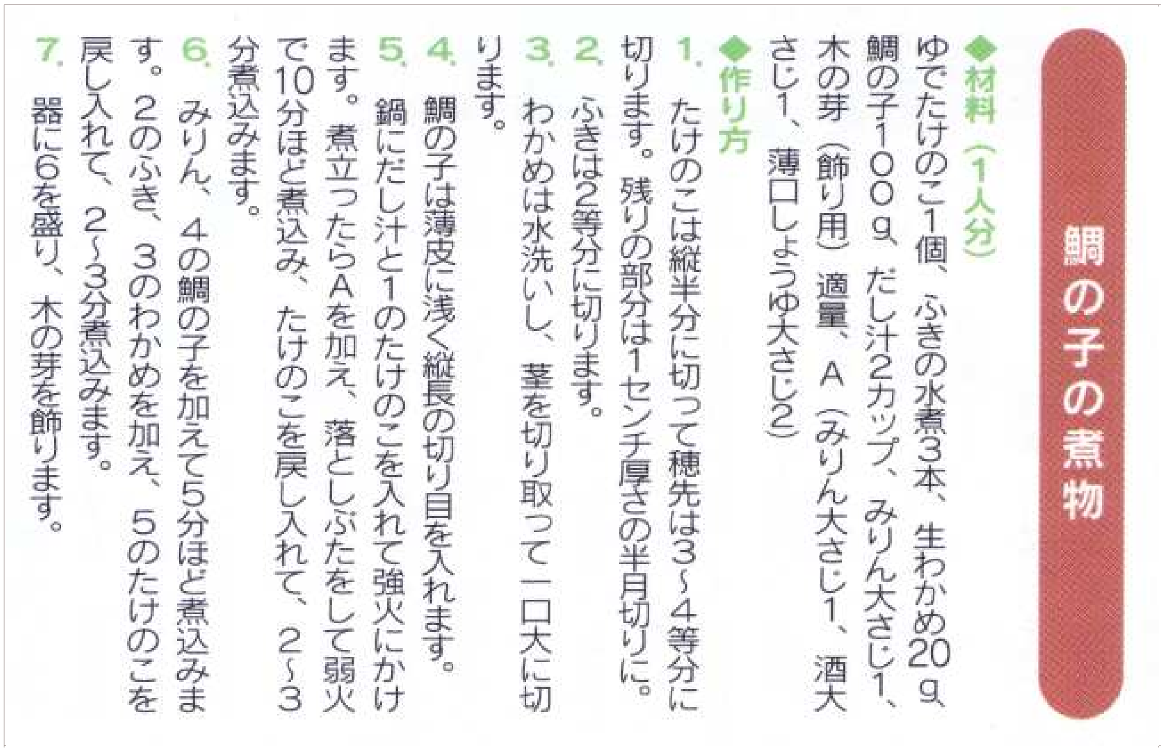
Exerpt from a vertically-set Japanese publication.

Vertical text is not simply graphical rotation.



Incorrect (graphically-rotated) and correct (upright stacking) vertical layout of Chinese text.

Vertical text can stack its columns in either direction



鯛の子の煮物

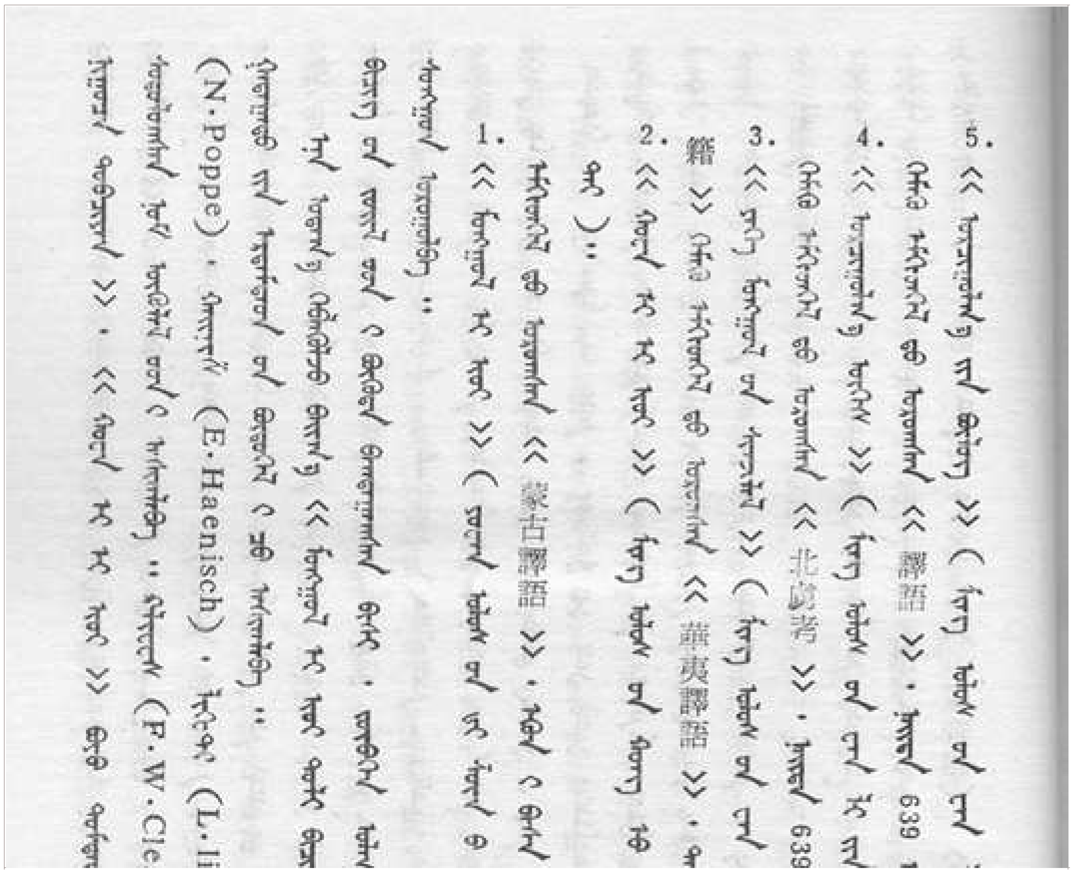
◆材料（1人分）

ゆでたけのこ1個、ふきの水煮3本、生わかめ20g、鯛の子100g、だし汁2カップ、みりん大さじ1、木の芽（飾り用）適量、A（みりん大さじ1、酒大さじ1、薄口しょうゆ大さじ2）

◆作り方

1. たけのこは縦半分に切って穂先は3~4等分に切ります。残りの部分は1センチ厚さの半月切りに。
2. ふきは2等分に切ります。
3. わかめは水洗いし、茎を切り取って一口大に切ります。
4. 鯛の子は薄皮に浅く縦長の切り目を入れます。
5. 鍋にだし汁と1のたけのこを入れて強火にかけます。煮立ったらAを加え、落としぶたをして弱火で10分ほど煮込み、たけのこを戻し入れて、2~3分煮込みます。
6. みりん、4の鯛の子を加えて5分ほど煮込みます。2のふき、3のわかめを加え、5のたけのこを戻し入れて、2~3分煮込みます。
7. 器に6を盛り、木の芽を飾ります。

Vertically-set Japanese.



5. << [illegible] >> ([illegible]) [illegible] << 譯語 >> · [illegible] 639 [illegible]

4. << [illegible] >> ([illegible]) [illegible] << 北虜考 >> · [illegible] 639

3. << [illegible] >> ([illegible]) [illegible] << 華夷譯語 >> · [illegible]

籍 >> [illegible]

2. << [illegible] >> ([illegible]) ::

[illegible] << 蒙古譯語 >> · [illegible]

1. << [illegible] >> ([illegible]

[illegible] ::

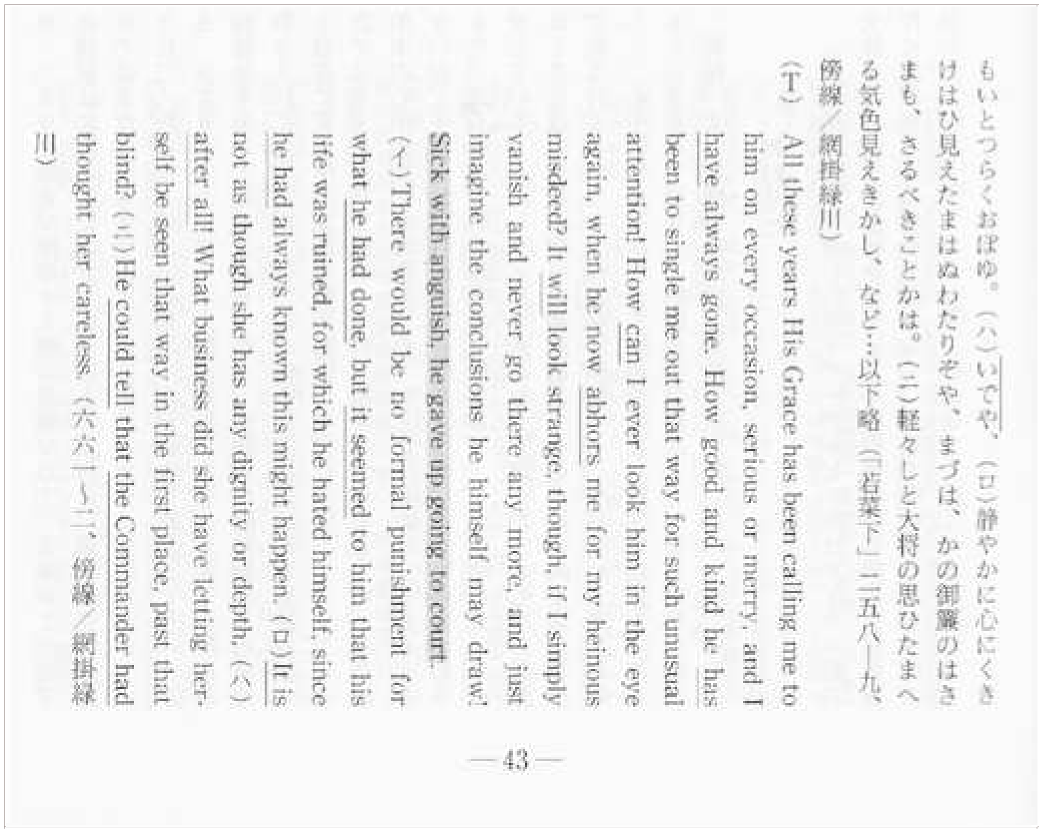
[illegible] (N·Poppe) · [illegible] (E·Haenisch) · [illegible] (L·Li

[illegible] :: [illegible] (F·W·Cle

[illegible] >> · << [illegible] >> [illegible]

Vertically-set Mongolian.

Vertical text layout is not just applied to vertical scripts



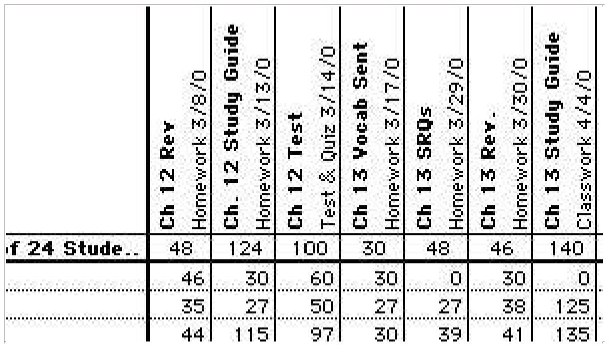
もいとつらくおぼゆ。(ハ)いでや、(ロ)静やかに心にくき
けはひ見えたまはぬわたりぞや、まづは、かの御簾のはさ
まも、さるべきことかは。(ニ)軽々しと大将の思ひたまへ
る気色見えきかし、など…以下略（「若菜下」二五八―九、
傍線／網掛線川）

(T) All these years His Grace has been calling me to
him on every occasion, serious or merry, and I
have always gone. How good and kind he has
been to single me out that way for such unusual
attention! How can I ever look him in the eye
again, when he now abhors me for my heinous
misdeed? It will look strange, though, if I simply
vanish and never go there any more, and just
imagine the conclusions he himself may draw!
Sick with anguish, he gave up going to court.
(イ) There would be no formal punishment for
what he had done, but it seemed to him that his
life was ruined, for which he hated himself, since
he had always known this might happen. (ロ) It is
not as though she has any dignity or depth. (ハ)
after all! What business did she have letting her-
self be seen that way in the first place, past that
blind? (ニ) He could tell that the Commander had
thought her careless. (六六一～二、傍線／網掛線
川）

—43—

Vertical English blockquote in vertically-set Japanese publication.

or just in vertical script contexts.



	Ch 12 Rev Homework 3/8/0	**Ch. 12 Study Guide** Homework 3/13/0	**Ch 12 Test** Test & Quiz 3/14/0	**Ch 13 Vocab Sent** Homework 3/17/0	**Ch 13 SRQs** Homework 3/29/0	**Ch 13 Rev.** Homework 3/30/0	**Ch 13 Study Guide** Classwork 4/4/0
f 24 Stude..	48	124	100	30	48	46	140
	46	30	60	30	0	30	0
	35	27	50	27	27	38	125
	44	115	97	30	39	41	135

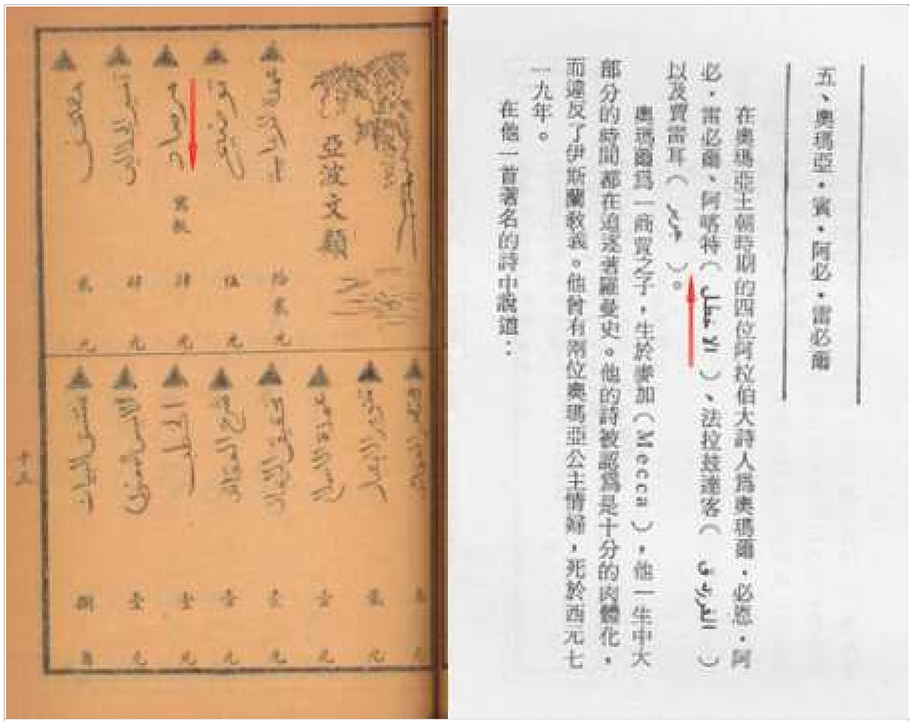
Vertical English in a teacher's gradebook.

There are constraints on how you can do it right.

.gnitcepxe saw I tahw etiuq t’nsi sihT
(This isn’t quite what I was expecting.)

Scrambled vertical English where the characters are ordered to face one way while the glyphs are individually rotated to face the opposite direction.

But there's not just one way to do it.


亞波文類

五、奧瑪亞・賓・阿必・雷必爾

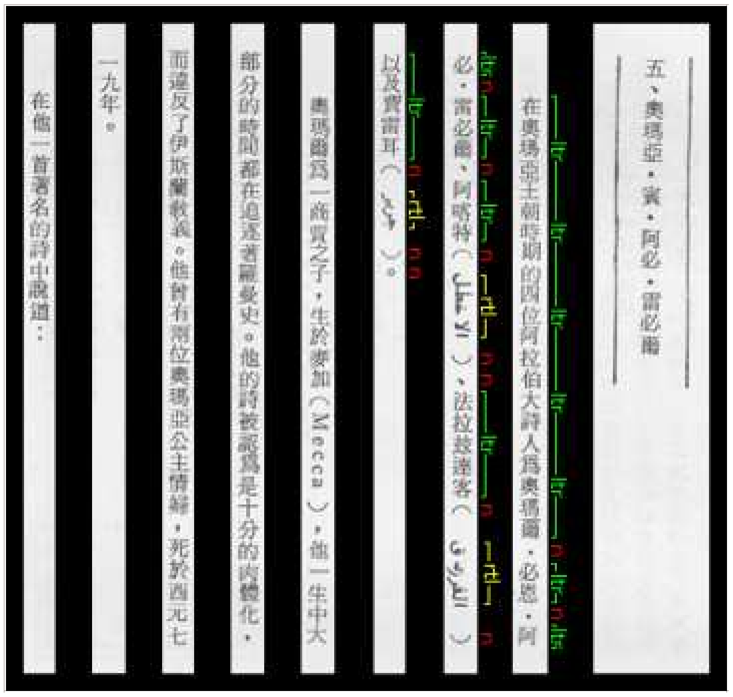
在奧瑪亞王朝時期的四位阿拉伯大詩人爲奧瑪爾・必恩・阿必・雷必爾、阿嗒特（الاخطل）、法拉茲達客（الفرزدق）以及賈雷耳（جرير）。

奧瑪爾爲一商賈之子，生於麥加（Mecca），他一生中大部分的時間都在追逐著羅曼史。他的詩被認爲是十分的肉體化，而違反了伊斯蘭教義。他曾有兩位奧瑪亞公主情婦，死於西元七一九年。

在他一首著名的詩中說道：

Two ways of handling vertical Arabic: rotating right (to read upwards) or rotating left (to read downwards).

Using a sideways version of the Unicode Bidirectional Algorithm, where `ltr` ('left-to-right') means 'top-to-bottom' and `rtl` ('right-to-left') means 'bottom-to-top', is one way to do it.



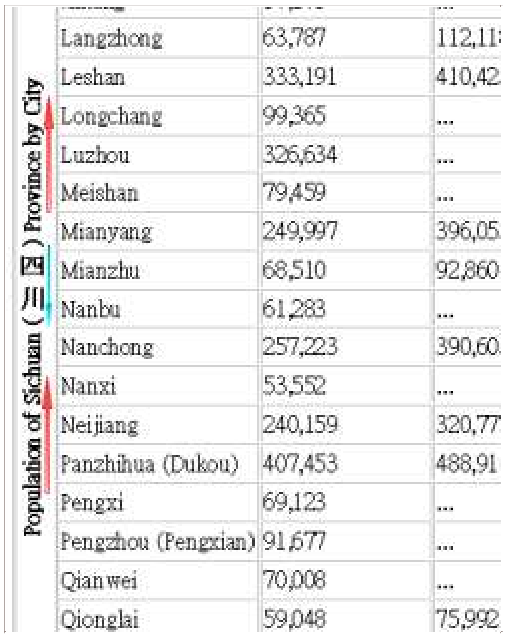
五、奧瑪亞・賓・阿必・雷必爾

在奧瑪亞王朝時期的四位阿拉伯大詩人爲奧瑪爾・必恩・阿必・雷必爾・阿嗒特（الأخطل）・法拉茲達客（الفرزدق）以及賈雷耳（جرير）。奧瑪爾爲一商賈之子，生於麥加（Mecca），他一生中大部分的時間都在追逐著羅曼史。他的詩被認爲是十分的肉體化，而違反了伊斯蘭教義。他曾有兩位奧瑪亞公主情婦，死於西元七一九年。

在他一首著名的詩中說道：

BIDI Analysis of vertically-set traditional Chinese with inline Arabic.

But it doesn't handle all cases.

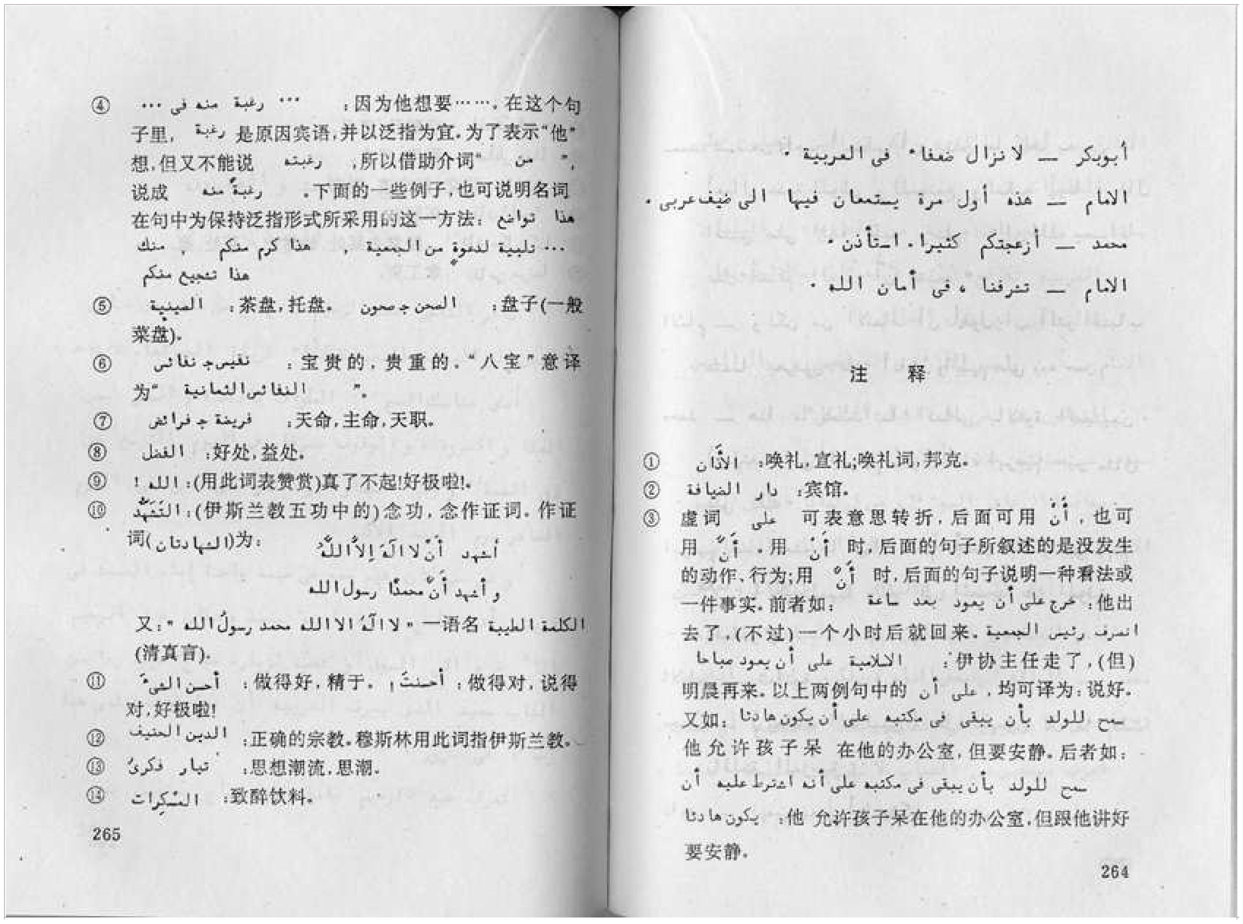

Population of Sichuan (四 川) Province by City

Langzhong	63,787	112,11
Leshan	333,191	410,42
Longchang	99,365	...
Luzhou	326,634	...
Meishan	79,459	...
Mianyang	249,997	396,05
Mianzhu	68,510	92,860
Nanbu	61,283	...
Nanchong	257,223	390,60
Nanxi	53,552	...
Neijiang	240,159	320,77
Panzhihua (Dukou)	407,453	488,91
Pengxi	69,123	...
Pengzhou (Pengxian)	91,677	...
Qianwei	70,008	...
Qionglai	59,048	75,992

Vertical English (bottom-to-top) header on the left of a city-population table; inlining the (top-to-bottom) Chinese characters corresponding to the region's name makes the text bidirectional despite both scripts having left-to-right directionality.

Vertical text makes it possible for two `ltr` scripts to go in opposite directions, and an `rtl` script to go in the same direction as an `ltr` script.

Perhaps this seems like stretching the limits of multi-script typography a little too much. But seemingly strange cases do exist.



أبوبكر — لا نزال ضعفاء في العربية .

الامام — هذه أول مرة يستمعان فيها الى ضيف عربي .

محمد — أزعجتكم كثيرا . استأذن .

الامام — تشرفنا ، في أمان الله .

注　释

① الأذان ：唤礼，宣礼；唤礼词，邦克。
② دار الضيافة ：宾馆。
③ 虚词　على　可表意思转折，后面可用 أنْ ，也可用 أنّ 。用 أنْ 时，后面的句子所叙述的是没发生的动作、行为；用 أنّ 时，后面的句子说明一种看法或一件事实。前者如：خرج على أن يعود بعد ساعة ：他出去了，(不过)一个小时后就回来。انصرف رئيس الجمعية الاسلامية على أن يعود صباحا ：伊协主任走了，(但)明晨再来。以上两例句中的 على أن ，均可译为：说好。又如：سمح للولد بأن يبقى فى مكتبه على أن يكون هادئا 他允许孩子呆在他的办公室，但要安静。后者如：سمح للولد بأن يبقى في مكتبه على أنه اشترط عليه أن يكون هادئا ：他允许孩子呆在他的办公室，但跟他讲好要安静。

264

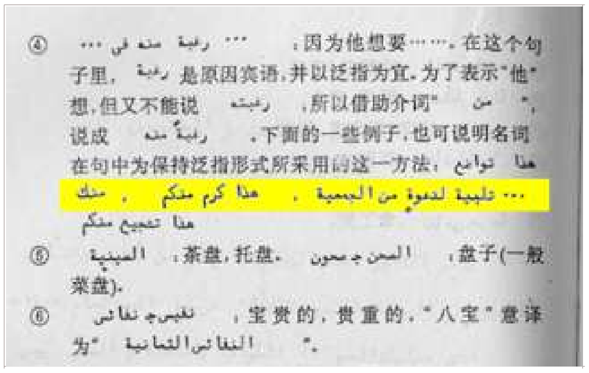
④ … رغبة منه في … ：因为他想要……，在这个句子里，رغبة 是原因宾语，并以泛指为宜。为了表示"他"想，但又不能说　رغبته　，所以借助介词"　من　"，说成　رغبةً منه　。下面的一些例子，也可说明名词在句中为保持泛指形式所采用的这一方法：هذا تواضع منك ，هذا كرم منكم ，… تلبية لدعوة من الجمعية هذا تشجيع منكم
⑤ الصينية ：茶盘，托盘。　الصحن ج صحون　：盘子(一般菜盘)。
⑥ نفيس ج نفائس　：宝贵的，贵重的。"八宝"意译为"　النفائس الثمانية　"。
⑦ فريضة ج فرائض　：天命，主命，天职。
⑧ الفضل　：好处，益处。
⑨ الله ! ：(用此词表赞赏)真了不起！好极啦！。
⑩ التَشَهُّد ：(伊斯兰教五功中的)念功，念作证词。作证词(الشهادتان)为：أشهد أنْ لا الهَ إلاّ اللّهُ و أشهد أنّ محمدًا رسول الله

又："لا الهَ الا الله محمد رسولُ الله" 一语名 الكلمة الطيبة (清真言)。
⑪ أحسن الشئَ　：做得好，精于。أحسنتَ ！：做得对，说得对，好极啦！
⑫ الدين الحنيف　：正确的宗教。穆斯林用此词指伊斯兰教。
⑬ تيار فكريّ　：思想潮流，思潮。
⑭ المُسْكِرات　：致醉饮料。

265

Chinese footnotes on an Arabic text.

Quick — from which side do I start reading this line?



④ … رغبة منه في … ：因为他想要……，在这个句子里，رغبة 是原因宾语，并以泛指为宜。为了表示"他"想，但又不能说　رغبته　，所以借助介词"　من　"，说成　رغبةً منه　。下面的一些例子，也可说明名词在句中为保持泛指形式所采用的这一方法：هذا تواضع منك ，هذا كرم منكم ，… تلبية لدعوة من الجمعية هذا تشجيع منكم
⑤ الصينية ：茶盘，托盘。　الصحن ج صحون　：盘子(一般菜盘)。
⑥ نفيس ج نفائس　：宝贵的，贵重的。"八宝"意译为"　النفائس الثمانية　"。

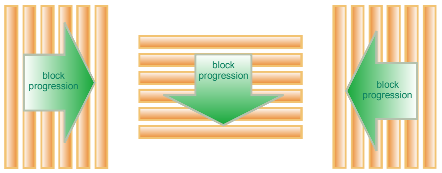
The trouble with reading bidirectional paragraphs.

These footnotes would perhaps be a bit more readable if the typographer took advantage of Chinese's vertical tradition to make both scripts read in the same direction: top to bottom within a mixed-script line and right to left between pages.

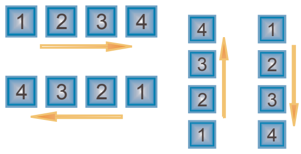
Analyzing Text Flow

The first step in creating a multi-script layout model is to define the relevant properties of the layout. Every run of text has three physical properties that describe the way the text flows:

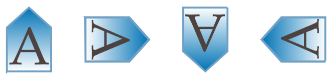
block progression
The direction the lines stack in.



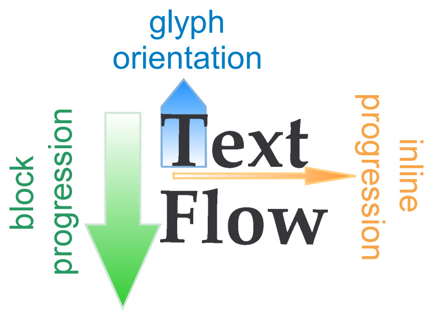
inline progression
The direction characters are ordered in within the line.



glyph orientation
The direction an individual glyph is facing.



Once all three properties are known, the text can be laid out.



Physical Text Layout

The most straightforward way of getting this information is to ask the user. But there are several problems with this:

- First of all, typesetting mixed-script text becomes a very tedious task. Each change in script would require manual intervention to adjust the text flow properties.
- Secondly, as we saw with the scrambled vertical English, not all combinations of inline-progression and glyph-orientation are valid for a given script. Good software should make invalid combinations of user input impossible.
- Thirdly, the system doesn't capture the interdependencies among the properties. Deciding “No, I think this passage would look better as vertical text” and switching the block-progression setting could throw off the careful setting of complex bidirectional text. This problem is even worse with web pages: if the browser does not support vertical text, it will fall back to horizontal, and of course the author has no chance at this point to reset the text for horizontal context.

.gnitcepxe saw I tahw etiuq t'nsi sihT
(This isn't quite what I was expecting.)

Fundamentally, these problems exist because interrelationships among the layout properties and the nature of the script are realized in the author's mind and not in the system.

Logical Text Layout

The set of inline-progression and glyph-orientation combinations that are valid for a run of text depends on inherent properties of the text's script. If we embed this information into the system we can use it to constrain the layout, making it possible to derive one layout property from the other. This lets us automate much of the inline-progression and glyph-orientation switching and allows defining layout switches in terms of the relationships among them.

Unicode systems already take advantage of this logical model in horizontal text. For example, you don't have to manually tell every run of Hebrew to order itself right-to-left because Unicode already provides that information through its character data tables (which provide directionality info) and the Unicode BIDI Algorithm (which defines how to use that directionality info to get a character ordering). We just need to extend the logic to handle vertical text.

Intrinsic Script Properties

There are three script properties we need to know for logical multi-directional text layout:

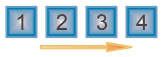
- horizontal directionality
- vertical directionality
- bi-orientational transformation

Directionality

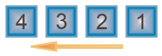
A script's *directionality* is the inline progression direction the script must take in valid text layout. There are two types of directionality:

horizontal directionality
is the directionality used in horizontal text layout. It can be

- left-to-right (`ltr`)



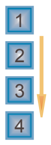
- right-to-left (`rtl`)



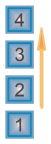
- none

vertical directionality
is the directionality used in vertical text layout. It can be

- top-to-bottom (`ttb`)



- bottom-to-top (`btt`)



- none

No directionality means the script does not have a preferred inline progression in that orientation. For example, while English must go from left to right in horizontal context, it can go either top to bottom or bottom to top in vertical text (since it doesn't have a vertical directionality). Like English, Japanese also has a left-to-right horizontal directionality. However, in vertical context Japanese must only go from top to bottom, even in a left-to-right block progression.

By script, the CJK ideographs (Han), Yi, Mongolian/Manchu, Hangul, Bopomofo, Hiragana, and Katakana all have top-to-bottom vertical directionality. Ogham has bottom-to-top vertical directionality. I believe all other scripts in Unicode have no vertical directionality.

Directionality in Unicode

Aside from generic punctuation, which is neutral, every character in Unicode has been assigned a horizontal directionality. Unfortunately the standard does not provide similar data for vertical directionality.

Because Unicode does not have a notion of vertical directionality, vertical-only scripts like Mongolian have been assigned a left-to-right horizontal directionality. This is the canonical directionality used in plain text, and it is this inline progression that defines the glyphs' reference orientation, which we will need later.

Extended values for the 'direction' property

In addition to per-character directionality values, the Unicode Bidirectional Algorithm needs to know the overall directionality of the block of text it is ordering. Plain text formatting uses an heuristic to guess the overall directionality from the first few characters. Higher-level protocols like HTML and CSS, however, use an explicit setting instead.

Like the Unicode override controls, and the HTML `dir` attribute, the CSS2 `direction` property only had the two horizontal directions: `ltr` and `rtl`. Adding new values allows it to express vertical directionality as well.

`direction`

Primary directionality. Can take the following values

`ltr`
Left-to-right directionality in horizontal text; No inherent directionality in vertical text. (Horizontal script) Examples: Latin, Tibetan

`rtl`
Right-to-left directionality in horizontal text; No inherent directionality in vertical text. (Horizontal script) Examples: Arabic, Hebrew

`ttb`
Top to bottom directionality in vertical text; No inherent directionality in horizontal text. (Vertical script) Example: traditional Mongolian

`ltr-ttb`
Left to right directionality in horizontal text; Top to bottom directionality in vertical text. (Bi-orientational script) Examples: Han, modern Yi

`ltr-btt`
Left to right directionality in horizontal text; Bottom to top directionality in vertical text. (Bi-orientational script) Example: Ogham

Classifying Scripts by Directionality

Scripts can be classified into three orientational categories:

horizontal
Scripts that have horizontal, but not vertical, directionality. Includes: Latin, Arabic, Hebrew, Devanagari

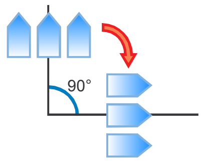
vertical
Scripts that have vertical, but not horizontal, directionality. Includes: Mongolian, Manchu

bi-orientational
Scripts that have both vertical and horizontal directionality. Includes: Han, Hangul, Yi, Ogham

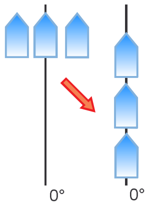
Bi-orientational Transformation

For vertical scripts, we also need to know how the glyphs transform when switching from their standard horizontal configuration to a vertical one. This property is the *bi-orientational transformation* and it can be

rotate
Rotate the glyph from horizontal to vertical



translate
Translate the glyph from horizontal to vertical



CJK (Chinese/Japanese/Korean) characters translate; they are always upright. Other scripts, such as Ogham and Mongolian, must be rotated.

Using Script Properties to Imply Directions

Text in a native orientation needs no additional stylistic hints for proper layout: its inline progression and glyph orientation are both intrinsically mandated by the script. The style system can figure out how to lay them out from the script properties, so settings for inline progression and glyph orientation are not necessary.

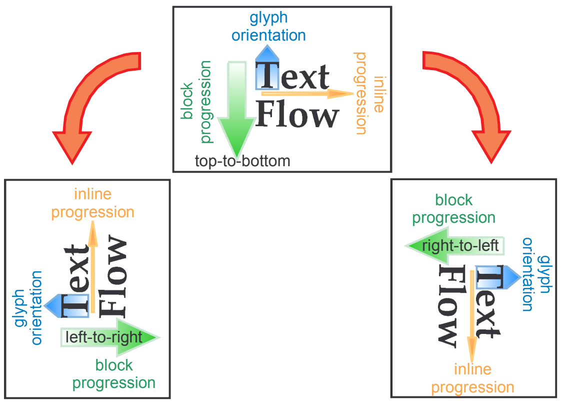
Text in a foreign orientation doesn't need directionality or glyph overrides either. It just needs a few hints: whether to translate upright, or, if it's rotated sideways, which side is "up". Given that, the rules for laying out the text in its native orientation are enough to determine the inline progression and exact glyph orientation.

With native scripts' layout pre-determined by the script's properties and the non-native scripts constrained enough that they only need a glyph-rotation hint, the typesetter's intervention reduces to just two settings for the *entire* block of text:

- what block progression to use
- how to orient non-native text
  (rotate right, rotate left, or stay upright)

Text Orientation Schemes

For horizontal scripts in a vertical orientation, the text is most comfortably laid out as if the whole text block were merely rotated from the horizontal:

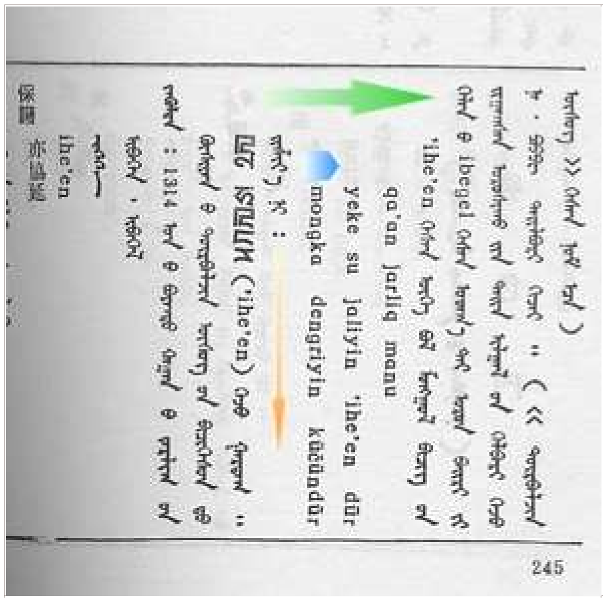


For example, English text in vertical lines that stack from left to right most naturally uses a glyph orientation that faces left. As a consequence of the script's constraints, the inline progression then runs from bottom to top. The same text, by the same logic, would in a right-to-left line stacking context face right and flow within each line from top to bottom.

To make this the default behavior, we can define a setting, ‘natural’ that depends on the block progression to make non-native text always face the top of the line stack. The glyph orientation and inline progression will thus adapt to whichever block progression happens to take effect.

This layout scheme is most appropriate for dealing with text that has been turned on its side for layout purposes—as for page headers or captions or table headings. However, a major reason for laying out text in a non-native orientation is mixing horizontal and vertical scripts, which introduces the requirement of making the secondary scripts flow well in the context of the primary script.

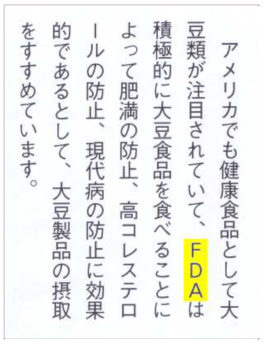
For example, a primarily Mongolian document, which has vertical lines stacking left to right, usually lays its Latin text with the glyphs facing the right.



Exerpt from a multi-script Mongolian dictionary. Columns read from left to right even though the individual lines of Latin are facing the opposite direction.

This makes the Latin run with the same inline progression as Mongolian and face the same direction it does in other East Asian layouts (which have vertical lines stacking right to left), but the glyphs are facing the *bottom* of the line stack rather than the top, something they wouldn't do in a primarily-English paragraph.

Yet another common layout is to keep the horizontal script's glyphs upright and order them from top to bottom; this is frequently done with Latin-script acronyms in vertical East Asian text.



アメリカでも健康食品として大
豆類が注目されていて、ＦＤＡは
積極的に大豆食品を食べることに
よって肥満の防止、高コレステロ
ールの防止、現代病の防止に効果
的であるとして、大豆製品の摂取
をすすめています。

Latin acronym in vertical Japanese

To handle these layouts, the style system needs to offer controls for choosing among these different layout schemes. Note, however, that scripts in their native orientations do not need these hints; only the non-native ones do. Also, this is only one simple scheme switch here: there's no need for the designer to set separate absolute inline progression and glyph orientation properties or to set styling properties on each individual text run of a different script.

CSS Properties for Text Layout

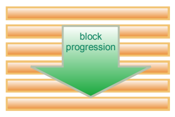
We can formalize these text layout settings into CSS properties:

- `block-progression` to set the block progression direction
- `text-orientation-vertical` to set the text orientation scheme for non-native scripts

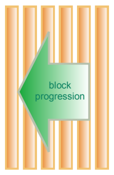
`block-progression`

Block progression (line stacking) direction. Can take the following values:

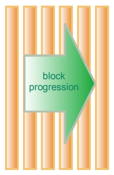
`TB`
Top-to-bottom line stacking (horizontal text).
Typically used for most non-East-Asian layout.



`RL`
Right-to-left line stacking (vertical text).
Typically used for traditional CJK layout.



`LR`
Left-to-right line stacking (vertical text).
Typically used for traditional Mongolian layout.

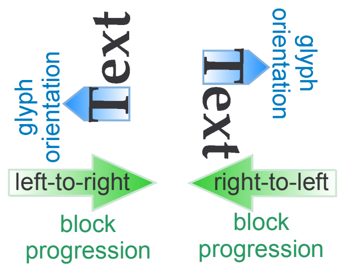


`text-orientation-vertical`

Glyph orientation scheme to use in vertical text. **Does not affect layout of vertical scripts, only horizontal ones.** Can take the following values:

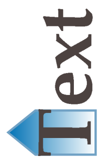
`natural`

Non-vertical script runs are laid out as if "up" was towards the top of the line stack (left or right, depending on the block progression in effect).



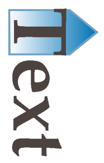
`left`

Non-vertical script runs are laid out as if "up" was towards the left side of the line stack.



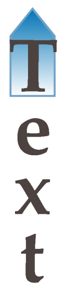
`right`

Non-vertical script runs are laid out as if "up" was towards the right of the line stack.



`upright`

Non-vertical scripts' characters read top to bottom, with each grapheme cluster oriented upright.



Note: For handling vertical-only scripts in horizontal layout, a `text-orientation-horizontal` *property is also necessary; it takes effect only when the block progression is top-to-bottom and only affects non-horizontal scripts like Monglian. To keep the discussion less verbose, we will only cover the vertical case here.*

Summary of Logical Layout Controls

In summary, to lay out a block of arbitrary, mixed-script text, the layout system needs to offer only three controls:

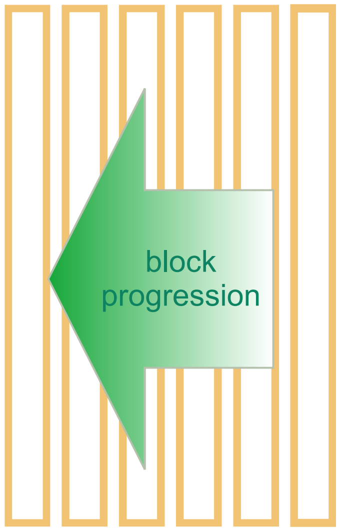
- overall directionality (BIDI property)
- block progression direction (layout property)
- text orientation scheme (stylistic property)

Because of the way correctness constraints are embedded in this layout system, any combination of the block-progression and text-orientation properties will result in a correct, if not optimally-configured, layout of any block of arbitrary mixed-script text.

The next step is detailing exactly how to implement such a system.

Line Stacking vs. Line Composition

Laying out text in lines is really two separate problems: composing the lines and stacking them. As far as line stacking is concerned, the lines are just boxes. The arrangement of items within the line box doesn't matter. The layout code just needs to rotate and stack the boxes according to the block-progression settings.



Figuring out what to do when different layout elements have different `block-progression` values is hard, especially in CSS's fluid layout model—but that's a box layout problem, not a text layout problem. Let's move on to putting together the text *inside* the line boxes.

Line Composition

Line composition is the process of laying out the text within a line.

We will assume that all sets of combining characters have been resolved into their respective grapheme clusters because that process depends only on the logical order of the characters and is not dependent on layout modes. With that out of the way, there are three other processes in line composition:

- Ordering the characters
- Rotating the characters
- Shaping the characters

Shaping only happens for some scripts and is wholly dependent on the order and orientation of the characters, so we will discuss it last.

The Unicode Bidirectional Algorithm

If all the characters in the line were guaranteed to have the same inline progression, then ordering the characters would be very straightforward. However, because different scripts have different directionalities, lines can contain a mixture of inline progressions. The *Unicode Bidirectional Algorithm* can resolve the order of characters within a mixed-direction line based on the directionalities assigned to each character.

Unicode Standard Annex #9 (UAX9) defines the BIDI algorithm in terms of the horizontal left-to-right (`ltr` or `L`) and right-to-left (`rtl` or `R`) directionalities. However, the character-ordering algorithm itself can handle a model extended with vertical directionalities if we just abstract it to apply to two arbitrary, opposing directionalities rather than just to right-to-left and left-to-right.

Mapping Directionalities

To resolve the horizontal and vertical directionalities to the two directionalities used in the BIDI algorithm, we'll use a map from the concrete directionalities to the abstract directionalities **high-to-low** and **low-to-high**.

The directions **left**, **right**, **top**, and **bottom** map to **high** or **low** based on the values of text-orientation and block-progression. **The mapping must apply to everything:**

- the individual character's directionality,
- embedding and override codes,
- the CSS direction values,
- HTML dir attributes,

- etc.

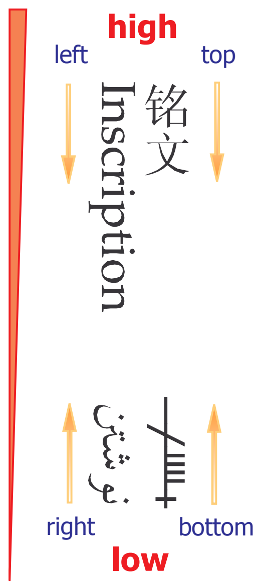
Note that **the algorithm remains unchanged when dealing with vertical text**. What changes is the directionality input.

In vertical context, bi-orientational scripts use their vertical directionality and behave as vertical, not horizontal, scripts. Han, for example, as a `ltr-ttb` script, is treated as `ttb` (top to bottom), *not* `ltr` (left to right). The `ltr-ttb` value for `direction` is correspondingly treated the same way as the value `ttb`.

Mapping for `text-orientation: right`

When the text-orientation is `right` or the text-orientation is `natural` and the block-progression is `RL`

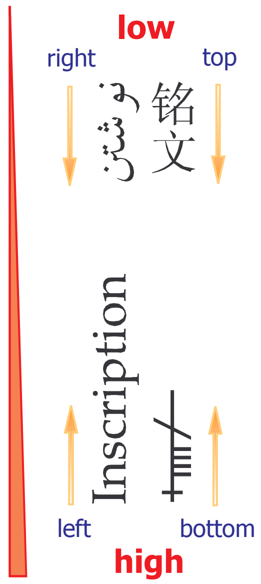
- Map **ttb** and **ltr** to **htl** (high to low)
- Map **btt** and **rtl** to **lth** (low to high)



Mapping for `text-orientation: left`

When the text-orientation is `left` or the text-orientation is `natural` and the block-progression is `LR`

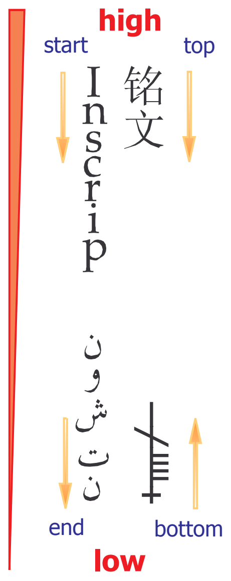
- Map **ttb** and **rtl** to **lth** (low to high)
- Map **btt** and **ltr** to **htl** (high to low)



Mapping for `text-orientation: upright`

When the text-orientation is `upright`

- Map **ttb**, **ltr** and **rtl** to **htl** (high to low)
- Map **btt** and to **lth** (low to high)



Running BIDI

When the BIDI algorithm is run on the text, the algorithm's notion of **right** will represent our **low** and **left** will represent our **high**. Notice that the **left-to-right** directionality always maps to **high-to-low** and **high-to-low** always maps to the algorithm's **left-to-right**. This is to make sure numbers, which are processed specially, always come out right.

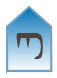
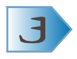
That takes care of the ordering. Next we need to determine each character's glyph orientation.

Glyph Orientation

Before the system can paint the text (or even do shaping and alignment), it needs to know how to rotate (or not rotate) the glyphs. For vertical and bi-orientational scripts, the glyph orientation is derived from the script properties. For horizontal scripts, it's given by the text-orientation property. Glyph rotation is always affects each grapheme cluster together as one unit.

For vertical scripts such as Mongolian, facing up ("upright") is defined to be the orientation used when the text is set *horizontally* using it's *Unicode-defined directionality* (in this case, `ltr`) even though this may not be the true upright position of the glyph. This is because fonts are generally encoded for use in horizontal text, not vertical text.

Mongolian glyph

facing up	**truly upright**
	

Transformations for punctuation should be handled by using the vertical glyph variants given in the font, but only when the primary direction of the text has a vertical directionality component or when the text-orientation is `upright`. (If the text is primarily horizontal text rotated sideways, then the punctuation should likewise be horizontal punctuation rotated sideways.)

Vertical Scripts: Deriving Glyph Orientation

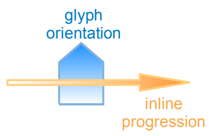
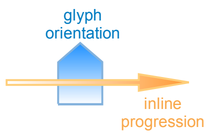
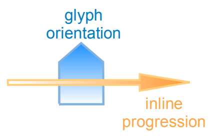
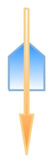
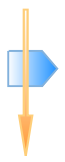
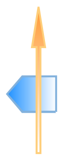
Han and Kana and Hangul and Yi need to be kept upright (0° rotation) because they use the same orientation in both horizontal and vertical text. Mongolian (and Ogham), however, rotate from one context to the other and so their glyphs must be rotated 90° from their horizontal orientation when used in vertical context. Given the script's horizontal and vertical directionalities and its bi-orientational transformation:

System's Knowledge of Vertical Scripts' Properties

	Han/Hangul/Kana/ Yi	**Mongolian/ Manchu**	**Ogham**
(cannonical) horizontal directionality	`ltr`	`(ltr)`	`ltr`
vertical directionality	`ttb`	`ttb`	`btt`
transformation	translation	rotation	rotation

the glyph orientation can be derived as follows:

System's Derivation of Vertical Scripts' Orientation

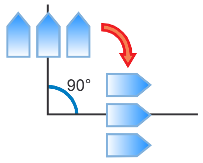
	Han/Hangul/Kana/ Yi	**Mongolian/ Manchu**	**Ogham**
horizontal orientation (vector direction)			
direction transformation	`ltr` to `ttb`	(`ltr`) to `ttb`	`ltr` to `btt`
glyph transformation	translate	rotate	rotate
vertical orientation			

Horizontal Scripts: Applying Text Orientation

Horizontal scripts get their glyph rotation directly from the style properties:

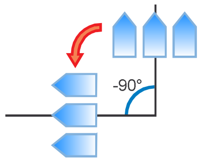
For `text-orientation: right`
(or `text-orientation: natural` when the block-progression is RL)

Rotate horizontal scripts' grapheme clusters 90° to the right.



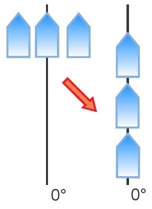
For `text-orientation: left`
(or `text-orientation: natural` when the block-progression is RL)

Rotate horizontal scripts' grapheme clusters 90° to the left.



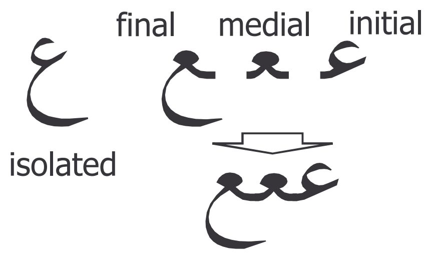
For `text-orientation: upright`

Keep glyphs for horizontal scripts upright and stack grapheme clusters vertically.



Character Shaping

Character shaping is the process of selecting, based on context, which of several glyph variants of a letter should be used. This is typical of cursive scripts like Arabic and Mongolian, in which the shape of a letter depends on whether it comes at the start of a word, in the middle of a word, or at the end of a word.



According to Unicode BIDI, character shaping occurs after BIDI reordering: the Arabic character shaped as an “initial” will always be on the right, even if the text is given a left-to-right override. This ensures that the letters always visually connect. (If shaping happened before reordering, an initial form on the right side of the word would wind up on the left and be trying to connect to nothing.)

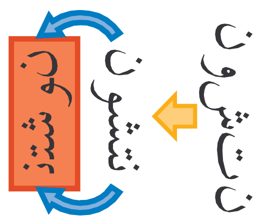
Reverse Shaping

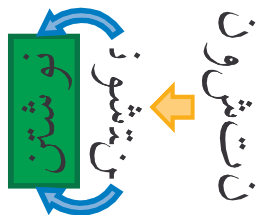
If a shaping script's characters are ordered in reverse because the text's directionality maps within the BIDI algorithm to a directionality other than its standard horizontal one, then the shaping needs to be done in reverse also. This happens, for example, with Mongolian when its **top-to-bottom** directionality maps to the BIDI algorithm's notion of **right-to-left**. Mongolian's canonical horizontal directionality is left-to-right, so normally it would map to **left-to-right** and the shaping algorithm is designed for that case.

To cope with this problem, we can isolate the affected string and either

- invert it, shape it, then un-invert it, or
- tell the shaping function to shape in reverse: to shape characters in the initial positions as finals and characters in the final positions as initials

The result of this process in a normal orientation would be a lot of disconnected letters. However, once the glyph orientation is applied, the glyphs will connect properly.





Arabic and Mongolian, both shaping scripts, can go in the same direction or in opposite directions, so blindly reverse-shaping the entire character string doesn't work. However, shaping occurs only within each directional level run, and it is also constrained to runs of text in the same script; Mongolian characters, from Arabic's point of view, form as concrete a boundary as Latin ones do. It is therefore possible to break up the text into pieces that have characters from no more than one shaping-affected script without compromising the accuracy of the shaping. Each of these pieces can then be shaped individually, in reverse if necessary.

Finishing Off

Once the line is composed, all that remains is to lock its **high** and **low** ends to the appropriate sides of the block and stack the lines according to the block-progression setting.

Data Missing from Unicode

In addition to knowing the text, its primary directionality, and its styling properties, the implementation needs to know something about the characters themselves to be able to take advantage of the logical model. For each character, the following information must be available to the text layout algorithm:

- *horizontal directionality*
- *vertical directionality*
- for vertical scripts, the *transformation* between horizontal and vertical orientations

Unicode currently provides horizontal directionality, but not vertical directionality or transformation. Also, for the model to be complete, Unicode would have to provide BIDI control characters for the extended directionality values equivalent to the ones for horizontal-only directionality settings.

Conclusion

There are a lot of cool, unusual, and *useful* ways of combining writing systems. By using constraints inherent in each script, *logical text layout* creates a framework for mixed-script typesetting that is flexible enough to handle more than just the basic possibilities, yet automatic and robust enough to be practical. Because it does not rely on directionality overrides or special fonts to work, the underlying text is portable and can be used, stripped of all styling information, in other Unicode contexts.

Acknowledgements

Thanks go out to:

- Martin Heijdra at the East Asian Library, for his guidance, expertise, and enthusiasm. I had never imagined that the bibliographer helping me find books would turn out to be an expert on international typography and Mongolian in particular.
- Ian Hickson, the members of the www-style mailing list, the members of the CSS Working Group, and the contributors to the Mozilla Project for tempering my technical skills and CSS knowledge over the years
- The CSS Working Group for giving me a chance to fix everything I found wrong in the CSS3 Text drafts.
- Håkon Wium Lie and Opera Software for supporting me in this project, to the point of even *paying* me over the summer to work on it. I only wish it hadn't taken so long so that I could spend more time on QA. ☺
- Brian W. Kernighan for taking on the role of my project advisor within the Princeton Computer Science Department.

Bibliography

- Bert Bos; Håkon Wium Lie; Chris Lilley; Ian Jacobs. *Cascading Style Sheets, level 2*. 1998. W3C Recommendation. URL: http://www.w3.org/TR/REC-CSS2
- Davis, Mark. *The Bidirectional Algorithm.* 17 April 2003. Unicode Standard Annex #9. URL: http://www.unicode.org/unicode/reports/tr9/tr9-11
- Heijdra, Martinus. Mongolian Text Layout. Spring 2003-present.
- Suignard, Michel. *CSS3 Text*. May 2003. W3C Candidate Recommendation. URL: http://www.w3.org/TR/2003/CR-css3-text-20030514/
- The Unicode Consortium. *The Unicode Standard: Version 4.0.0*. URL: http://www.unicode.org/versions/Unicode4.0.0/