

Canonical Ordering of Marks in Thai Script

Peter Constable

July 7, 2018

This document examines canonical ordering of marks in Thai script, and some issues that arise. It arrives at a set of rendering rules to use in a Thai rendering engine. These rules maintain principles that canonically-equivalent sequences have the same display, and that non-canonically-equivalent sequences have distinct displays. Not all sequences are treated as valid, however, for purposes of Thai shaping.

This analysis was motivated by evaluating requirements for Patani Malay orthography, as described in Unicode 10, [Chapter 16](#), and by document [L2/10-451, Proposal not to encode 4 minority Thai letters for Patani Malay](#).

Background: the canonical ordering algorithm

Unicode normalization was created to deal with certain issues that arise from dynamic composition, and from the need to accommodate legacy encodings. In particular, a character sequence can include marks in distinct positions that cannot interact typographically, and so different orderings of those marks in a sequence do not lead to visual distinction.

For example, consider a sequence of a base character followed by an above mark, then a below mark; for instance, < “a”, “̂”, “̣” >. Since the marks occupy different positions relative to the base, they do not interact typographically, and reversing the order of marks in the encoded representation — < “a”, “̣”, “̂” > — would result in the same presentation: “ã”.

Because there is no visual distinction between these differently-ordered sequences, there cannot be a useful and safe semantic distinction. Hence, these differently-ordered sequences are effectively equivalent. Unicode normalization is an algorithm intended to reflect such equivalence. The canonical ordering algorithm is the specific portion of the normalization algorithm intended to establish equivalence between differently-ordered combining mark sequences.

In contrast, different combining marks that occupy the same positions relative to a base do interact typographically, and so different orderings of those marks in a sequence *do* lead to a visual distinction. For example, a tilde can be above an acute, as in “ã”; or the acute can be above the tilde, as in “ă”. For marks that interact in this way, marks stack outward from the base in the order they occur in the encoded sequence. Since, in such cases, the different ordering of encoded sequences do correspond to visual distinctions, the canonical ordering algorithm is intended to establish *non-equivalence* between the differently-ordered sequences.

Every combining mark is assigned to a canonical combining class. When marks fall into distinct positions relative to a base (e.g., tilde above versus dot below), they are put into distinct classes. Because relative ordering does not result in a visual distinction, these are to be considered equivalent. During normalization, the canonical ordering algorithm will re-order marks into a canonical order based on their class assignments, thereby neutralizing or “folding away” the semantically-irrelevant difference in ordering.

In contrast, marks that are similarly positioned relative to a base, and so can interact typographically, are assigned to the same canonical combining class. Because they interact typographically, different orders of the marks within an encoded sequence do correspond to different relative positions of those marks. During normalization, marks in the same class are *not* re-ordered, but maintain their relative ordering. In this way, canonical ordering preserves the differently-ordered sequences as a semantically-significant distinction.

Note that this ordering of marks is considered *canonical* solely for purposes of comparison and determining equivalence. In general, Unicode does not consider any particular ordering of marks to be preferred, and there is no expectation that marks would be re-ordered in user data, though this can happen.

Canonical order of marks is determined by a numeric value assigned to each class: marks in a lower-value class are re-ordered before marks in a higher-value class.

Some combining marks are assigned to a class with numeric value 0, which has special properties. In particular, class 0 is exempt from re-ordering. Moreover, it creates an opaque boundary for canonical re-ordering.

For example, if the classes in a mark sequence are *230 220*, normalization will reorder them as *220 230*. But for a sequence *230 0 220*, no re-ordering occurs: marks in class 0 are not re-ordered, and they block re-ordering of marks that precede and follow. Because of this, sequences with classes *230 0 220* are not deemed by Unicode normalization to be equivalent to the differently-ordered sequences *220 0 230* even though there may be no visual distinction.

Some general issues

While canonical ordering is not intended to be a preferred order for marks in user data, it does have a bearing on what kinds of sequences can be considered useful for user data. For one thing, some applications *may* treat canonical ordering as preferred, and transform any user data that isn't so-ordered into canonical order. Also, even if applications do not reorder user data into canonical order, they will often use canonical ordering to determine when strings are deemed to be the same (equivalent) or different. Because of this, different ordering of mark sequences that are folded under canonical ordering cannot be considered viable for semantic distinctions since it should be expected that applications will not treat these as distinct, and there is no guarantee that the ordering differences will be retained.

Now, it so happens that there are cases of marks assigned to classes in such a way that different orders of the marks are folded under canonical ordering even though the different orders of the marks *do* result in visual distinctions and so otherwise would potentially be useful for semantic distinctions. An example of this for Thai will be considered below. These cases (generally accidents of history, not a result of specific design intent) are unfortunate: a potentially-useful distinction cannot be considered viable. They also add complexity for implementations since the implementations should ensure that the ordering distinctions are folded and are not reflected in any usable way. For example, the marks must be presented with a fixed positioning relative to the base, regardless of the order in the encoded representation.

Issues could also arise if marks that have distinct positions relative to a base, hence cannot interact typographically, yet were not re-ordered during normalization: that would mean that different encoded orderings are preserved as non-equivalent, and so a semantic distinction could be assumed, even though no visible distinction is possible. This would create phishing vulnerabilities, and so would be a bad thing. This arises in certain cases due to some marks being assigned to class 0. Because of the way class 0 is handled in normalization, these are treated as non-equivalent in Unicode normalization, yet no visual distinction is possible. Examples of this for Thai will be considered below.

One other general scenario to consider are marks that can interact typographically, yet some are assigned to class 0. For example, consider a situation involving above marks, some of which are assigned to class 230, and some assigned to class 0. Since marks in the same class retain their ordering, and since marks in class 0 are not re-ordered, this situation is no different than if all the marks had been in class 230, or all had been in class 0. This case is not problematic precisely because there is only one non-zero class involved. But if class 0 marks combine with marks in *two* other classes, then issues will arise. Examples of this for Thai will be considered below.

Background: combining classes of marks used in Thai script

Combining marks used with Thai script include marks used for Thai language and for other languages, such as Pattani Malay. They fall into six categories involving six combining classes:

A. Thai-specific above vowel/rhyme marks, combining class 0:

- U+0E31 THAI CHARACTER MAI HAN-AKAT “◌̂”
- U+0E34 THAI CHARACTER SARA I “◌̃”
- U+0E35 THAI CHARACTER SARA II “◌̄”
- U+0E36 THAI CHARACTER SARA UE “◌̅”
- U+0E37 THAI CHARACTER SARA UEE “◌̆”
- U+0E47 THAI CHARACTER MAITAIKHU “◌̇”
- U+0E4C THAI CHARACTER THANTHAKHAT “◌̈”
- U+0E4D THAI CHARACTER NIKHAHIT “◌̉”
- U+0E4E THAI CHARACTER YAMAKKAN “◌̊”

B. Thai-specific nukta, combining class 9:

- U+0E3A THAI CHARACTER PHINTHU “◌̋”

C. Thai-specific below vowel marks, combining class 103:

- U+0E38 THAI CHARACTER SARA U “◌̌”
- U+0E39 THAI CHARACTER SARA UU “◌̍”

D. Thai-specific above tone marks, combining class 107:

- U+0E48 THAI CHARACTER MAI EK “◌̎”
- U+0E49 THAI CHARACTER MAI THO “◌̏”

- U+0E4A THAI CHARACTER MAI TRI “◌̑”
 - U+0E4B THAI CHARACTER MAI CHATTAWA “◌̑̑”
- E. Generic below marks, combining class 220:
- One known case, used for a consonant modifier (nukta) in Patani Malay: U+0331 COMBINING MACRON BELOW “◌̑̑”
- F. Generic above marks, combining class 230:
- One known case, used for a vowel in Patani Malay: U+0303 COMBINING TILDE “◌̑̑”

In examples below, one representative mark from each class will be used (SARA I for class A, SARA U for class C, MAI EK for class D).

Interactions between marks of different classes




Some of the more problematic cases described in this section are not known to be used in the orthography of any language. That doesn’t ensure that users won’t want to use them in some way. General-purpose implementations can’t predict what sequences will be used, and need to be able support any possible sequences.

Examples will be shown on the same base, “ก”.

Case 1: Non-interacting marks in distinct, non-zero classes

This case involves an above mark and a below mark in distinct, non-zero classes. In principle, these cases should not be problematic: the differently-ordered sequences cannot be visually distinct, and they are folded by normalization.




| Classes | Sequences | Appearance |
|---|--|------------|
| Phintuu (9), tones (107) | <0E01 0E3A 0E48> “ก̑” ≡ <0E01 0E48 0E3A> “ก̑̑” | |
| Phintuu (9), generic above marks (230) | <0E01 0E3A 0303> “ก̑̑̑” ≡ <0E01 0303 0E3A> “ก̑̑̑” | |
| Below vowels (103), tones (107) | <0E01 0E38 0E48> “ก̑̑” ≡ <0E01 0E48 0E38> “ก̑̑” | |

| | | |
|---|--|---|
| Below vowels (103), generic above marks (230) | <OE01 0E38 0303> “กึ” ≡ <OE01 0303 0E38> “กึ” |  |
| Generic below marks (220), tones (107) | <OE01 0331 0E48> “กึ” ≡ <OE01 0E48 0331> “กึ” |  |
| Generic below marks (220), generic above marks (230) | <OE01 0331 0303> “กึ” ≡ <OE01 0303 0331> “กึ” |  |

Note that the shaping engine in Windows is differentiating between the different orders in each case except the last, which has no Thai-specific marks. This is because the Thai engine in Windows was designed to follow Thai-language conventions and legacy-implementation behaviours in which Thai marks must be entered in a particular order (phinthu, then vowel, then tone), and because it wasn't designed to accommodate generic marks — it hasn't been updated to support Patani Malay — and breaks the cluster before a Thai-specific mark that follows a non-Thai-specific character.

Case 2: Non-interacting marks, one in class 0

This case involves an above vowel mark in class 0 with a below mark. Because class 0 marks are not re-ordered during normalization, the differently-ordered sequences are not canonically equivalent. Yet there is no possible visible distinction.

| Classes | Sequences | Appearance |
|--|--|---|
| Phintuu (9), above vowels (0) | <OE01 0E3A 0E34> “กึ” ≠ <OE01 0E34 0E3A> “กึ” |  |
| Below vowels (103), above vowels (0) | <OE01 0E38 0E34> “กึ” ≠ <OE01 0E34 0E38> “กึ” |  |
| Generic below marks (220), above vowels (0) | <OE01 0331 0E34> “กึ” ≠ <OE01 0E34 0331> “กึ” |  |

Because the differently-ordered sequences are not canonically equivalent yet have no visual distinction, this can lead to spoofing vulnerability.

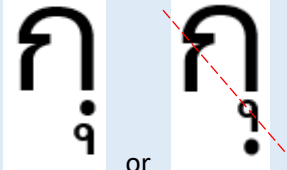
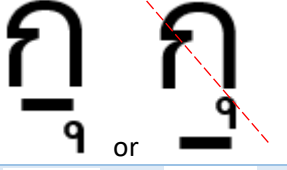
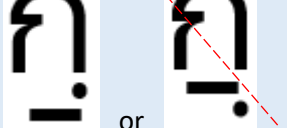
In this case, the Windows Thai engine is designed to accept only one order as valid — the below marks must occur before the above vowel mark — and it treats the other order as invalid (the cluster breaks before the second mark). In this way, a spoofing vulnerability is avoided, and users are motivated to use only one encoded representation.

The second sub-case has vowel marks below and above. I am not aware of any attested usage of such combinations, though that does not imply it will never be useful for some situation. The other two sub-cases occur in Patani Malay, and it so happens that the order *below mark + above vowel* is the one that is most useful since the below marks are used as consonant diacritics.

Case 3: Below mark + below mark, distinct classes

This case involves two below marks from distinct classes. Because the marks are all positioned below a base, they can interact typographically and have distinct relative positions. Yet because they are in distinct canonical combining classes, different relative ordering in encoded representation is folded away in normalization. Hence, visual distinctions from different ordering cannot be preserved.

Since no visual distinction can be preserved, one of the two possible relative positions of the marks should be preferred over the other. A priori, there is no reason to prefer either over the other. But actual usage in particular writing systems may provide a basis to prefer one over the other.

| Classes | Sequences | Appearance |
|--|--|---|
| Phintuu (9), below vowels (103) | <0E01 0E3A 0E38> “ᨧᨶ” ≡ <0E01 0E38 0E3A> “ᨧᨶ” |  |
| Generic below marks (220), below vowels (103) | <0E01 0331 0E38> “ᨧᨶ” ≡ <0E01 0E38 0331> “ᨧᨶ” |  |
| Phintuu (9), generic below marks (220) | <0E01 0E3A 0331> “ᨧᨶ” ≡ <0E01 0331 0E3A> “ᨧᨶ” |  |

It so happens that the first two of these combinations are used in Patani Malay, using the appearance on the left in each case. In the first case, the visual appearance is consistent with the canonical ordering — the phintuu (9) is closer to the consonant than the vowel mark. In the second case, however, the canonical order is contrary to the visual appearance: the macron below must be closer to the consonant, but in canonical order it follows the vowel.

The third case is not used in Patani Malay; assuming no unforeseen requirements of other orthographies, the preferred appearance can be chosen solely on the basis of consistency with the canonical ordering.

Thus, Thai rendering engines can be designed to display the left-hand result in each case. Note that ensuring only one of these possibilities is displayed adds complexity into a Thai rendering engine.

Also note that implementations will face serious problems if there are ever any writing systems that require these marks to be displayed with the other relative positioning.

Note: In my opinion, it would have been better to encode a new Thai-specific consonant diacritic, or new consonant letters, as was done for Devanagari script: four consonant letters were added for Sindhi, each with a precomposed macron below and no decomposition. The Thai block has plenty of room for three additions.

The Windows Thai engine currently does not support using generic combining marks, and so breaks a cluster before a Thai-specific mark when it follows a generic mark. In the case of phintu + below vowels, it displays these the same, but in the opposite order to what is required for Patani Malay.

Case 4: Above mark + above mark, one in class 0

This case involves two above marks, one of them in class 0. Because the marks are all positioned above a base, they can interact typographically and have distinct relative positions. And because one of the marks is in class 0, no re-ordering occurs during normalization, and different orders are preserved.

Note that this is like the case of two marks from the same class: by definition, they interact typographically and can have different relative positions; but no re-ordering occurs in during normalization, and different orders are preserved.

These cases should not be problematic.

| Classes | Sequences | Appearance |
|--|-------------------------|------------|
| Above vowels (0), tones (107) | <0E01 0E34 0E48> “กั” | |
| | ≠ <0E01 0E48 0E34> “กั” | |
| Above vowels (0), generic above marks (230) | <0E01 0E34 0303> “กั” | |
| | ≠ <0E01 0303 0E34> “กั” | |

Note that the Windows Thai engine handles the different orders of Thai-specific marks. It does not yet handle generic marks, however, and breaks a cluster before a Thai mark when it follows a generic mark.

Case 5: Above mark + above mark, distinct non-zero classes

This case involves two above marks in distinct, non-zero classes: a generic above mark, and a tone. Because the marks are all positioned above a base, they can interact typographically and have distinct relative positions. Yet because they are in distinct canonical combining classes, different relative ordering in encoded representation is folded away in normalization. Hence, visual distinctions from different ordering cannot be preserved.

Since no visual distinction can be preserved, one of the two possible relative positions of the marks should be preferred over the other. A priori, there is no reason to prefer either over the other. But actual usage in particular writing systems may provide a basis to prefer one over the other.

| Classes | Sequences | Appearance |
|---|--|------------|
| Generic above marks (230), tones (107) | <code><0E01 0303 0E48> “ก็”</code> \equiv <code><0E01 0E48 0303> “ก”</code> | |

It so happens that this combination is used in Patani Malay, using the appearance on the left. The canonical order, however, is contrary to the visual appearance: the tone must be farther from the consonant since it modifies the syllable as a whole; but in canonical order, it precedes the vowel.

Thus, Thai rendering engines can be designed to display the left-hand result. Note that ensuring only one of these possibilities is displayed adds complexity into a Thai rendering engine.

Also note that implementations will face serious problems if there are ever any writing systems that require these marks to be displayed with the other relative positioning.

Note that the Windows Thai engine does not yet handle generic marks and breaks a cluster before a Thai mark when it follows a generic mark.

Case 6: Three above marks in distinct classes

This case involves three above marks in distinct classes: a Thai above vowel sign, a generic above mark, and a tone.

Note: These combinations are not known to be in use. Nevertheless, general-purpose implementations must allow for any possible sequences.

Because the marks are all positioned above a base, they can interact typographically and have distinct relative positions. Because these have distinct canonical combining classes, some ordering distinctions are folded during normalization. Yet because one mark is class 0, that will cause some ordering distinctions to be retained.

Three marks (representative examples are 0E34, 0E48 and 0303) have six possible orders in encoded representations. These fall into four equivalence classes under normalization.

| Equivalence class | Sequences | Appearance |
|-------------------|--|------------|
| A | <0E01 0E34 0E48 0303> “กั” ≡ <0E01 0E34 0303 0E48> “กั” | |
| B | <0E01 0E48 0E34 0303> “กั” | |
| C | <0E01 0E48 0303 0E34> “กั” ≡ <0E01 0303 0E48 0E34> “กั” | |
| D | <0E01 0303 0E34 0E48> “กั” | |

For equivalence classes A and C, we can adopt the same principle used in case 5 to decide which possible display to use: the tone marks always display outside the generic above marks.

So, we end up allowing tones to go above or below the grouping of the two other marks. But the relative ordering within that grouping is constrained if the tone is below, yet unconstrained if the tone is above. And the tone can occur between the other two only if the generic mark is below it and the Thai vowel mark is above it.

Re-evaluation

The above approach has tried to maintain these general principles:

- That two canonically-equivalent sequences are equally valid and should display the same.
- That two sequences that are not canonically equivalent and that have distinct displays can be equally-valid sequences.

Given the nature of the combining marks used in Thai and their canonical combining classes, however, it has been necessary to make choices that remove any possibility of displaying certain visual arrangements of marks, and we have needed to introduce some non-trivial complexity into rendering logic.

Suppose we were to revisit case 4, specifically the sub-case of Thai above vowels and tones. Because the differently-ordered sequences are not canonically equivalent and can have a distinct display, it was assumed above that each should be treated as valid. Yet this resulted in an interaction between tones and other above marks that could not be maintained consistently when we came to case 5. There is no known use case in which tone marks should be positioned below vowel marks, however. So, perhaps if we are willing to rule out that possibility — as we have been required to do for other plausible arrangements of marks because the character sequences are canonically equivalent — then perhaps it might allow some simplification of logic. The new constraint would be not to treat as valid a sequence in which Thai above vowel sign (CCC = 0) occurs within the same sequence as a preceding tone mark.

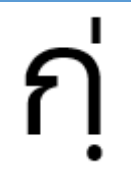

But if we

Thai rendering rules

With the above analysis, we arrive at certain tentative rules for a Thai rendering engine:




- If a combining sequence includes a tone mark, and a Thai above vowel is encountered within the same combining sequence but after the tone mark, then break the cluster before the Thai above vowel. (The vowel and any following marks will then comprise an invalid cluster.)
- If a combining sequence has a Thai above vowel, and if a below mark (phintuu, Thai below vowel or generic below mark) is encountered within the same combining sequence but after the above vowel, then break the cluster before the below mark. (These will then comprise an invalid cluster.)
- If a valid cluster sequence contains any tone marks, move the tone marks (maintaining their relative order) to the end of the sequence.
- If (after the preceding steps) a valid cluster sequence ends with a tone mark and the tone mark is followed by U+0E33 THAI CHARACTER SARA AM, then replace the sara am with the sequence <0E4D 0E32>, and then move 0E4D directly before the tone mark.
- Within a valid cluster sequence, if there is a phintuu, move it immediately after the base glyph.
- Within a valid cluster sequence, if there are generic below marks, move them (maintaining their relative order) to a position following the base glyph and, if present, following phintuu.
- Other marks within a valid cluster sequence retain their relative ordering.

This is how each of the example of sequences given above would display following these rules:

| Reference | Sequence(s) | Appearance |
|--|--|---|
| Case 1, phintuu + tones | <0E01 0E3A 0E48> “กั” ≡ <0E01 0E48 0E3A> “กั” |  |
| Case 1, phintuu + generic above marks | <0E01 0E3A 0303> “กั” ≡ <0E01 0303 0E3A> “กั” |  |

| | | |
|---|--|----|
| Case 1, below vowels + tones | <0E01 0E38 0E48> “ဂံ” ≡ <0E01 0E48 0E38> “ဂံ” | ဂံ |
| Case 1, below vowels + generic above marks | <0E01 0E38 0303> “ဂံ” ≡ <0E01 0303 0E38> “ဂံ” | ဂံ |
| Case 1, generic below marks + tones | <0E01 0331 0E48> “ဂံ” ≡ <0E01 0E48 0331> “ဂံ” | ဂံ |
| Case 1, generic below marks + generic above marks | <0E01 0331 0303> “ဂံ” ≡ <0E01 0303 0331> “ဂံ” | ဂံ |
| Case 2, phintuu + above vowels, order (i) | <0E01 0E3A 0E34> “ဂံ” | ဂံ |
| Case 2, phintuu + above vowels, order (ii) | <0E01 0E34 0E3A> “ဂံ” | ဂံ |
| Case 2, below vowels + above vowels, order (i) | <0E01 0E38 0E34> “ဂံ” | ဂံ |
| Case 2, below vowels + above vowels, order (ii) | <0E01 0E34 0E38> “ဂံ” | ဂံ |
| Case 2, generic below marks + above vowels, order (i) | <0E01 0331 0E34> “ဂံ” | ဂံ |

| | | |
|---|--|--|
| Case 2, generic below marks + above vowels, order (ii) | <0E01 0E34 0331> “နီ” | |
| Case 3, phintuu + below vowels | <0E01 0E3A 0E38> “န့” ≡ <0E01 0E38 0E3A> “န့” | |
| Case 3, generic below marks + below vowels | <0E01 0331 0E38> “န့” ≡ <0E01 0E38 0331> “န့” | |
| Case 3, phintuu + generic below marks | <0E01 0E3A 0331> “န့” ≡ <0E01 0331 0E3A> “န့” | |
| Case 4, above vowels + tones, order (i) | <0E01 0E34 0E48> “နီ” | |
| Case 4, above vowels + tones, order (ii) | <0E01 0E48 0E34> “နီ” | |
| Case 4, above vowels + generic above marks, order (i) | <0E01 0E34 0303> “နီ” | |
| Case 4, above vowels + generic above marks, order (ii) | <0E01 0303 0E34> “နီ” | |
| Case 5, generic above marks + tones | <0E01 0303 0E48> “နီ” ≡ <0E01 0E48 0303> “နီ” | |
| Case 6, above vowels + tones + generic above marks, equivalence class “A” | <0E01 0E34 0E48 0303> “နီ” ≡ <0E01 0E34 0303 0E48> “နီ” | |

| | | |
|--|---|---|
| <p>Case 6, above vowels + tones + generic above marks, equivalence class "B"</p> | <p><0E01 0E48 0E34 0303> “နီ”</p> |  |
| <p>Case 6, above vowels + tones + generic above marks, equivalence class "C"</p> | <p><0E01 0E48 0303 0E34> “နီ” ≡ <0E01 0303 0E48 0E34> “နီ”</p> |  |
| <p>Case 6, above vowels + tones + generic above marks, equivalence class "D"</p> | <p><0E01 0303 0E34 0E48> “နီ”</p> |  |