

TEI Beyond TEI

Kevin Graaf

Visual Interactions in Early Writing Systems (VIEWS),
University of Cambridge*

*Starting 1 October 2026

11 August 2026



1. Introduction

100% Human-made

Everything in this presentation is 100% human-made.

No “AI.”

No LLMs.

- I'm interested in digital typography and text encoding, especially as they relate to ancient, minoritized, and emerging scripts
- I'll be continuing this work come this October as a part of the Visual Interactions in Early Writing Systems (VIEWS) Project (Faculty of Classics, University of Cambridge)

Questions I want to answer

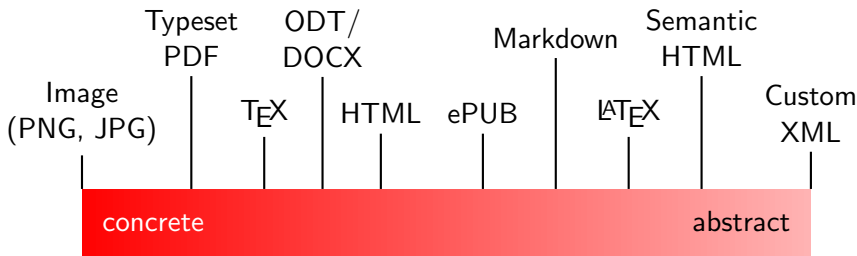
- Text encoding may seem a solved problem, but is it actually?
- What is the relationship between text and encoding and markup?
- How can we think *methodically* about text encoding?
- How can encoding schemes be more inclusive of minoritized and/or emerging (i.e. new) scripts?

2. Some Observations About Markup

- XML frequently criticized for child vs. attribute distinction
- Indeed, when used (poorly!) as a serialization format, not obvious where to use what
- But as a document format, XML is well suited to its role!
- A simple, analytical test: “text” is what remains if you ask a proficient reader to enumerate every character/glyph/grapheme/letter on a printed, typeset page.
- Paratext is everything else
- I argue: the child/attribute distinction in XML *just is* a/the text-paratext distinction
- Stripping markup = eliminating paratext

Concrete vs Abstract Representations of Text

- Could also say “semantic vs non-semantic,” etc.
- We can imagine a continuum, like so



TEI's Crucial Insight

- TEI is noticeably absent from this diagram
- TEI is *neither* fully abstract *nor* fully concrete w.r.t. formatting
- Instead, allows users to be as specific *as they want* (up to a point)
- Very, very difficult to define fully abstract/semantic encodings of text
- Common to suggest that document formats should “separate content from presentation,” all fine and good except. . .
- There is no single boundary between “content” and “presentation” which applies to all text!
- As a result, formats like HTML (and encodings like Unicode!) are an incoherent mishmash of both
- Worse still, difficult to extract exact information desired according to different criteria

Content vs. Presentation

- Instead of positing a universal content/presentation distinction. . .
- When encoding text, ask what information must be preserved
- Then consider which *views* into text “shake out” everything but those aspects salient to problem at hand
- For example:
 - In corpus linguistics, everything but words might be irrelevant
 - A casual reader just wants a nicely typeset page, doesn’t care about critical apparatus
 - Grapholinguists want to know *exactly* what was written down
- I call this a *telescoping* representation

Anglocentricity of XML

- Markup like XML allows us to cordon off textual complexity
- Describe structure in terms of a simple character set (ASCII) that's easy to edit, then render to {PDF, HTML, ePub, ...}
- Transparency of XML-based formats is a serious advantage over opaque rich text formats (Word, etc.)
- Except, this requires all metadata to be in English (or Basic Latin)
- Ok, let's let tags be in Unicode
- Now we have to worry about text editing in every writing system
- Greek, Cyrillic, CJK aren't so bad!
- What about Nastaliq? Bidirectional Text? Egyptian?
- Now we're back to where we started!
- Defining all higher structure in terms of a simple character is a good idea! ASCII just isn't an equitable choice

Markup vs Text Encoding

- My claim: boundary between markup and text encoding is unnatural
- Let's say we have an unencoded glyph, we could:
 - ① Declare an entity in our DTD
 - ② Use TEI's `<char>` or `<glyph>` feature
 - ③ Use the Private Use Area
 - ④ Beg the Unicode Consortium for encoding space
- Which is the right choice?
- We can argue till we're blue in the face, but...
- Obviously no fact of the matter, overlap in functionality is a result of historical contingencies
- I'm not the first to notice this, see Atkin's *Metacode* from 2001.
- Many, many more examples of this phenomenon (as we shall see)

Some interesting features of `<char>`

- TEI's `<char>` let's us do a couple unique things:
 - ① We can define arbitrary properties
 - ② We can define arbitrary mappings
- Difficult problems become easy with better representations
- Properties and mappings fill in the missing information that makes text processing hard!
- With this kind of power, what do we even need Unicode for?
- (Wink, hint, nudge. . .)

3. Flaws of Unicode

Unicode: A Critique

- *Unicode* really is *universal*
- This was hard won! Decades of work and advocacy
- A monumental achievement to be sure, but...
- ...No technology should be sacrosanct!
- I want to offer some critiques of Unicode

Unicode: A Critique

- *Unicode* really is *universal*
- This was hard won! Decades of work and advocacy
- A monumental achievement to be sure, but...
- ...No technology should be sacrosanct!
- I want to offer some critiques of Unicode
- Now, excuse me while I duck...



The Long Shadow of Legacy Encodings

- An explicit goal of Unicode was round-trip compatibility with legacy character sets and encodings, essentially the “Cartesian product” of previous encodings
- A worthy goal perhaps, but problematic. . .
- The source of much of Unicode’s complexity
- In particular, the ultimate reason for its multiple normal forms
- Unicode is now older than many/most of its predecessors were when it (Unicode) was new
- Nobody remembers or cares about e.g. Classic Mac OS codepages, must we pay the price of compatibility forever?
- This isn’t free! Security issues (w.r.t. string equivalence), bugs, immense programmer effort

3.1. The Myth of Plain Text

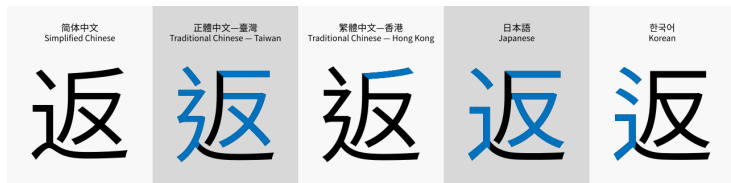
What is Plain Text?

Plain text is the underlying content stream to which formatting can be applied. . . [It] must contain enough information to permit the text to be rendered legibly, and nothing more

[Unicode] inherently assumes that the geometry of 1-dimensional graphetic sequences—as long as there is no 2- dimensional higher structure such as a list or a table—carries no syntactic or semantic information. . . [but]. . . [t]here are cases where human creativity has transcended this model. . .

—Yannis Haralambous and Martin Dürst,
“Unicode from a linguistic point of view”

Some Considerations



- Correctly formatted, mixed Chinese-Japanese text is practically *impossible* in plain Unicode
- Similar problem with Russian/Ukrainian-Bulgarian Cyrillic
- Scripts like Nastaliq need more formatting than Unicode provides to be correctly rendered at a baseline, writing communities resultingly abuse spaces, etc.
- Some notations (e.g. Algol 60) used bolding to indicate syntactically basic distinctions*

*Unicode has legacy bolded Latin presentation forms, doesn't work for other scripts

```

procedure euler (fct, sum, eps, tim) ; value eps, tim ;
integer tim ; real procedure fct ; real sum, eps ;
comment euler computes the sum of fct(i) for i from zero up to
infinity by means of a suitably refined euler transformation. The
summation is stopped as soon as tim times in succession the absolute
value of the terms of the transformed series are found to be
less than eps. Hence, one should provide a function fct with one
integer argument, an upper bound eps, and an integer tim. The
output is the sum sum. euler is particularly efficient in the case
of a slowly convergent or divergent alternating series. ;
begin integer i, k, n, t ; array m[0:15] ; real mn, mp, ds ;
i := n := t := 0 ; m[0] := fct(0) ; sum := m[0]/2 ;
nextterm: i := i+1 ; mn := fct(i) ;
    for k := 0 step 1 until n do
        begin mp := (mn+m[k])/2 ; m[k] := mn ;
            mn := mp end means ;
    if (abs(mn)<abs(m[n]))^(n<15) then
        begin ds := mn/2 ; n := n+1 ; m[n] :=
            mn end accept
    else ds := mn ;
    sum := sum + ds ;
    if abs(ds)<eps then t := t+1 else t := 0 ;
    if t<tim then go to nextterm
end euler

```

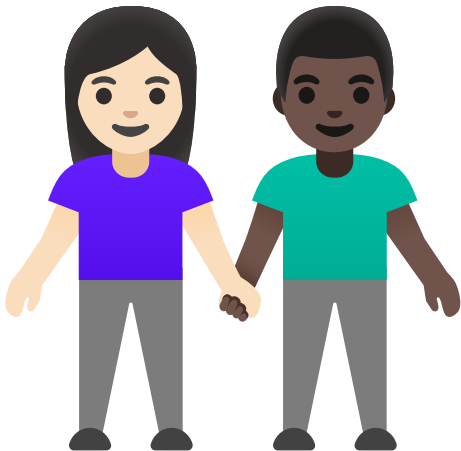
- So is Unicode's "plain text" actually just text (vs paratext)?
- No, not really, at least not in practice
- Recall the legibility criterion
- Unicode has many things firmly in the territory of paratext, lacks others in text
- "Text" should not be assumed to be legible on its own, nor is paratext "extra" formatting
- Conclusion: "plain text" is *not* a trans-historical, cross-scriptal category

The Necessity of Structure

- Unicode is nominally “flat,” linear, and unstructured
- But as we shall see, it contains many forms of structure (each with a different syntax!)
- The perceived simplicity of unstructured text is deceptive
- Merely rearranges where the complexity is located.
- Simpler yes, but simpler for whom?
- Can we cut this Gordian Knot and simply accept structure?

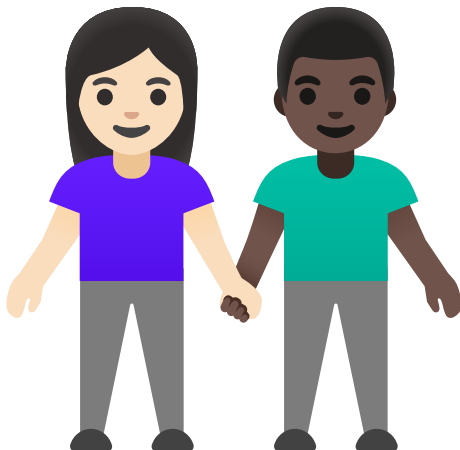
Emoji

Many emoji are built from other emoji with “ZWJ” sequences



Emoji

Many emoji are built from other emoji with “ZWJ” sequences



- 1  *Woman*
- 2  *Skin Color Modifier*
- 3  *Zero Width Joiner*
- 4  *Handshake*
- 5  *Zero Width Joiner*
- 6  *Man*
- 7  *Skin Color Modifier*

Egyptian Hieroglyphs Layout Control

- Unicode includes layout control characters for Egyptian
- *Structured* design, complete with nesting brackets

- E.g. <-i-mn:n-m-HAt:t->



- Seems like a natural fit for XML!
- Conversely, if using “native” Unicode Egyptian in TEI docs, must deal with two kinds of structure, one of which is totally opaque in most editors
- Very awkward overlap in functionality
- Could use pure TEI + Gardiner numbers (or the like), except for this being:
 - 1 ugly
 - 2 unergonomic

	1343	1344	1345
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
A			
B			
C			
D			
E			
F			

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Unicode can *almost* handle many (simple) instances of math notation
- But this obviously overlaps with T_EX, MathML, OpenMath, etc.
- XML is *much* nicer to work with than Unicode!

3.2. Think Systematically About Text Operations

Underdeterminedness

- Why are common text processing operations difficult in Unicode?
- Consider case folding:
 - In German, $\text{SS} \mapsto \text{ß}$ (sometimes)
 - in Turkish, $\text{İ} \mapsto \text{i}$, and $\text{I} \mapsto \text{i}$
- Unicode case folding is really broken, because downcasing is not the inverse of upcasing!
- Simply put, such operations are *underdetermined*!
- Alternatively, there's a hidden parameter here
- Instead of $\text{fold}(\sigma : \text{string}) \rightarrow \text{string} \dots$
- $\dots \text{fold}(\sigma : \text{string}, \ell : \text{locale}) \rightarrow \text{string}$
- In other words, mere Unicode text doesn't contain enough information by itself to correctly perform such operations

Well known in e.g. type theory that equality is the hardest problem!

Equality is the basis of many (most?) common string operations:

- Search? “Are there are substrings which match (i.e. are equal to) some string?”
- Hash table indexing
- The list goes on. . .

Normalization

- Unicode *does* have a notion of normalization
- Poor “algebraic” properties, not closed under concatenation (!!)
- Much more about legacy considerations

Textual Equalities

There is no single equality operator over text which can cover all needs

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style**

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style**

equality *modulo* style

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style**
- blessed vs blessèd

equality *modulo* style

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style**

equality *modulo* style

- blessed vs blessèd

equality *modulo* diacritics

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style**
- blessed vs blessèd
- red vs red

equality *modulo* style
equality *modulo* diacritics

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style**
- blessed vs blessèd
- red vs red

equality *modulo* style

equality *modulo* diacritics

equality *modulo* color

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style** equality *modulo* style
- blessed vs blessèd equality *modulo* diacritics
- red vs red equality *modulo* color
- CAPITAL vs capital vs cApItAL

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style** equality *modulo* style
- blessed vs blessèd equality *modulo* diacritics
- red vs red equality *modulo* color
- CAPITAL vs capital vs cApItAL equality *modulo* case

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style** equality *modulo* style
- blessed vs blessèd equality *modulo* diacritics
- red vs red equality *modulo* color
- CAPITAL vs capital vs cApItAL equality *modulo* case
- a vs α

Textual Equalities

There is no single equality operator over text which can cover all needs

- *style* vs **style** equality *modulo* style
- blessed vs blessèd equality *modulo* diacritics
- red vs red equality *modulo* color
- CAPITAL vs capital vs cApItAL equality *modulo* case
- a vs α equality *modulo* variant

4. Inclusivity

How Unicode Fails Ancient/Minoritized/Emerging Scripts



- Unicode encoding space is large, but finite
- Must be centrally coordinated:
 - Ensure everyone uses disjoint ranges of codepoints
 - Don't waste encoding space on spurious, vanity, or immature encodings
- **HOWEVER**, Unicode Consortium is now de facto arbiter of what is and is not a “real” script, what distinctions are worth encoding, etc
- No matter how we dice it, this *is* normative
- Requirements for encoding are effectively circular:
 - Must show organic uptake to be a candidate
 - Being able to use a script for ephemera (e.g. texting) is practically required for such uptake!
- Unicode encodings are final, so difficult to know when scholarship is sufficiently mature to encode ancient or undertheorized scripts

5. Introducing Telecode

Prolegomena to Any Future Text Representation

- I want to sketch a new way to represent text—a scheme I call Telecode
- If we were to do Unicode over again, how would we?
- Difficult to untangle different spheres of interest. These categories can't always be neatly separated
- Linguists, paleographers, programmers, everyday users all want different things
- Sometimes you want the text in itself, other times you need metadata to interpret the text (cf. Han unification)
- So don't choose! Have both!
- This is a *telescoping* encoding—a single mechanism which represents structure at all levels of organization

Telecode In Brief

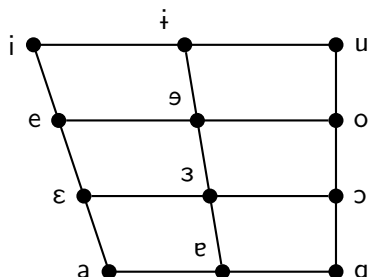
- Rather than a text *encoding*, Telecode is a *data structure* for text
- In other words: get the data model right first, worry about concrete representation later
- The Telecode data model is a superset of XML
- The Big Ideas:
 - Delimit metadata/markup with dedicated codes instead of “in-band” syntax
 - Represents graphemes/glyphs as *names* instead of codepoints for an unbounded encoding space, just like XML tags
 - Eliminate as much central coordination as possible, use namespaces to avoid collisions
 - Decompose all properties and components
 - Be self-describing, derive properties directly from the encoding itself (no Unicode database required)
 - Make common text operations simple and reliable, without requiring knowledge of specific languages

Universal Naming Alphabet (UNA)

- How then do we encode names in any of the world's languages?
- Latin Alphabet is neither neutral, nor well-suited to this role
- UNA is my proposed solution to this problem
- UNA is a set of 56 codes or “letters” in which mostly reasonable “orthographies” can be designed for most languages
- Somewhat similar to the International Phonetic Alphabet (IPA), with some simplifications
- I use IPA-like symbols to represent UNA codes, merely notational
- cf. relationship between numbers and numerals, $XVI = 16_{10} = F_{16}$
- Not a human writing system, nor intended as a perfect phonetic/phonemic transcription scheme
- Rather, a linguistically/culturally neutral system for phonetic approximations
- Still under active development!

UNA Letters

	Lab.	Alv.	Retro.	Pal.	Vel.	Uvu.	Laryn.
Nasal	m	n	ɲ	ɲ	ŋ	ɴ	
Lateral		l	ɭ	ʎ	ʟ		
Tap	v	r					
Stop	p	t	ʈ	c	k	q	ʔ
Sibilant		s	ʂ	ʃ			
Fricative	f	θ		ç	x	χ	ħ



Modifier	Symbol
Click	!
Nasal	ñ
Non-Default	h
Breathy	ḥ
Creaky	ḥ̣
Harsh	ḥ̥
Faucalized	ḥ̠
Retracted	ḥ̡
Rounded	ḥ̜
Non-Syllabic	ḥ̚
Stressed	ḥ̌

Tone	Symbol
1	˩
2	˨˩˩
3	˨˩˨˩˩
4	˨˩˨˩˩˩
5	˨˩˨˩˩˩˩

	0	1	2	3	4	5	6	7
0	m	n	ɲ	ɲ	ŋ	ɴ	l	ɭ
1	ʎ	ʟ	v	r	p	t	ʈ	c
2	k	q	ʔ	s	ʂ	ʃ	f	θ
3	ç	x	χ	ħ	i	ɨ	u	e
4	ə	o	ɛ	ɜ	ɔ	a	ɐ	ɑ
5	!	ñ	h	ḥ	ḥ̣	ḥ̥	ḥ̠	ḥ̡
6	w	ḥ̜	ʔ	˩	˨˩˩	˨˩˨˩˩	˨˩˨˩˩˩	˨˩˨˩˩˩˩

Attribute Model

- In XML, physical order of attributes is not semantically significant, i.e. does not contribute to the meaning of a document
- In Unicode, the physical order of combining characters (i.e. attributes) is *partially* significant.
- XML attributes can take an arbitrary string as an argument
- Unicode attributes are essentially binary (present or not)
- Telecode model is a synthesis of these two designs
- Attribute order is insignificant, but payload can be structured
- Elements in attributes are tagged with set of equivalence classes
- Thus, easy to compare text *modulo* some set of properties by ignoring all elements in attributes tagged by pertinent class(es)
- Can therefore indicate “what” vs “how” w.r.t diacritic modification, accent grave is doing something different in English *blessèd* than French *la* vs *là*

A Taste of Telecode

Let's look at how we might encode “Straße” (*street* in German) in Telecode. I will use the abstract syntax, with English instead of UNA names for readability.

$$\mathcal{G} \langle \text{latin : es} \rangle \left[\text{latin : case} \mapsto \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:upper} \rangle} \right],$$

$$\mathcal{G} \langle \text{latin : tee} \rangle \left[\text{latin : case} \mapsto \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:lower} \rangle} \right],$$

$$\mathcal{G} \langle \text{latin : ar} \rangle \left[\text{latin : case} \mapsto \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:lower} \rangle} \right],$$

$$\mathcal{G} \langle \text{latin : ei} \rangle \left[\text{latin : case} \mapsto \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:lower} \rangle} \right],$$

$$\mathcal{A} \langle \text{latin : case} \rangle \left\{ \begin{array}{l} \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:upper} \rangle} \mapsto \mathcal{G} \langle \text{latin : es} \rangle \left[\text{latin : case} \mapsto \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:upper} \rangle} \right], \\ \mathcal{G} \langle \text{latin : es} \rangle \left[\text{latin : case} \mapsto \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:upper} \rangle} \right] \\ \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:lower} \rangle} \mapsto \mathcal{G} \langle \text{german : eszett} \rangle \left[\text{latin : case} \mapsto \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:lower} \rangle} \right] \end{array} \right\}$$
$$\left[\text{latin : case} \mapsto \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:lower} \rangle} \right],$$

$$\mathcal{G} \langle \text{latin : ee} \rangle \left[\text{latin : case} \mapsto \mathcal{M}_{|\text{latin:typo}|}^{\langle \text{latin:upper} \rangle} \right],$$

TCS-8 Encoding

In addition to the 56 UNA Letters, there are also 8 control characters:

- 0 — NUL reserved
- 1 — END (E) delimiter
- 2 — SCR (S) begin structure
- 3 — GRF (G) begin grapheme
- 4 — ALT (A) begin alternate/mapping
- 5 — CMP (C) begin compound
- 6 — ATR (M) begin attribute
- 7 — SPC (W) white space

Total of 64 basic codes. Conveniently, if inefficiently, store each in a byte. TCS-8 is also *self-normalizing*, there is only one way to represent the same Telecode text, encoding enforces that attributes are always sorted

A Taste of Telecode – TCS-8 Encoding

G latiinum E es E

M latiinum E kasus E S latiinum E mai^hor E E M latiinum E tupus E E

G latiinum E tee E

M latiinum E kasus E S latiinum E minor E E M latiinum E tupus E E

G latiinum E er E

M latiinum E kasus E S latiinum E minor E E M latiinum E tupus E E

G latiinum E aa E

M latiinum E kasus E S latiinum E minor E E M latiinum E tupus E E

A latiinum E kasus E

S latiinum E minor E E M latiinum E tupus E

G thoit^f E eset E

M latiinum E kasus E S latiinum E minor E E M latiinum E tupus E E

E

S latiinum E m³h⁷ E E M latiinum E tupus E

G latiinum E es E

M latiinum E kasus E S latiinum E mai^hor E E M latiinum E tupus E E

G latiinum E es E

M latiinum E kasus E S latiinum E mai^hor E E M latiinum E tupus E E

E

E

M latiinum E kasus E S latiinum E minor E E M latiinum E tupus E E

G latiinum E ee E

M latiinum E kasus E S latiinum E minor E E M latiinum E tupus E E

- Now to address the elephant in the room
- Obviously, coding overhead of TCS-8 is . . . significant
- Isn't this massively inefficient?
- A few points to consider first:
 - Efficient for whom? UTF-8 privileges English
 - To what extent does this matter in the era of TB-sized SSDs?
 - Many protocols, filesystems already transparently compress everything
- TCS-8 isn't intended as a wire format, rather a maximally simple representation
- I have a few ideas on how to make things more efficient
 - TCS-8 already DEFLATE's well
 - Design Telecode-aware protocols
 - Design more sophisticated transmission formats

TCS-C Compression Scheme

- Full description is beyond scope of this talk
- However, essence is very simple
- Use back references to refer to previously used names and/or structures
- Somewhat similar to SCSU (if anyone remembers *that*)
- My experiments suggest this largely eliminates coding overhead for at least alphabets and syllabaries
- More work needed on CJK

Bringing Order to Chaos

- Obviously, would be total chaos if every user made their own decision about how to encode their writing system
- My thought is that writing communities would internally standardize encoding
- What's important is that individual communities are sovereign, can make their own decisions
- Ideally would publish schemata, akin to RELAX NG (but for Telecode)

What's Next?

- So, did we defeat Unicode?
- (pause for laughs, jeers)
- I don't know if Unicode can or should be replaced, but. . .
- The assumption that computer text is “solved” is stultifying—prevents us from better understanding the domain
- I hope I have at least shown that the *theory* of text encoding is interesting—and subtle!
- Maybe Unicode won't be replaced, but we might develop better data structures and better libraries for handling text?

Questions?

email: `kevingraaf@kbg.sh`

website: `https://kbg.sh`

code: `https://fossil.kbg.sh`