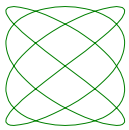
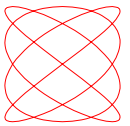
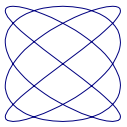
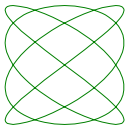
ஒருங்குறி –
தோற்றமும் கோட்பாடும்

முனைவர் ஸ்ரீரமணஸர்மா
இந்தியவியல்/தொழில்நுட்ப
ஆய்வாளர்
தமிழ்நாடு

Genesis and Philosophy of Unicode

Shriramana Sharma, Ph D
Indology/Technology Research Scholar
Tamil Nadu
jamadagni at gmail dot com

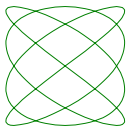
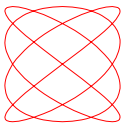



தமிழ்
இணையக் கல்விக்கழகம்

மற்றும்

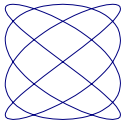
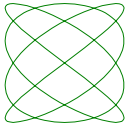
உத்தமம் (இந்தியக்கிளை)

வழங்கும்
சொற்பொழிவு

History of text encoding

(Background of Unicode)

Anatomy of digital text

- Well known: computer understands only 0s/1s -> numbers
- Text represented as sequences of numbers
- Each “letter” i.e. smallest textual unit given a number – mapping should be one to one
- Pre-agreement on this mapping – “character encoding model”

ASCII

- One of oldest encodings – first edition 1963
- 7 bits (0/1) => 2^7 = 128 combinations
- Max capacity 128 – 0 to 127 used as “codepoints” for various characters
- A-Z, a-z, 0-9 and other symbols (for punctuation etc)
- Other technical “control code” characters

7-bit vs 8-bit

- 7 bits = 128 chars sufficient for English with basic punctuation
- 8 bits (powers of two in general) are efficient in computer data storage
- 128 additional characters can be denoted using 8 bits
- total 256 ($=2^8$) characters
- most post-ASCII legacy encodings

Which additional characters?

- ASCII not sufficient for various European and other languages
- Supplementary characters for sets of languages (East European, West European) added in extra 128
- To switch from East to West European (for example) one had to select another “code page”

ISCII – 1

- 8-bit encoding for Indian scripts
- Evolved by GOI in 1980s
- ASCII in first 128 characters
- Indic characters in next 128

01	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
16	32	48	64	80	96	112	128	144	160	176	192	208	224	240	
NUL	DLE	SP	0	@	P	`	p				ओ	ढ	ऱ	ॐ	EXT
SOH	DC1	!	1	A	Q	a	q			ँ	औ	ण	ल	॑	०
STX	DC2	"	2	B	R	b	r			ं	ऑ	त	ळ	॒	१
ETX	DC3	#	3	C	S	c	s			ः	क	थ	ऴ	॓	२
EOT	DC4	$	4	D	T	d	t			अ	ख	द	व	॔	३
ENQ	NAK	%	5	E	U	e	u			आ	ग	ध	श	ॕ	४
ACK	SYN	&	6	F	V	f	v			इ	घ	न	ष	ॖ	५
BEL	ETB	'	7	G	W	g	w			ई	ङ	ऩ	स	ॗ	६
BS	CAN	(	8	H	X	h	x			उ	च	प	ह	़	७
HT	EM	)	9	I	Y	i	y			ऊ	छ	फ	INV	ऽ	८
LF	SUB	*	:	J	Z	j	z			ऋ	ज	ब	ा	।	९
VT	ESC	+	;	K	[	k	{			ऌ	झ	भ	ि		
FF	FS	,	<	L	\	l	|		ऍ	ञ	म	ी		
CR	GS	-	=	M	]	m	}			ऎ	ट	य	ु		
SO	RS	.	>	N	^	n	~			ए	ठ	य़	ू		
SI	US	/	?	O	_	o	DEL			ऐ	ड	र	ृ	ATR	

ISCII – 2

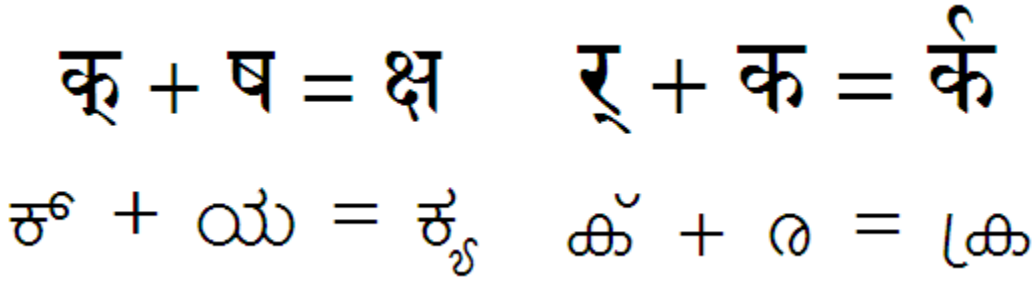
- Common set of characters KA KHA etc -> unified encoding for all Indic scripts
- Per-script unique characters (Tamil/Malayalam LLLA etc) also added
- Classification – independent vowel, consonant, vowel sign, combining marks
- Control characters used to change script, add bold etc formatting

Disadvantage of 8-bit encodings

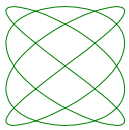
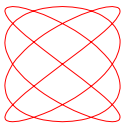
- 8-bits = 256 characters cannot accommodate the huge number of different writing systems of the world (or even of Europe)
- Code page selection problems – not all systems had support for all codepages
- Different standards for codepages
- Universal interchange of data/information very difficult

Additional difficulties in ISCII

- ISCII depends on software to support properly displaying combinations of letters (ligatures like K·SSA, combining forms like reph)

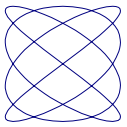
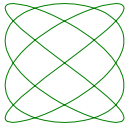


- Complex text layout
- Prevalent software did not support it

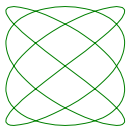
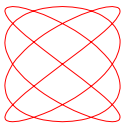
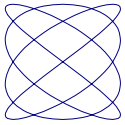
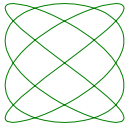
Result

- Developers of Indic fonts and software developed own encodings
- Full 256 character set allotted to Indic
  - Means font has to be changed for Indic vs English
- One script per font
  - Means font has to be changed per Indic script
- Ligatures and combining forms allotted separate codepoints

Resultant confusion

- Each vendor developed different encoding
- Text composed in one font/software would not be compatible in another
- Email/web – attach fonts too (legalities!)
- Text is meaningless without font
- Junk text displayed on screen
- Not searchable! -> Not useful for research!

Unicode - Introduction

What Unicode is and is not!

- Unicode is a standardized mapping of “characters” to codepoints
- Unicode Consortium and Unicode Technical Committee are responsible for maintaining this standard and this only
- Unicode is not the software that displays text on your screen!
- Unicode Standard/Consortium/Committee are not responsible for badly displayed text!

Unicode != its implementations

- Software that is based on Unicode is of many kinds – browsers, wordprocessors, PDF makers, publishing/graphics tools
- These (or underlying libraries) are responsible for correct display of text
- If your text is not displaying correctly, complain to Microsoft/Adobe/Apple's i18n division, not to Unicode Consortium!

Unicode = Universal code

- One codepoint for each “written form” in all the scripts of the world (all, meaning current and historic, majority and minority)
- Scripts of the world analysed into constituent orthographic units called “characters”
- Each character assigned codepoint
- (Some invisible meta-characters – ZWJ etc)

Capacity of Unicode

- Obviously, 8-bits = 256 not enough
  - Hexadecimal notation 00_{16} to FF_{16}
- Initial plan, 16-bits = 65536 (0000_{16} to $FFFF_{16}$)
- Later expanded upto $10FFFF_{16}$ (1,114,112 characters)
- Hopefully we don't have more than 1 million distinct written forms in the world [even considering Han (Chinese)!]

Scripts, not languages

- English, French, German etc languages -> single Latin script with extensions
- Arabic, Farsi, Urdu etc -> Arabic
- Hindi, Marathi, Nepali etc -> Devanagari
- Encoding a character in Unicode is done when the character is attested in writing that script; it does not mean the language is changing or being changed in any way

Current repertoire of Unicode

- Western scripts – Latin, Cyrillic etc
- Middle East scripts – Arabic, Hebrew etc
- Far Eastern scripts – Chinese, Japanese etc
- South Asian scripts:
  - Indian scripts – Tamil, Devanagari, Bengali etc
  - South-East Asian scripts – Myanmar, Balinese etc
- Historic scripts – Aramaic, Brahmi etc

Pros of Unicode - 1

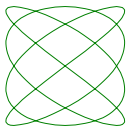
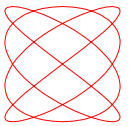
- No more codepages
- One encoding for all scripts
- Font/software/system independent
- Interoperability! Most important, since data is useless if it cannot be shared.

Pros of Unicode - 2

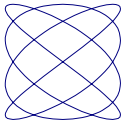
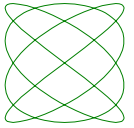
- Unicode-aware applications (browser, database, email client, GUI application, programming language interpreters) need not know anything about individual languages/scripts but just be compliant to Unicode – Python program demo
- Automatically all scripts will get supported (over time)

Perceived cons of Unicode

- Like ISCII, needs complex text layout (CTL) support in software for display (not storage)
  - Problems with Adobe software
  - PDF problem has solution
- Sometimes character model differs from native perception: Not only Tamil, Khmer
- Reason: Uniformity in model for all scripts so those without knowledge of individual scripts can develop CTL software

Principles of Unicode

Characters, not glyphs

- Character – orthographic unit
- Glyph – visual unit

- All the above glyphs (re)present the same ‘character’ “Devanagari Letter A”.
- Unicode allots it the code number 0905_{16}

Each character has...

- A “codepoint” - hex number: 0_{16} - $10FFFF_{16}$
- A character name, representative glyph
- Character properties:
  - General category – Lo, Mc, Mn, No, So, Po
  - Numeric value
  - Indic syllabic/matra category – position in Indic syllable
  - Canonical decomposition

Stability policy

- Reason: Reliability, Backward compatibility
- After encoding, following cannot be changed/removed:
  - Codepoint
  - Character name
  - Numeric value
  - Canonical decomposition
- Glyph or general category can be corrected

In case of errors...

- Only for names we have a remedy:
- FE18 – “Lenticular Brakcet”
- Added Normative Alias reading “Bracket”
- For other cases, use annotations discouraging usage of character or recommending use of another character
- Annotations - Usage notes, clarifications

Other aspects of the code charts

- Informative alias - Translations/Alternate names
- Ex: Virama -> Halant, Pulli, Pollu, Chandrakala
- Only normative alias in programs/libraries
- Remember: non-native programmers
- Cross references – related characters
- Code chart is mainly for programmers
- It is not a dictionary of the language!

No duplicate encoding

- Due to stability policy, same script will be not encoded twice with different codepoints
- Under rare circumstances “canonical decompositions” making a character equivalent to another character/sequence
- Duplicates in Latin/Cyrillic (A etc) due to compatibility issues with legacy encoding
- Security issue (if no decomposition)

Security issues

- Characters with identical/similar appearance:

  அனத அதை

- Within scripts, between scripts
- Spoofing (steal your passwords and money!)
- Strict rules that such domain names not allowed – central regulatory authority – ICANN (Internet Corp for Assigned Names/Numbers)

Planes

- 000000_{16} to $10FFFF_{16}$ divided into 17 planes of 65536 characters (0000_{16} to $FFFF_{16}$) each
- 0 0000 to F FFFF makes 16 planes, 10 xxxx is the 17th; planes 0 to 16
- Plane 0 = Basic Multilingual Plane = BMP; almost filled up, all major/current scripts
- Plane 1 = Supplementary Multilingual Plane = SMP = rarer scripts, also important for us!

Blocks

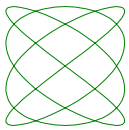
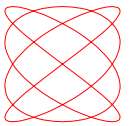
- Ranges of codepoints in multiples of 16
- Often (but not always) same old 128 per script
- If reqd, multiple blocks per script
- Latin script: 7 blocks and counting!
- Arabic: 6, Cyrillic: 4
- Winner: Han: 13, Over 70,000 characters
- Devanagari, Deva. Extd., Vedic Extensions

Encoding forms

- Important technical detail
- Unicode is an encoding; means it maps abstract characters to codepoints
- The codepoints must be stored as sequences of bytes (group of 8 bits)
- Mapping of codepoints to sequences of bytes is encoding form UTF-8, UTF-16, UTF-32
- UTF-x – minimum no. of bits per codepoint

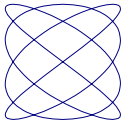
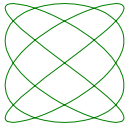
Encoding schemes

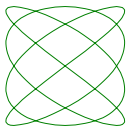
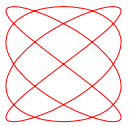
- When more than one byte is required per codepoint, the order of bytes is important
- $1000_{10} = 03E8_{16}$
- Big-endian = priority to big end of value
- 03E8 stored as 03 E8.
- Little-endian = priority to little end of value
- 03E8 stored as E8 03.
- UTF-16 LE/BE, UTF-32 LE/BE (not in UTF-8)

Careful with these

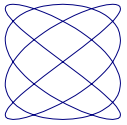
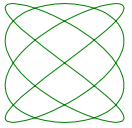
- Unicode text is sequence of codepoints
- It can be saved in any of these encoding forms/schemes
- Use the same encoding form/scheme each time you save/read a file
- Otherwise data becomes junk
- To avoid confusion, use UTF-8 always

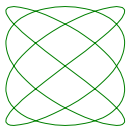
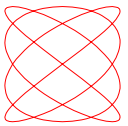
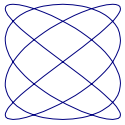
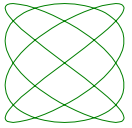
Legacy vs modern software

- People still use PageMaker!
- Such old software does *not* support Unicode
- Many free alternatives like LibreOffice, XeTeX
- Major scripts pretty well supported
- Some difficulties in support of minor scripts
- Situation will get better and better as we use these software and submit bug reports

More details

- http://unicode.org
- Unicode mailing list for doubts and clarifications
- Mostly, the problem is with your software or font and not with Unicode! ☺
- Legitimate requirements for new scripts/characters will be welcome
- On this last point, more presently...

Unicode and Indic

Current coverage of Indic scripts

- 24 scripts listed under “South Asian scripts”
- Major Indic: Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Telugu, Kannada, Tamil, Malayalam
- Old/minor Indic: Brahmi, Kharoshthi, Sharada, Kaithi, Takri, Meetei Mayek etc
- Other related: Sinhala etc

Indic encoding model

- ISCII-based model
- But obviously, separate encoding for each script – no awkward control characters
- Major scripts encoded back in Unicode 1.1 in 1993, many minor scripts and other per-script character additions after that
- Major scripts enjoy one-offset mapping from ISCII i.e. ISCII_code + offset -> Unicode

Ongoing proposals

- Various North Indian minority script proposals by Anshuman Pandey
- Miscellaneous South Indian proposals by myself and others (in date-wise order):
  - Grantha (+ Vedic)
  - Tamil fractions and symbols
  - Malayalam fractions
  - Telugu/Kannada misc. characters

General encoding of Indic scripts 1

- Independent vowels: A, AA, I, II etc

  அ ஆ இ ஈ

- Consonants: KA, KHA, GA, GHA etc

  க ஙா ச ஞ

- Vowel signs: -AA, -I, -II, -U etc

  ◌ா ◌ி ◌ீ

General encoding of Indic scripts 2

- Numerals

  க உ ங சு ஞ சூ எ அ கூ ஐ

- Other symbols (ongoing proposal)

  ஹ ஷ ஶ ஸ ஜ ஜூ ஸ ஸூ ஷ ஷூ க்ஷ க்ஷூ

  ப ஊ ஓ வ்ர ஈ ஈ சு வ ஔ ஹ

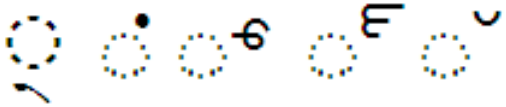
  ஐ ஸ்ரூ ள்ர ஹ வி ...

  டீ ஆ கூ டு கி குி

“Virama”

- Mark added to consonants to remove inherent vowel is called “Virama” in Unicode parlance

- Various shapes and names in various Indic scripts:



Example Code Chart - Telugu

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0C0		◌ఁ	◌ం	◌ః		అ	ఆ	ఇ	ఈ	ఉ	ఊ	ఋ	ఌ		ఎ	ఏ
0C1	ఐ		ఒ	ఓ	ఔ	క	ఖ	గ	ఘ	ఙ	చ	ఛ	జ	ఝ	ఞ	ట
0C2	ఠ	డ	ఢ	ణ	త	థ	ద	ధ	న		ప	ఫ	బ	భ	మ	య
0C3	ర	ఱ	ల	ళ		వ	శ	ష	స	హ				ఽ	◌ా	◌ి
0C4	◌ీ	◌ు	◌ూ	◌ృ	◌ౄ		◌ె	◌ే	◌ై		◌ొ	◌ో	◌ౌ	◌్		
0C5						◌ి	◌ె		◌ే	◌ై						
0C6	ౠ	ౡ	◌ౢ	◌ౣ			౦	౧	౨	౩	౪	౫	౬	౭	౮	౯
0C7									౸	౹	౺	౻	౼	౽	౾	౿

Contextual forms

- All previous characters have separate codepoints.
- However, character sequences can take *contextual forms*:

र् + क = र्क ◌ं "repha"

ಕ್ + ಯ = ಕ್ಯ ◌್ಯ "YA-phalaa"

ಕ್ + ರ = ಕ್ರ ◌್ರ "RA-vattu"

Contextual forms not encoded

- Such contextual written forms are not encoded separately:

क् + ष = क्ष क् + () = क्()

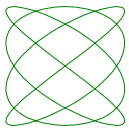
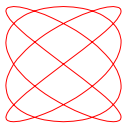
ज् + ञ = ज्ञ त् + () = त्()

ङ् + य = ङ्य () + य = ()्य

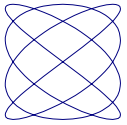
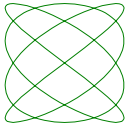
- They are produced by Complex Text Layout

Advantage of CTL

- Encoding is freed from presentation issues (Tamil ൈല vs ്ല) to represent semantic content
- NLP, text-to-speech applications based on such an encoding need not be concerned with variation in written forms and can concentrate on semantic content

Complex Text Layout

Need for complex text layout

- Basic Latin script A-Z, a-z – one character, one glyph, no combining behaviour
- Extended Latin script – combining diacritics
- Correct positioning of diacritics on base
- Indic scripts, many to many relationship of underlying text content and written forms
- Complex text layout (CTL) unavoidable!

What does CTL do? – 1

- Software processes input sequence of characters and outputs sequence of glyphs
- General rules for Indic
- Per-script rules
  - Devanagari has vowel sign I on left
  - Bengali/Oriya/Tamil/Malayalam take VS E on left and split-position VS O
  - Position of other combining forms

What does CTL do? – 2

- Text and font are input for CTL
- Glyphs with positions are output of CTL
- Text will be in usually phonetic order (as per the encoding model chosen for the script)
- Glyph display can be in any order as per the unique rules of the script in question
- “Kombu” goes left!

CTL – OpenType - 1

- Most prevalent system – OpenType
- Glyph manipulation rules are in software
- Font marks glyphs as belonging to particular categories: ligature, half-form etc
- Software reads this and produces result
- MS Windows: UniScribe
- Cross-platform: ICU, Pango, now HarfBuzz

CTL – OpenType - 2

- Disadvantages to OpenType:
- Script grammar and glyph manipulation rules cannot be easily (or legally) added, edited or tailored as per one's unique requirements
- Especially an issue with minor scripts/orthographies
- Software maker may not see commercial value in supporting your minor script

CTL – Graphite – 1

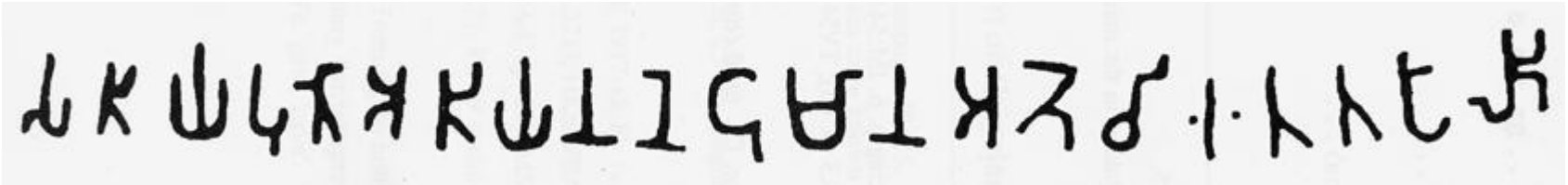
- Alternate rendering technology
- Full control of glyph processing in your hands
- If you have the requisite font glyphs, you can define the rendering as you require
- Powerful Graphite Description Language (GDL) to describe glyph processing rules
- Free software!

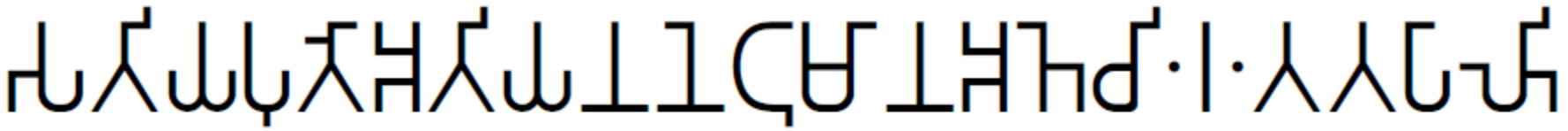
CTL – Graphite – 2

- Current support – all open-source software:
  - Firefox
  - LibreOffice
  - XeTeX
  - HarfBuzz
- Professional quality documents can be done
- Support from commercial vendors unlikely
- Again, bugs will be ironed out if we report

CTL – Graphite – 6

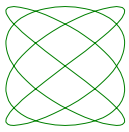
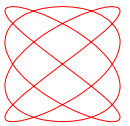
- Tamil Brahmi font Adinatha (me and friends):



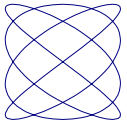
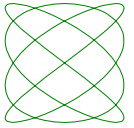


ஸதியபுதோஅதியநநேடுமாநஅஞசிஈததபாளி[1]

satiyaputōatiyananēṭumānañciītatapāḷi

Adding new scripts/characters to Unicode

Unicode proposals – 1

- Identify, standardize and categorize letters of newly proposed script
- Check that it's not already encoded
- Font making – At least glyphs for chart to be submitted with proposal
- Submit to Unicode Technical Committee

Unicode proposals – 2

- UNICODE TECHNICAL COMMITTEE:
- Experts from Google, Yahoo, Microsoft, Apple, Sybase, Adobe – search engines, database, operating system libraries, publication/archival software
- Input from academic sources / user bodies
- Meetings conducted every 3 months

Unicode proposals – 2

- CONTENTS OF PROPOSAL:
- Official proposal summary form
- Brief history of script
- Overall nature of script
- Character repertoire to be encoded – names and shapes
- Orthographic features (ligatures etc) of script to be described

Unicode proposals – 3

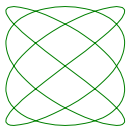
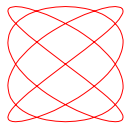
- CONTENTS OF PROPOSAL (contd.):
- Technical issues re combining behaviour etc if any need to be discussed
- Unicode Character Properties
- Both the above require good understanding of Unicode technicalities, or guidance of experienced person
- Proper references and attestations!

Standardization process

- Clearance of technical issues at Unicode Technical Committee
- Often you're asked to revise document
- Passed on to WG2 of ISO 10646 (ISO standard tracking Unicode)
- Largely formality-sake process, but two years
- Possible +ve/-ve feedback from members countries

Final publication and after

- Publication from Unicode Consortium as Unicode Standard
- (Usually later) publication from ISO as ISO 10646 revised/amendment version
- Fast/slow implementation by software vendors
- Feedback from users, bug fixes
- Settle down...

Unicode is not perfect.

But Unicode is good.

It wants to help us.

Let's help it back!