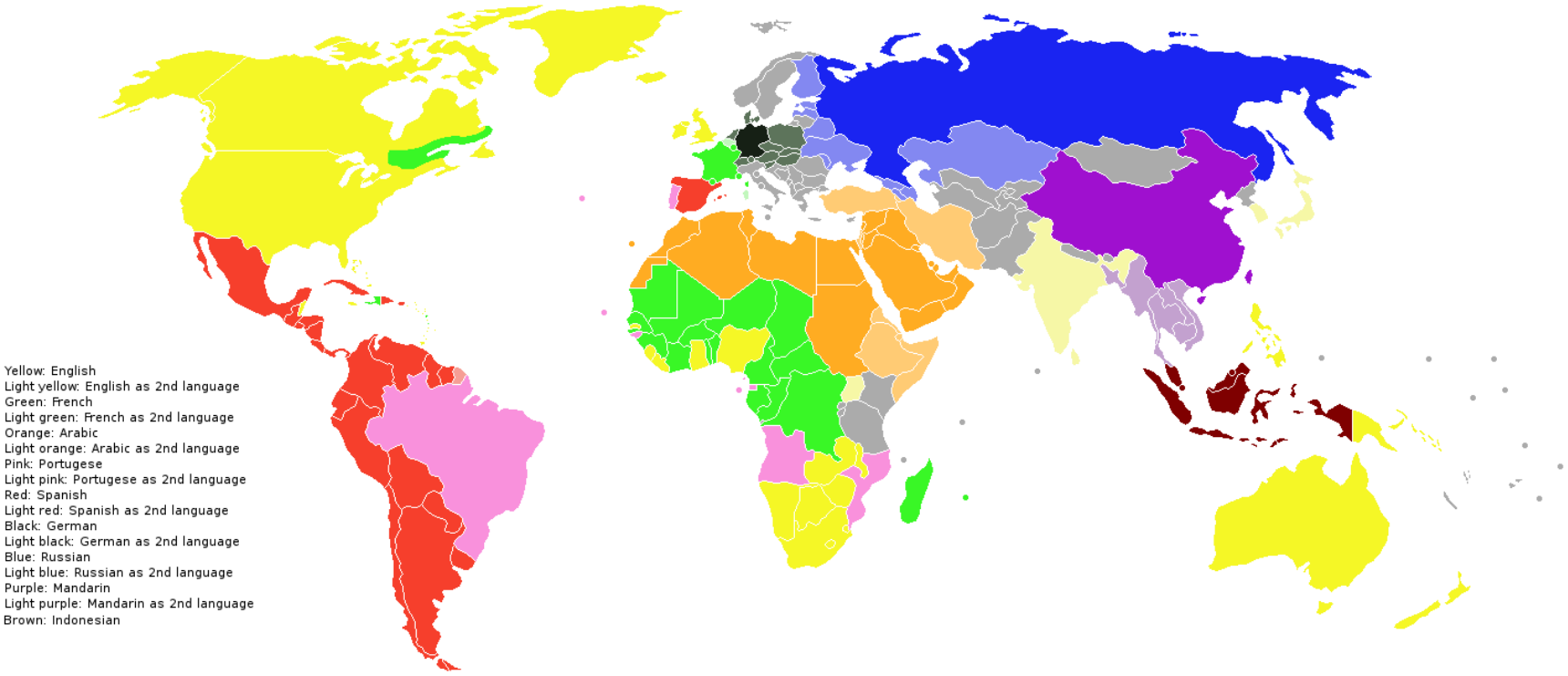
Exploring Programming Clojure in Other Human Languages

Elango Cheran

Clojure/West 2015

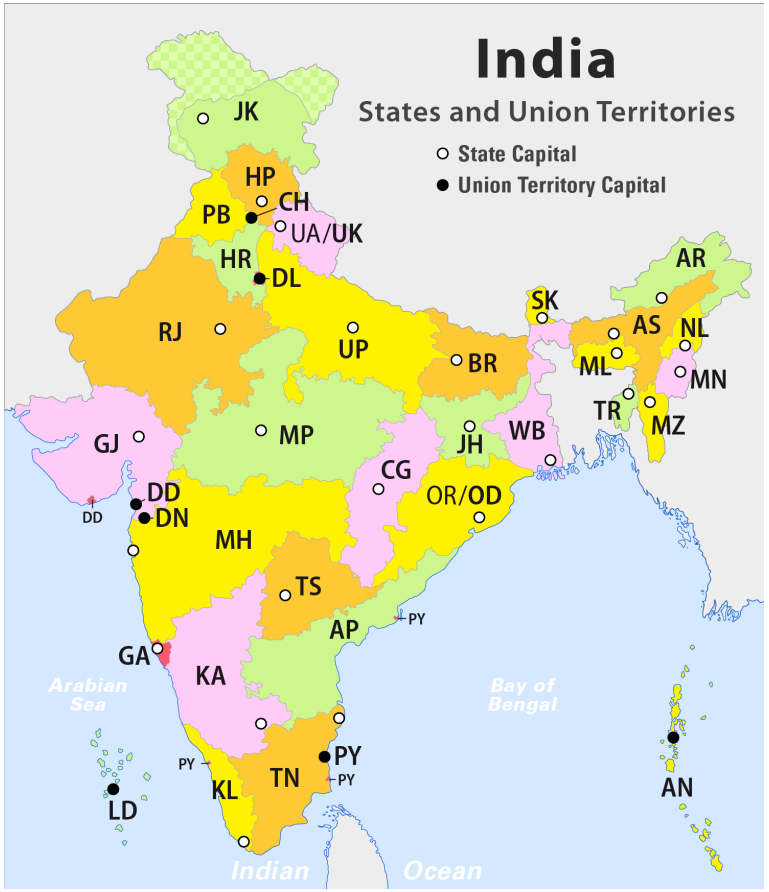
Language and the World Around Us



Language and the World Around Us

1951

2014



Language and Identity

- Large-scale changes
  - Social / religious movements
  - What about academic papers, laws, etc.?

Language and Identity

- Large-scale changes
  - Social / religious movements
  - What about academic papers, laws, etc.?
- Boundaries
  - Conflicts inside nation-states
  - Connected diasporas across borders

No Good Analogies Here...

Term	Real Life	Clojure
Language	‘Protocol’ for communication	‘Protocol’ for computer instructions
Identity	A personal label applied by self and/or others	Meaning we give to a binding whose value changes over time
State	Apolitical entity with defined borders; or a personal status at a timepoint	A binding’s value at a particular timepoint

- Language influences identity in real-life, but in programming?

Original Problem

Human Language	Unicode
Word is a seq of letters	String is a seq of codepoints (characters)
English ● alphabetic ● 26 letters	● Letter ↔ char ● Lexicographic order ↔ Codepoint numerical order
Thamil ● syllabary ● 12 vowels, 18 consonants, 12x18 CV letters	● Codepoints represent graphemes (writing), not phonemes (grammar) ● Letters ⟶ 1 or 2 codepoints ⟶ strings

Problem Example

Letter	Phonemes (Grammar)		Unicode	
க் k	க்		க 0B95	◌் 0BCD
	k		ka	?
க ka	க்	அ	க 0B95	
	k	a	ka	
கு ku	க்	உ	க 0B95	◌ு 0BC1
	k	u	ka	?

Clojure and Unicode

```
user> (def π 3.14159)
#'user/π
user> (/ π 4)
0.7853975
```

Clojure and Unicode

```
user> (def எடு take)
#'user/எடு
user> (எடு 2 [5 7 11 13 17])
(5 7)

user> (def விடு drop)
#'user/விடு
user> (விடு 2 [5 7 11 13 17])
(11 13 17)
```

When You Can’t Resolve To Values

```
user> (def எனில் if)
CompilerException java.lang.
RuntimeException: Unable to resolve
symbol: if in this context
```

When You Can't Resolve To Values

```
user> (defmacro எனில்
        [& body]
        `(if ~@body))
#'user/எனில்

user> (எனில் (pos? π)
        "π is positive"
        "π is negative or 0")
"π is positive"
```

```
(def எடு take)
(def விடு drop)
...
```

```
(def எடு take)
(def விடு drop)
...
```

```
(defmacro எனில்
  [& body]
  `(if ~@body))
(defmacro மற்றும்
  [& body]
  `(and ~@body))
...
```

Refactored defs

```
(defmacro translate-fn
  [old-name new-name]
  `(def ~new-name ~old-name))

(translate-fn take எடு)
(translate-fn drop விடு)
...
```

Refactored defmacros

```
(defmacro translate-form
  [old-name new-name]
  `(defmacro ~new-name
     [~'& body#]
     `(~'~old-name ~@body#)))

(translate-form if எனில்)
(translate-form and மற்றும்)
...
```

The Underlying Data

```
{'take 'எடு
 'drop 'விடு
  ... }
```

```
{'if 'எனில்
 'and 'மற்றும்
  ... }
```

Second Pass - Refactored defs

```
(defmacro translate-fn-symbol
  [old-name new-name]
  `(def ~(eval new-name) ~(eval old-name)))

(translate-fn-symbol 'take 'எடு)
(translate-fn-symbol 'drop 'விடு)
```

Second Pass - Refactored defmacros

```
(defmacro translate-form-symbol
  [old-name new-name]
  `(defmacro ~(eval new-name)
     [~'& body#]
     `(~'~(eval old-name) ~@body#)))

(translate-form-symbol 'if 'எனில்)
(translate-form-symbol 'and 'மற்றும்)
```

Second Pass - Main macros

```
(defmacro translate-fns
  [symb-map]
  `(do
     ~@
     (for [[old-form# new-form#] (eval symb-map)]
       `(translate-fn-symbol '~old-form# '~new-form#))))

(defmacro translate-forms
  [symb-map]
  `(do
     ~@
     (for [[old-form# new-form#] (eval symb-map)]
        `(translate-form-symbol '~old-form#
                                '~new-form#))))
```

Second Pass - Run

```
;; "translation" for fns, values, ...
(translate-fns fns-map)

;; "translation" for macros,
;; special forms
(translate-forms forms-map)
```

Result - Example

```
(வரையறு-செயல்கூறு தன்னால்-பெருக்கு
  [அ]
  (* அ அ))

(வரையறு எண்கள் [108 1008 18 64 6 12 247])

(வரையறு-செயல்கூறு சுற்று-01
  []
  (வரி-அச்சிடு "எண்களையும் அவற்றின் சதுர ஆக்கங்களையும்(?)
அச்சிடுவது:")
  (செய்வரிசை [எண் எண்கள்]
              (வரி-அச்சிடு (தொடை "[" எண் "] -> ["
                                   (தன்னால்-பெருக்கு எண்) "]"))))

clj-thamil.demo.trans-demo-01> (சுற்று-01)
எணகளையும் அவற்றின் சதுர ஆக்கங்களையும் அச்சிடுவது:
[108] -> [11664]
[1008] -> [1016064]
...
```

Quick Demo

Looking Back

- Translation macros entirely “user-level” code

Looking Back

- Translation macros entirely “user-level” code
- Always has been possible in Lisp

Looking Back

- Translation macros entirely “user-level” code
- Always has been possible in Lisp
- Clojure & ClojureScript & target runtimes support Unicode
  - Using cljx to compile to both (waiting for Clojure 1.7)

Looking Back

- Translation macros entirely “user-level” code
- Always has been possible in Lisp
- Clojure & ClojureScript & target runtimes support Unicode
  - Using cljx to compile to both (waiting for Clojure 1.7)
- Need full stack support
  - Unicode, OS, encodings, fonts, apps
    - Thank you Aquamacs!
    - Need: fixed-width fonts, new Thamil words

Looking Forward

- Softer learning curve for programming for non-English speakers
  - How easy was learning programming?
  - How easy was learning a foreign language?
  - Composing both operations?

Looking Forward

- Softer learning curve for programming for non-English speakers
  - How easy was learning programming?
  - How easy was learning a foreign language?
  - Composing both operations?
- Make child coding literacy feasible globally

Looking Forward

- Softer learning curve for programming for non-English speakers
  - How easy was learning programming?
  - How easy was learning a foreign language?
  - Composing both operations?
- Make child coding literacy feasible globally
- More programmers
  - More Clojure programmers

Looking Forward

- Softer learning curve for programming
  - Lisp + turtle - () = ?

Looking Forward

- Softer learning curve for programming
  - Lisp + turtle - () = ?
    - Logo > BASIC

```
to square
repeat 4 [forward 50 right 90]
end
```

```
10 LET X = 3
20 PRINT X
30 GOTO FAIL
```

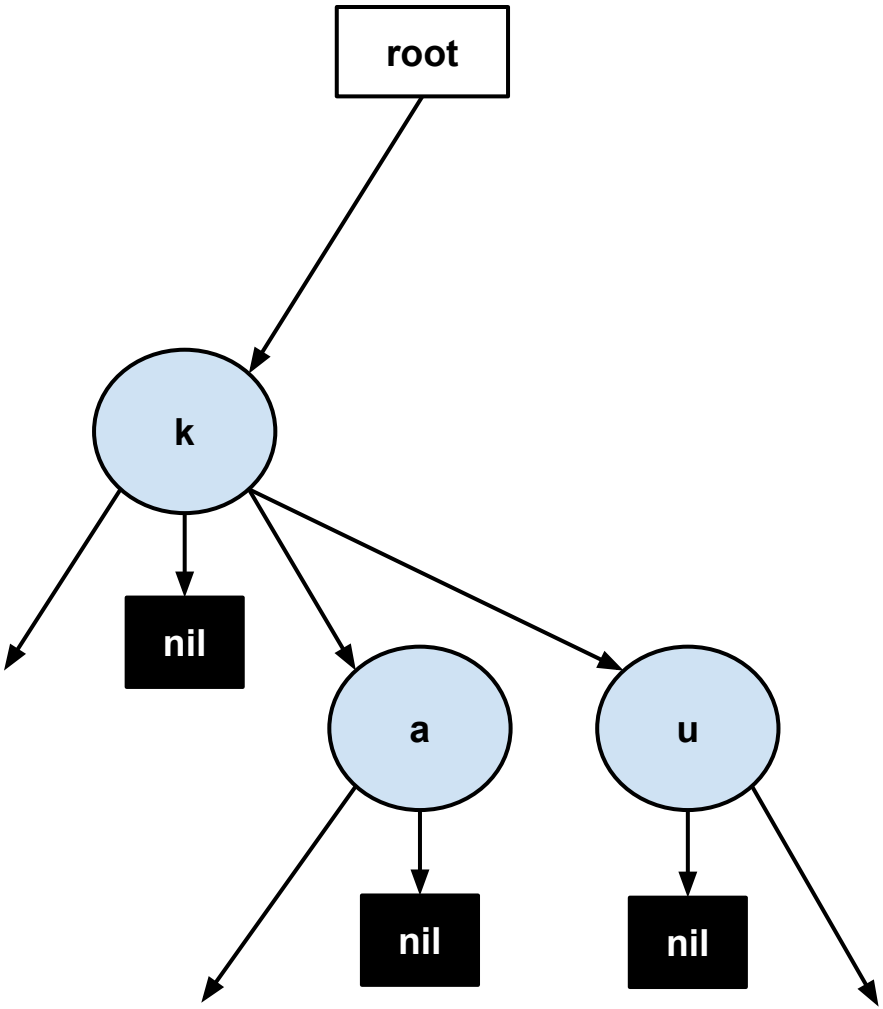
Current Limitations

- Bootstrapping `require`ing the ns with new form defs
- Handling translated literals
  - Numerals
  - true / false
  - :as, :else, :only, ...
- Stacktraces -> Java / JS -> English
- JS compiler optimizations break on 1+ char letters?

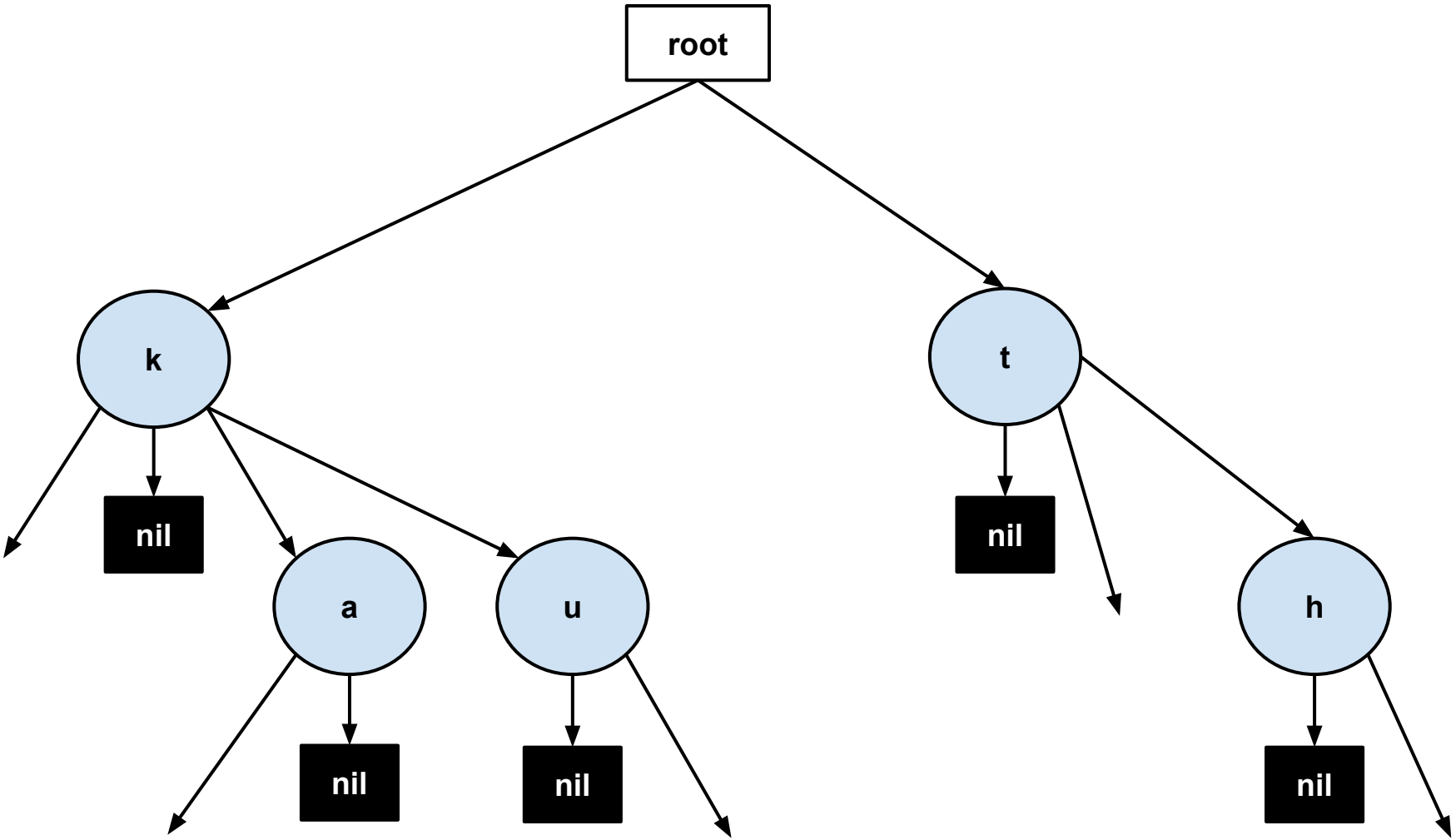
Solving the Alphabet Parse Problem

- Letters' encoding in Unicode not based on alphabet
  - Letter encoding 1+ chars -> use a string (seq)
  - Encodings of some letters are prefixes of others
- Maintain all letters (seqs) in prefix tree!
  - A.k.a. trie

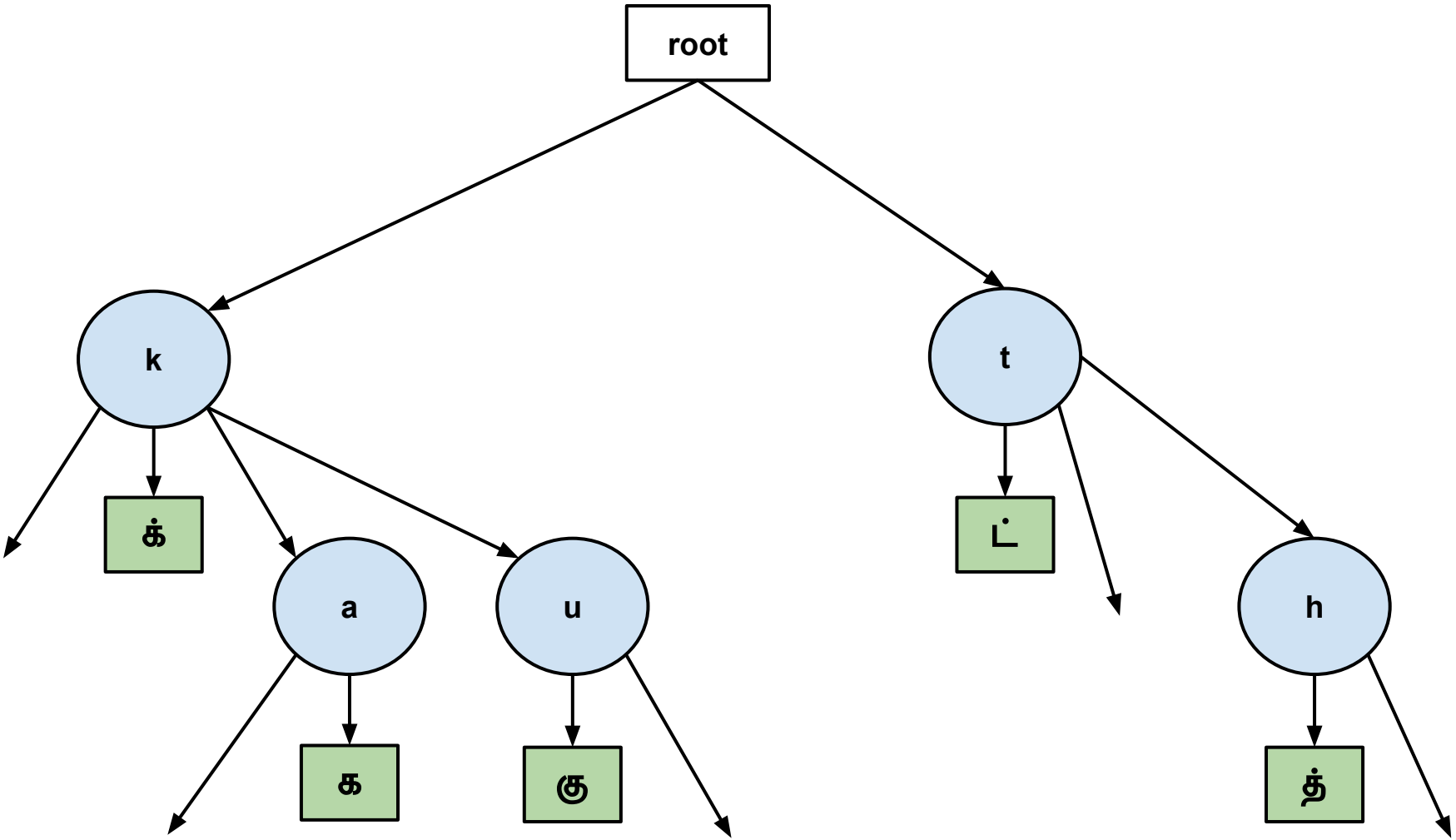
Trie Example



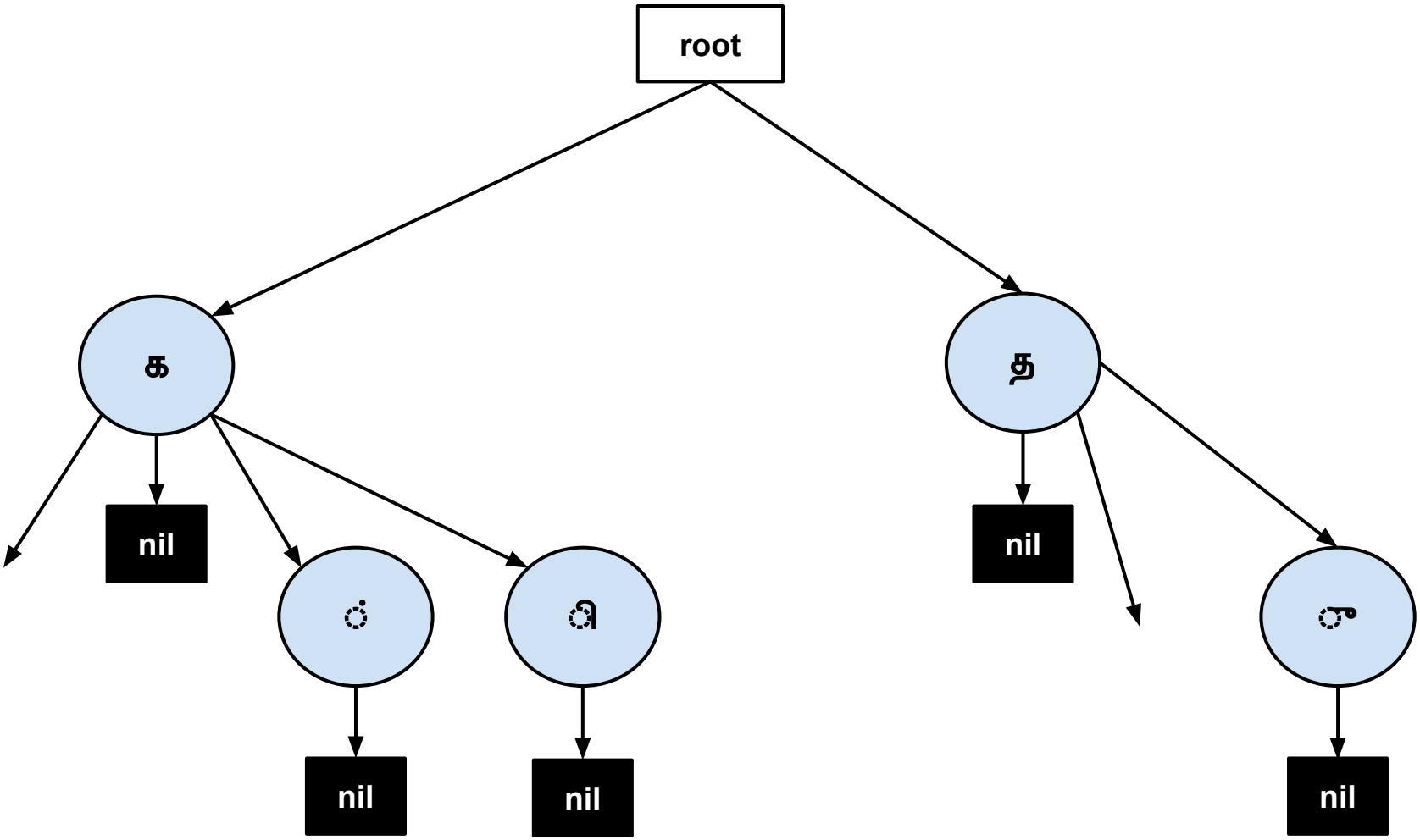
Trie Example



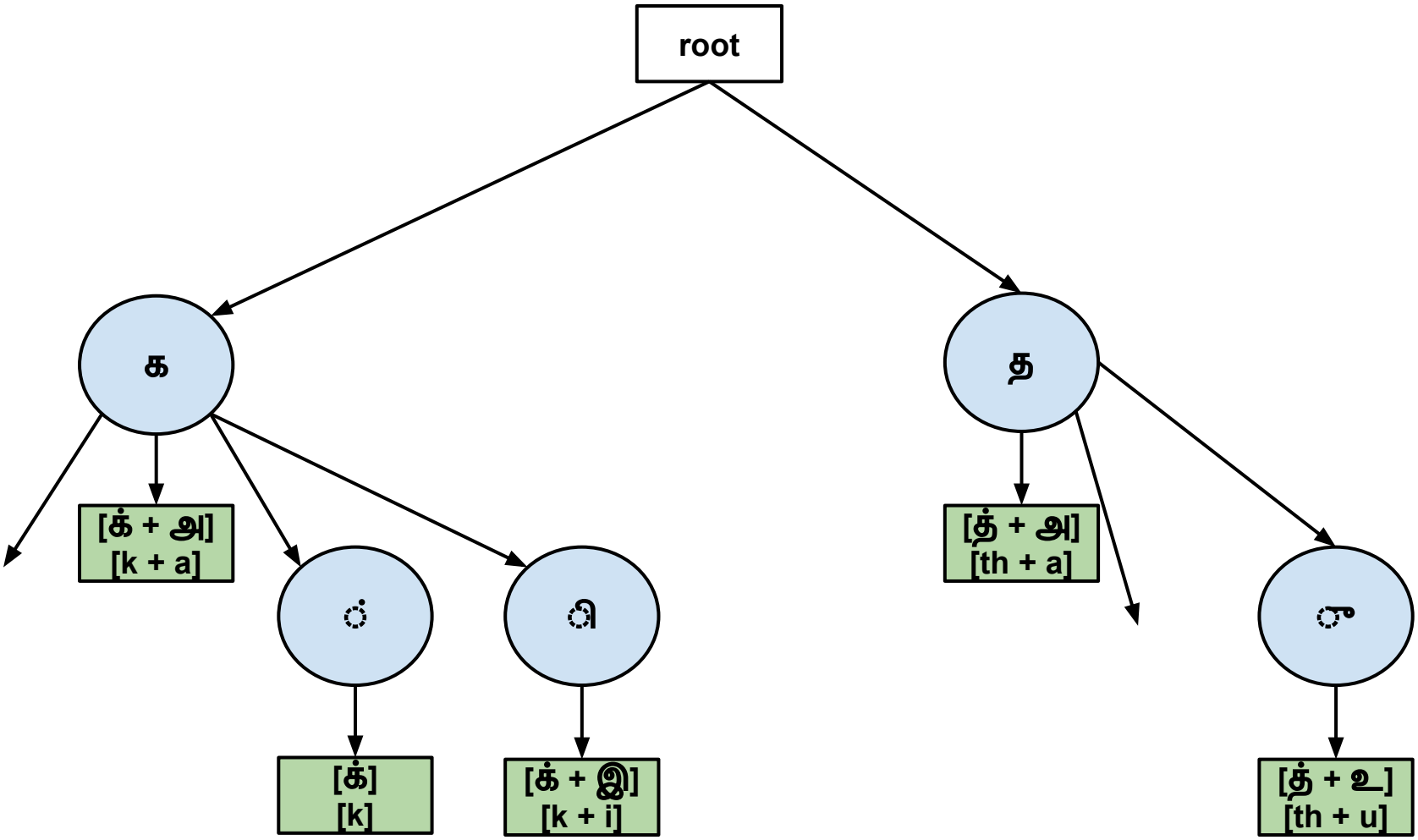
Trie Example - Associating Values



In Practice, Parsing Letters in Thamil



In Practice, Parsing and Converting



Challenges for non-Western Langs.

- Each language has its own specific set of functionality needed to handle Unicode spec
  - 1 letter -> 1+ chars
  - Right-to-left
  - Logographs
  - etc.
- Some languages require more effort than others

Bridging Across Languages

- Would multiple languages fragment the userbase?

Bridging Across Languages

- Would multiple languages fragment the userbase?
- Much to be gained by learning from each other
  - Thamil ↔ Japanese (& Korean)

Yayoi Period

Sangam Period

Bridging Across Languages

- Would multiple languages fragment the userbase?
- Much to be gained by learning from each other
  - Thamil ↔ Japanese (& Korean)
  - Thamil ↔ Thai

ผัด ไทย - p^{h}ad thai
ผ ด ฑ ย
p d th y

பட் தைய் - “pat thaiy”
ப் ட் த் ய்
p t th y

Yayoi Period

Sangam Period

Bridging Across Languages

- Would multiple languages fragment the userbase?
- Much to be gained by learning from each other
  - Thamil ↔ Japanese (& Korean)
  - Thamil ↔ Thai
- Already: Clojure ↔ CLisp, Haskell, Go, etc.

ผัด ไทย - p^had thai
ผ ด ฑ ย
p^h d th y

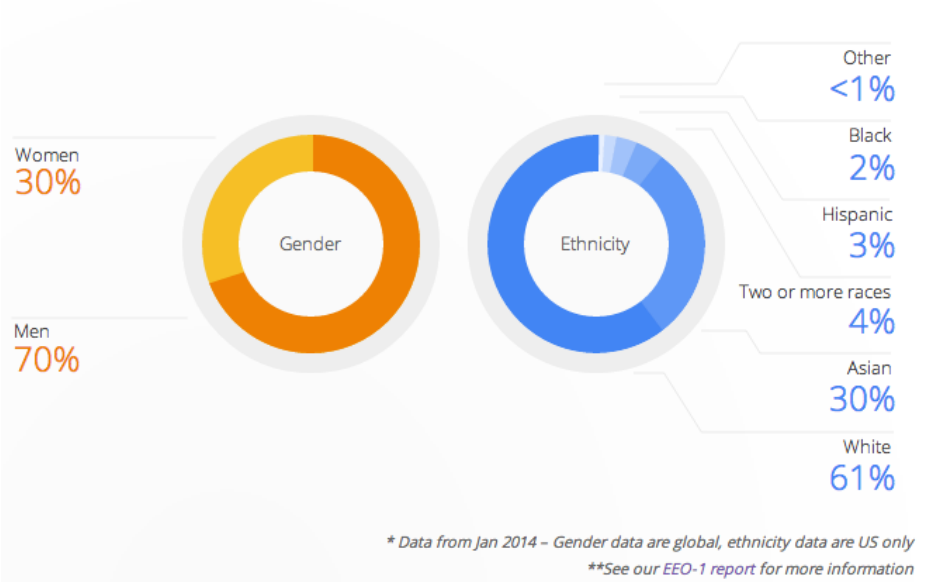
பட் தைய் - “pat thaiy”
ப் ட் த் ய்
p t th y

Yayoi Period

Sangam Period

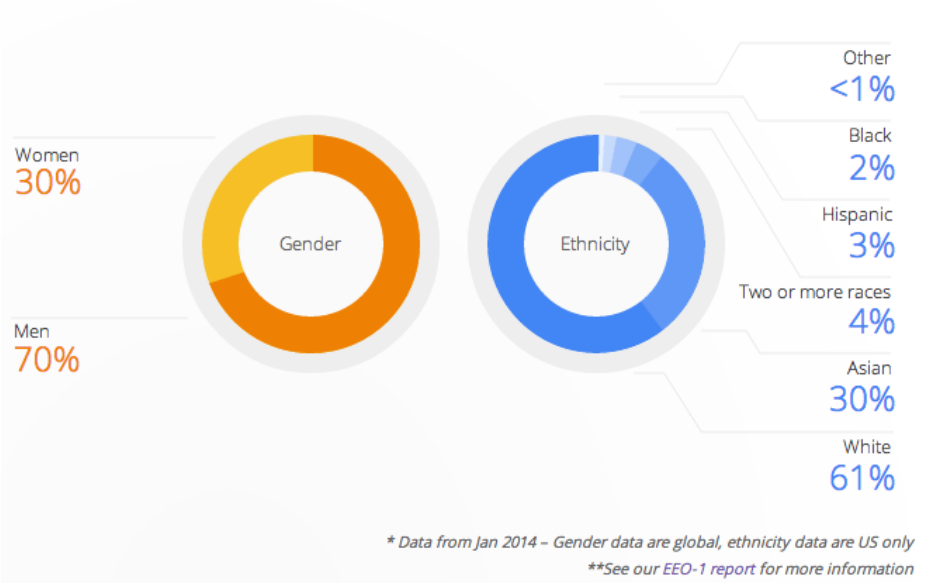
Diversity and Programming

- “Getting to work on diversity at Google” - Google Blog, Mar. 2014



Diversity and Programming

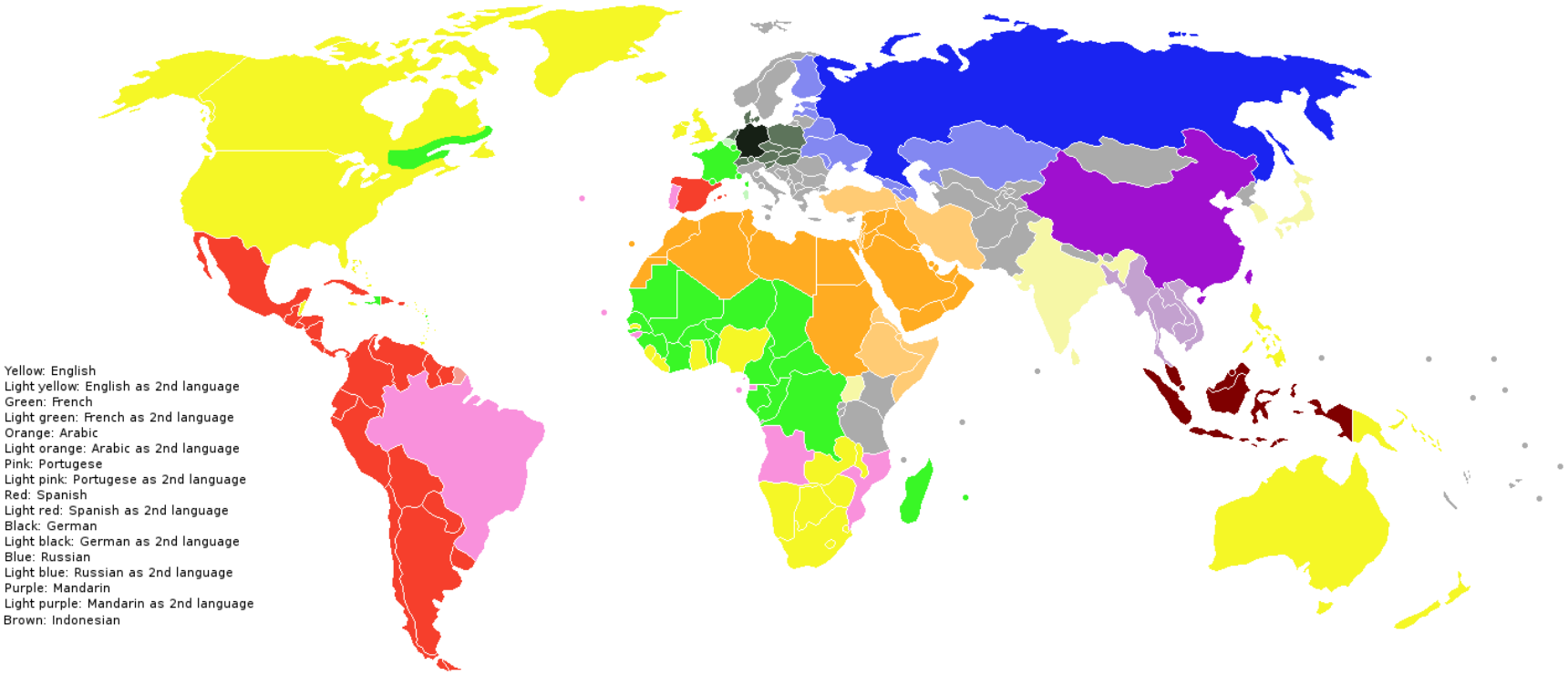
- “Getting to work on diversity at Google” - Google Blog, Mar. 2014
- Grassroots efforts on changing culture
  - RailsBridge, ClojureBridge



RailsBridge

ClojureBridge

Language and Global Diversity



Thanks

- Google
- Aston and Steve Rumble
- For being awesome:
  - Clojure community
  - RailsBridge & ClojureBridge

More Info

clj-thamil project:

https://github.com/echeran/clj-thamil

ClojureBridge: http://www.clojurebridge.org/

RailsBridge: http://www.railsbridge.org/

random Thamil history info:

http://www.ibiblio.org/tamil/history