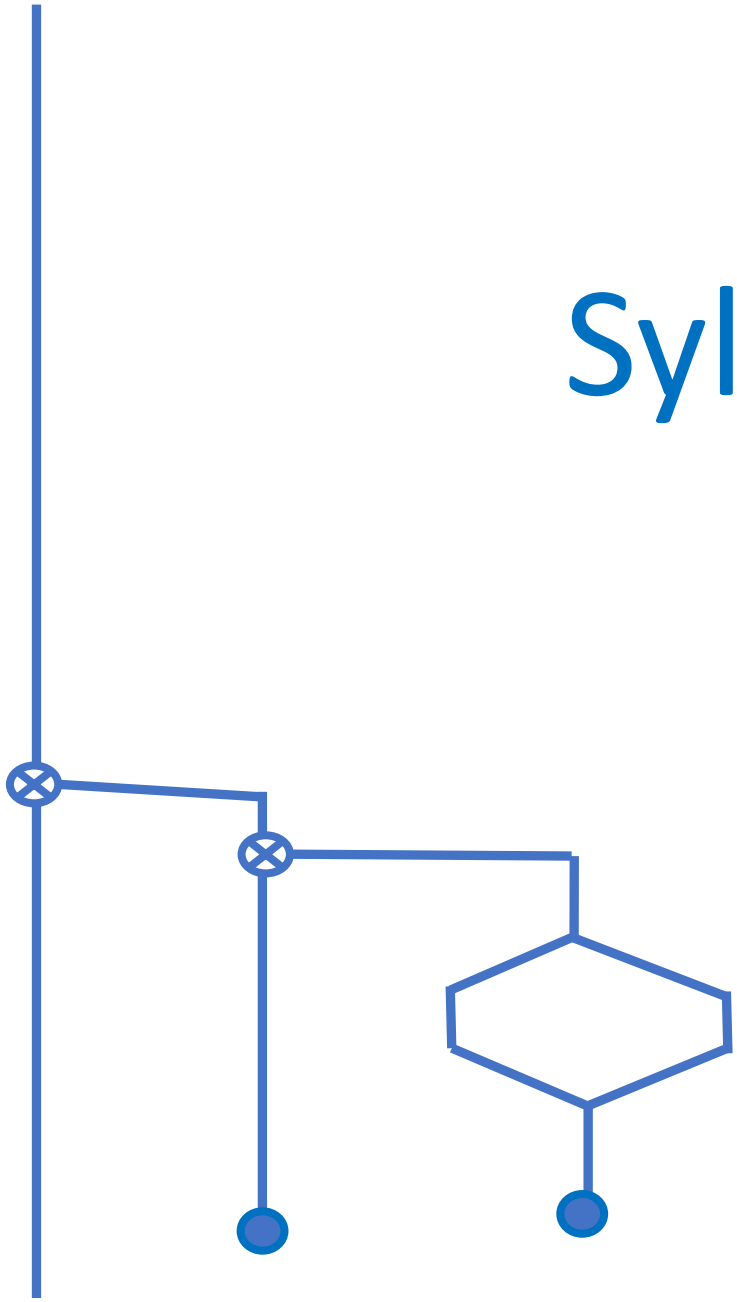
# Syllabification



SVFIG

Oct. 22, 2022

Bill Ragsdale

# The Challenge

Create a list of about 20 words.

Create a set of rules to parse them into syllables.

Consider if your words should be diverse or similar (cat & dog vs. baby & babies).

I have no idea if this is easy or difficult.

# Introduction

My earliest examination shows syllable detection relies on sound more than letter patterns. Thus, we need access to a dictionary with word sounds and an expanded symbol set (i.e. hard and soft vowels, silent letters and much more)

It appears this challenge would be more suitable for a semester project in computer science: data base, parsing, patterns, etc.

This today's project will barely touch the process of syllable detection.

The most I can hope for is to develop the supporting (parsing) code and a rule or two.

# Full Rule Set

**Rule #1 A syllable is formed by at least one vowel (a, e, i, o, u).**
*For example: a, th**e**, pl**a**nt, b**a**-n**a**-n**a**, ch**i**l-dr**e**n, c**a**m-**e**r-**a**.*

**Exceptions:**

**a) Silent e is not counted as a vowel in a syllable.**
*For example: tap**e**, lik**e**, lov**e**, ex-treme, tak**e**, blu**e.***

**b) When two vowels carry one sound (diphthong), they cannot be divided.**
*For example: c**oi**n, l**ou**d, br**ea**d, m**oo**n, s**ou**nd, b**eau**-ti-ful, a-v**oi**d.*

**c) *The letter "y" is not strictly a vowel but behaves like one.***
*For example: man-**y**, bi-c**y**-cle, i-v**y**.*

**Rule #2 Divide the syllable between two same consonants.**
*For example: ra**b-b**it, le**t-t**er, bu**f-f**et, de**s-s**ert, be**r-r**y.*

# Full Rule Set II

**Rule #3 Vowel with long/short vowel sound . . .**

a) The consonant goes with the second vowel if the **first vowel has a long vowel sound**.

*For example: ba-**sic**, ro-**bot**, wa-**ter.***

b) The consonant goes with the first vowel if **it has a short vowel sound**.

*For example: **riv**-er, **mod**-el, **pan**-el.*

**Exception:**
Never split **two consonants that make only one sound** (ch, sh, ph, th, wh & gh) when pronounced together and aren't the same letter (diagraphs).
*For example: tea**ch**-er, la**sh**-es, gra**ph**-ic, pan-**th**er, ba**th**-tub.*

**Rule #4 Divide between two vowels that make two sounds.**

*Fo example: di-et, di-aer-e-sis.*

**Exception:**
**Don't split two vowels that make one sound.**
*For example: c**oa**t, b**oa**t, m**ee**t.*

# Full Rule Set III

**Rule #5 Use prefixes and suffixes to separate syllables.**
*For example: **re**-turn, **un**-u-su-al, **pre**-paid, end-**less**, pay-**ing,** hap-pi-**ness, un**-kind-**ly.***

**Rule #6 Compound nouns are always divided between the two words.**
*For example: **some**-thing, **cup**-cake, **with**-out, **how**-ev-er, **ba-by**-sit-ter, **class**-room, **break**-fast, **sun**-flow-er.*

**Rule #7a**
**Divide before the consonant before an "-le" syllable and sounds like “-el”.**
*For example: a-b**le**, can-d**le**, fum-b**le**, ap-p**le**, ta-b**le**, cas-t**le**.*

**Rule #7b Exception: Words which end with “ckle”.**
*For example: ti**ck-le**, ta**ck-le**.*

# Questions

Question: How to track rule usage?
Discovery : Use a rule number as a syllable separator? Unruly becomes: un3ru5ly

Question: To analyze letter by letter or by letter groups?
Discovery:  Use letter by letter. Gives a simpler, linear logic.

Question: Should we catalog letter clusters (diphthongs, coin & loud; there, thought) and give them unique symbols?
Discovery:  No answer yet.

Question: scan left to right or right  to left?
Discovery: Develop  support words for both.

Question: What decision structure to use: If/else rules, table driven, a parse tree, or ??”
Discovery: An easy start is if/else rules.  I suspect a parse tree will be more general.

# Word List

the, plant, banana, children, camera, tape, like, love, extreme, take, blue, coin, loud, bread, moon, sound, beautiful, avoid, many, bicycle, ivy, basic, robot, water, river, model, panel, teacher, lashes, graphic, panther, bathtub, diet, diaeresis, coat, boat, meet, return, unusual, prepaid, end-less, paying, happiness, unkindly, something, cupcake, without, however, babysitter, classroom, breakfast, sunflower, **able, candle, fumble, apple, table, castle, tickle, tackle,** sabre, saber, savor, savior.

# Target Result

the, plant, ba-na-na, chil-dren, cam-er-a.
tape, like, love, ex-treme, take, blue.
coin, loud, bread, moon, sound, beau-ti-ful, a-void.
man-y, bi-cy-cle, i-vy.
ba-sic, ro-bot, wa-ter.
ri-v-er, mod-el, pan-el.
teach-er, lash-es, graph-ic, pan-ther, bath-tub.
di-et, di-aer-e-sis.
coat, boat, meet.
re-turn, un-u-su-al, pre-paid, end-less, pay-ing, hap-pi-ness, un-kind-ly.
some-thing, cup-cake, with-out, how-ev-er, ba-by-sit-ter, class-room, break-fast, sun-flow-er.

a-ble, can-dle, fum-ble, ap-ple, ta-ble, cas-tle. [Rule 7a]
tick-le, tack-le. [Rule 7b]

Sab-re, sab-er, sa-vor, sav-ior

# Conversion Algorithm

Key elements

1. Develop a list of target words.
2. From the list, extract the next word to scratch workspace.
3. Develop the rule structure, incrementally for 7 rules.
4. Form a rule for the simplest case.
5. Apply the rule.
6. Repeat forming the next rule until exhausted.
7. Add the parsed word to an output report.
8. Repeat for the next word at 2.

# Applying Rule #7

**Rule #7a  Divide before the consonant before an "-le" syllable and sounds like “-el”.**
*For example: a-b**le**, can-d**le**, fum-b**le**, ap-p**le**, ta-b**le**, cas-t**le**.*

**Rule #7b exception: Words which end with “ckle”.**
*For example: ti**ck-le**, ta**ck-le***

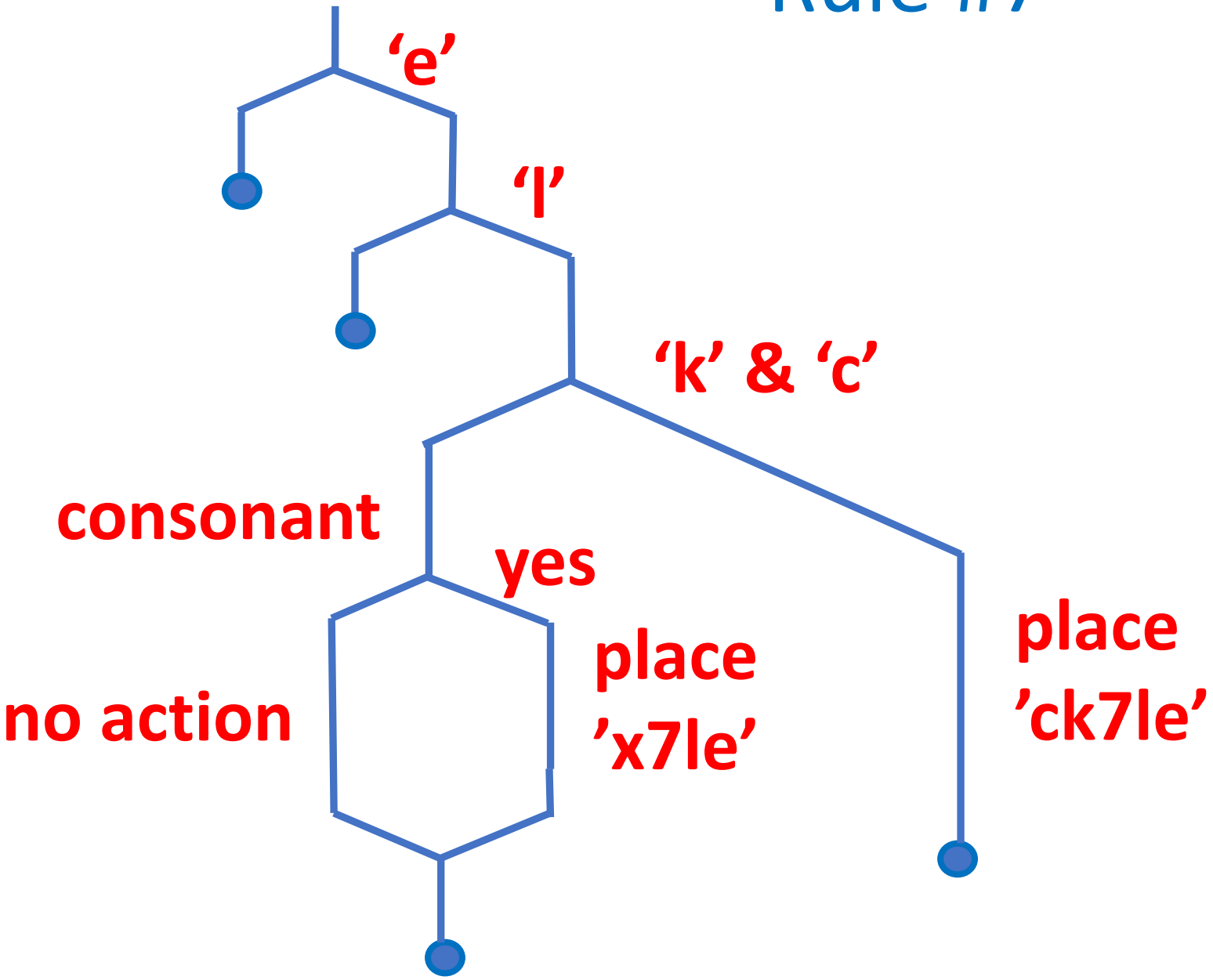
**If letter~0 is not ‘e’ exit**

**If letter~1  is not  ‘l’  exit**

**If letter~ 2  is ‘k’ and letter ~3 is ‘c’  place 7 as divider ‘ck7le’, exit**

**If  letter~2 is a consonant place 7 as a divider ‘7xle’.**

# Rule #7



# Rule 7

```
: Rule7  ( --- )
  fetch~0 ascii e <> if  exit then
  fetch~1 ascii l <> if  exit then
  fetch~2 ascii k = fetch~3 ascii c = and
      if 2 Insert-Rule-7 exit then
  fetch~2 consonant?
      if 3 Insert-rule-7 then ;

Looks for   tickle  to tick7le
  and       apple   to ap7ple
```

# Fetching Ending Letters

```
: fetch~    ( n  --- )
   create ,
   does>  @ target-size swap -
          1- target-buffer + c@  ;

\ Words to fetch characters from the end of the
target buffer
0 fetch~ fetch~0   1 fetch~ fetch~1
2 fetch~ fetch~2   3 fetch~ fetch~3
```

# Reporting Rule 7

```
: Insert-rule-7 ( n --- )
    target-buffer target-size +
    over -  dup  dup 1+
    3 roll   cmove>
    ascii 7 swap c!  \ punch in '7'
    1 +to target-size   ;
```

Spread contents of the target buffer and insert the marker ‘7’.

# Applying the Rules

```
: Run-Tests ( --- )
    restart-words  cr cr
    my-file-words 0
    do  get-word
        Rule7 type-target-buffer cr  loop ;
```

More rules would be added after ‘Rule7’

# Getting A Word

```
: get-word  ( --- )
   demark-word  copy-demarked  eat-demarked ;
```

Copy the next word from a file buffer.

# Workspace

```
( Workspace for files )
0 value My-File-Name     \ holds path and name
0 value My-File-ID       \ file handle
0 value My-File-Size     \ byte count
0 value My-File-Location \ pointer to ram buffer
0 value My-file-Offset   \ current processing point
0 value My-file-Words    \ count of file words

( Workspace for target word )
  create target-buffer  20 allot
0 value  target-size      \ byte count in target word
```

# Output Example

```
babysitter
classroom
breakfast
sunflower
    a7ble
 can7dle
 fum7ble
  ap7ple
  ta7ble
 cas7tle
tick7le
tack7le
sabre  saber  savor  savior  ok
```

# Summary

I am used to parsing computer text.

This full project was much more than I expected.

Developing the file access and support took about four hours.

Rule 7 development took about 5 minutes.

A full project would require dictionary access and an expanded notation or symbol set.

# Questions?