1. **Design sketch:**
   The system includes 6 FST automata, each handling both "if" and "only if" for four standard spelling change rules, one special spelling change rule for an irregular (hopefully applicable to others), and the rule for pluralization. The system also includes a limited lexicon automaton, with separate paths for nouns, the three classes of verbs, and irregulars. Each class of verb endings is broken into two subclasses, "class A" (infinitive and present non-third-person plural) and "class B" (remaining present forms and all present subjunctive forms), to facilitate handling certain common irregular patterns. Three non-surface characters are used in the lexicon, C (c that changes to zc before back vowels), C' (c that changes to g before back vowels), and G (g that changes to j before back vowels), each with a dedicated FSA to handle the spelling change. The remaining spelling change automata, "u insertion" and "z-c mutation", are assumed to apply globally. No additional character subsets are used, just surface consonants, front vowels, and back vowels.

   The "G-j mutation", "C'-g mutation", and "z-c mutation" automata all work in the same fashion, branching from a start state into one state for each of the possible surface characters and then verifying that the required vowel class follows. The "C-zc mutation" automaton is similar but slightly more complicated, passing through an intermediate state along one of the branches to insert an extra character.

   The pluralization automaton works by tracking the presence of an immediately prior consonant using two states and then branching from the "found_consonant" state, one branch for +s suffixes (inserting an e) and the other branch for all other suffixes. The branches simply verify that they've received the expected input.

   The system should be easy to augment with additional regular vocabulary, since there are very few dependencies and constraints imposed on the regulars. One simply needs to add them to the appropriate tables in the lexical automaton. Adding additional irregulars would be more challenging, unless they happen to fit similar patterns to the ones already incorporated.

2. **Listings:**

   **myspanish.yaml:**
   ```yaml
   boundary: '#'
   lexicon: myspanish.lex
   defaults: "a e i o u a' e' i' o' u' b c d f g h j k l m n~ p q r s t v w x y z +:0 #"
   subsets:
     "CONS": "b c d f g h j k l m n~ p q r s t v w x y z"
     "V": " a a' e e' i i' o o' u u'"
     "FRONT": "e i e' i'"
     "BACK": "u o a u' o' a'"
     "LOW": "e o a a' e' o'"
     "HIGH": "i i' u u'"
     "@": "a e i o u a' e' i' o' u' b c C C' d f g G h j k l m n~ p q r s t v w x y z +:0 #"
   rules:
     u insertion:
       start:
         'g': found_g
         '0': reject
       found_g:
         '0': inserted
         '0': reject
       inserted:
         '0': inserted
   ```
G-j mutation:
start:
  'G:j': found_G_expecting_back
  'G:g': found_G_no_back
  '@': start
found_G_expecting_back:
  '@:BACK': start
  '@:0': found_G_expecting_back
  '@': reject
found_G_no_back:
  '@:BACK': reject
  'G:j': found_G_expecting_back
  'G:g': found_G_no_back
  '@:0': found_G_no_back
  '@': start

C'-g mutation:
start:
  "C':g": found_C'_expecting_back
  "C':c": found_C'_no_back
  '@': start
found_C'_expecting_back:
  '@:BACK': start
  '@:0': found_C'_expecting_back
  '@': reject
found_C'_no_back:
  '@:BACK': reject
  "C':g": found_C'_expecting_back
  "C':c": found_C'_no_back
  '@:0': found_C'_no_back
  '@': start

C-zc mutation:
start:
  'C:z': found_C_insert_c_expecting_back
  'C:c': found_C_no_back
  '0:c': reject
  '@': start
found_C_insert_c_expecting_back:
  '0:c': found_C_expecting_back
  '@': reject
found_C_expecting_back:
  '@:BACK': start
  '@:0': found_C_expecting_back
  '@': reject
found_C_no_back:
  '@:BACK': reject
  '0:0': found_C_no_back
  '0:c': reject
  'C:z': found_C_insert_c_expecting_back
  'C:c': found_C_no_back
  '@': start

z-c mutation:
start:
  'z:c': found_z_expecting_front
  'z': found_z_no_front
  '@': start
found_z_expecting_front:
  '@:FRONT': start
  '@:0': found_z_expecting_front
  '@': reject
found_z_no_front:
  '@:FRONT': reject
  'z:c': found_z_expecting_front
  'z': found_z_no_front
'@:0': found_z_no_front
'@': start

pluralization:
start:
'@:CONS': found_consonant
'+:e': reject
'@': start
found_consonant:
'@:CONS': found_consonant
'+:e': subst_e
'+:@': nonsubst_e_no_s
'@:0': found_consonant
'@': start
rejecting state subst_e:
's': found_consonant # FIXME: might want to check for word boundary
'@': reject          # FIXME? sure hope no other rules need to do +:e
state nonsubst_e_no_s:
's': reject
'+:e': subst_e
'@:CONS': found_consonant
'@:0': nonsubst_e_no_s
'@': start

myspanish.lex:
Begin: Verb Noun
Verb: V_REG_AR  V_REG_ER  V_REG_IR  V_IRREG
AfterVerbAR: VERB_AR_SUFFIX_A  VERB_AR_SUFFIX_B
AfterVerbER: VERB_ER_SUFFIX_A  VERB_ER_SUFFIX_B
AfterVerbIR: VERB_IR_SUFFIX_A  VERB_IR_SUFFIX_B
AfterVerbSuffix: End
Noun: N_ROOT
AfterNoun: PLURAL End

V_REG_AR:
cruz  AfterVerbAR  Verb(cruzar)
atac  AfterVerbAR  Verb(atacar)
lleg  AfterVerbAR  Verb(llegar)
pag  AfterVerbAR  Verb(pagar)
trabajar  AfterVerbAR  Verb(trabajar)

V_REG_ER:
conoC  AfterVerbER  Verb(conocer)
pareC  AfterVerbER  Verb(parecer)
venz  AfterVerbER  Verb(vencer)
ejerz  AfterVerbER  Verb(ejercer)
cO  AfterVerbER  Verb(coger)

V_REG_IR:

V_IRREG:
coz  VERB_ER_SUFFIX_A  Verb(cocer)
cuez  VERB_ER_SUFFIX_B  Verb(cocer)
deC' VERB_IR_SUFFIX_A  Verb(decir)
diC' VERB_IR_SUFFIX_B  Verb(decir)

VERB_AR_SUFFIX_A:
+ar  AfterVerbSuffix  +INFIN
+emos  AfterVerbSuffix  +1pl+PSUBJ

VERB_AR_SUFFIX_B:
+o  AfterVerbSuffix  +1sg+PRES
+as  AfterVerbSuffix  +2sg+PRES
+a  AfterVerbSuffix  +3sg+PRES
+an  AfterVerbSuffix  +3pl+PRES
+e  AfterVerbSuffix  +1sg+PSUBJ
AfterVerbSuffix +2sg+PSUBJ
+e AfterVerbSuffix +3sg+PSUBJ
+amos AfterVerbSuffix +1pl+PRES
+en AfterVerbSuffix +3pl+PSUBJ

VERB_ER_SUFFIX_A:
+er AfterVerbSuffix +INFIN
+emos AfterVerbSuffix +1pl+PRES

VERB_ER_SUFFIX_B:
+o AfterVerbSuffix +1sg+PRES
+es AfterVerbSuffix +2sg+PRES
+e AfterVerbSuffix +3sg+PRES
+en AfterVerbSuffix +3pl+PRES
+a AfterVerbSuffix +1sg+PSUBJ
+as AfterVerbSuffix +2sg+PSUBJ
+am AfterVerbSuffix +3sg+PSUBJ
+amos AfterVerbSuffix +1pl+PRES
+an AfterVerbSuffix +3pl+PSUBJ

VERB_IR_SUFFIX_A:
+ir AfterVerbSuffix +INFIN
+imos AfterVerbSuffix +1pl+PRES

VERB_IR_SUFFIX_B:
+o AfterVerbSuffix +1sg+PRES
+es AfterVerbSuffix +2sg+PRES
+e AfterVerbSuffix +3sg+PRES
+en AfterVerbSuffix +3pl+PRES
+a AfterVerbSuffix +1sg+PSUBJ
+as AfterVerbSuffix +2sg+PSUBJ
+am AfterVerbSuffix +3sg+PSUBJ
+amos AfterVerbSuffix +1pl+PRES
+an AfterVerbSuffix +3pl+PSUBJ

N_ROOT:
la'piz AfterNoun Noun(la'piz)
bota AfterNoun Noun(bota)
ciudad AfterNoun Noun(ciudad)

PLURAL:
+s End +PL

End:
'#' End None

3. >>> test_recognize(load("myspanish.yaml"), "spanish.rec")
[]
[['coG+er', 'Verb(coger)+INFIN']]
[['coG+o', 'Verb(coger)+1sg+PRES']]
[['coG+es', 'Verb(coger)+2sg+PRES']]
[['coG+e', 'Verb(coger)+3sg+PRES']]
[['coG+emos', 'Verb(coger)+1pl+PRES']]
[['coG+en', 'Verb(coger)+3pl+PRES']]
[['coG+a', 'Verb(coger)+1sg+PSUBJ'], ['coG+a', 'Verb(coger)+3sg+PSUBJ']]
[['lleg+ar', 'Verb(llegar)+INFIN']]
[['lleg+o', 'Verb(llegar)+1sg+PRES']]
[['lleg+as', 'Verb(llegar)+2sg+PRES']]
[['lleg+an', 'Verb(llegar)+3pl+PRES']]
[['pag+ar', 'Verb(pagar)+INFIN']]
[['pag+o', 'Verb(pagar)+1sg+PRES']]
[['pag+as', 'Verb(pagar)+2sg+PRES']]
[['pag+en', 'Verb(pagar)+3pl+PRES']]
[['pag+e', 'Verb(pagar)+1sg+PSUBJ'], ['pag+e', 'Verb(pagar)+3sg+PSUBJ']]
[['conoC+er', 'Verb(conocer)+INFIN']]
[['conoC+o', 'Verb(conocer)+1sg+PRES']]
[['conoC+es', 'Verb(conocer)+2sg+PRES']]
[['conoC+e', 'Verb(conocer)+3sg+PRES']]

Page 4