MORPHO-PHONOLOGICAL ANALYZER FOR TURKISH

Turkish is a suffixal language. That is, morphemes are right-attached to the words. The suffixes in Turkish can be divided into two paradigms: a noun paradigm and a verb paradigm.
I will handle the verb paradigm with a Turkish text fragment .
Verbs are the red ones.

" Kendi sağlığımızı ve toplum sağlığını korumak için uyguladığımız izolasyon sürecinde bazı olumsuz ruhsal sonuçlar yaşarız. Çoğu zaman günümüzün kısalığından yakındığımız rutin hayatımızdan uzaklaşıp, artık zamanın geçmemesine kızarız.
Her ne kadar izolasyon sürecinde çok fazla vaktimiz var gibi görünse de bunun tamamından verim almayabiliyoruz.
Kapalı kaldığımız zamanın artmasına ters orantılı olarak verim alabileceğimiz zaman azalır.
Yine de bugünlerin geçeceğini unutmayıp, umutsuzluğa düşmemeliyiz . "

---

yaşarız	:	yaşa - r - ız stem aor ps
kızarız	:	kız - ar - ız stem aor ps
almayabiliyoruz	:	al - ma - y - abil - i - yor - uz stem neg bl abl prog ps
azalır	:	az - al - ır stem NtV aor
düşmemeliyiz	:	düş - me - meli - yiz stem neg nec ps

neg: Negation
aor: Aorist
nec: Necessary
prog: Progressive
abl: Ability

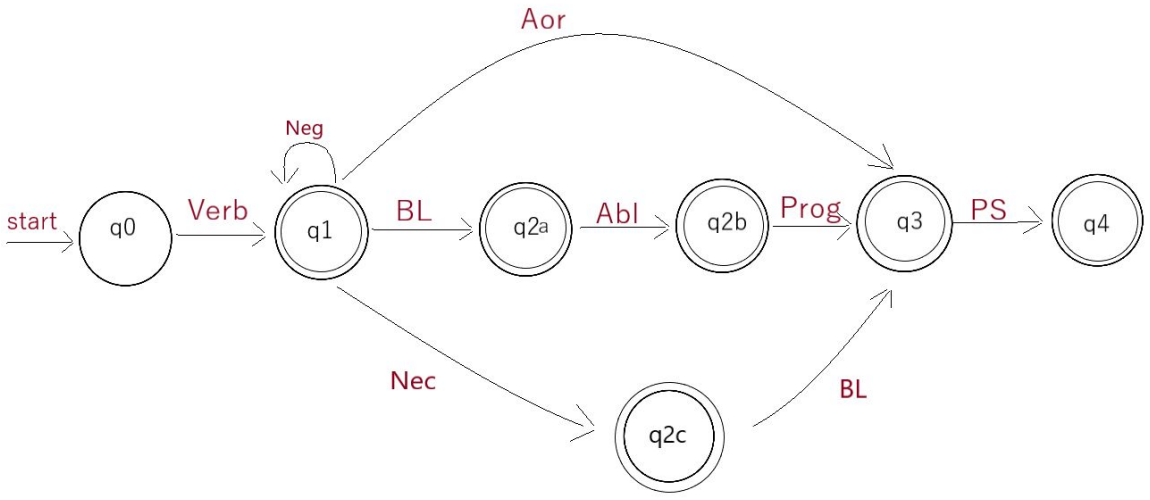
ps: Personal Suffix
bl: Buffer Letter
NtV: Noun to Verb

Buffer Letters (bl)

“The buffer letters" have the function of connecting letters, as they usually connect two vowels. In Turkish, two vowels cannot be next to each other for it is a language that is read as it's written. When a word that ends with a vowel takes a suffix that starts with a vowel, we put the buffer letter in between them. In Turkish we call buffer
letters “kaynaştırma harfi”, the meaning of which is "combining letter."
There are 4 buffer letters in Turkish : -y , -ş ,- s, - n .

Finite-State Automaton

Here is a finite-state automaton that models how the verbs which on the text can be morphologically analyzed.



-Now , I will write a Prolog program that can morpho-phonologically analyze these verbs.

```
initial(q0).
final(q1).
final(q2a).
final(q2b).
final(q2c).
final(q3).
final(q4).


t(q0,verb,q1).
t(q1,neg,q1).
t(q1,buffer,q2a).
t(q1,aor,q3).
t(q1,nec,q2c).
t(q2a,abl,q2b).
t(q2b,prog,q3).
t(q2c,buffer,q3).
t(q3,ps,q4).

allomorph(yaşa,verb).
allomorph(kız,verb).
allomorph(al,verb).
allomorph(azal,verb).
allomorph(düş,verb).

allomorph(y,buffer).
```

```
allomorph(ma,neg).
allomorph(me,neg).

allomorph(r,aor).
allomorph(ar,aor).
allomorph(ır,aor).
allomorph(ir,aor).
allomorph(er,aor).

allomorph(abil,abl).
allomorph(ebil,abl).

allomorph(yor,prog).
allomorph(iyor,prog).


allomorph(malı,nec).
allomorph(meli,nec).

allomorph(ız,ps).
allomorph(uz,ps).
allomorph(iz,ps).
allomorph(yiz,ps).
```

- To make query we will use analyzer ;

```
analyzer(String,List_of_Morphemes):-
    initial(State),
    analyzer(String,State,List_of_Morphemes).

analyzer('',State,[]):- final(State).

analyzer(String,CurrentState,[Morpheme|Morphemes]):-
    concat(Prefix,Suffix,String),
    allomorph(Prefix,Morpheme),
    t(CurrentState,Morpheme,NextState),
    analyzer(Suffix,NextState,Morphemes).
```

-Results of queries ;

```
?- analyzer(ya□arız,X).
X = [verb, aor, ps] .

?- analyzer(kızarız,X).
X = [verb, aor, ps] .

?- analyzer(almayabiliyoruz,X).
X = [verb, neg, buffer, abl, prog, ps] .

?- analyzer(azalır,X).
X = [verb, aor] .

?- analyzer(dü□memeliyiz,X).
X = [verb, neg, nec, buffer, ps]
```

-Here is the Prolog code of the program ı used.

```
initial(q0).

final(q1).
final(q2a).
final(q2b).
final(q2c).
final(q3).
final(q4).

t(q0,verb,q1).
t(q1,neg,q1).
t(q1,buffer,q2a).
t(q1,aor,q3).
t(q1,nec,q2c).
t(q2a,abl,q2b).
t(q2b,prog,q3).
t(q2c,buffer,q3).
t(q3,ps,q4).



allomorph(yaşa,verb).
allomorph(kız,verb).
allomorph(al,verb).
allomorph(azal,verb).
allomorph(düş,verb).

allomorph(y,buffer).

allomorph(ma,neg).
allomorph(me,neg).

allomorph(r,aor).
allomorph(ar,aor).
allomorph(ır,aor).
allomorph(ir,aor).
allomorph(er,aor).

allomorph(abil,abl).
allomorph(ebil,abl).

allomorph(yor,prog).
allomorph(iyor,prog).
```

```
allomorph(malı,nec).
allomorph(meli,nec).

allomorph(ız,ps).
allomorph(uz,ps).
allomorph(iz,ps).
allomorph(yiz,ps).


analyzer(String,List_of_Morphemes):-
   initial(State),
   analyzer(String,State,List_of_Morphemes).

analyzer('',State,[]):- final(State).

analyzer(String,CurrentState,[Morpheme|Morphemes]):-
   concat(Prefix,Suffix,String),
   allomorph(Prefix,Morpheme),
   t(CurrentState,Morpheme,NextState),
   analyzer(Suffix,NextState,Morphemes).
```

Melike DEMİRDAĞ