Ianna Alvita Lewis

Final Project

LIN 177

Proposed topic - Observing and creating a model for the syllable structure seen in French using SWI-Prolog.

1. A detailed write-up which clearly explains the various aspects of the project.

(a) What is the linguistic phenomenon to be modeled? Be clear in this and include examples.

The linguistic phenomenon being modeled in this project is the syllable structure seen in French. I have tried to observe and create a model that generates the most common syllables produced in a dialect of the French language (Parisian French), and I have examined variants with respect to syllabic structure.

In French, each syllable contains only one vowel, and a consonant cannot solely constitute a syllable. Each consonant belongs to the same syllable as the vowel immediately following. Thus, the **CV**, **VC, V**, **CCV**, **CVC** and **CCVC** syllable structures are the most common spoken syllable types in French.

According to the common French syllable structure, the **onset** can be absent, simple (one consonant) or complex (two consonants or more), and the **coda** is always simple (one consonant) if present. The **rhyme** thus consists of at most two positions, one being the **mandatory nucleus**, the other a simple **coda**.

The **onset** can consist of all French consonants, except for the stressed nasals (ŋ,ɲ). It can also consist of clusters of two consonants:

- an obstruent followed by a nasal vowel (for example: tm in [atmosphere])
- an obstruent followed by a liquid (for example: pl in [plaie])
- a nasal followed by a glide (for example: mj in [mien])
- a liquid followed by a glide (for example: lj in [liaison])
- a stop followed by a liquid followed by a glide (for example: tʁw in [trois], tʁɥ in [truie])

The **nucleus** in French consists only of vowels, as French does not allow syllabic consonants. The **coda** may consist of all French consonants.

The variation in syllabic structure in French is due to the optional schwa which may or may not be spoken. For example, in the word “appeler” (to address, to call by name), there are three syllables “a-ppe-ler”, corresponding to either three phonological syllables /a-pə-le/, or two /ap-le/, depending on whether the schwa is maintained or not. In my program, I have assumed that the schwa is always spoken.

Below is a deductive theory that I have written for the French syllable:

POSTULATES
P1. The empty sequence is an onset.

P2. The empty sequence is a coda.

RULES OF INFERENCE
R1. The sequence of an onset and a rhyme is a syllable.

R2. Any consonantic nonnasal phone is an onset (p, b, t, d, k, g, f, v, s, z, ʃ, ʒ, l, ʁ, j, w, ɥ).

R3. Any nonvelar nonpalatal nonstressed nasal phone is an onset (so m and n are both onsets).

R4. Any consonantic nonsonorant phone followed by a sonorant, nonnasal, alveolar, nonpalatal phone is an onset (obstruents followed by the liquid /l/ are onsets).

R5. Any consonantic, nonsonorant phone followed by a sonorant, nonnasal, nonalveolar, nonpalatal, nonlabial phone is an onset (obstruents followed by the liquid /ʁ/ are onsets).

R6. Any consonantic, nasal nonstressed phone followed by a sonorant, labial, palatal, nonnasal phone is an onset (nasals followed by the glide /ɥ/ are onsets).

R7. Any consonantic, nasal nonstressed phone followed by a sonorant, labial, nonnasal, nonpalatal phone is an onset (nasals followed by the glide /w/ are onsets).

R8. Any consonantic, nasal nonstressed phone followed by a sonorant, palatal, nonanterior, nonnasal, nonlabial phone is an onset (nasals followed by the glide /j/ are onsets).

R9. Any sonorant, nonnasal, nonlabial, nonpalatal phone followed by a sonorant, labial, palatal, nonnasal phone is an onset (liquids followed by the glide /ɥ/ are onsets).

R10. Any sonorant, nonnasal, nonlabial, nonpalatal phone followed by a sonorant, labial, nonpalatal, nonnasal phone is an onset (liquids followed by the glide /w/ are onsets).

R11. Any sonorant, nonnasal, nonlabial, nonpalatal phone followed by a sonorant, palatal, nonanterior, nonnasal, nonlabial phone is an onset (liquids followed by the glide /j/ are onsets).

R12. Any nonsonorant, noncontinuous phone followed by a sonorant, nonnasal, nonlabial, nonpalatal, followed by a sonorant, nonnasal, nonalveolar, nonvelar is an onset. (plosives followed by liquids followed by glides are onsets).

R13. The sequence of a nucleus and a coda is a rhyme.

R14. Any nonconsonantic phone is a nucleus. (a, ɑ, e, ɛ, ə, i, œ, ø, o, ɔ, u, y, ɑ̃, ɛ̃, œ̃, ɔ̃ are all nuclei)

R15. Any consonantic nonsonorant phone is a coda (p, b, t, d, k, g, f, v, s, z, ʃ, ʒ are all codas).

R16. Any nasal, nonstressed phone is a coda (m, n, ŋ, ɲ are codas).

R17. Any sonorant, nonnasal, nonlabial, nonpalatal phone is a coda. (l, ʁ are codas).

R18. Any sonorant, nonnasal, nonalveolar, nonvelar phone is a coda. (j, w, ɥ are codas).

I used this deductive theory above and wrote a Prolog program to generate all possible syllables in French (frenchsyllables.pl).

(b) What was the impetus for choosing this particular topic? Why did you choose it?

Being a multilingual speaker, it took me a while to decide which language I wanted to choose and what aspect of the language I wanted to model. I spent six years from middle school through high school learning French and fell in love with the language, not only because it would help fulfill my 13 year old self's dream of getting around France speaking the language fluently, but also because it was very different from the languages I learnt and was exposed to growing up, such as Kannada and Konkani (local Indian languages).

After a few years of learning French, I became a fairly fluent speaker and was able to engage in conversation with other speakers. However, due to linguistic influences from English and the Indian languages that I spoke, it took me a while to articulate syllables and words in French the way native speakers do, but with practice I got better at pronunciations.

This is what influenced me and got me interested to take up the topic of observing syllable structures in French, and I wanted to make an attempt at creating an accurate model of these syllables using the concepts I learnt in this course.

(c) How does the class material relate to this? Was anything similar done in class or discussed in the text?

In lecture we went through rules of syllable structure in English and also created a model for syllables in Senufo as an assignment, however this language was more restricted. The text discusses in detail about syllable structure in English as well. It shows us how a deductive theory, containing postulates and rules of inference, was formulated to write out a syllable program in Prolog.

(d) How do we query your project to get the intended output?

Using the findall function, you can make Prolog generate all possible syllables in French – 'findall(X,syllable(X),Y).'

You can also do the same for each component of the syllable – 'findall(X,onset(X),Y).' , 'findall(X,nucleus(X),Y).' or 'findall(X,coda(X),Y).'

To find the number of syllables generated, I used 'findall(X,syllable(X),Y),length(Y,Z).' The number of phones generated was 36,652.

(e) What were some of the challenges faced in developing your implementation? Does everything work as intended? Were any issues left unsolved?

At first it was challenging trying to figure and write out all the phones produced in French and distinguishing them based on natural class. After I wrote out *frenchphones.pl* and *frenchproperties.pl*, I tried to figure out the placement of different phones in syllables in French, and created a few rules about generalizations that I observed. Having a French dictionary around helped me a lot.

This syllable program was very tedious to write, and it does have its shortcomings. With regards to undergeneration, *frenchsyllable.pl* fails to generate syllables with more than one consonant in its coda, as observed in words like test (CVCC), filtre (CVCCC), etc. when the schwa vowel is not produced (which would otherwise be CVC-CV and CVC-CCV when the extra schwa is produced). Trying to figure out codas with more than one consonant would be very time consuming, which is why I assumed the schwa to be always produced. This would not be allowed in well-formed syllables of French.

(f) Is your project principled relative to what you have learned about linguistic theory and the frameworks we have used in class and/or the framework you have decided to follow in your project? Why or why not?

I do think my project is in sync with the linguistic frameworks we have used in class. This project closely aligns with the English syllable program shown in the textbook, though French has very different phones, with varying properties and syllable structure. The predicate structure in my program follows the syllable structure we see in French. The number of rules of inference match the number of predicates in my program, and linguistically, I believe my program is principled.

2. Program code written in Prolog which includes all necessary components to make your project work.

frenchphone.pl

```
% French Phone definitions
% The numbers correspond to unicode indices
:- ['fulldisplay.pl'].

phone(A):- name(A,[112]). %p
phone(A):- name(A,[098]). %b
phone(A):- name(A,[109]). %m
phone(A):- name(A,[116]). %t
phone(A):- name(A,[100]). %d
phone(A):- name(A,[110]). %n
phone(A):- name(A,[107]). %k
phone(A):- name(A,[103]). %g
phone(A):- name(A,[331]). %ŋ
phone(A):- name(A,[102]). %f
phone(A):- name(A,[118]). %v
phone(A):- name(A,[115]). %s
phone(A):- name(A,[122]). %z
phone(A):- name(A,[643]). %ʃ
phone(A):- name(A,[658]). %ʒ
phone(A):- name(A,[108]). %l
phone(A):- name(A,[626]). %ɲ
phone(A):- name(A,[641]). %ʁ
phone(A):- name(A,[106]). %j
phone(A):- name(A,[119]). %w
phone(A):- name(A,[613]). %ɥ
phone(A):- name(A,[097]). %a
phone(A):- name(A,[593]). %ɑ
phone(A):- name(A,[101]). %e
phone(A):- name(A,[603]). %ɛ
phone(A):- name(A,[601]). %ə
phone(A):- name(A,[105]). %i
phone(A):- name(A,[339]). %œ
phone(A):- name(A,[248]). %ø
phone(A):- name(A,[111]). %o
phone(A):- name(A,[596]). %ɔ
phone(A):- name(A,[117]). %u
phone(A):- name(A,[121]). %y
phone(A):- name(A,[593,771]). %ɑ̃
phone(A):- name(A,[603,771]). %ɛ̃
phone(A):- name(A,[339,771]). %œ̃
phone(A):- name(A,[596,771]). %ɔ̃
phone(A):- name(A,[7869]). %ẽ
```

frenchproperties.pl

```
% Properties of French phones

:- ['frenchphone.pl'].

% Consonants
cns(A):- name(A,[112]). %p
cns(A):- name(A,[098]). %b
cns(A):- name(A,[116]). %t
cns(A):- name(A,[100]). %d
cns(A):- name(A,[110]). %n
cns(A):- name(A,[107]). %k
cns(A):- name(A,[103]). %g
cns(A):- name(A,[331]). %ŋ
cns(A):- name(A,[102]). %f
cns(A):- name(A,[118]). %v
cns(A):- name(A,[115]). %s
cns(A):- name(A,[122]). %z
cns(A):- name(A,[643]). %ʃ
cns(A):- name(A,[658]). %ʒ
cns(A):- name(A,[108]). %l
cns(A):- name(A,[109]). %m
cns(A):- name(A,[626]). %ɲ
cns(A):- name(A,[641]). %ʁ
% Semivowels
cns(A):- name(A,[106]). %j
cns(A):- name(A,[119]). %w
cns(A):- name(A,[613]). %ɥ

% Sonorant
snt(A):- name(A,[109]). %m
snt(A):- name(A,[110]). %n
snt(A):- name(A,[331]). %ŋ
snt(A):- name(A,[626]). %ɲ
snt(A):- name(A,[108]). %l
snt(A):- name(A,[641]). %ʁ
snt(A):- name(A,[106]). %j
snt(A):- name(A,[119]). %w
snt(A):- name(A,[613]). %ɥ

% Nasal
nas(A):- name(A,[109]). %m
nas(A):- name(A,[110]). %n
nas(A):- name(A,[331]). %ŋ
nas(A):- name(A,[626]). %ɲ
nas(A):- name(A,[593,771]). %ɑ̃
nas(A):- name(A,[603,771]). %ɛ̃
nas(A):- name(A,[339,771]). %œ̃
nas(A):- name(A,[596,771]). %ɔ̃

% Voiced
voi(A):- name(A,[112]). %p
voi(A):- name(A,[098]). %b
voi(A):- name(A,[100]). %d
voi(A):- name(A,[110]). %n
voi(A):- name(A,[103]). %g
voi(A):- name(A,[331]). %ŋ
voi(A):- name(A,[118]). %v
voi(A):- name(A,[122]). %z
voi(A):- name(A,[658]). %ʒ
voi(A):- name(A,[108]). %l
voi(A):- name(A,[109]). %m
voi(A):- name(A,[626]). %ɲ
voi(A):- name(A,[641]). %ʁ
voi(A):- name(A,[106]). %j
voi(A):- name(A,[119]). %w
voi(A):- name(A,[613]). %ɥ

% Continuant
cnt(A):- name(A,[102]). %f
cnt(A):- name(A,[118]). %v
cnt(A):- name(A,[115]). %s
cnt(A):- name(A,[122]). %z
cnt(A):- name(A,[643]). %ʃ
cnt(A):- name(A,[658]). %ʒ
cnt(A):- name(A,[641]). %ʁ
cnt(A):- name(A,[108]). %l
cnt(A):- name(A,[106]). %j
cnt(A):- name(A,[119]). %w
cnt(A):- name(A,[613]). %ɥ
cnt(A):- name(A,[097]). %a
cnt(A):- name(A,[593]). %ɑ
cnt(A):- name(A,[101]). %e
cnt(A):- name(A,[603]). %ɛ
cnt(A):- name(A,[601]). %ə
cnt(A):- name(A,[105]). %i
cnt(A):- name(A,[339]). %œ
cnt(A):- name(A,[248]). %ø
cnt(A):- name(A,[111]). %o
cnt(A):- name(A,[596]). %ɔ
cnt(A):- name(A,[117]). %u
cnt(A):- name(A,[121]). %y

% Labial
lab(A):- name(A,[112]). %p
lab(A):- name(A,[098]). %b
lab(A):- name(A,[109]). %m
lab(A):- name(A,[102]). %f
lab(A):- name(A,[118]). %v
lab(A):- name(A,[119]). %w
lab(A):- name(A,[613]). %ɥ

% Alveolar
alv(A):- name(A,[116]). %t
alv(A):- name(A,[100]). %d
alv(A):- name(A,[110]). %n
alv(A):- name(A,[115]). %s
alv(A):- name(A,[122]). %z
alv(A):- name(A,[108]). %l

% Palatal
pal(A):- name(A,[643]). %ʃ
pal(A):- name(A,[658]). %ʒ
pal(A):- name(A,[106]). %j
pal(A):- name(A,[613]). %ɥ
pal(A):- name(A,[626]). %ɲ

% Anterior
ant(A):- name(A,[112]). %p
ant(A):- name(A,[098]). %b
ant(A):- name(A,[109]). %m
ant(A):- name(A,[116]). %t
ant(A):- name(A,[100]). %d
ant(A):- name(A,[110]). %n
ant(A):- name(A,[102]). %f
ant(A):- name(A,[118]). %v
ant(A):- name(A,[115]). %s
ant(A):- name(A,[122]). %z
ant(A):- name(A,[108]). %l

% Velar
vel(A):- name(A,[107]). %k
vel(A):- name(A,[103]). %g
vel(A):- name(A,[331]). %ŋ
vel(A):- name(A,[641]). %ʁ

% Coronal
cor(A):- name(A,[116]). %t
cor(A):- name(A,[100]). %d
cor(A):- name(A,[110]). %n
cor(A):- name(A,[115]). %s
cor(A):- name(A,[122]). %z
cor(A):- name(A,[643]). %ʃ
cor(A):- name(A,[658]). %ʒ
cor(A):- name(A,[108]). %l
cor(A):- name(A,[106]). %j
cor(A):- name(A,[626]). %ɲ

% Sibilant
sib(A):- name(A,[115]). %s
sib(A):- name(A,[122]). %z
sib(A):- name(A,[643]). %ʃ
sib(A):- name(A,[658]). %ʒ

% High
hih(A):- name(A,[105]). %i
hih(A):- name(A,[117]). %u
hih(A):- name(A,[121]). %y

% Mid (high-low)
mid(A):- name(A,[101]). %e
mid(A):- name(A,[248]). %ø
mid(A):- name(A,[111]). %o
mid(A):- name(A,[601]). %ə
mid(A):- name(A,[603]). %ɛ
mid(A):- name(A,[339]). %œ
mid(A):- name(A,[596]). %ɔ

% Low
low(A):- name(A,[097]). %a

% Back
bck(A):- name(A,[117]). %u
bck(A):- name(A,[111]). %o
bck(A):- name(A,[596]). %ɔ

% Central (front-back)
ctr(A):- name(A,[601]). %ə
ctr(A):- name(A,[097]). %a

% Tense
tns(A):- name(A,[105]). %i
tns(A):- name(A,[117]). %u
tns(A):- name(A,[121]). %y
tns(A):- name(A,[101]). %e
tns(A):- name(A,[111]). %o
tns(A):- name(A,[248]). %ø

% Stressed
str(A):- name(A,[097]). %a
str(A):- name(A,[593]). %ɑ
str(A):- name(A,[101]). %e
str(A):- name(A,[603]). %ɛ
str(A):- name(A,[601]). %ə
str(A):- name(A,[105]). %i
str(A):- name(A,[339]). %œ
str(A):- name(A,[248]). %ø
str(A):- name(A,[111]). %o
str(A):- name(A,[596]). %ɔ
str(A):- name(A,[117]). %u
str(A):- name(A,[121]). %y
str(A):- name(A,[593,771]). %ɑ̃
str(A):- name(A,[603,771]). %ɛ̃
str(A):- name(A,[339,771]). %œ̃
str(A):- name(A,[596,771]). %ɔ̃
```

---

frenchsyllable.pl

```
/*
CV, V, CCV, CVC, VC and CCVC syllables are the
most frequent spoken syllable types in French.
*/

% Syllable structure of French

:- ['frenchproperties.pl'].

% This predicate yields valid syllables of French
syllable(A):- %(R1)
    onset(B),
    rhyme(C),
    append(B,C,A).
    /* The concatenation of an onset and a rhyme is a syllable */

% This predicate yields valid onsets of French using the phones and properties
% loaded from frenchproperties.pl
onset([A]):- %(R2)
    phone(A),cns(A),not(nas(A)).

    /* p,b,t,d,k,g,f,v,s,z,ʃ,ʒ,l,ʁ,j,w,ɥ are onsets */

onset([A]):- %(R3)
    phone(A),nas(A),not(vel(A)),not(pal(A)),not(str(A)).

    /* m,n are onsets */

onset([A,B]):- %(R4)
    phone(A), cns(A),not(snt(A)),
    phone(B),snt(B),not(nas(B)),alv(B),not(pal(B)).
    /* obstruent+liquid l are onsets */

onset([A,B]):- %(R5)
    phone(A), cns(A),not(snt(A)),
    phone(B),snt(B),not(nas(B)),not(lab(B)), not(pal(B)), not(alv(B)).

    /* obstruent+liquid ʁ are onsets */

onset([A,B]):- %(R6)
    phone(A), cns(A), nas(A), not(str(A)),
    phone(B),snt(B), lab(B), pal(B), not(nas(B)).

    /* nasal+glide ɥ are onsets */

onset([A,B]):- %(R7)
    phone(A), cns(A), nas(A), not(str(A)),
    phone(B),snt(B), not(nas(B)), lab(B), not(pal(B)).

    /* nasal+glide w are onsets */

onset([A,B]):- %(R8)
    phone(A), cns(A), nas(A), not(str(A)),
    phone(B),snt(B), pal(B), not(ant(B)),not(nas(B)), not(lab(B)).

    /* nasal+glide j are onsets */

onset([A,B]):- %(R9)
    phone(A), snt(A),not(nas(A)),not(lab(A)), not(pal(A)),
    phone(B),snt(B), lab(B), pal(B), not(nas(B)).

    /* liquid+glide ɥ are onsets */

onset([A,B]):- %(R10)
    phone(A), snt(A),not(nas(A)),not(lab(A)), not(pal(A)),
    phone(B),snt(B), not(nas(B)), lab(B), not(pal(B)).

    /* liquid+glide w are onsets */

onset([A,B]):-  %(R11)
    phone(A), snt(A),not(nas(A)),not(lab(A)), not(pal(A)),
    phone(B),snt(B), pal(B), not(ant(B)),not(nas(B)), not(lab(B)).

    /* liquid+glide j are onsets */

onset([A,B,C]):- %(R12)
    phone(A), cns(A),not(snt(A)), not(cnt(A)),
    phone(B), snt(B),not(nas(B)),not(lab(B)), not(pal(B)),
    phone(C), snt(C), not(nas(C)), not(alv(C)), not(vel(C)).
    /* plosive+liquid+glide are onsets */

onset([]).  %(P1)

    /* onsets may be empty */

% This predicate yields valid rhymes of French
rhyme(C):-  %(R13)
    nucleus(A),
    coda(B),
    append(A,B,C).

    /* the concatenation of a nucleus and a coda is a rhyme */

% This predicate yields valid nuclei of French using the phones and properties
% loaded from frenchproperties.pl
nucleus([A]):-  %(R14)
    phone(A),not(cns(A)).

    /* any vowel is a nucleus */

% This predicate yields valid coda of French using the phones and properties
% loaded from frenchproperties.pl
coda([]). %(P2)

    /* codas may be empty */

coda([A]):- %(R15)
    phone(A),cns(A),not(snt(A)).

    /* p,b,t,d,k,g,f,v,s,z,ʃ,ʒ are codas */

coda([A]):- %(R16)
    phone(A),nas(A),not(str(A)).

    /* m,n,ŋ,ɲ are codas */

coda([A]):- %(R17)
    phone(A), snt(A),not(nas(A)),not(lab(A)), not(pal(A)).
    /* l,ʁ are codas */

coda([A]):- %(R18)
    phone(A), snt(A), not(nas(A)), not(alv(A)), not(vel(A)).
    /* j,w,ɥ are codas */

% This predicate helps us get ordered results
syllable(A,B):-
```

```
    length(A,B),
    syllable(A).
```