A SYLLABIC PARSER IN PROLOG

Raquel Fernández Rovira
Universitat Autònoma de Barcelona

0. INTRODUCTION

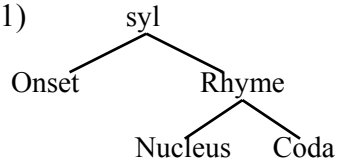
The main objetive of this work is to construct and implement a Prolog program capable of analyzing the syllable structure of phonic sequences in Catalan. As a consequence of that, we expect to be able to start a reflection about the theoretical implications that derive from the election of a particular implementation. In the first section, we present a number of common assumptions concerning syllable structure in general and about the structure of the syllable in the Catalan language. In the second and third sections, we analyze the phenomenon and provide a Prolog implementation that captures the observed regularities. We proceed gradually, presenting, first, a simplfied version (Syllabic Parser I) and, later, a slightly more elaborated one (Syllabic Parser II); the code for both versions is included in the two appendices found at the end of this paper. Last section is an attempt to establish some connections between the implementation strategies we used and a number of theoretical generalizations suggested from the perspetive of a theory of markedness and a number of concepts developed within the framework of Optimality Theory.

1. THE SYLLABLE

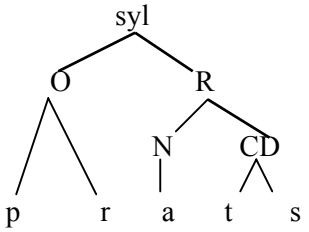
The syllable is a phonological unit that plays a fundamental role in the phonology of natural languages. Despite the difficulties one faces at the time of finding a clear and uniform phonetic correlate of syllables, it is obvious that, an empirical level, speakers possess an intuitive knowledge about how the phonetic stream is to be syllabified in their own languages and whether a sequence of sounds is well-formed or not with respect to syllable structure. Moreover, on the theoretical side, the syllable has become one of the key units of analysis: a large number of phonological processes makes reference to syllable structure in a crucial way, to the extent that, if this notion were not part of the theory, an important number of generalizations would be missed.[1]
The syllable brings together in a single unit one or more segments that are grouped into different constituents playing different roles within the syllable. In general, it is assumed that the constituents making up a syllable are onset, nucleus and coda, although within many theoretical frameworks it is also assumed that nucleus and coda are gathered together to form an intermediate constituent called rhyme. In this paper, following Jiménez (1999), we have opted for the latter, as shown in (1), where the Catalan word *prats* ('fields') is analyzed according to that constituent structure:

[1]A typical, but fairly illustrative, example is the variety of voice assimilation processes observed in several languages (see Mascaró 1995 for a thorough description) of which Catalan is an example. In this language, voice assimilation affects obstruents only in coda position: cap limit [kab.li.mit] ('no limit') vs. triple [tri.ple] ('triple').

(1)



Example:



In general, it is assumed that the distribution of segments within a syllable obeys the so called Sonority Principle (Clements 1990), according to which the nucleus is the peak of maximum sonority and that it gradually decreases towards the syllable margins. The universal sonority scale is set in (2):

(2) Low vowel > High vowel > Liquid > Nasal > Voiced fricative > Unvoiced fricative > Voiced stop > Unvoiced stop

1.1. *The syllabic structure of Catalan*

In Catalan, the syllable nucleus must always be filled with a vowel, whereas the onset and the coda may be made up by a number of elements bearing the feature [+consonantal].[2] Both the onset and the coda are optional, and both may be simple or complex; the presence of the nucleus, however, is mandatory for a syllable to be well-formed.

a) Onsets: As already noted in the previous paragraph, in Catalan, onsets may be simple or complex. Complex onsets may contain a maximum of two segments: the first consonant (C1) can only be a stop or /f/, while the second consonant (C2) must be either /l/ or /r/. In principle, any consonant may be part of a simple onset.

b) Codas: Catalan codas may also be simple or complex. The latter may contain a maximum of three consonantal segments in a sequence that, in principle, obeys the sonority scale.[3] The exception to that generalization is codas made up of three segments, where the last segments is usually an /s/ or a /z/. As for simple codas, there are no restrictions as to what consonants are allowed in them.

---

[2]The segments that may appear in an onset and in a coda may be both consonants and glides. For the sake of simplicity, I will not deal with those cases in which glides appear, be it as part of syllabic onsets or as part of codas as in the case of diphtongs.

[3]Mascaró (1989) has proposed the following simplefied sonority scale for Catalan:

(i) Vowels > Approximants > Liquids > Nasals > Fricatives > Stops

As we will se below, our implementation does not contain a direct corelate of the sonority scale, since we have solved this problem using a different, but functionally equivalent, strategy involving rules einriched with features.

2.- THE ANALYSIS OF SYLLABLE STRUCTURE

Segments in the phonic sequence of languages are not distributed at random. A typical explanation for that fact is to assume that every language incorporates a set of constraints, tacitly known by the speaker, that determines what is a possible combination in that language, which explains why certain sequences are not acceptable. In this sense, the set of constraints in charge of determining the syllable structure of a language is a *grammar*. Therefore, our main goal here will be to build a grammar, that is, a set of rules specifying what sequences of segments are acceptable as possible sequences in Catalan and, moreover, capable of assigning them a syllabic structure.

A program capable of performing such a task is typically called *parser* and it is most commonly associated with the syntactic analysis of natural languages. The very same techniques used to solve the problem of the analysis of syntactic structure may, however, be applied to any other problem involving an ordered sequence of elements that must be grouped into constituents, the organization of which may be specified by means of a set of rules (Clocksin & Mellish 1981). More specifically, then, our goal is to construct what may be called a Syllabic Parser.

A classical strategy to deal with the analysis problem is to make use of Context-Free Grammars (CFG). Prolog is particularly well-suited for the implementation of CFGs, since it incorporates the -> functor, which allows the programmer to abbreviate Prolog clauses and to use a much simpler notation of grammar rules as rules of a Definite Clause Grammar (DCG). As we show in the following section, our parser takes advantage of this additional functionality of Prolog in order to capture the basic facts of Catalan syllable structure.

3.- THE IMPLEMENTATION

Our main goal is to write a program that is capable of:

(i) Deciding whether some sequence of segments (a syllable, a word or a sequence of words) is a possible sequence in Catalan; and
(ii) If it is, assigning to that sequence one and only one syllable structure.

First, we need a lexicon that, in this particular case, will be a repertoire of segments. To simplify a bit the implementation, we have opted for reducing a bit the inventory with respect to the real phonological inventory of Catalan. Thus, we have included five vowels only (i.e., a, e, i, o and u) instead of eight, we have substantially simplified the fricatives, as we do not capture the difference between /s/ and /z/ and we dispense with the palatals. As for the liquids, we have included just one lateral /l/ and we do not distinguish between trill and flap realizations of rhotics. Finally, we have only included two nasals, /m/ and /n/, instead of four (i.e., the palatal and the velar have been dispensed with), whereas the list of stops is complete. This simplification is motivated by notational limitations only and it does not affect the performance of the program, which would be able to produce the adequate structures, should we be able to find a friendly way to represent the missing segments and, as a consequence, to extend our lexicon accordingly. The lexicon, as currently used in the implementation is set in (3):

(3)

```
v --> [a]        c(líquida) --> [r]          c(oclusiva) --> [t]
v --> [e]        c(líquida) --> [l]          c(oclusiva) --> [k]
v --> [o]        c(fricativa) --> [f]        c(oclusiva) --> [b]
v --> [i]        c(fricativa) --> [s]        c(oclusiva) --> [d]
v --> [u]        c(nasal) --> [m]                 c(oclusiva)  -->
[g]
                 c(nasal)  --> [n]
```

What we want is to be able to analyze sequences with any number of syllables. This can be accomplished by a recursive rule like the following:

(4)
```
cadena --> sil.
cadena --> sil, cadena.
```

It is not entirely obvious that syllable structure is actually recursive, rather it most likely seems that potential infinity of sequences is a consequence of mere iteration. That is, a string is a potentially infinite sequence of concatenated syllables. This particular theoretical and descriptive point notwithstanding, it appears that, in order to capture iterative structures (a phenomenon very much within the scope of regular grammars) with a DCG (and with CFGs in general), we are forced to make use of recursive rules (Gazdar & Mellish 1989).

In addition to that, it will be necessary to include a number of rules in charge of capturing the internal structure of syllabic constituents:

(5)
```
sil --> ob, rima.
sil --> rima.
rima --> nucli.
rima --> nucli, coda.
nucli --> v.
```

These rules need be complemented by a set of constraints over the internal structure of onsets and codas, according to the description given in section 1.1.

(6)
```
ob --> c(_).
ob --> c(oclusiva), c(líquida).
ob --> f, c(líquida).

coda --> c(_).
coda --> c(_), c(_).
coda --> c(_), c(_), s.
```

The result is a first version of our syllabic parser (Syllabic Parser I) whose code can be found in the Appendix I of this paper.

3.1. *Syllabic Parser I*

This first version of the parser accomplishes in a fairly satisfactory way the goals listed at the beginning of this section, namely,

(i) Deciding whether some sequence of segments (a syllable, a word or a sequence of words) is a possible sequence in Catalan; and

(ii) If it is, assigning to that sequence one and only one syllable structure.

This is, for example, how the program responds to a query like the following:

(7)
```
?- sil(X,[s,l,u,r,p],[]).
no
```

"'slurp' is not a possible syllable in Catalan."

We have also included an additional argument to all rules, which makes possible to construct the tree structure of every syllable:

(8)
```
?- analisi (Estructura, [k,o,r,p],[]).
Estructura = cadena (sil( ob(c(k)), rima(nucli(v(o)),
coda(c(r), c(p)))).
```

"'korp' es a monosyllabic string, the internal structure of which contains an onset, with the consonant k, and a rhyme; in the rhyme, the nucleus is filled with the vowel o and the coda is made up by a cluster containing r and p."

There are, however, a number of remaining shortcomings with this version of the parser. A minor problem has to do with the sequences of consonants that are permitted in coda position: the Syllabic Parser I not only accepts a correctly assigns a structure to a string like 'korp' (orthographically, corb Eng: 'crow'), but also accepts such nonexistent syllables as 'kopr'. As we will see presently, this is easily solved by further constraining the combinations of consonant clusters permitted in coda position.

A potentially more damaging problem has to do with the second of our goals above. The point is better appreciated if we take a look at the answer the parser provides when it is asked to syllabify a simple word like *casa* ('house'). This is reproduced verbatim in (9):

(9)
```
?- analisi(Estrucutra, [k,a,s,a],[]).
Estructura = cadena(sil(ob(c(k)), rima(nucli(v(a))),
cadena(sil(ob(c(s)), rima (nucli(v(a))))).

Estructura = cadena(sil(ob(c(k)), rima(nucli(v(a)),
coda(c(s)))), cadena(sil(rima (nucli(v(a))))).
```

The program treats this sequence as ambiguous and, accordingly, it provides two possible structures, one in which 's' is the onset of the second syllable and another where it appears as the coda of the first syllable. The same situation arises in words with a similar structure like *cofre* ('chest'), *simple* ('simple') or *obert* ('open'), to which the program respectively assigns the following syllabifications, where the first in each set is the correct one (the dot stands for a syllable edge): *co.fre* / *cof.re* / *cofr.e*; *sim.ple* / *simp.le*; *o.bert* / *ob.ert*.

The diagnostic is obvious: the grammar we have written for Syllabic Parser I overgenerates, while our goal number (ii) is very explicit in stating that the parser must assign 'one and only one syllable structure' to a sequence. Syllable structure, unlike syntactic structure, is not ambiguous and we need to capture this fact in our implementation. The diagnostic is obvious, but the cure not so.

3.2. *A syllabification algorithm*

Most theoretical models in phonology are derivational in nature and, as a consequence, tend to conceive of syllabification as a process that may be described algorithmically. These algorithms typically proceed as follows: (i) firstly, syllabic nuclei are identified; (ii) secondly, as many segments as possible are adjoined as syllable onsets; and (iii) finally, all remaining unsyllabified segments are incorporated as syllable codas. The idea behind all algorithms of this type is to capture a well documented tendency among the languages of world to maximize the number of onsets in a sequence and to minimize the number of codas, subject to the constraints imposed by the sonority scale. That is, an onset is always preferred over a coda, as long as the onset does not violate the sonority scale. As such, our parser is completely unable to capture this fact.

A possible alternative to implement an algorithm like the one we have just described would involve the introduction of rules like the following:

(10)
```
comprova ([X|R], [(X,ob)|R2]):-  c(X, posició(X,N),
                                 v(V, posició (V, N+1),
                                 comprova (R,R2).
```

That is, the predicate comprova would assign a segment X to an onset, just in case X is a consonant in the Nth position and there is a vowel in the Nth+1 position. Of course, this strategy would also require the incorporation of a rule in charge of numbering the positions of segments:

(11)
```
numera (N, [X|R], [posició(X,N)|R2]):-  N+1=N1,
numera (N1, R, R2).
```

Whatever the virtues of this strategy may be, in the end we have decided to try with a much simpler, but equally effective, option.

3.3. *Syllabic Parser II*

Before we get into the ambiguity problem, it may be worthwhile to point out that Syllabic Parser II (see Appendix II) is much more explicit in all that has to do with the combinations of consonantal segments that are allowed in coda position. Here, we have used a collection of features that make reference to the mode of articulation of each segment and that have been incorporated as arguments in the category consonant. This simple move is sufficient to adequately constrain the set of possible codas at the time that it eliminates the need, for instance, of implementing a rule assigning sonority values to consonants plus an additional constraint forcing that clusters be organized in a decreasing order. This second option would certainly come much closer to the notion of a 'sonority scale', although, in this case, the more extensional solution we have adopted is equally functional and, at the same time, not excessively prolix, given the relative simplicity of the problem. This point is reinforced if, as the reader may remember from section 1.1, the sonority scale can only be interpreted as describing a general tendency for the organization of segmental sequences, never as an absolute constraint. In effect, in any case, we would be forced to write a number of special rules in order to deal with

the awkward behavior of the fricative consonant /s/, which, in some specific contexts, violates the sonority principle. Thus, in Catalan /s/ may be part of a coda in such a way that syllables like taps ('caps, corks') or marcs ('frames') are totally acceptable, even though stops are less sonorous than nasals.[4]

Turning now to the other problem, recall that Syllabic Parser I assigned more than one structure to some sequences. Thus, for a work like *tapa* ('lid') the program yields the two logically possible syllabifications, namely, ta.pa and tap.a. As already pointed out, this overgeneration is inadmissible, as syllable structure is not ambiguous, the only acceptable syllabification being in this particular case the first one. In the implementation of Syllabic Parser II, in order to eliminate such structural ambiguities, we have opted for a strategy that makes use of the cut (!) predicate of Prolog.

This predicate ensures that the parser will come up with a single solution, when there is more than one. In order to accomplish this, we insert a 'cut' at the end of the first clause:

(12)
```
analisi (A,X,Y) :-    cadena (A,X,Y),
                       !.
```

Now, if Prolog is able to satisfy a goal, it will not look for additional alternatives through a series of reevaluations. In a way, we are telling the system that if he managed to reach a certain point, it means that it used the adequate rules to satisfy that goal. Let us consider in some detail how the system now treats the example we saw in (9) above:

(13)
```
?- analisi (Estructura, [k,a,s,a],[]).
```

To solve this goal, Prolog uses the only clause containing the analisi predicate. In order to satisfy the subgoal cadena appearing in the first clause, it will first try to use the first rule containing the cadena predicate, but, since it cannot gather all segments into a single syllable (i.e., it cannot satisfy the goal), it will backtrack in order to call the following rule:

(14)
```
?- analisi (Estructura, [k,a,s,a],[]).
```

At this point, the system will try to satisfy the goal sil(A), which will take him through the following collection of rules:

(15)
```
sil (sil (O,R)) --> ob(O), rima(R).
ob(ob(C)) --> c(C,_).
c(c(k), oclusiva) --> [k].
rima ( rima (N)) --> nucli (N).
nucli (nucli (V)) --> v(V).
v (v(a)) --> [a].
```

To satisfy the subgoal cadena(B), the parser will use more or less the same rules as before, starting with:

(16)
```
cadena(cadena(A)) --> sil(A).
```

[4]This is a fairly common phenomenon in a great variety of languages. In English, for example, such violations of the sonority scale are observed both in codas (as in 'cats') and in onsets (as in 'special').

In the end, it will be able to satisfy the goal cadena(A,X,Y), and, as a consequence, the initial goal too, with the variables instantiated as in (17):

(17) `cadena((sil(ob(c(k)),rima(nucli(v(a))),`
`cadena(sil(ob(c(s)),rima (nucli(v(a))))), [k,a,s,a],[]).`

At this point, the system reaches the 'cut', which tells it that it must not go on, because it already found the correct answer.

With the incorporation of the cut predicate, Syllabic Parser II is able to assign the correct structure to all those sequences that were a problem for Syllabic Parser I, given their potential ambiguity. In this case, this predicate serves as a very powerful tool, making it possible for the program to work correctly and eliminating the need to introduce additional modifications in the form of the clauses. It is important to point out, however, that this strategy relies heavily on a specific rule ordering without which it would be impossible to get the desired results. One has to bear in mind, then, that Prolog applies rules in the order in which they have been introduced in the data-base, a point that must not be forgotten at the time of planning possible extensions of the program.

4. THE THEORY OF MARKEDNESS AND THE PRIORITIZATION OF RULES

We already noted that there exists a universal preference among the world's languages for syllables with onsets. The syllabification algorithm described above and a series of processes of insertion of consonants observed in some languages[5] are direct reflex of this fact. Along the same lines, the so-called resyllabification or postlexical syllabification in Catalan, also shows this general tendency.[6] These and other similar observations are supposed to be captured by a theory of markedness for syllabic structure and, more concretely, by such principles like the Onset Principle (Itô 1989), according to which onsetless syllables are to be avoided. Along with this principle, Optimality Theory[7] incorporates other constraints like, for example, NO CODA, with the aim of capturing a generalized preference among the world's languages for codaless syllables.
Thus, it is assumed that CV (consonant-vowel) is the universally unmarked syllable, which means that, given an underlying sequence of the form CVCV, a syllabification

[5]In German, for example, when syllabification of a word would result in the presence of an onsetless syllable, a glottal stop is inserted as an onset of that syllable. Thus, the word Theater ('theater') syllabifies as The.[?a].ter, where the second syllable gets its onset through the insertion of the glottal stop.

[6]This is a process that presumably takes place at the level of the utterance. When a word starts with a vowel, tha is, it is onsetless, and the preceding word ends in a consonant (i.e., it has a coda), the consonant in coda position is reanalyzed as the onset of the following syllable, even though it was lexically onsetless. Thus, in the sequence *mal home* ('bad man'), the /l/ in coda position becomes the onset of the first syllable of the second word, ma.lo.me (orthographic <h> is never pronounced in Catalan). Unfortunately, postelixal syllabification in Catalan is not totally consisten with this principle, since in sequences where a word ends in a stop and the following word starts with a liquid (e.g., *cuc lent*, lit: 'worm slow'), no resyllabification takes place in order to produce a complex onset (*kuk.len* vs. **ku.klen*). Our system does not take into account this irregularity, an implementation of which we leave for further research.

[7]See Prince & Smolensky 1993, McCarthy & Prince 1993, among others.

like CV.CV will always be favored over other alternative analyses like CVC.V, CVC.CV (with insertion of an onset) or CV.V (with deletion of a consonant). There is, moreover, another general constraint that limits the complexity of syllabic constituents, what Prince & Smolensky, again within the OT framework, have termed NO-COMPLEX.

These descriptive principles that are part of the theory of markedness find their way in our implementation through the strategy of rule ordering, which, along with the action of the 'cut' predicate, act as mechanisms of prioritization. Thus, by ordering the rules in charge of describing syllable-types, we manage to give priority to syllables with onsets over syllables without them (The Onset Principle), whereas the ordering of rules describing syllable rhymes penalizes syllables with codas (NO CODA).

(18)
```
sil (sil (O, R)) --> ob(O), rima(R).
sil (sil (R)) --> rima(R).

rima (rima(N)) --> nucli (N).
rima (rima(N, CD)) --> nucli (N), coda(CD).
```

On the other hand, the organization of the clauses describing the internal structure of onsets and codas (onset with a single consonant, followed by complex onset with two consonants; coda with a single consonant, followed by complex coda with two consonants, followed by complex coda with three consonants) favor simple onsets an codas over complex ones (NO-COMPLEX). Thus, by prioritizing syllables with onsets and without codas, we favor (as long as the resulting cluster agrees with the phonotactic constraints of the language) structures of the type CV.CCV over those of type CVC.CV (but see note 6):

(19)
```
?- analisi (Estructura, [k,o,f,r,e],[]).
     Estructura = cadena (sil(ob(c(k)), rima(nucli(v(o))),
cadena(sil (ob(c(f), c(r)), rima(nucli(v(e)))))).
```

Thus, with rule ordering in the data-base establishing a prioritization of certain structures, the rest of the rules is only accessed by the system when, given the properties of the input string, it is impossible to satisfy some goal by applying only prioritized rules. For example, the string *bon* ('good') can only have the following structure:

(20)
```
?- analisi (Estructura, [b,o,n],[]).
     Estructura = cadena (sil (ob(c(b)), rima (nucli(v(o)),
coda (c(n)))).
```

In order to satisfy the goal, the system had to use a rule in which the rhyme is made up of a nucleus and a coda; the properties of the input make necessary the emergence of the marked form. Now, the prioritization of rules makes it possible that the coda of this syllable becomes an onset in a sequence like *bon home* ('good man'), which is assigned the optimal analysis CV.CV.CV:

(21)
```
?- analisi (Estructura,[b,o,n,o,m,e],[]).
     Estructura = cadena (sil(ob(c(b)), rima(nucli(v(o))),
cadena (sil(ob(c(n)), rima(nucli(v(o)))), cadena (sil(ob(c(m)),
rima (nucli(v(e)))))).
```

5. CONCLUSIONS

Our program is able to simulate a specific linguistic skill that appears to be part of the implicit knowledge of speakers. The implementation provides a partial but adequate analysis of this capacity in the sense that it is able to detemine whether a specific string of segments constitutes a possible sequence in Catalan or not and, in case it is, it is able to assign a single and correct syllabic structure. This is accomplished by making use of the Prolog cut predicate and a specific rule ordering that gives priority to unmarked forms over the marked ones, that is those where syllables have no codas an their margins are as simple as possible. This does not block marked structures completely, as these may eventually emerge when this is required by the properties of the input string. In these cases, rules occupying a lower-ranked position in the data-base are activated and syllables with codas and/or complex margins emerge.

Apart from this interpretations in terms of the theory of markedness, we have found interesting the attempt of establishing possible connections with some basic tenets of Optimality Theory, despite the ambiguous interpretation, in this and other frameworks, of such theoretical terms as 'marked/unmarked' and 'default'. This is not the place to start a discussion of this topics, but it is clear that a clarification of these issues is necessary, a topic that we expect to treat in our future research.

6. Bibliography

Clements, G. N. (1990). "The role of the sonority cycle in core syllabification". In J. Kingston & M. Beckman, eds. *Papers in Laboratory Phonology I. Between the Grammar and Physics of Speech*. Cambridge University Press, Cambridge, pp. 283-333.

Clocksin, R. & C. Mellish (1981). *Programming in Prolog*. Springer Verlag, Berlin.

Gazdar, G. & C. Mellish (1989). *Natural Language Processing in Prolog*. Addison Wesley.

Itô, J. (1989). "A prosodic theory of epenthesis". *Natural Language and Linguistic Theory* 7(2):217-260.

Jiménez, J. (1999). *L'estructura sil·làbica del català*. Institut Interuniversitari de Filologia Valenciana/Publicacions de l'Abadia de Montserrat, València/Barcelona.

Mascaró, J. (1989). "On the form of deletion and insertion rules". *Probus* 1:31-61.

Mascaró, J. (1995). "A reduction and spreading theory of voicing and other sound effects". *Catalan Working Papers in Linguistics* 4(2):267-328.

McCarthy, J. J. & A. S. Prince (1993). Prosodic Morphology I: Constraint Interaction and Satisfaction. Ms. University of Massachusetts and Rutgers University, Amherst and New Bruswick.

Prince, A. S. & P. Smolensky (1993). Optimality Theory. Constraint Interaction in Generative Grammar. Technical Report #2 of the Rutgers Center for Cognitive Science, Rutgers University, New Brunswick, NJ.

APPENDIX I

Syllabic Parser I

```
v(v(a)) --> [a].
v(v(e)) --> [e].
v(v(o)) --> [o].
v(v(i)) --> [i].
v(v(u)) --> [u].

c(c(p),oclusiva) --> [p].
c(c(t),oclusiva) --> [t].
c(c(k),oclusiva) --> [k].
c(c(b),oclusiva) --> [b].
c(c(d),oclusiva) --> [d].
c(c(g),oclusiva) --> [g].

c(c(r),liquida) --> [r].
c(c(l),liquida) --> [l].

c(c(f),fricativa) --> [f].
c(c(s),fricativa) --> [s].

c(c(m),nasal) --> [m].
c(c(n),nasal) --> [n].

analisi(A,X,Y):-
                cadena(A,X,Y).

cadena(cadena(A)) --> sil(A).
cadena(cadena(A,B)) --> sil(A), cadena(B).

sil(sil(O,R)) --> ob(O), rima(R).
sil(sil(R)) --> rima(R).

ob(ob(C)) --> c(C,_).
ob(ob(C1,C2)) --> c(C1,oclusiva), c(C2,liquida).
ob(ob(c(f),C2)) --> c(c(f),_), c(C2,liquida).

rima(rima(N)) --> nucli(N).
rima(rima(N,CD)) --> nucli(N), coda(CD).

nucli(nucli(V)) --> v(V).

coda(coda(C)) --> c(C,_).
coda(coda(C1,C2)) --> c(C1,_), c(C2,_).
coda(coda(C1,C2,c(s))) --> c(C1,_), c(C2,_), c(c(s),_).
```

APPENDIX II

Syllabic Parser II

```
v(v(a)) --> [a].
v(v(e)) --> [e].
v(v(o)) --> [o].
v(v(i)) --> [i].
v(v(u)) --> [u].

c(c(p),oclusiva) --> [p].
c(c(t),oclusiva) --> [t].
c(c(k),oclusiva) --> [k].
c(c(b),oclusiva) --> [b].
c(c(d),oclusiva) --> [d].
c(c(g),oclusiva) --> [g].

c(c(r),liquida) --> [r].
c(c(l),liquida) --> [l].

c(c(f),fricativa) --> [f].
c(c(s),fricativa) --> [s].

c(c(m),nasal) --> [m].
c(c(n),nasal) --> [n].

analisi(A,X,Y):-
                cadena(A,X,Y),
                !.

cadena(cadena(A)) --> sil(A).
cadena(cadena(A,B)) --> sil(A), cadena(B).

sil(sil(O,R)) --> ob(O), rima(R).
sil(sil(R)) --> rima(R).

ob(ob(C)) --> c(C,_).
ob(ob(C1,C2)) --> c(C1,oclusiva), c(C2,liquida).
ob(ob(c(f),C2)) --> c(c(f),_), c(C2,liquida).

rima(rima(N)) --> nucli(N).
rima(rima(N,CD)) --> nucli(N), coda(CD).

nucli(nucli(V)) --> v(V).

coda(coda(C)) --> c(C,_).
coda(coda(C1,c(s))) --> c(C1,_), c(c(s),_).

coda(coda(C1,C2)) --> c(C1,liquida),(c(C2,nasal); c(C2,oclusiva)).
coda(coda(c(s),C2)) --> c(c(s),_), c(C2,oclusiva).

coda(coda(C1,C2,c(s))) --> c(C1,liquida),(c(C2,oclusiva);
c(C2,nasal)), c(c(s),_).
coda(coda(c(s),C2,c(s))) --> c(c(s),_), c(C2,oclusiva), c(c(s),_).
```