Ejercicio 1: analizador morfológico

- **Analizador morfológico de verbos regulares de la 1ª conjugación en modo indicativo (tiempos simples)**
  - Identificar el infinitivo del verbo, el tiempo, la persona y el número
  - Por ejemplo
    - Hablará
    - Hablar, futuro, 3ª persona, singular

Ejercicio 1: diccionario

- Necesitamos incluir información en el diccionario
  - Verbos regulares de la 1ª conjugación
    - Infinitivo = Raíz ++ “ar”
    - Conjugaciones = Raíz ++ Terminación
  - Debemos identificar las terminaciones

```
% es_terminación(Termin, Tiempo, Persona, Número)

es_terminación(o, presente, 1, singular).
es_terminación(as, presente, 2, singular).
...
es_terminación(aré, futuro, 1, singular).
...
```

Ejercicio 1: diccionario

- Necesitamos incluir información en el diccionario
  - Verbos regulares de la 1ª conjugación
    - Infinitivo = Raíz ++ “ar”
    - Conjugaciones = Raíz ++ Terminación
  - Debemos enumerar las raíces de los verbos que queremos reconocer conjugados
    - Podemos asociarlas con los infinitivos
    - O no (y tendremos que construirlos en la DCG)

```
% es_verbo(Raíz, Infinitivo)
es_verbo(am, amar).
es_verbo(habl, hablar).
es_verbo(jug, jugar).
es_verbo(peg, pegar).
...
```

```
% es_raíz(Raíz)
es_raíz(am).
es_raíz(habl).
es_raíz(jug).
es_raíz(peg).
...
```

Ejercicio 1: DCG analizador morfológico

- Sólo necesitamos una regla
  - Verbos regulares de la 1ª conjugación
    - Infinitivo = Raíz ++ “ar”
    - Conjugaciones = Raíz ++ Terminación
  - Estructura básica

```
verbo --> [V].
```

  - Necesitamos parámetros para devolver el resultado y hacer todo el trabajo de descomposición

```
verbo(Inf, Tpo, Pers, Num) --> [V], { ... }.
```

  - Leeremos un átomo y tenemos que descomponerlo. Para ello, necesitamos saber cómo hacerlo en Prolog
    - Transforma un átomo en una lista de caracteres (códigos ascii)

```
% name(?Átomo0Int, ?Cadena)
```

      Necesita al menos 1 argumento instanciado: sirve en los 2 sentidos

      Así podemos usar predicados de listas para la descomposición

Ejercicio 1: DCG analizador morfológico

- Necesitamos parámetros para devolver el resultado y hacer todo el trabajo de descomposición

```
verbo(Inf, Tpo, Pers, Num) -->
        [V],
        {
          name(V, VerboCad),
          append(RaízCad, TermCad, VerboCad),
          name(Raíz, RaízCad),
          es_verbo(Raíz, Inf),
          name(Term, TermCad),
          es_terminación(Term, Tpo, Pers, Num)
        }.
```

Ejercicio 1: DCG analizador morfológico

- Con la segunda alternativa de representación del diccionario, habría que hacer un poquito más de trabajo

```
verbo(Inf, Tpo, Pers, Num) -->
        [V],
        {
          name(V, VerboCad),
          append(RaízCad, TermCad, VerboCad),
          name(Raíz, RaízCad),
          es_raíz(Raíz),
          name(ar, ArCad),
          append(Raíz, ArCad, Inf),
          name(Term, TermCad),
          es_terminación(Term, Tpo, Pers, Num)
        }.
```

Ejercicio 1: uso de la DCG

- Como el símbolo inicial de la gramática es

```
verbo
```

y tiene 4 parámetros, el uso en Prolog es

```
?- verbo(Inf, Tpo, Pers, Num, [hablarás], []).
Inf = hablar
Tpo = futuro
Pers = 2
Num = singular ;

No
?- verbo(Inf, Tpo, Pers, Num, [marajá], []).

No
```

Ejercicio 3: analizador sintáctico

- **Construir un analizador Prolog basado en una DCG que permita analizar frases como las siguientes**
  - *David habla con Ana*
  - *Julia lee libros en el jardín*
  - *Los niños leen*
  - *Pedro juega*
  - *Elvira juega en la piscina*

Ejercicio 3: analizador sintáctico

- **Cada verbo puede llevar como máximo un complemento circunstancial**
  - Los complementos circunstanciales siempre empiezan por una preposición
  - Para cada verbo se establece una única proposición permitida o ninguna (si el verbo no admite complementos)
  - Aunque el verbo admita un complemento circunstancial puede no llevar ninguno

Ejercicio 3: analizador sintáctico

- **Las frases con verbos transitivos pueden llevar o no complemento directo**
  - Si hay complemento directo, el complemento circunstancial aparecerá después
- **Las frases con verbos intransitivos no pueden tener complemento directo**

Ejercicio 3: analizador sintáctico

- **Se debe garantizar**
  - La concordancia sujeto-verbo
  - La compatibilidad verbo-complemento circunstancial
- **Si el análisis de la frase es correcto, se devolverá el árbol de análisis sintáctico correspondiente**

Ejercicio 3: estructura básica

```
frase       --> g_nominal, g_verbal.

g_nominal   --> nombre.
g_nominal   --> articulo, nombre.
g_nominal   --> nombre_propio.

g_verbal    --> verbo_trans, g_nominal, complemento.
g_verbal    --> verbo_trans, g_nominal.
g_verbal    --> verbo_trans, complemento.
g_verbal    --> verbo_trans.

g_verbal    --> verbo_int, complemento.
g_verbal    --> verbo_int.
```

Estructura de frase

Posibles formas de grupos nominales

Posibles formas de grupos verbales, distinguiendo tipos de verbos

Ejercicio 3: árbol de análisis sintáctico

```
frase(f(GN, GV))  --> g_nominal(GN), g_verbal(GV).

g_nominal(gn(N))          --> nombre(N).
g_nominal(gn(A, N))       --> articulo(A), nombre(N).
g_nominal(gn(N))          --> nombre_propio(N).

g_verbal(gv(V, CD, C))  --> verbo_tr(V),
                            g_nominal(CD), complem(C).
g_verbal(gv(V, CD))       --> verbo_tr(V), g_nominal(CD).
g_verbal(gv(V, C))        --> verbo_tr(V), complem(C).
g_verbal(gv(V))           --> verbo_tr(V).

g_verbal(gv(V, C))        --> verbo_int(V), complem(C).
g_verbal(gv(V))           --> verbo_int(V).
```

Ejercicio 3: concordancia

```
frase(f(GN, GV)) --> g_nominal(GN, Num), g_verbal(GV, Num).
```

concordancia sujeto-verbo

```
g_nominal(gn(N), Num)       --> nombre(N, Num, _).
g_nominal(gn(A, N), Num)    --> articulo(A, Num, Gen),
                                nombre(N, Num, Gen).
g_nominal(gn(N), singular) --> nombre_propio(N).

g_verbal(gv(V, CD, C), Num)--> verbo_tr(V, Num),
                               g_nominal(CD, _), complem(C).
g_verbal(gv(V, CD), Num)    --> verbo_tr(V, Num),
                                g_nominal(CD, _).
g_verbal(gv(V, C), Num)     --> verbo_tr(V, Num), complem(C).
g_verbal(gv(V))             --> verbo_tr(V, Num).

g_verbal(gv(V, C), Num)     --> verbo_int(V, Num), complem(C).
g_verbal(gv(V), Num)            --> verbo_int(V, Num).
```

concordancia artículo-nombre

Ejercicio 3: complementos circunstanciales

```
g_verbal(gv(V, CD, C), N)  --> verbo_trans(V, N, Lprep),
                               g_nominal(CD, _N),
                               complemento(C, Lprep).
g_verbal(gv(V, CD), N)     --> verbo_trans(V, N, _L),
                               g_nominal(CD, _N).
g_verbal(gv(V, C), N)      --> verbo_trans(V, N, Lprep),
                               complemento(C, Lprep).
g_verbal(gv(V), N)         --> verbo_trans(V, N, _L).
g_verbal(gv(V, C), N)      --> verbo_int(V, N, Lprep),
                               complemento(C, Lprep).
g_verbal(gv(V), N)         --> verbo_int(V, N, _L).

complemento(c(P, GN), [Prep]) --> preposicion(P, [Prep]),
                                  g_nominal(GN, _N).
```

¡sólo lista unitaria en este ej.!

```
preposicion(p(P), [P])     --> [P], {es_preposicion(P)}.
```

Ejercicio 3: resto de la DCG

```
nombre_propio(np(P))      --> [P],
                              {es_nombre_propio(P)}.

nombre(n(P), Num, Gen)    --> [P],
                              {es_nombre(P, Num, Gen)}.

articulo(a(P), Num, Gen) --> [P],
                              {es_articulo(P, Num, Gen)}.

verbo_int(v(P), Num, L) --> [P],
                              {es_verbo_intr(P, Num, L)}.

verbo_trans(v(P), Num, L) --> [P],
                              {es_verbo_trans(P, Num, L)}.
```

Ejercicio 3: diccionario

```
es_nombre_propio(ana).
es_nombre_propio(julia).
es_nombre_propio(david).
es_nombre_propio(pedro).
es_nombre_propio(elvira).

es_nombre(libros, plural, masc).
es_nombre(niños, plural, masc).
es_nombre(jardín, singular, masc).
es_nombre(piscina, singular, fem).

es_articulo(el, singular, masc).
es_articulo(la, singular, fem).
es_articulo(los, plural, masc).
```

Ejercicio 3: diccionario

```
es_verbo_intr(habla, singular, [con]).
es_verbo_intr(juega, singular, [en]).

es_verbo_trans(lee, singular, [en]).
es_verbo_trans(leen, plural, [en]).
es_verbo_trans(escriben, plural, []).

es_preposicion(en).
es_preposicion(con).
```

Ejercicio 3: uso de la DCG

- Como el símbolo inicial de la gramática es

```
frase
```

y tiene 1 parámetro, el uso en Prolog es

```
?- frase(Árbol, [david, habla, con, ana], []).
Árbol = frase(gn(np(david)),
              gv(v(habla), c(p(con), gn(np(ana)))))

?- frase(Árbol, [los, niños, leen], []).
Árbol = frase(gn(a(los), n(niños)), gv(v(leen)))

?- frase(Árbol, [pedro, juega], []).
Árbol = frase(gn(np(pedro)), gv(v(juega)))

?- frase(A, [elvira, juega, en, la, piscina], []).
A = frase(gn(np(elvira)),
   gv(v(juega), c(p(en), gn(a(la), n(piscina)))))
```

Ejercicio 6: interfaz en LN a BD

- **Construir una interfaz en LN a una BD correspondiente a un catálogo de material informático**
- **El sistema debe ser capaz de contestar a preguntas sencillas y, además, para que sea más cómodo de usar se permitirán algunos tipos de elipsis, que se resolverán mediante un mecanismo de foco de atención**
  - *¿Qué precio tiene la impresora HX-851?*
  - *¿Y la HX-853?*
  - *¿Cuál es el precio del módem R-411?*
  - *¿Y la velocidad?*

Ejercicio 6: interfaz en LN a una BD

- Novedad de este ejercicio
  - Para la comodidad de uso del sistema, se permitirán algunos tipos de elipsis *(omisión de parte de la pregunta)*
  - Se resolverán mediante un mecanismo de foco de atención
- BD simplificada con cláusulas Prolog

```
% valor(Nombre, Atributo, Valor)
valor('HX-851', precio, 10000).
valor('HX-853', precio, 12000).
valor('R-411', precio, 5000).
valor('R-411', velocidad, 56).
...
```

Ejercicio 6: ciclo pregunta-respuesta

- Ciclo pregunta-respuesta con foco de atención

```
consulta(Foco) :-
        write('Pregunta:    '),
        read(P),
        analiza(R, Foco, NuevoFoco, P, []),
        % Llamada al símbolo inicial de la DCG
        write('Respuesta:    '),
        write(R), nl,
        consulta(Nuevofoco).
```

Ejercicio 6: análisis de consultas

```
analiza(Valor, _FAnt, foco(Nombre, Atributo)) -->
                comienzo,
                [Atributo],
                sigue,
                [Nombre],
                { valor(Nombre, Atributo, Valor) }.

comienzo --> ([dame] ; [dime ] ; [cuál, es ]),
             determinante.

sigue --> [del] | [de], determinante.

determinante --> [el] | [la] | [un] | [una] | [].
```

Ejercicio 6: análisis de consultas

```
analiza(Valor, _FAnt, foco(Nombre, Atributo)) -->
                [qué, Atributo, tiene],
                determinante,
                [Nombre],
                { valor(Nombre, Atributo, Valor) }.

analiza(Valor, foco(N, A), foco(N, Atributo)) -->
                [y], determinante,
                [Atributo],
                { valor(N, Atributo, Valor) }.

analiza(Valor, foco(N, A), foco(Nombre, A)) -->
                [y], determinante,
                [Nombre],
                { valor(Nombre, A, Valor) }.
```

Ejercicio 6: ejemplo de uso

```
| ?- consulta(_).
Pregunta:    [qué, precio, tiene, la, 'HX-851'].
Respuesta:     10000
Foco:             foco(HX-851, precio)
Pregunta:    [y, la, 'HX-853'].
Respuesta:     12000
Foco:             foco(HX-853, precio)
Pregunta:    [cuál, es, el, precio, del, 'R-411'].
Respuesta:     5000
Foco:             foco(R-411, precio)
Pregunta:    [y, la, velocidad].
Respuesta:     56
Foco:             foco(R-411, velocidad)
Pregunta:    [].
No
```