Teoría de Autómatas y Lenguajes Formales (2008/09)

Práctica 2

(Gramáticas independientes del contexto, forma normal de Chomsky y algoritmo CYK)

Enunciado

Esta práctica se centra en el algoritmo CYK (Cocke, Younger y Kasami), el cual, dada una cadena de símbolos de entrada y una gramática independiente del contexto en forma normal de Chomsky, nos dice si la cadena pertenece o no al lenguaje generado por dicha gramática.

El algoritmo se enuncia como sigue:

Sea G = (N,T,P,S) una gramática independiente del contexto, que no tiene reglas epsilon y que está en forma normal de Chomsky. Sea w una cadena de T^* de longitud n. Sea w_{ij} la subcadena de w que comienza en la posición i y tiene longitud j.

- Para cada i = 1, 2, ..., n, sea $N_{i1} = \{A \mid A \rightarrow w_{i1}\}$.

  Es decir, N_{i1} es el conjunto de los no terminales que producen directamente el i-ésimo símbolo de w.

- Para j = 2, 3, ..., n, hacer lo siguiente:
  - Para i = 1, 2, ..., n-j+1, hacer lo siguiente:
    - Inicializar N_{ij} al conjunto vacío.
    - Para k = 1, 2, ..., j-1, añadir a N_{ij} todos los símbolos no terminales A para los cuales A -> B C es una regla de P, con B perteneciente a N_{ik} y C perteneciente a $N_{i+k,j-k}$.
- Si S pertenece a N_{1n}, entonces w pertenece a L(G).

Ejemplo:

- Consideremos la grámatica independiente del contexto:

```
S -> A B | B C
A -> B A | a
B -> C C | b
C -> A B | a
```

  Para la cadena w = bbab, se obtiene la siguiente tabla, donde cada casilla representa al conjunto N_{ij}:

j = 4	S, C			
j = 3	A	S, C		
j = 2	-	A, S	S, C	
j = 1	B	B	A, C	B

	i = 1	i = 2	i = 3	i = 4
	b	b	a	b

Dado que S está en N_{14}, w puede ser generada a partir del símbolo inicial S. Por tanto, w está en el lenguaje generado por esta gramática.

La librería `ocaml-talf` incluye las siguientes funciones para construir cómodamente valores de los tipos de datos involucrados en esta práctica:

- Se proporcionan dos funciones, `gic_of_string` y `gic_of_file`, para construir valores del tipo de dato `gic`.
- Se proporcionan dos funciones, `cadena_of_string` y `cadena_of_file`, para construir valores del tipo de dato `simbolo list`.

Se pide implementar lo siguiente:

1. Una función `es_FNC : gic -> bool`, que indique si una gramática dada está o no en formal normal de Chomsky. Valor de este apartado: 1 punto.

2. Un tipo de dato `tabla_CYK` que dé soporte a la tabla triangular que necesita el algoritmo CYK para funcionar. Valor de este apartado: 2 puntos.

3. Una función `parse_CYK : simbolo list -> gic -> tabla_CYK`, que dada una cadena de símbolos de entrada y una gramática, devuelva la tabla triangular resultante de aplicar el algoritmo CYK sobre esa gramática y sobre esa cadena. Lo primero que debe hacer esta función es comprobar que la cadena tiene longitud 1 o mayor, y que la gramática está en forma normal de Chomsky. Si no es así, la función debe activar una excepción. Valor de este apartado: 3 puntos.

4. Una función `arbol_CYK : int -> int -> tabla_CYK -> string list`, que devuelva todos los posibles sub-árboles de análisis (secuencias parentizadas de símbolos) que se pueden generar a partir de los contenidos de la celda N_{ij}, y que por tanto dan cobertura a la subcadena w_{ij} (siendo w la cadena de entrada que dio lugar a la tabla CYK, y siendo i y j los dos parámetros de tipo `int` que debe recibir la función). Valor de este apartado: 4 puntos.

Ejemplos de ejecución:

```
# let g = gic_of_string
     "S A B C;
      a b;
      S;
      S -> A B | B C;
      A -> B A | a;
      B -> C C | b;
      C -> A B | a";;
val g : Auto.gic =
  Gic
   (Conj.Conjunto
     [No_terminal "S"; No_terminal "A"; No_terminal "B"; No_terminal "C"],
    Conj.Conjunto [Terminal "a"; Terminal "b"],
    Conj.Conjunto
     [Regla_gic (No_terminal "S", [No_terminal "A"; No_terminal "B"]);
      Regla_gic (No_terminal "S", [No_terminal "B"; No_terminal "C"]);
      Regla_gic (No_terminal "A", [No_terminal "B"; No_terminal "A"]);
      Regla_gic (No_terminal "A", [Terminal "a"]);
      Regla_gic (No_terminal "B", [No_terminal "C"; No_terminal "C"]);
      Regla_gic (No_terminal "B", [Terminal "b"]);
      Regla_gic (No_terminal "C", [No_terminal "A"; No_terminal "B"]);
      Regla_gic (No_terminal "C", [Terminal "a"])],
    No_terminal "S")
```

```
# let c1 = cadena_of_string "b b a b";;
val c1 : Auto.simbolo list =
  [Terminal "b"; Terminal "b"; Terminal "a"; Terminal "b"]

# let t1 = parse_CYK c1 g;;
- : tabla_CYK = ...

# arbol_CYK 1 4 t1;;
- : string list =
  ["(S (A (B b) (A (B b) (A a))) (B b))";
   "(S (B b) (C (A (B b) (A a)) (B b)))"]

# let c2 = cadena_of_string "b b b b";;
val c1 : Auto.simbolo list =
  [Terminal "b"; Terminal "b"; Terminal "b"; Terminal "b"]

# let t2 = parse_CYK c2 g;;
- : tabla_CYK = ...

# arbol_CYK 1 4 t2;;
- : string list = []
```

Realiza todas las implementaciones en un fichero llamado `cyk.ml`, que es el único que debes entregar después.

Tiempo para la realización de la práctica: 3 semanas.
