Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Pràctica PAA - Algoritme CKY

Programació i Algorítmia Avançades

Grau en Intel·ligència Artificial
Daniel Álvarez 23857151X
Albert Roca 48106974J

23/05/2025

Contents

1 Introducció

En aquesta pràctica hem desenvolupat l'algorisme CKY, Cocke–Kasami–Younger, aquest és un dels clàssics per a detectar llenguatges generats per Context-Free-Grammars en forma normal de Chomsky (CNF).

Hem desenvolupat tres extensions. En primer lloc, una extensió base on s'executa l'algoritme CKY normal i corrent. En segon lloc, hem fet una primera extensió la qual fa la conversió de CFG a CNF per tal de poder executar CKY. I finalment, hem implementat una versió probabilística de l'algorisme CKY, que ens permet calcular la probabilitat que una paraula sigui generada per una gramàtica probabilística donada (PCFG).

A part de les extensions, hem creat un generador automàtic de gramàtiques i paraules. Aquest ens permet construir gramàtiques aleatòries que poden ser tant en CNF com en CFG general, així com assignar probabilitats de manera coherent en cas de voler fer servir la versió probabilística.

Així doncs, el nostre CKY rep una gramàtica que si està en CFG la transforma a CNF i aplica CKY. Si no ho està aplica directament l'algoritme, i retorna True o False depenent de si un string donat pertany a la gramàtica. Aquesta és l'estructura dels arxius:

- **CODIS:**
  - **Main.py:** Menú principal d'execució
  - **Extensió_base.py:** Algoritme CKY
  - **Extensió_1.py**: Codi de conversió de CFG a FNC
  - **Extensió_2.py**: Algoritme CKY probabilístic
  - **Generador_gramàtiques.py**: Generador de gramàtiques aleatòries
  - **Generador_Paraula.py**: Generador de paraules aleatòries, per comprovar si pertanyen a una gramàtica.
  - **Experimentació.py**: Arxiu que executa totes les possibilitats de l'experimentació aleatòria, per a més tard desar els resultats en un arxiu .txt.
  - **Utils.py**: Arxiu amb dos funcions per llegir gramàtiques i passar-les al format de llistes de tuples.
- **DADES:**
  - **gramatica_cfg.txt**: Gramàtica CFG d'exemple.
  - **gramatica_cfg2.txt**: Segona CFG per proves.
  - **gramatica_probabilistica.txt**: Gramàtica probabilística.
  - **gramatica.txt**: Gramàtica base.
  - **gramatica2.txt**: Gramàtica CNF.
  - **paraula.txt**: Paraula a analitzar.
- **JOCS DE PROVA:**
  - **jocs_de_proves1.txt**: Resultats de l'experiment aleatori 1.
  - **jocs_de_proves2.txt**: Resultats de l'experiment aleatori 2.

2 Format de les gramàtiques

Gramàtiques no probabilístiques:
Les gramàtiques sense probabilitats les representem com una llista de tuples. Cada tupla conté dues parts: el símbol no terminal del costat esquerre de la regla, i una llista ordenada amb els símbols del costat dret.

```
regles = [
    ('S', ['X1', 'X1']),
    ('S', ['b']),
    ('X1', ['S', 'X2']),
    ...
]
```

Així, cada element de la llista correspon a una regla de la gramàtica, on el primer valor és el no terminal i el segon és la seqüència de símbols (terminals o no terminals) que pot generar.

Gramàtiques probabilístiques:
En el cas de gramàtiques probabilístiques, hem afegit a cada regla la seva probabilitat associada. Per això utilitzem una llista on cada element és una tupla formada per dues parts: una sub-tupla amb la regla (head i body), i la probabilitat de producció (un valor `float` entre 0 i 1).

```
regles_prob = [
    (('S', ['X1', 'X1']), 0.41),
    (('S', ['m']), 0.20),
    (('S', ['S', 'X2']), 0.39),
    (('X1', ['S', 'X2']), 0.5),
    ...
]
```

En resum, la nostra representació es basa sempre en llistes de tuples, diferenciant el cas probabilístic del determinista amb la presència d'un segon element a la tupla (la probabilitat). Aquest sistema ens ha permès una gran flexibilitat a l'hora de generar, transformar gramàtiques.

Lectura de gramàtiques:
A més, hem desenvolupat un arxiu **utils.py** específic per llegir gramàtiques escrites en un format convencional. Un exemple típic d'aquest format seria:

```
S -> a | X A | A X | b
A -> R B
B -> A X | b | a
X -> a
R -> X B
```

Aquest arxiu lector processa línia per línia, identificant el símbol no terminal a l'esquerra de la fletxa i totes les possibles produccions a la dreta, separant-les per l'operador `|`. D'aquesta manera, podem treballar fàcilment amb fitxers de gramàtiques escrites de manera convencional, sense necessitat d'adaptar manualment l'entrada al format intern del nostre programa.

3 Explicació dels codis

3.1 Fitxer `main.py`

Aquest fitxer actua com a punt d'entrada principal al sistema, i serveix com a interfície d'usuari per provar i experimentar amb diferents funcionalitats relacionades amb el reconeixement de paraules a partir de gramàtiques. L'objectiu principal del projecte és analitzar si una paraula pot ser generada per una gramàtica lliure de context (CFG), en particular utilitzant la seva transformació a Forma Normal de Chomsky (CNF), i aplicar-hi l'algorisme CKY — tant en la seva versió clàssica com en la probabilística.

El fitxer organitza diferents escenaris d'execució que cobreixen diversos objectius:

3.1.1 Objectius generals del sistema

- Verificar si una paraula concreta pot ser generada per una gramàtica donada.
- Explorar el funcionament de l'algorisme CKY sobre gramàtiques en CNF.
- Entendre com transformar una CFG a CNF per tal que sigui apta per CKY.
- Analitzar la probabilitat d'una paraula sota una gramàtica probabilística.
- Generar de forma automàtica gramàtiques i paraules amb finalitats d'experimentació.

3.1.2 Execucions disponibles

El sistema ofereix quatre modes d'execució diferents, cadascun amb una finalitat pedagògica i experimental específica:

Execució bàsica (CKY sobre gramàtica CNF donada)

Aquesta execució aplica l'algorisme CKY clàssic sobre una gramàtica que ja ha estat escrita directament en Forma Normal de Chomsky (CNF). És útil per comprovar el funcionament directe de CKY quan ja disposem d'una gramàtica preparada. El que busquem és explorar com CKY analitza la derivació d'una paraula en CNF, sense haver de fer transformacions prèvies.

Execució amb transformació CFG → CNF

Aquest mode treballa amb una gramàtica original en forma de CFG (més expressiva i menys restringida), i la transforma a CNF abans d'aplicar CKY. Això reflecteix el cas pràctic habitual, on les gramàtiques es defineixen de forma lliure i cal adaptar-les abans d'utilitzar-les amb CKY. El que busquem és entendre el procés de conversió de CFG a CNF i veure com afecta el reconeixement de paraules.

Execució amb CKY probabilístic (PCFG)

Aquesta execució fa servir una versió probabilística de CKY per tractar amb gramàtiques probabilístiques (PCFGs). Cada regla de producció ve associada amb una probabilitat, i l'algorisme cerca la derivació amb màxima probabilitat per a la paraula donada. El que busquem és no només determinar si una paraula pot ser generada, sinó quantificar amb quina probabilitat, afavorint interpretacions més realistes i útils en contextos com el processament del llenguatge natural.

Execució aleatòria (experimentació)

Aquest mode és especialment útil per a la pràctica i l'anàlisi empírica. Permet generar automàticament gramàtiques, transformar-les si cal, i generar paraules que pertanyen o no al llenguatge definit per aquestes gramàtiques. El que busquem és facilitar l'experimentació massiva amb casos nous, per validar el comportament dels algorismes i identificar casos límit o d'error.

3.1.3 Altres funcionalitats

Generació d'exemples vàlids

El sistema també permet generar i mostrar exemples concrets de paraules que pertanyen al llenguatge d'una gramàtica seleccionada. Aquesta funcionalitat és útil per visualitzar què pot generar una gramàtica concreta i ajudar a entendre la seva estructura. El que busquem és proporcionar intuïció sobre el llenguatge generat per una gramàtica, especialment útil quan la gramàtica és complexa o generada aleatòriament.

3.1.4 Funció principal `main()`

Aquesta funció mostra un menú d'opcions per escollir quina execució es vol realitzar. Actua com a coordinador que dirigeix el flux cap a la funció corresponent. Aquesta estructura modular permet separar la lògica de cada cas i facilita tant la lectura com el manteniment del codi.

3.2 Fitxer `extensio_base.py`: Algorisme CKY clàssic

Aquest mòdul conté la classe `CKY`, que implementa l'algorisme de reconeixement CKY per a gramàtiques en Forma Normal de Chomsky. L'algorisme CKY és un mètode fonamental en el camp del processament de llenguatge natural i l'anàlisi sintàctica, ja que permet determinar si una paraula pot ser generada per una gramàtica lliure de context transformada a CNF.

3.2.1 Objectiu del mòdul

L'objectiu principal d'aquest fitxer és oferir una implementació funcional i eficient de CKY per a ser utilitzada sobre gramàtiques ja en CNF. Aquesta implementació es pot fer servir per:

- Verificar si una paraula donada pertany al llenguatge generat per la gramàtica.
- Avaluar diferents estratègies de generació i transformació de gramàtiques (des de CFG).
- Servir com a base per a extensions com CKY probabilístic.

3.2.2 Estructura de la classe `CKY`

- **Constructor** `__init__`: rep la llista de regles de producció en CNF i el símbol inicial. També comprova si aquest símbol inicial pot generar epsilon (cadena buida). Aquesta comprovació és rellevant per casos especials com paraules de longitud zero.
- **Mètode** `parse()`: és una interfície general que crida internament el mètode `parse_quiet()`.
- **Mètode** `parse_quiet()`: és la implementació concreta i silenciosa de l'algorisme CKY. Es basa en una taula bidimensional de dimensions $n \times n$, on n és la longitud de la paraula d'entrada. Cada cel·la conté els símbols no terminals que poden generar la subcadena corresponent.

3.2.3 Funcionament de l'algorisme

L'algorisme s'estructura en dues fases:

1. **Inicialització de la diagonal principal:** per a cada símbol terminal de la paraula, es comprova si alguna regla de producció pot generar-lo directament (és a dir, regles de la forma $A \to a$). Si és així, es guarda el no terminal A en la cel·la corresponent (i, i).

2. **Ompliment de la taula:** per a cada subcadena de longitud 2 fins a n, es combinen resultats parcials de subcadenes més curtes. Per cada partició de la subcadena $[i, j]$ en dos segments $[i, k]$ i $[k + 1, j]$, es comprova si existeixen regles binàries de la forma $A \to BC$ tals que B estigui a la posició (i, k) i C a $(k + 1, j)$. Si és així, s'afegeix A a la posició (i, j).

Finalment, es retorna si el símbol inicial es troba a la cel·la $(0, n - 1)$, que representa la paraula completa. Això indica si la paraula pot ser derivada completament des del símbol inicial de la gramàtica.

3.2.4 Casos especials tractats

L'algorisme contempla de manera explícita el cas d'una paraula buida. En aquest cas, retorna `True` si el símbol inicial pot derivar λ (epsilon), i `False` en cas contrari. Aquesta comprovació és important, ja que el CKY normalment no pot analitzar paraules de longitud 0 si no es tracta aquest cas fora de la taula.

3.3 Fitxer `extensio_1.py`: Conversió de CFG a CNF

Aquest mòdul conté la classe `CFGtoCNF`, que té com a objectiu transformar una gramàtica lliure de context (CFG) en Forma Normal de Chomsky (CNF). Aquesta transformació és necessària perquè l'algorisme CKY només pot operar sobre gramàtiques que estiguin en CNF, una forma restringida però equivalent pel que fa al poder generatiu.

3.3.1 Objectiu de la conversió

Tot i que una CFG pot expressar qualsevol llenguatge lliure de context, el format CNF imposa restriccions que permeten tractament algorítmic eficient. Concretament, en CNF totes les regles han de ser de la forma:

- $A \to BC$, on B i C són no terminals
- $A \to a$, on a és un símbol terminal
- (opcional) $S \to \lambda$, només si la paraula buida és part del llenguatge

Aquestes formes permeten que l'algorisme CKY pugui construir de manera eficient una taula de derivacions a partir de subcadenes.

3.3.2 Procés general de conversió

La conversió es realitza mitjançant una seqüència de passos successius que simplifiquen i normalitzen les regles:

1. **Afegir un nou símbol inicial (opcional)**: Si el símbol inicial apareix en alguna producció del costat dret, es crea un nou símbol inicial que el substitueixi. Això evita ambigüitats durant la derivació.

2. **Eliminació de produccions lambda (λ)**: Es detecten tots els símbols no terminals que poden derivar la cadena buida i s'actualitzen les produccions per tenir en compte totes les combinacions possibles on aquests símbols apareixen. Es garanteix que només el símbol inicial pugui tenir una producció λ, si escau.

3. **Eliminació de produccions unàries**: Aquestes són regles del tipus $A \to B$, on tant A com B són no terminals. Aquestes produccions s'eliminen substituint-les per les produccions de B, mantenint la capacitat generativa.

4. **Substitució de terminals en regles llargues**: En produccions de dues o més posicions, no es poden barrejar terminals amb no terminals. Així doncs, es creen nous símbols auxiliars (ex. $T\_A \to a$) per representar els terminals, i es fan servir en lloc del símbol terminal original.

5. **Descomposició de produccions llargues**: Les regles amb més de dos símbols al costat dret es descomponen en produccions binàries fent ús de nous no terminals auxiliars. Per exemple, una regla com $A \to BCD$ es transforma en $A \to BY1$ i $Y1 \to CD$.

3.3.3 Detalls importants del funcionament

- Es fa ús d'estructures com conjunts, diccionaris i codificació de produccions per evitar duplicats i assegurar consistència.
- L'ordre dels passos és crític: per exemple, eliminar λ abans d'eliminar produccions unàries, ja que aquestes poden dependre de símbols nul·lables.
- El mètode `convert()` encapsula tot el procés, aplicant totes les transformacions en l'ordre correcte i retornant la gramàtica ja convertida en CNF.

3.3.4 Aplicació pràctica

Aquest mòdul és essencial per permetre l'ús de CKY sobre gramàtiques generals (CFG). En contextos educatius o d'aprenentatge automàtic, sovint disposem de gramàtiques definides de forma més expressiva, i cal automatitzar aquesta conversió per integrar-les amb sistemes d'anàlisi sintàctica com CKY o les seves variants probabilístiques.

3.4 Fitxer `extensio_2.py`: Algorisme CKY probabilístic

Aquest mòdul implementa una extensió de l'algorisme CKY clàssic per tal de treballar amb **gramàtiques lliures de context probabilístiques** (PCFG). La classe `ProbabilisticCKY` adapta CKY perquè, a més de comprovar si una paraula pot ser derivada, calculi també la probabilitat màxima amb què aquesta paraula pot ser generada segons la gramàtica donada.

En moltes aplicacions pràctiques, com el processament del llenguatge natural, no n'hi ha prou amb saber si una frase és gramatical: volem saber *quines interpretacions són més probables*. Les PCFGs ens permeten assignar una probabilitat a cada producció, i així poder calcular, per una paraula donada, la derivació més probable.

L'algorisme de CKY probabilístic conserva l'estructura del CKY clàssic, però treballa amb **taules que acumulen probabilitats**, i calcula per cada subcadenes la millor derivació (és a dir, la més probable).

3.4.1 Descripció general de la classe

- `_build_rules_dict()`: Prepara la gramàtica en forma de diccionari, convertint la llista de tuples de la forma `((A, body), p)` en una estructura que permet accedir ràpidament a la probabilitat d'una producció concreta.
- `parse(word)`: Implementa l'algorisme CKY adaptat al cas probabilístic. Construeix una taula tridimensional on cada cel·la conté un diccionari amb les probabilitats màximes d'arribar a un símbol no terminal des de certa subcadena.

3.4.2 Funcionament de l'algorisme

El funcionament segueix l'esquema del CKY clàssic amb les següents diferències claus:

1. **Inicialització de la diagonal:** per cada símbol terminal de la paraula, es col·loquen a la taula totes les regles $A \to a$ que coincideixen, i es registra la seva probabilitat.
2. **Construcció de subestructures:** per cada subcadena de longitud ≥ 2, es consideren totes les possibles particions de la subcadena, i totes les regles binàries $A \to BC$. Si B i C poden derivar les parts respectives de la subcadena, es calcula la probabilitat del producte:

   $$P(A \to BC) \cdot P(B \Rightarrow \alpha) \cdot P(C \Rightarrow \beta)$$

   i es guarda només la màxima trobada.
3. **Resultat final:** la cel·la que cobreix tota la paraula (de 0 a n) conté les probabilitats màximes de generar-la des de cada símbol. Retornem la probabilitat associada al símbol inicial. Si no hi és, la paraula no pot ser generada (probabilitat zero).

3.4.3 Comportament davant paraules no generables

Si una paraula no pot ser generada per la gramàtica (és a dir, no hi ha cap seqüència de regles que la cobreixi), l'algorisme retorna `False`. Això facilita la seva integració amb altres sistemes que poden actuar diferent segons si una entrada és admissible o no.

3.4.4 Aplicació pràctica

Aquest algorisme és essencial per treballar amb models estadístics del llenguatge. Permet:

- Avaluar la probabilitat d'una frase.
- Seleccionar la derivació més probable entre moltes possibles.
- Integrar-se amb sistemes d'aprenentatge supervisat per estimar les probabilitats de les regles a partir de corpus etiquetats.

3.5 Fitxer `generador_gramatiques.py`: Generació automàtica de gramàtiques

Aquest mòdul conté la classe `GrammarMaker`, que permet generar automàticament gramàtiques lliures de context, tant en forma estàndard (CFG) com en Forma Normal de Chomsky (CNF). Aquesta funcionalitat és clau per a la realització d'experiments, tests i validacions dins el sistema, ja que permet crear múltiples gramàtiques de prova de manera aleatòria i controlada.

3.5.1 Objectiu del generador

Les gramàtiques generades serveixen per a:

- Provar l'algorisme CKY i el CKY probabilístic amb múltiples casos diferents.
- Simular entorns reals amb estructures gramaticals variades.
- Observar el comportament de derivacions llargues i recursives.
- Generar corpus sintètics amb o sense probabilitats per entrenament i test.

3.5.2 Estructura i estratègia de generació

El generador segueix una estratègia dissenyada per afavorir la **recursivitat i la complexitat estructural**, de manera que les paraules generades siguin més riques i el procés d'anàlisi sigui més significatiu.

- `__init__()`: Inicialitza les estructures internes: conjunt de no terminals, llista de regles i terminals usats.
- `crea_gramatica()`: Punt d'entrada principal. Permet indicar si la gramàtica ha d'estar en CNF i si ha de ser probabilística. Internament crida `crea_gramatica_recursiva()`.
- `crea_gramatica_recursiva()`: Genera la gramàtica a partir de regles inicials i va afegint regles amb preferència per produccions recursives o de longitud mitjana-alta. Es garanteix que tots els no terminals acabin tenint almenys una producció terminal.
- `_regla_cnf_recursiva()`: Crea regles binàries o terminals seguint les restriccions de la CNF, afavorint la reutilització de no terminals ja creats (per fomentar recursivitat).
- `_regla_cfg_recursiva()`: Crea regles amb 2 a 4 símbols al costat dret (termes terminals i no terminals), la qual cosa facilita la generació de paraules més llargues i variades en gramàtiques CFG.
- `_nou_no_terminal() i _nou_terminal()`: Generen nous símbols no terminals i terminals evitant repeticions.
- `Probabilitats (opcional)`: Si l'usuari sol·licita una gramàtica probabilística, es distribueixen probabilitats aleatòries normalitzades a les regles que tenen la mateixa capçalera. Això permet que la gramàtica es pugui usar amb `ProbabilisticCKY`.

3.5.3 Mecanismes de control i garantia estructural

Per assegurar la consistència i l'utilitat de les gramàtiques generades:

- Es controla el nombre mínim i màxim de regles.
- S'assegura que cada no terminal tingui almenys una derivació terminal.
- Es prioritza la creació de regles que mantinguin la recursió i diversitat.
- S'afegeixen explícitament algunes regles recursives addicionals com a reforç estructural.

3.5.4 Utilitat en el sistema

Aquest generador és fonamental per al mode d'**execució aleatòria** que permet:

- Crear gramàtiques CNF per CKY clàssic.
- Crear PCFGs per CKY probabilístic.
- Generar gramàtiques CFG per experimentar amb la conversió CFG $\rightarrow$ CNF.
- Estudiar com el disseny d'una gramàtica influeix en la derivació de paraules.

L'objectiu de la classe és doble:

- **Generar paraules vàlides** (és a dir, derivables a partir del símbol inicial) utilitzant una estratègia recursiva sobre les regles de la gramàtica.
- **Generar paraules invàlides**, a partir de la modificació controlada d'una paraula vàlida, per tal de garantir que no pertanyi al llenguatge.

Aquesta funcionalitat és especialment útil per fer experiments sistemàtics: verificar que l'algorisme reconeix les paraules correctes i rebutja les incorrectes.

3.5.5 Estrategia de generació

La generació es fa mitjançant una derivació recursiva des del símbol inicial:

- En cada pas, es tria una producció aleatòria que tingui com a capçalera el símbol actual.
- Es controla la **profunditat de la derivació** per evitar derivacions infinites (límit de profunditat).
- Es fa un seguiment de la **longitud de la paraula** per garantir que compleixi restriccions com longitud mínima o màxima.
- En funció de si es vol prioritzar llargada, es pot optar per produccions binàries (que deriven més llargament) o terminals (que acaben la derivació).

3.5.6 Funcionalitat de generació de paraules vàlides

La funció `crea_paraula()` intenta construir una paraula que sigui derivable segons la gramàtica:

- Prioritza paraules de longitud entre 3 i 5, que són més informatives per a l'anàlisi.
- Si no pot generar cap paraula d'aquestes longituds, fa intents addicionals amb qualsevol longitud vàlida.
- Garanteix que la paraula final compleixi les restriccions de longitud mínima i màxima.

3.5.7 Generació de paraules invàlides

Quan es vol generar una paraula que **no pertanyi** al llenguatge:

1. Primer es genera una paraula vàlida.
2. A continuació, es modifica lleugerament (canvi d'un caràcter o afegit d'un símbol no esperat).
3. Aquesta estratègia garanteix que la nova paraula estigui fora del llenguatge, però sigui semblant a una de vàlida, cosa que la fa útil per tests realistes.

3.5.8 Mètodes interns destacats

- `_construeix_amb_preferencia()`: construeix paraules intentant ajustar-se a una longitud objectiu.
- `_construeix()`: construeix paraules sense preferències de longitud, útil com a solució general.
- `_expandir_cos()`: aplica recursivament les produccions de cada part del costat dret de les regles.
- `_modificar_paraula()`: altera una paraula vàlida per trencar-ne la gramaticalitat, utilitzant inserció o substitució de caràcters.

3.5.9 Utilitat pràctica

Aquest mòdul es fa servir especialment en:

- L'experiment aleatori (opció 4 del `main.py`) per generar tant gramàtiques com paraules.
- La funció `mostra_exemples_pertanyents()`, que mostra exemples de paraules vàlides per a una gramàtica concreta.
- Validació de l'algorisme CKY amb entrada generada artificialment.

4 Experimentació

Abans de començar amb la experimentació, anem a comentar què és el que hem dut a terme. Hem creat un codi anomenat `experimentacio.py`. Aquest script s'encarrega d'executar totes les combinacions possibles dins de l'opció de generació aleatòria. És a dir, en quant a la generació de gramàtiques (probabilística o no), la conversió a forma normal de Chomsky (CNF) i la pertinença o no de la paraula generada al llenguatge corresponent.

Els resultats de cada execució es guarden en un fitxer `joc_de_proves.txt`, on bàsicament apareix tota la informació de cada experiment, la gramàtica original, la gramàtica transformada (si cal), la paraula generada i la sortida True o False. Aquestes experimentacions les hem fet sota dues llavors fixades **random.seed(1234) i random.seed(5678)**, per a poder assegurar que els experiments es puguin tornar a executar. és a dir, hem executat dos vegades cada experiment. El primer experiment de cada combinació el demostrarem amb la seva corresponent taula de programació dinàmica del CKY.

Abans de començar la experimentació, per a poder comprovar que les extensions anaven bé, vam provar d'utilitzar-les amb les gramàtiques donades en l'enunciat de la pràctica i una en cfg feta manualment i la paraula **bababa**.

Gramàtiques de proves bàsiques:

```
Gramàtica 1 (CNF):
S -> A B | C D | C B | S S
A -> B C | a
B -> S C | b
C -> D D | b
D -> B A

Gramàtica 2 (CNF):
S -> a | X A | A X | b
A -> R B
B -> A X | b | a
X -> a
R -> X B

Gramàtica 3 (CFG):
S -> A B C | a
A -> a |
B -> b | S
C -> c c

Gramàtica 4 (CFG):
S -> X Y Z | d
X -> x |
```

```
Y -> y | Z
Z -> z z

Gramàtica 5 (CNF + Probabilística):
S -> A B 0.25
S -> C D 0.25
S -> C B 0.25
S -> S S 0.25
A -> B C 0.5
A -> a 0.5
B -> S C 0.5
B -> b 0.5
C -> D D 0.5
C -> b 0.5
D -> B A 1.0
```

Resultats:

Gramàtica	**bababa**
Gramàtica 1	**TRUE**
Gramàtica 2	**FALSE**
Gramàtica 3	**FALSE**
Gramàtica 4	**FALSE**
Gramàtica 5	**TRUE**, prob = 0.00195

Table 1: Resultats per a la paraula `bababa` en cada gramàtica

4.1 Experiment 1

4.1.1 Prova 1

Opció	Valor
Probabilística	False
CNF	False
Paraula pertany	False

Table 2: Configuració de l'Experiment 1

Gramàtica original generada en CFG:

$$S \to d\ a \quad S \to X1\ c\ X1 \quad S \to l\ a \quad X1 \to S\ X2\ X3$$
$$X2 \to v\ s\ r \quad X2 \to u \quad S \to j \quad X1 \to p$$
$$X3 \to l \quad X1 \to X1\ u\ X2 \quad S \to S\ q\ X1$$

Gramàtica transformada a CNF:

$$S \to T_D\ T_A \quad S \to T_L\ T_A \quad X2 \to T_V\ Y3 \quad X2 \to u$$
$$S \to X1\ Y1 \quad X1 \to S\ Y2 \quad Y3 \to T_S\ T_R \quad S \to j$$
$$Y1 \to T_C\ X1 \quad Y2 \to X2\ X3 \quad X1 \to p \quad X3 \to l$$
$$X1 \to X1\ Y4 \quad Y4 \to T_U\ X2 \quad S \to S\ Y5 \quad Y5 \to T_Q\ X1$$
$$S \to X1\ Y6 \quad Y6 \to T_C\ X1 \quad X1 \to S\ Y7 \quad Y7 \to X2\ X3$$
$$X2 \to T_V\ Y8 \quad Y8 \to T_S\ T_R \quad X1 \to X1\ Y9 \quad Y9 \to T_U\ X2$$
$$S \to S\ Y10 \quad Y10 \to T_Q\ X1 \quad S\_ST \to T_D\ T_A \quad S\_ST \to X1\ Y11$$
$$Y11 \to T_C\ X1 \quad S\_START \to T_L\ T_A \quad S\_ST \to j \quad S\_ST \to S\ Y12$$
$$Y12 \to T_Q\ X1 \quad T_D \to d \quad T_A \to a \quad T_C \to c$$
$$T_L \to l \quad T_V \to v \quad T_S \to s \quad T_R \to r$$
$$T_U \to u \quad T_Q \to q$$

Paraula generada que no pertany: **lwq**
Taula CKY per a comprovar si realment no pertany a la gramàtica:

Figure 1: Taula CKY

Sortida del programa: **FALSE**

Com es pot veure a la taula CKY, el símbol inicial S no apareix a la casella de dalt a la dreta, ni en cap casella rellevant de la taula. Això ens diu que la paraula `lwq` no pot ser generada per la gramàtica, per això retorna FALSE.

4.1.2 Prova 2

Gramàtica original generada en CFG:

$S \to a\ q$	$S \to X1\ p\ X1$	$S \to X1\ X2$	$X2 \to X3\ S\ S$
$X2 \to S\ X4\ p\ l$	$X4 \to u$	$X2 \to h$	$S \to j$
$X3 \to d$	$X1 \to l$	$X1 \to X1\ p\ X3$	$X1 \to X1\ n\ S$

Gramàtica transformada a CNF:

$S \to T_A\ T_Q$	$S \to X1\ Y1$	$Y1 \to T_P\ X1$	$S \to X1\ X2$
$X2 \to X3\ Y2$	$Y2 \to S\ S$	$X2 \to S\ Y3$	$Y3 \to X4\ Y4$
$Y4 \to T_P\ T_L$	$X4 \to u$	$X2 \to h$	$S \to j$
$X3 \to d$	$X1 \to l$	$X1 \to X1\ Y5$	$Y5 \to T_P\ X3$
$X1 \to X1\ Y6$	$Y6 \to T_N\ S$	$S \to X1\ Y7$	$Y7 \to T_P\ X1$
$X2 \to X3\ Y8$	$Y8 \to S\ S$	$X2 \to S\ Y9$	$Y9 \to X4\ Y10$
$Y10 \to T_P\ T_L$	$X1 \to X1\ Y11$	$Y11 \to T_P\ X3$	$X1 \to X1\ Y12$
$Y12 \to T_N\ S$	$S_{START} \to T_A\ T_Q$	$S_{START} \to X1\ Y13$	$Y13 \to T_P\ X1$
$S_{START} \to X1\ X2$	$S_{START} \to j$	$T_A \to a$	$T_Q \to q$
$T_P \to p$	$T_L \to l$	$T_N \to n$	

Paraula generada que no pertany: **pq**

Sortida del programa: **FALSE**

Tot i que la paraula generada, **pq**, conté només símbols terminals que apareixen a les produccions de la gramàtica, no hi ha cap regla que generi les regles **T_P i T_Q** , que són les que es necessitarien per a generar la paraula pq. Per tant, queda justificat que **pq** no pertany al llenguatge generat per aquesta gramàtica.

4.2 Experiment 2

4.2.1 Prova 1

Opció	Valor
Probabilística	False
CNF	False
Paraula pertany	True

Table 3: Configuració de l'Experiment 1

Gramàtica original generada en CFG:

$S \to f\ i$ $\quad X2 \to n$ $\quad X1 \to y$ $\quad S \to S\ u\ X1$
$S \to X1\ e\ X1$ $\quad S \to n$ $\quad X2 \to X2\ p\ S$
$X1 \to S\ S$ $\quad X1 \to p$
$S \to X1\ X2\ X1$

Gramàtica transformada a CNF:

$S \to T_F\ T_I$ $\quad X1 \to S\ S$ $\quad X2 \to n$ $\quad S \to S\ Y4$
$S \to X1\ Y1$ $\quad S \to X1\ Y2$ $\quad S \to n$ $\quad S \to X1\ Y5$
$Y1 \to T_E\ X1$ $\quad Y2 \to X2\ X1$ $\quad X1 \to y$ $\quad Y4 \to T_U\ X1$
$S \to X1\ Y6$ $\quad X2 \to X2\ Y3$ $\quad Y3 \to T_P\ S$ $\quad Y5 \to T_E\ X1$
$Y6 \to X2\ X1$ $\quad X2 \to X2\ Y7$ $\quad Y7 \to T_P\ S$ $\quad S \to S\ Y8$
$Y8 \to T_U\ X1$ $\quad S\_ST \to T_F\ T_I$ $\quad S\_ST \to X1\ Y9$ $\quad Y9 \to T_E\ X1$
$S\_ST \to X1\ Y10$ $\quad Y10 \to X2\ X1$ $\quad S\_ST \to n$ $\quad S\_ST \to S\ Y11$
$Y11 \to T_U\ X1$ $\quad T_F \to f$ $\quad T_I \to i$ $\quad T_E \to e$
$T_P \to p$ $\quad T_U \to u$

Paraula generada que pertany: **fiyeyenn**

Taula CKY per a comprovar si realment pertany a la gramàtica:

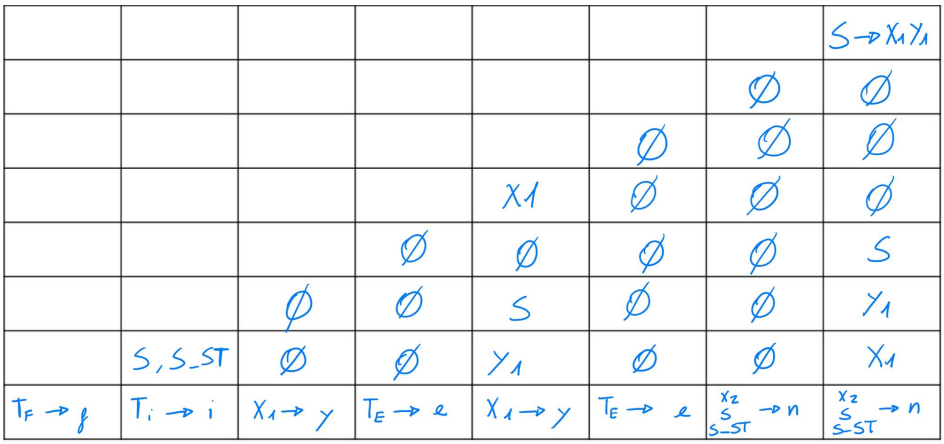

Figure 2: Taula CKY

Sortida del programa: **TRUE**

Amb la taula CKY, veiem que el símbol inicial S apareix a la casella superior dreta de la taula. Això ens diu que la paraula que hem mirat pot ser generada a partir del símbol inicial segons la gramàtica. Per tant, **la paraula pertany al llenguatge**.

4.2.2 Prova 2

Gramàtica original generada en CFG:

$$S \to x\ h \quad S \to X1\ p\ X1 \quad S \to o\ X1 \quad X1 \to S\ u\ v$$
$$X1 \to k \quad S \to p \quad S \to S\ h\ X1 \quad S \to S\ k\ X1$$

Gramàtica transformada a CNF:

$$S \to T_X\ T_H \quad S \to X1\ Y1 \quad Y1 \to T_P\ X1 \quad S \to T_O\ X1$$
$$X1 \to S\ Y2 \quad Y2 \to T_U\ T_V \quad X1 \to k \quad S \to p$$
$$S \to S\ Y3 \quad Y3 \to T_H\ X1 \quad S \to S\ Y4 \quad Y4 \to T_K\ X1$$
$$S \to X1\ Y5 \quad Y5 \to T_P\ X1 \quad X1 \to S\ Y6 \quad Y6 \to T_U\ T_V$$
$$S \to S\ Y7 \quad Y7 \to T_H\ X1 \quad S \to S\ Y8 \quad Y8 \to T_K\ X1$$
$$S_{\text{START}} \to T_X\ T_H \quad S_{\text{START}} \to X1\ Y9 \quad Y9 \to T_P\ X1 \quad S_{\text{START}} \to T_O\ X1$$
$$S_{\text{START}} \to p \quad S_{\text{START}} \to S\ Y10 \quad Y10 \to T_H\ X1 \quad S_{\text{START}} \to S\ Y11$$
$$Y11 \to T_K\ X1 \quad T_X \to x \quad T_H \to h \quad T_P \to p$$
$$T_O \to o \quad T_U \to u \quad T_V \to v \quad T_K \to k$$

Paraula generada que pertany: **okhkkpuv**

Sortida del programa: **TRUE**

La paraula **okhkkpuv** pertany a la gramàtica, ja que a l'aplicar l'algorisme CKY amb la taula corresponent, hem comprovat manualment que és possible construir aquesta paraula a partir de les regles donades. La sortida del programa reforça aquest resultat, retornant **TRUE**.

4.3 Experiment 3

4.3.1 Prova 1

Opció	Valor
Probabilística	False
CNF	True
Paraula pertany	False

Table 4: Configuració de l'Experiment 1

Gramàtica original generada en CNF:

$$S \to X1\ X1 \quad S \to X2\ X2 \quad X2 \to w \quad X2 \to X2\ S$$
$$S \to q \quad S \to X1\ X2 \quad X1 \to n \quad S \to S\ X2$$

Paraula generada que no pertany: **nf**
Taula CKY per a comprovar si realment no pertany a la gramàtica:

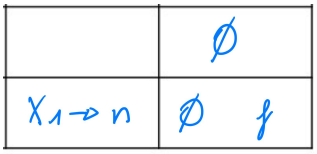

Figure 3: Taula CKY

Sortida del programa: **FALSE**

En analitzar la taula CKY, la casella superior dreta, està buida. Això implica que no podem obtenir la paraula a partir de cap derivació vàlida del símbol inicial de la gramàtica. Per aquest motiu, podem concloure que **la paraula no forma part del llenguatge generat per la gramàtica**.

4.3.2 Prova 2

Gramàtica original generada en CNF:

$$S \to X1\ X1 \quad S \to p \quad S \to X1\ X2 \quad S \to X2\ X1$$
$$X1 \to n \quad X2 \to p \quad S \to S\ X2 \quad S \to S\ X1$$

Paraula generada que no pertany: **nu**
Taula CKY per a comprovar si realment no pertany a la gramàtica:

Sortida del programa: **FALSE**

4.4 Experiment 4

4.4.1 Prova 1

Opció	Valor
Probabilística	False
CNF	True
Paraula pertany	True

Table 5: Configuració de l'Experiment 4

Gramàtica original generada en CNF:

$$S \to X1\ X1 \quad X2 \to X1\ S \quad X1 \to c \quad X2 \to p$$
$$S \to q \quad S \to X2\ X1 \quad X1 \to X1\ X2$$
$$S \to X1\ X2$$

Paraula generada que no pertany: **cpcqcpcq**

Taula CKY per a comprovar si realment pertany a la gramàtica:

							X_2, S
						S	$\emptyset$
					X_2, S	$\emptyset$	S
				X_2, S	$\emptyset$	$\emptyset$	$\emptyset$
			S, X_1	$\emptyset$	S	$\emptyset$	S, X_1
		X_2, S	$\emptyset$	S	$\emptyset$	X_2, S	$\emptyset$
	S, X_1	S	X_2	$\emptyset$	S, X_1	S	X_2
$X_1 \to c$	$X_2 \to p$	$X_1 \to c$	$S \to q$	$X_1 \to c$	$X_2 \to p$	$X_1 \to c$	$S \to q$

Figure 4: Taula CKY feta a mà per comprovar si la paraula pertany

Sortida del programa: **TRUE**

En la casella de a dalt de tot de la taula, trobem el símbol inicial S. Per tant, podem afirmar que **la paraula pertany al llenguatge de la gramàtica**.

4.4.2 Prova 2

Gramàtica original generada en CNF:

$$S \to X1\ X1 \quad S \to r \quad X1 \to S\ X2 \quad S \to X1\ X2$$
$$X1 \to d \quad X1 \to t \quad X1 \to X3\ X3 \quad X3 \to d$$
$$X2 \to g \quad S \to S\ X3 \quad S \to S\ X1$$

Paraula generada que no pertany: **rgddd**

Sortida del programa: **TRUE**

Com es pot veure a la taula, la paraula **rgddd** pot ser derivada segons les regles de la gramàtica, de manera que el programa retorna correctament **TRUE**.

4.5 Experiment 5

4.5.1 Prova 1

Opció	Valor
Probabilística	True
CNF	True
Paraula pertany	False

Table 6: Configuració de l'Experiment 4

Gramàtica original generada en CNF amb probabilitats:

$S \to X1\ X1\ \ (0.16)$ $X1 \to S\ X2\ \ (0.72)$ $X2 \to n\ \ (0.26)$ $X3 \to v\ \ (1.00)$
$S \to b\ \ (0.23)$ $X1 \to k\ \ (0.28)$ $X2 \to X2\ X1\ \ (0.74)$
$S \to X3\ X3\ \ (0.32)$
$S \to S\ X3\ \ (0.30)$

Paraula generada que no pertany: **cv**

Taula CKY per a comprovar si realment no pertany a la gramàtica:

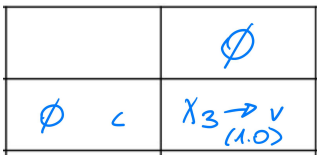

Figure 5: Taula CKY

Sortida del programa: **FALSE, Probabilitat = 0**

En observar la taula probabilística, comprovem que la casella superior dreta no conté el símbol inicial S. De fet, només apareix $X_3 \to v$ amb una probabilitat de 1.0, però no s'obté cap derivació des de S que cobreixi tota la paraula. Per tant, podem concloure que la paraula no pertany al llenguatge generat per la gramàtica.

4.5.2 Prova 2

Gramàtica original generada en CNF amb probabilitats:

$S \to X1\ X1\ \ (0.25)$ $S \to u\ \ (0.24)$ $S \to X3\ X1\ \ (0.19)$ $S \to S\ X1\ \ (0.31)$
$X1 \to S\ X2\ \ (0.56)$ $X1 \to X3\ X3\ \ (0.42)$ $X1 \to o\ \ (0.02)$
$X3 \to s\ \ (1.00)$ $X2 \to w\ \ (0.95)$ $X2 \to X2\ X3\ \ (0.05)$

Paraula generada que no pertany: **gssuww**

Taula CKY per a comprovar si realment no pertany a la gramàtica:
Sortida del programa: **FALSE, Probabilitat = 0**

4.6 Experiment 6

4.6.1 Prova 1

Opció	Valor
Probabilística	True
CNF	True
Paraula pertany	True

Table 7: Configuració de l'Experiment 6

Gramàtica original generada en CNF amb probabilitats:

$$S \to X1\ X1 \quad (0.41) \qquad X1 \to S\ X2 \quad (0.50) \qquad X2 \to X1\ S \quad (0.23) \qquad S \to S\ X2 \quad (0.39)$$
$$S \to m \quad (0.20) \qquad X1 \to z \quad (0.50) \qquad X2 \to x \quad (0.24)$$
$$X2 \to X2\ X1 \quad (0.53)$$

Paraula generada que pertany: **mxxzxzmx**

Taula CKY per a comprovar si realment pertany a la gramàtica:

							$S \to S\ X_2$ $1{,}56 \cdot 10^{-7}$
						S $4\cdot10^{-9}$ | X_1 $6\cdot10^{-7}$ | X_2 $2\cdot10^{-6}$	$\emptyset$
					S $1\cdot10^{-5}$ | X_1 $2\cdot10^{-8}$ | S $1\cdot10^{-5}$	$\emptyset$	$\emptyset$
				S $5\cdot10^{-5}$ | X_1 $6\cdot10^{-5}$	$\emptyset$	$\emptyset$	$\emptyset$
			X_1 $6\cdot10^{-4}$ | S $5\cdot10^{-4}$ | S $5\cdot10^{-4}$	$\emptyset$	$\emptyset$	$\emptyset$	X_2 $8\cdot10^{-4}$
		S, 0.0019 | X_1 0.0023	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	X_2 0,002 | S 0,004
	S 0,019 , X_1 0,024	$\emptyset$	X_2 0,0636	$\emptyset$	X_2 0,0636	X_2 0,023	S 0,019, X_1 0,024
$S \to m$ (0.2)	$X_2 \to x$ 0.24	$X_2 \to x$ 0.24	$X_1 \to z$ 0.5	$X_2 \to x$ 0.24	$X_1 \to z$ 0.5	$S \to m$ 0.2	$X_2 \to x$ 0.24

Figure 6: Taula CKY

Sortida del programa: **TRUE, Probabilitat = 1.4649958723645444e-07**

En analitzar la taula CKY probabilística per a la paraula donada, observem que a la darrera casella, en la que només hem anotat la derivació que proporciona la probabilitat més alta, corresponent a la regla $S \to S\ X_2$ amb una probabilitat de $1{,}56 \cdot 10^{-7}$. Aquesta probabilitat final pràcticament coincideix amb la que retorna el programa ($1{,}4649958723645444 \cdot 10^{-7}$), de manera que ens confirma que el càlcul manual és correcte. Per tant, podem concloure que la paraula pertany al llenguatge generat per la gramàtica, amb la probabilitat indicada.

4.6.2 Prova 2

Gramàtica original generada en CNF amb probabilitats:

$S \to X1\ X1$ (0.0100)	$S \to m$ (0.3400)	$S \to X1\ X2$ (0.3400)	$S \to X2\ X1$ (0.0300)
$S \to w$ (0.2700)	$X1 \to X2\ S$ (0.2300)	$X1 \to v$ (0.5300)	$X1 \to X1\ S$ (0.2500)
$X2 \to S\ X1$ (0.4400)	$X2 \to q$ (0.5100)	$X2 \to X2\ X1$ (0.0500)	

Paraula generada que pertany: **vwqvwvvw**

Sortida del programa: **TRUE, Probabilitat = 5.096429055943536e-08**

La paraula generada **vwqvwvvw** pertany a la gramàtica. A més, la probabilitat calculada pel CKY probabilístic és de **5.096429055943536e-08**.

5 Conclusions

En primer lloc, hem vist que l'algorisme CKY és realment fiable per determinar la pertinença d'una paraula al llenguatge generat per una gramàtica, sempre que aquesta es trobi en forma normal de Chomsky. La taula de programació dinàmica que genera l'algorisme permet justificar formalment qualsevol decisió, cosa que és un avantatge tant a nivell d'anàlisi com de depuració.

D'altra banda, la transformació de CFG a CNF, tot i ser teòricament senzilla, en la pràctica pot complicar considerablement la gramàtica original, generant una gran quantitat de regles auxiliars i nous símbols no terminals. Això obliga a ser rigorosos tant en la interpretació de la gramàtica resultant com en la definició del símbol inicial a l'hora d'aplicar CKY, ja que una mínima confusió pot portar a errors que no haurien d'aparèixer.

A nivell d'implementació, la modularitat del codi s'ha demostrat clau per poder experimentar amb diversos tipus de gramàtiques i paraules. El fet de disposar de mòduls específics per generar gramàtiques, transformar-les i generar paraules (tant vàlides com no vàlides) ens ha permès automatitzar tot el procés de proves i explorar molts casos diferents, cosa que seria inviable manualment.

A més, la incorporació de la versió probabilística de CKY ens ha permès no només avaluar la pertinença d'una paraula, sinó també calcular la probabilitat de la derivació més probable depenent de la gramàtica donada.

Finalment, després de realitzar aquesta pràctica podem dir que l'ús combinat de l'algorisme CKY, la transformació a CNF i les eines de generació automàtica de gramàtiques i paraules constitueix una base sòlida i flexible per a l'anàlisi de llenguatges formals. Aquesta experiència ens ha permès entendre que la clau està en l'automatització i la rigorositat en tot el procés, aspectes essencials tant per a l'aprenentatge com per a aplicacions pràctiques reals en el camp del processament de llenguatge.

6 Bibliografia

- Noe Hernandez. (2020, 4 diciembre). 06. Forma Normal de Chomsky. Algoritmo CKY [Vídeo]. YouTube. https://www.youtube.com/watch?v=K1vuHCHGgBs
- Colaboradores de Wikipedia. (2025, 22 mayo). Gramática libre de contexto probabilística. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/Gram3A1ticalibredecontextoprobabilC3ADstica
- Jorge, C. R., C, C. J. R. (2010). Gramáticas probabilísticas para la desambiguación sintáctica. Dialnet. https://dialnet.unirioja.es/servlet/tesis?codigo=66853
- Contributors to Wikimedia projects. (2024, 2 enero). Forma normal de Chomsky. Viquipèdia, L'enciclopèdia Lliure. https://ca.wikipedia.org/w/index.php?title=Forma_normal_de_Chomskyoldid=32888197
- Turmo Borràs, J. (s.f.). Parsing: Constituents. Universitat Politècnica de Catalunya. Recuperat de https://www.cs.upc.edu/ turmo/plh/lectures/08-parsing-constituents.pdf