

Teoría de Autómatas y Lenguajes Formales

Unidad 2 — Sesión práctica: Derivación vs árbol + ambigüedad + precedencia (AP2 P-C1)

Docente: Helder Octavio Fernández Guzmán

- Revisión operativa: **derivación** vs **árbol de derivación**.
- Leftmost vs rightmost: regla práctica (qué se expande).
- Evidencia de **ambigüedad**: una cadena con **dos árboles distintos**.
- Desambiguación: fijar **precedencia** (y asociatividad cuando sea necesario) con niveles (E,T,F).
- Trabajo del estudiante: **AP2 P-C1** (4 etapas) en Moodle.

Recordatorio mínimo: evidencia (IN / NOT IN)

Periodo 1/6

- Derivación (un paso): si $A \rightarrow \alpha$, entonces $xAy \Rightarrow x\alpha y$.
- Derivación (cero o más pasos): \Rightarrow^* .
- **IN** requiere evidencia: derivación o árbol.
- **NOT IN** requiere argumento estructural breve (2–4 líneas).

Derivación vs Árbol (idea clave)

Periodo 1/6

- **Derivación:** secuencia de **formas sentenciales** (procedimiento paso a paso).
- **Árbol:** estructura jerárquica que muestra **cómo se agrupan** expansiones.
- Un mismo árbol puede leerse con derivación leftmost o rightmost.
- Si hay **dos árboles distintos** para la misma cadena, hay **ambigüedad**.

Derivación: $S \Rightarrow^* w$

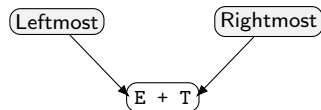
Árbol de derivación

Ambos sirven como evidencia de pertenencia (IN)

Leftmost vs Rightmost (regla operativa)

Periodo 1/6

- **Leftmost (izquierdista):** expandir siempre el **no terminal** más a la izquierda.
- **Rightmost (derechista):** expandir siempre el **no terminal** más a la derecha.
- Mismo objetivo (llegar a terminales), distinto **orden de expansión**.



Forma sentencial: mezcla terminales y no terminales

Leftmost: expandir primero el no terminal más a la izquierda

Rightmost: expandir primero el no terminal más a la derecha

Ejemplo de práctica: operadores + y *

Periodo 2/6

Usaremos un ejemplo de práctica para revisar **ambigüedad** y **precedencia**:

Gramática ambigua G_a^{+*}

$$E \rightarrow E + E \mid E * E \mid (E) \mid N$$

$$N \rightarrow D \mid D N$$

$$D \rightarrow 0 \mid 1 \mid \dots \mid 9$$

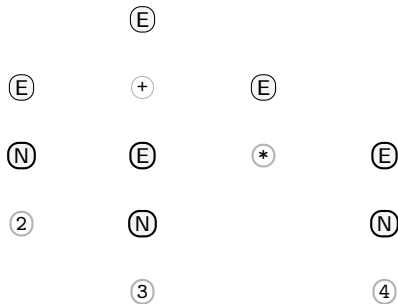
Cadena fija para evidenciar ambigüedad

$$w = 2+3*4$$

Enfocaremos la evidencia formal (árbol/derivación), no el “significado”.

Ambigüedad: Árbol 1 para $2+3*4$

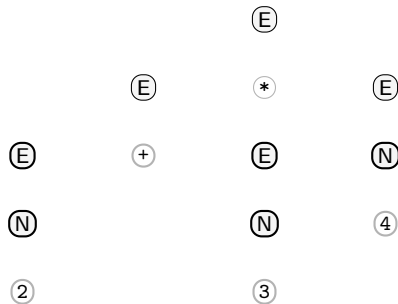
Periodo 2/6



Lectura: raíz +; el * queda en el subárbol derecho.

Ambigüedad: Árbol 2 para $2+3*4$

Periodo 2/6



Lectura: raíz *; el + queda en el subárbol izquierdo.

Conclusión: evidencia de ambigüedad

Periodo 2/6

Criterio práctico

Una CFG es **ambigua** si existe una cadena w con **dos árboles de derivación distintos**.

- Para $w = 2+3*4$ mostramos **dos árboles diferentes**.
- Por tanto, G_a^{+*} es **ambigua**.

Desambiguación por precedencia (niveles)

Periodo 3/6

- Para fijar precedencia, organizamos la gramática en **niveles** (no terminales distintos).
- Idea típica: E (suma) \rightarrow T (producto) \rightarrow F (número o paréntesis).
- En este ejemplo: * se agrupa antes que +.

E: combina con +



T: combina con *



F: número o paréntesis

La estructura queda determinada por el nivel

Ejemplo: gramática corregida G_c^{+*} (no ambigua)

Periodo 3/6

Por niveles (sin usar primas)

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid N$$

$$N \rightarrow D \mid D N$$

$$D \rightarrow 0 \mid 1 \mid \dots \mid 9$$

Idea: T resuelve multiplicaciones ($*$) antes de que E combine con $+$.

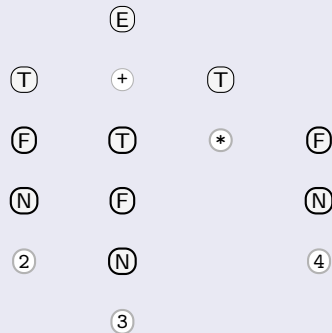
Derivación (secuencia) \leftrightarrow Árbol (estructura)

Periodo 3/6

Derivación (conceptual, leftmost)

$E \Rightarrow E + T$
 $\Rightarrow T + T$
 $\Rightarrow F + T$
 $\Rightarrow N + T$
 $\Rightarrow 2 + T$
 $\Rightarrow 2 + T * F$
 $\Rightarrow 2 + 3 * 4$

Árbol de derivación (misma estructura)



- Actividad en Moodle: **AP2 P-C1 — CFG: precedencia, ambigüedad y corrección.**
- Herramienta: **UC Davis Automaton Simulator (CFG).**
- En el simulador: evita no terminales con prima (como E'); usa letras como E,T,F,

Mapa de trabajo (entregables por etapa)

- 1 Etapa 1: demostrar **ambigüedad** con 2 árboles distintos.
- 2 Etapa 2: proponer una **gramática corregida** con precedencia > sobre | + justificación (3–5 líneas).
- 3 Etapa 3: con la gramática corregida: 1 derivación **leftmost** + 1 **rightmost** + 1 árbol.
- 4 Etapa 4: 2 IN + 2 NOT IN + argumento breve + **capturas** (1 IN y 1 NOT IN).

Plantilla de cabecera (al inicio del archivo .cfg de cada etapa)

```
---
Autor: "Apellido, Nombre"
Curso: "TAyLF - Intensivo 2026"
Actividad: "AP2 P-C1"
Etapa: "X"
Tool: "UC Davis Automaton Simulator (CFG)"
---
```

Checklist rápido:

- Cada etapa incluye su YAML.
- Etapa 4 incluye 2 capturas legibles (1 IN y 1 NOT IN).
- NOT IN lleva argumento estructural breve (2–4 líneas).

- **Min 0–15:** Etapa 1 lista (dos árboles distintos).
- **Min 15–30:** Etapa 2: gramática corregida + justificación.
- **Min 30–40:** Etapa 3: leftmost + rightmost + árbol.
- **Min 40–45:** Etapa 4: IN/NOT IN + argumento + capturas mínimas.

Checklist final antes de enviar

- ☐ YAML en cada etapa
 - ☐ Etapa 1: 2 árboles distintos
 - ☐ Etapa 3: leftmost + rightmost + 1 árbol
 - ☐ Etapa 4: 2 IN + 2 NOT IN + 2 capturas