Teoría de Autómatas y Lenguajes Formales

Unidad 2 — Sesión 11: Derivaciones (izq./der.) + Árbol de derivación + práctica + quiz

Docente: Helder Octavio Fernández Guzmán

Sesión 11: ruta del día

Periodo 1/6

- Definiciones operativas: derivación **izquierdista** vs **derechista**.
- CFG central para `id+id+...+id` (sin + final).
- Ejemplo guiado: derivaciones para `id+id+id`.
- Árbol de derivación (parse tree) del mismo ejemplo.
- Quiz de cierre (Moodle): **10 preguntas** (8–10 min).
- Actividad individual (mixta): cuaderno + simulador.

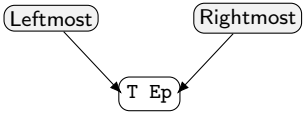
Recordatorio mínimo: CFG + evidencia

Periodo 1/6

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

Izquierdista vs derechista (idea operativa)

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



Forma sentencial: contiene terminales y no terminales

`T` es no terminal (se expande a `id`)

`Ep` representa E' (cola), y puede terminar con ϵ

CFG central (notación teórica con E')

Lenguaje: una o más `id` separadas por +, sin + final.

$$E \to T\ E'$$
$$E' \to +\ T\ E' \mid \epsilon$$
$$T \to id$$

Lectura: E' es la "cola" de la expresión: o agrega `+id` y continúa, o termina (ϵ).

Nota de herramienta: en el simulador usaremos `Ep` en lugar de E'.

Ejemplo guiado: cadena objetivo

Cadena objetivo:

$$w = \texttt{id+id+id}$$

Plan:

- Derivación **izquierdista** completa.
- Derivación **derechista** completa (misma cadena).
- Árbol de derivación del mismo w.

Derivación izquierdista (leftmost): pasos 1–4

$$
\begin{aligned}
E &\Rightarrow T\ E' \\
&\Rightarrow id\ E' \\
&\Rightarrow id + TE' \\
&\Rightarrow id + id\ E'
\end{aligned}
$$

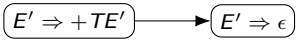
Regla aplicada: siempre expandimos el **no terminal más a la izquierda**.

Observación: E' aún no termina; puede seguir agregando `+id`.

Derivación izquierdista (leftmost): pasos 5–7

$$
\begin{aligned}
id + id\, E' &\Rightarrow id + id + TE' \\
&\Rightarrow id + id + id\, E' \\
&\Rightarrow id + id + id
\end{aligned}
$$

Paso clave: el cierre usa $E' \Rightarrow \epsilon$.



"E' agrega otro `+id` o termina"

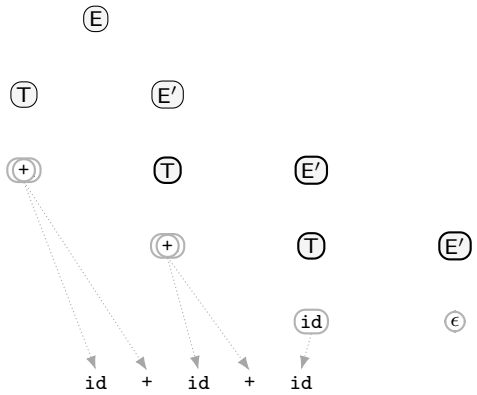
Derivación derechista (rightmost): pasos completos

$$
\begin{aligned}
E &\Rightarrow T\,E' \\
&\Rightarrow T + TE' \\
&\Rightarrow T + T + TE' \\
&\Rightarrow T + T + T \\
&\Rightarrow T + T + id \\
&\Rightarrow T + id + id \\
&\Rightarrow id + id + id
\end{aligned}
$$

Regla aplicada: siempre expandimos el **no terminal más a la derecha**.

Idea: mismo resultado final; el orden de expansión cambia.

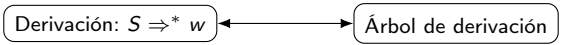
Árbol de derivación y *yield* para `id+id+id`



Lectura: el árbol describe la estructura; el *yield* concatena terminales (izq→der).

Derivación y árbol: misma evidencia, distinto enfoque

- **Derivación:** procedimiento paso a paso para llegar a terminales.
- **Árbol:** estructura jerárquica que explica cómo se construye la cadena.



Ambos sirven como evidencia de pertenencia (IN)

Quiz de cierre (antes de la actividad)

Periodo 4/6

- Duración: **10 minutos**.
- Total: **10 preguntas**.
- Cubre: notación ($\Rightarrow, \Rightarrow^*, \epsilon$), leftmost/rightmost, e IN/NOT IN rápidos con justificación mínima.

Revisión express: errores típicos (2–3 minutos)

Periodo 4/6

- Olvidar que E' puede terminar con ϵ (la cadena se queda "colgando").
- Aceptar `id+` por accidente (no debe ser posible).
- Confundir evidencia: **IN** sin derivación/árbol; **NOT IN** sin argumento estructural.

Actividad individual (mixta): cuaderno + simulador

Parte A (cuaderno): para `id+id`

- 1 derivación **izquierdista** completa.
- 1 árbol de derivación (puede ser simple).

Parte B (simulador): 2 capturas legibles

- **IN:** `id+id`
- **NOT IN:** `id+` (o `+id`)

Checklist de evidencia

☐ Foto del cuaderno (derivación + árbol)

☐ Captura IN en simulador

☐ Captura NOT IN en simulador

Simulador UC Davis: YAML listo para copiar (usa Ep)

CFG en YAML

```
# CFG: id(+id)*
E:  [TEp]
Ep: ['+TEp', '']
T:  [id]
```

Notas:

- Ep corresponde a E' (solo cambia el nombre).
- La cadena vacía se escribe como ''.
- Si un símbolo entra en conflicto con YAML (p.ej. ,), usa comillas.

E' (notación) ⟷ Ep (simulador)

Cierre: checklist de calidad (evidencia y coherencia)

Periodo 6/6

Antes de cerrar, verifica:

- La derivación termina en **solo terminales** (no quedan no terminales).
- **IN** tiene evidencia: derivación o árbol.
- **NOT IN** tiene argumento estructural breve (2–3 líneas).
- La CFG no permite + final (por ejemplo, `id+` debe ser NOT IN).

Ambigüedad: una cadena, dos árboles distintos

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

Ejemplo (ambigua):

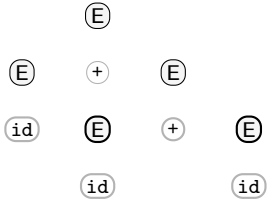
$$E \to E + E \mid id$$

Árbol A

$(id + id) + id$

Árbol B

$id + (id + id)$



Ambos generan `id+id+id`, pero con distinta estructura (asociatividad).

Idea clave: la gramática no fuerza una agrupación única; por eso aparecen dos árboles.