Traducción Dirigida por Sintaxis

CI4721 – Lenguajes de Programación II

Ernesto Hernández-Novich
<emhn@usb.ve>

Universidad "Simón Bolívar"

Copyright ©2012-2016

UNIVERSIDAD SIMÓN BOLÍVAR

Definición Dirigida por Sintaxis

Una **Definición Dirigida por Sintaxis** (*Syntax-Directed Definition* o SDD por sus siglas en inglés) es la combinación de:

- Una $G = (N, \Sigma, P, S)$ libre de contexto.
- Un conjunto de **atributos** asociados a los símbolos de la gramática.
  - El símbolo $X \in (N \cup \Sigma)$ puede tener cero o más atributos asociados.
  - De cualquier tipo conveniente: números, cadenas, valores de verdad, tablas, referencias, conjuntos, ...
  - Usaremos $X.a$ para denotar que a es uno de los atributos de X.
- Un conjunto de **reglas semánticas** asociadas a las producciones.
  - También se les llama **acciones** o **reglas de etiquetado**.
  - Establecen dependencias entre atributos de los símbolos.
  - Sin efectos de borde – **Gramática de Atributos**.
  - Con efectos de borde – **Esquemas de Traducción**.

UNIVERSIDAD SIMÓN BOLÍVAR

Tipos de Atributos

Cada $A \to \alpha$ tiene asociado un conjunto de reglas semánticas de la forma

$$b \leftarrow f(c_1, c_2, \ldots, c_k)$$

donde f es una función

- b es **sintetizado** para A si todos los c_i pertenecen a símbolos en α.
- b es **heredado** para B si B ocurre en α y los c_i pertenecen a *cualquier* símbolo de la producción.
- Diremos que b *depende* de los c_i.
- Los terminales sólo tienen atributos sintetizados.
  - También se les llama **intrínsecos**.
  - Su valor depende del lexema particular.

UNIVERSIDAD SIMÓN BOLÍVAR

Expresiones, de nuevo

Para reconocimiento ascendente

$$
\begin{array}{ll}
S \to E\# & \{S.val \leftarrow E.val\} \\
E \to E_1 + T & \{E.val \leftarrow E_1.val + T.val\} \\
E \to T & \{E.val \leftarrow T.val\} \\
T \to T_1 * F & \{T.val \leftarrow T_1.val * F.val\} \\
T \to F & \{T.val \leftarrow F.val\} \\
F \to (E) & \{F.val \leftarrow E.val\} \\
F \to \mathbf{n} & \{F.val \leftarrow \mathbf{n}.val\}
\end{array}
$$

- Todos los símbolos tienen un atributo *val* numérico.
- El atributo *val* es sintetizado.
- El atributo $\mathbf{n}.val$ corresponde al valor numérico asociado al lexema.

UNIVERSIDAD SIMÓN BOLÍVAR

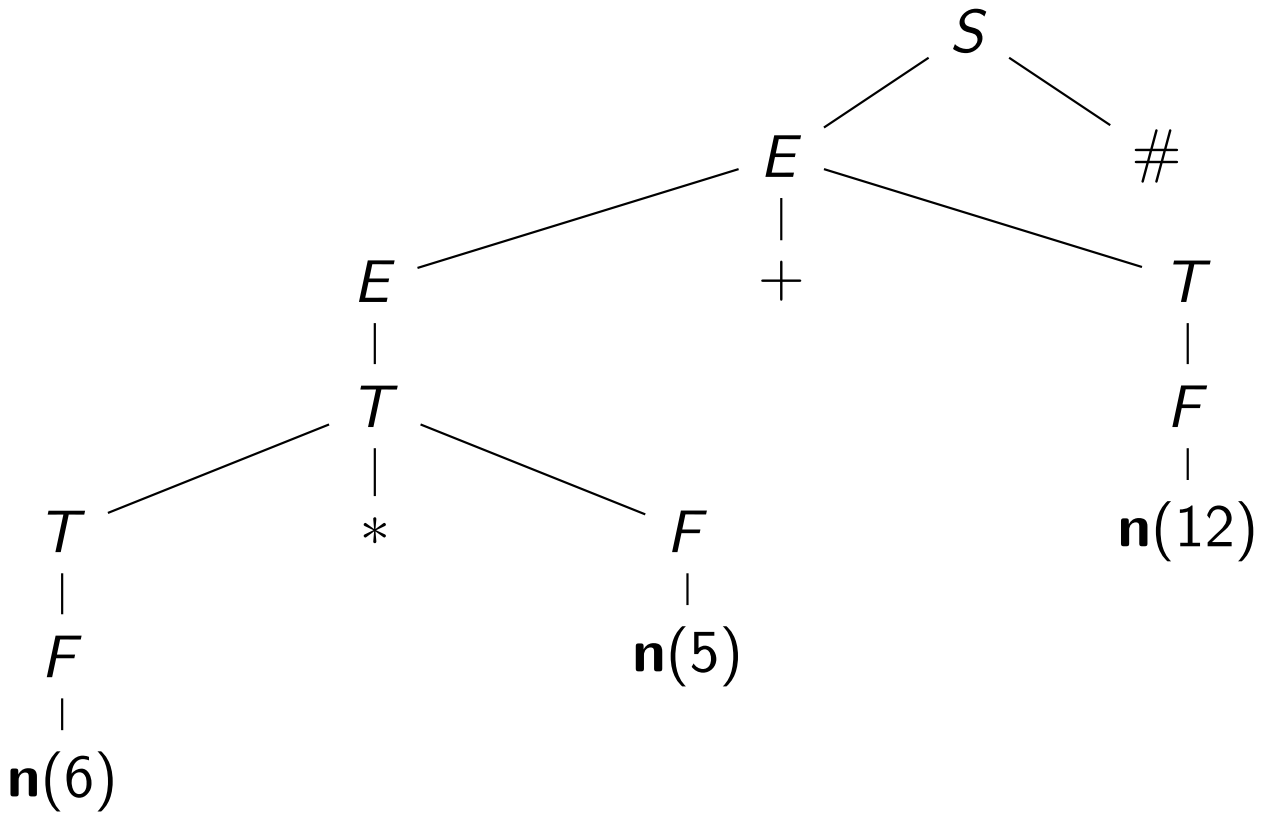
Decorando el árbol

- La gramática de atributos **no** es un programa – es un sistema de ecuaciones simultáneas que requiere solución.
- Para un $w \in L(G)$ cualquiera
  - Se construye el árbol de derivación.
  - Se **decoran** los nodos con las ecuaciones de cálculo de atributos.
  - Se evalúan los atributos en algún orden, hasta evaluarlos todos.
- Para evaluar un atributo es necesario haber evaluado previamente todos aquellos de los cuales depende.

UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

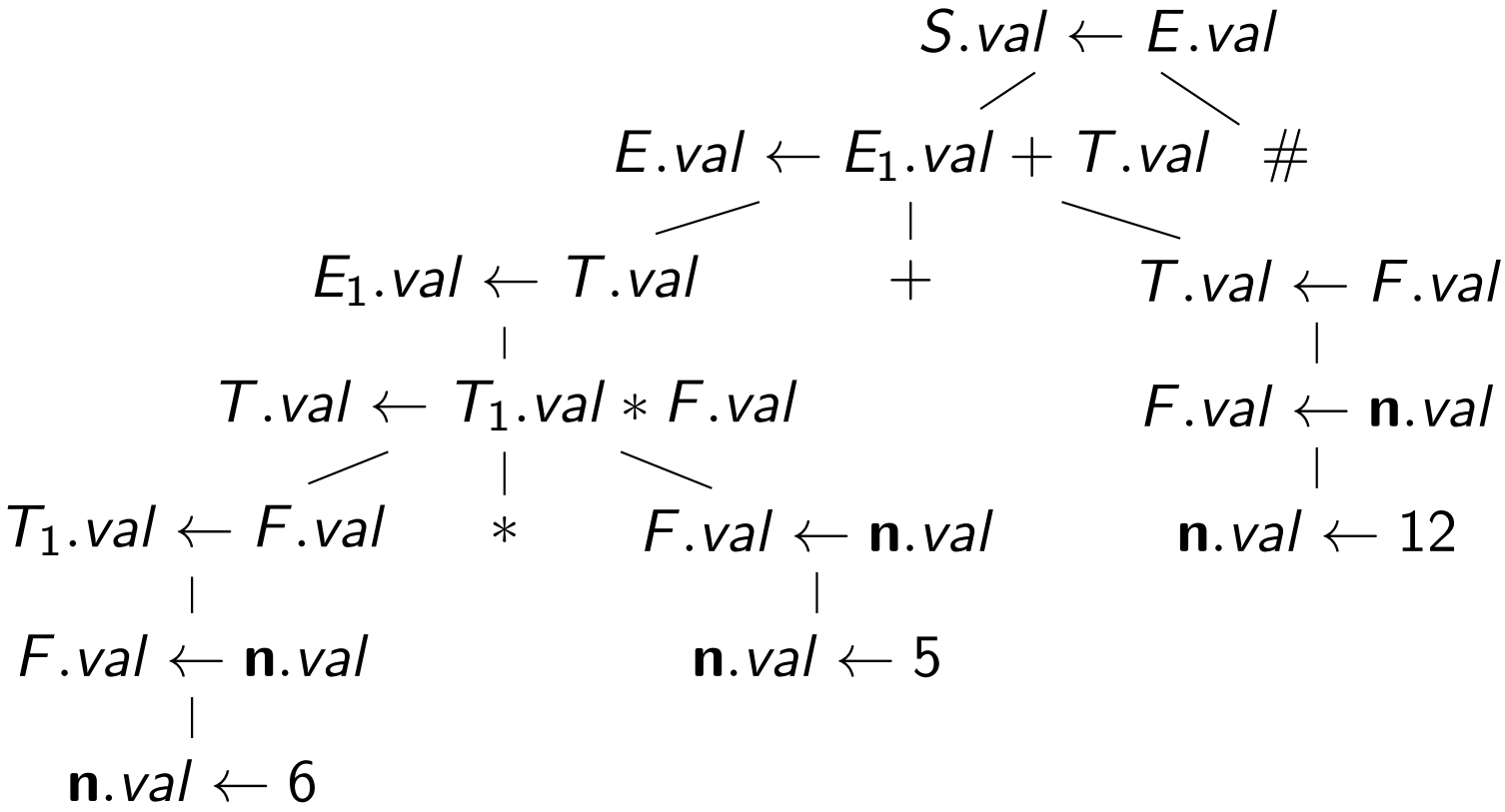
Paso 1 – La derivación para 6*5+12



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

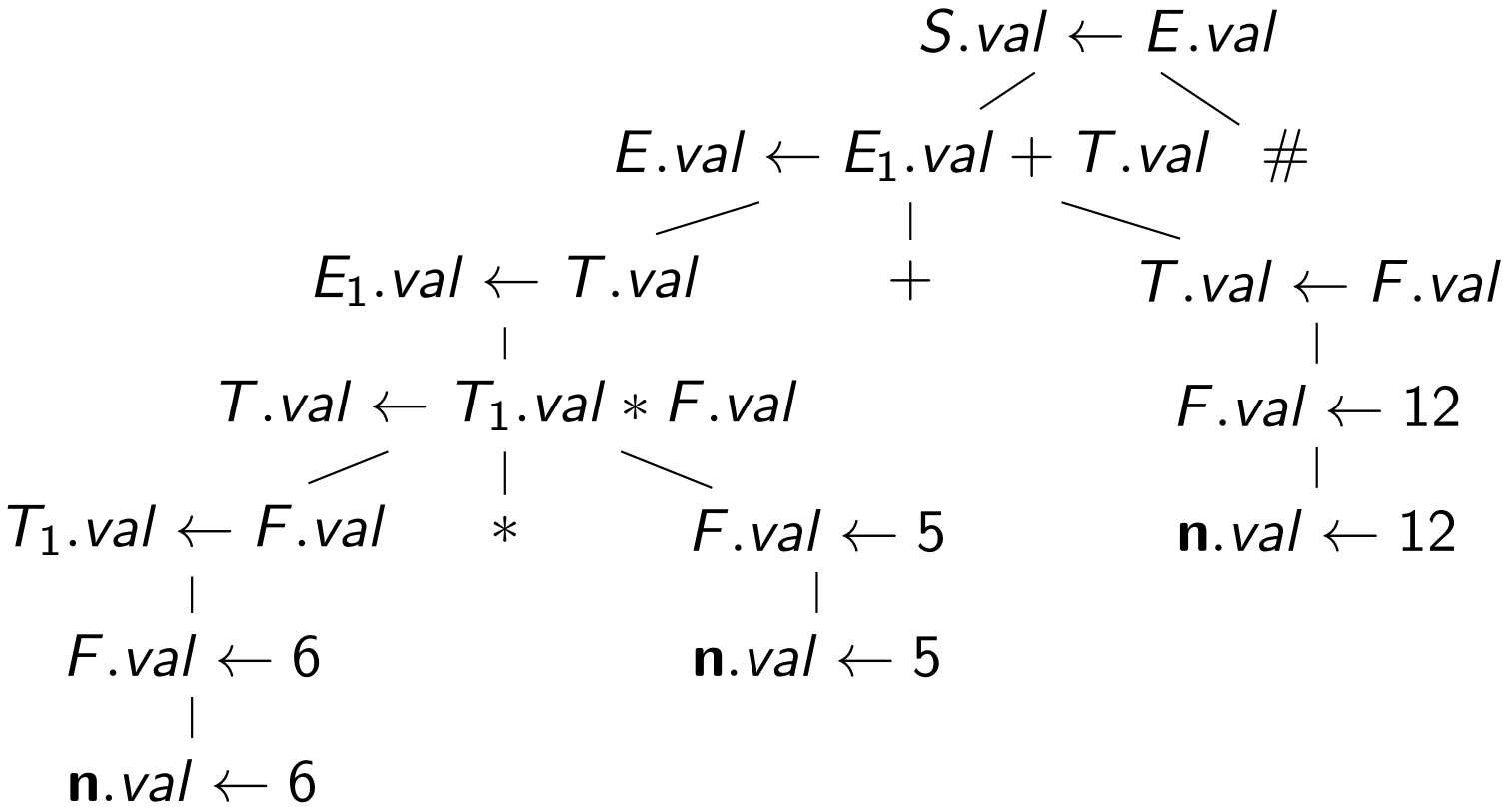
Paso 2 – Las ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

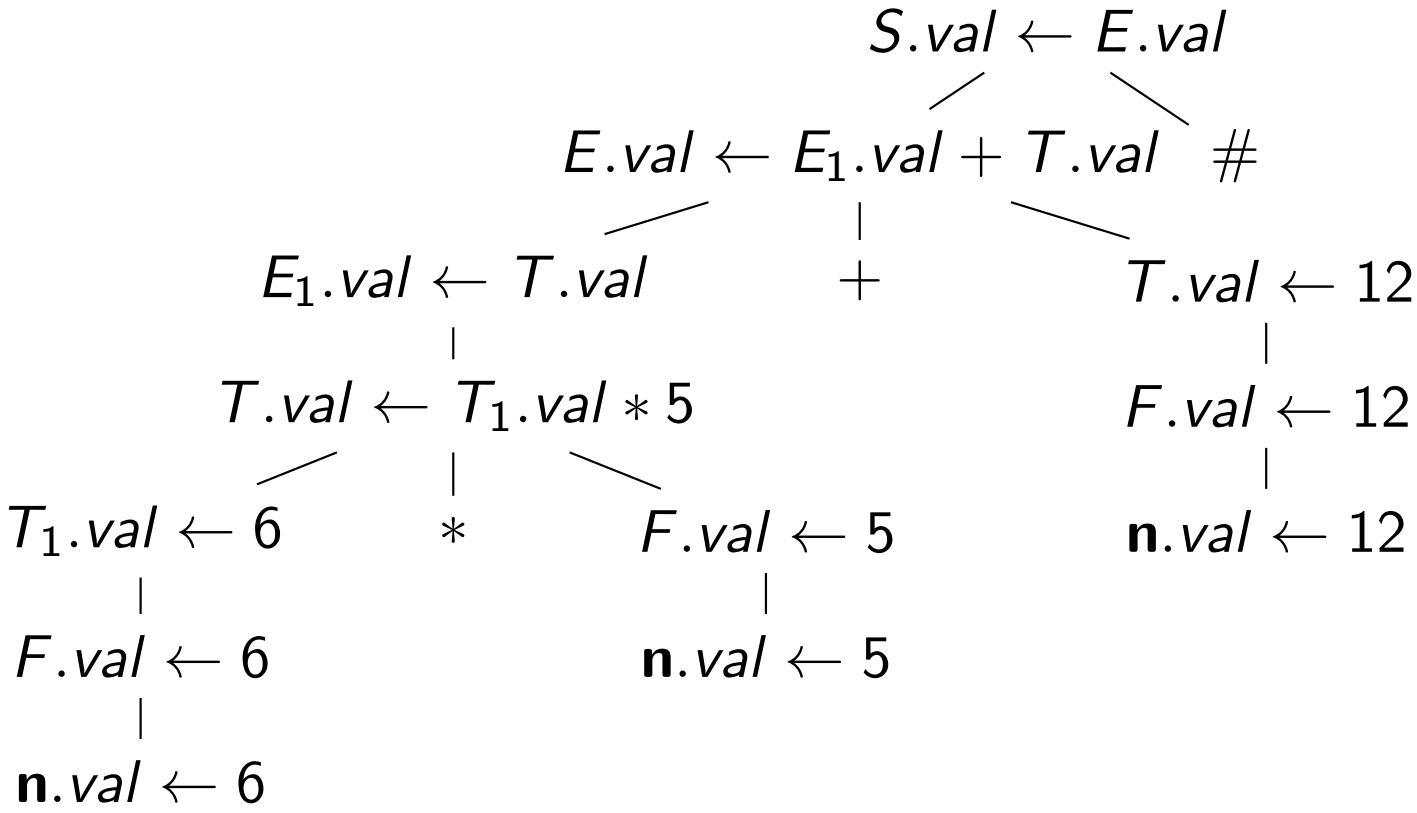
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

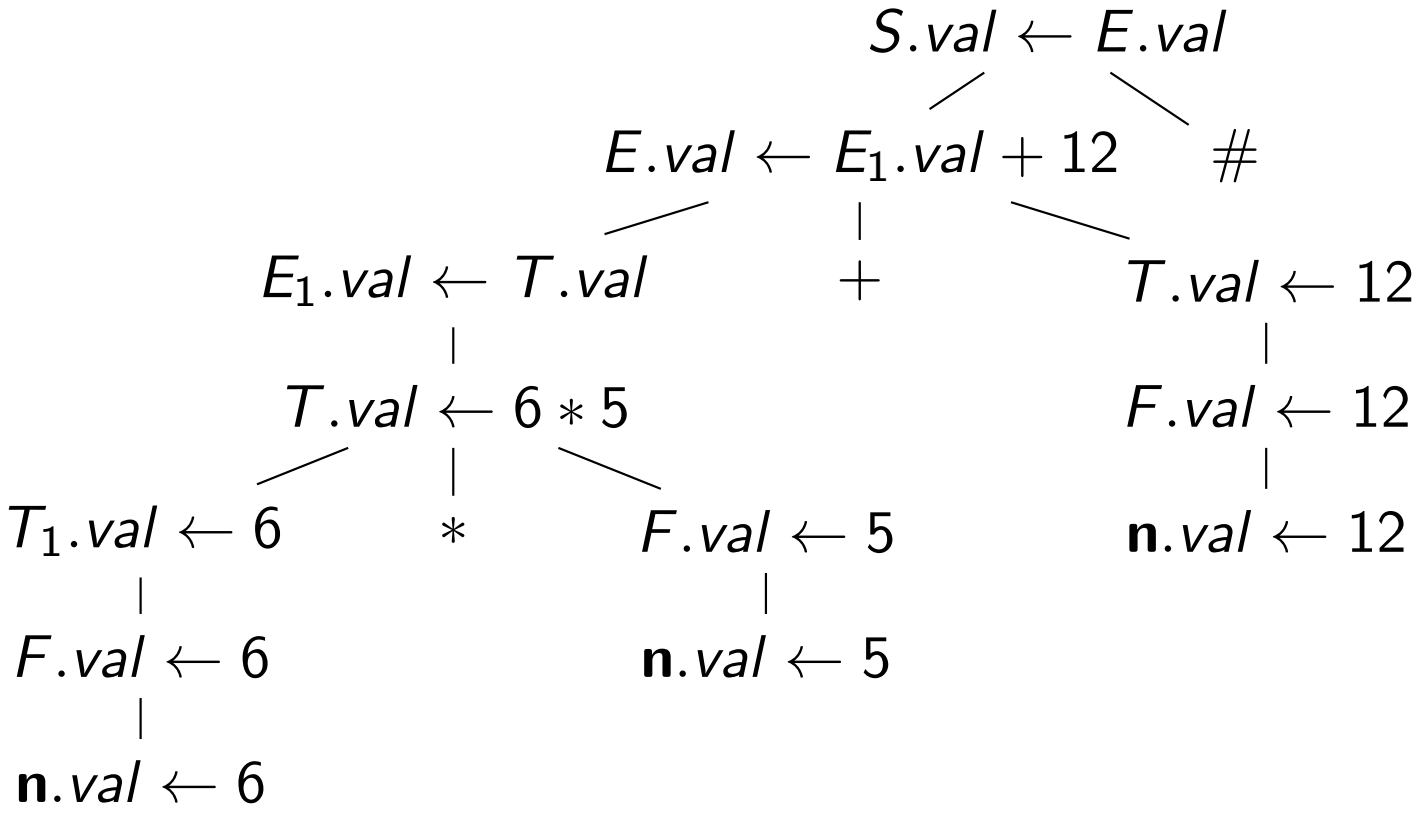
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

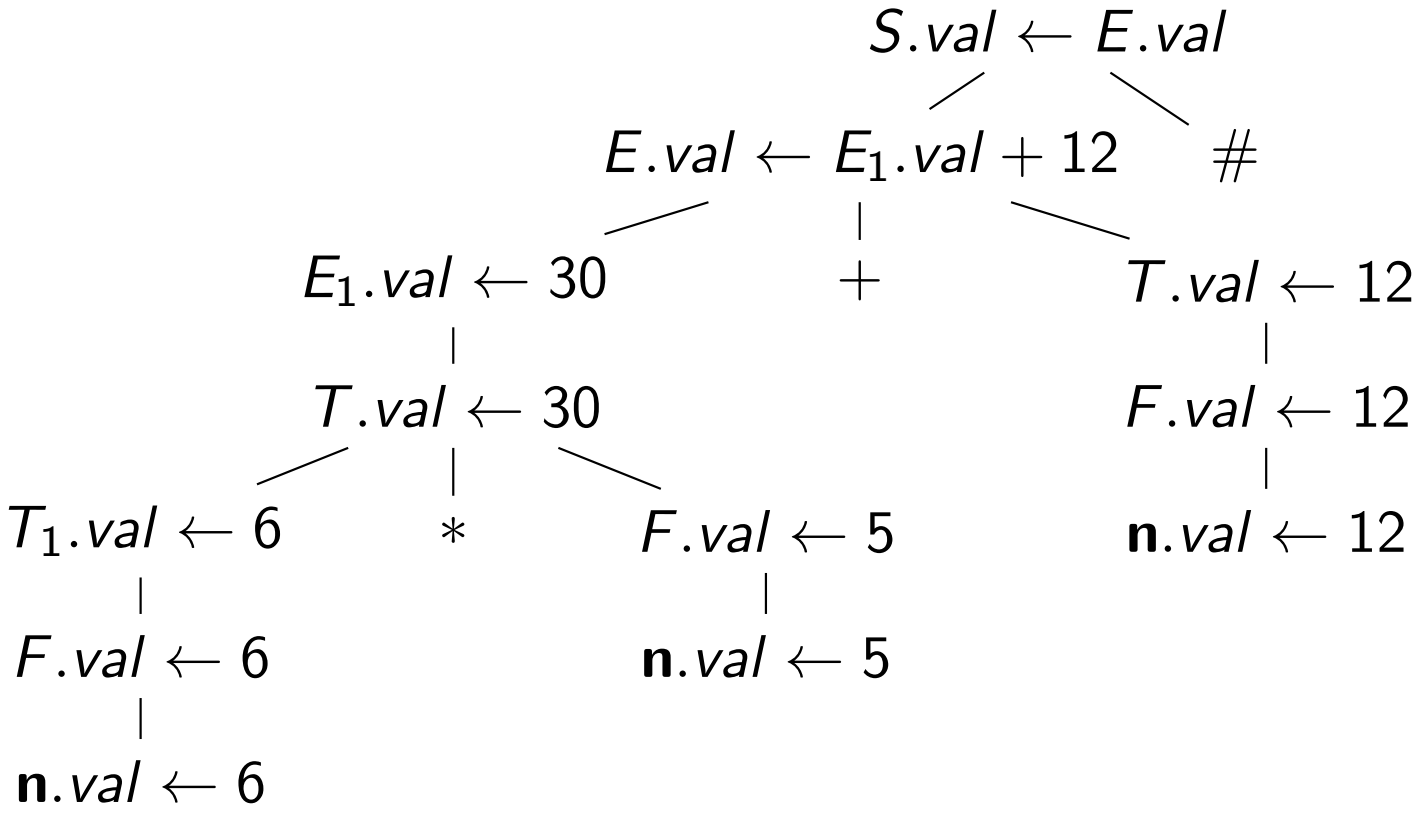
Decorando el árbol

Paso 3 – Resolver ecuaciones



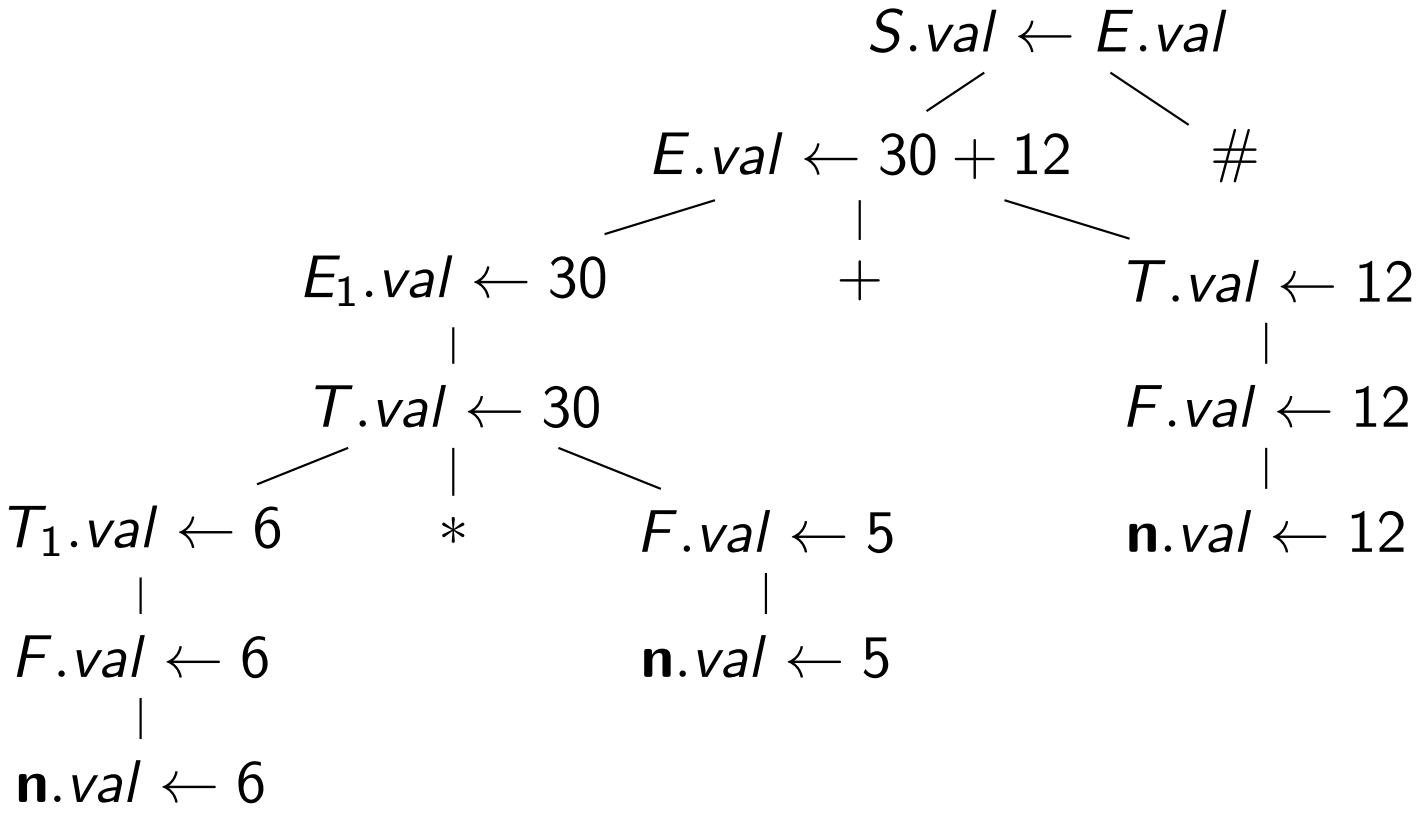
Decorando el árbol

Paso 3 – Resolver ecuaciones



Decorando el árbol

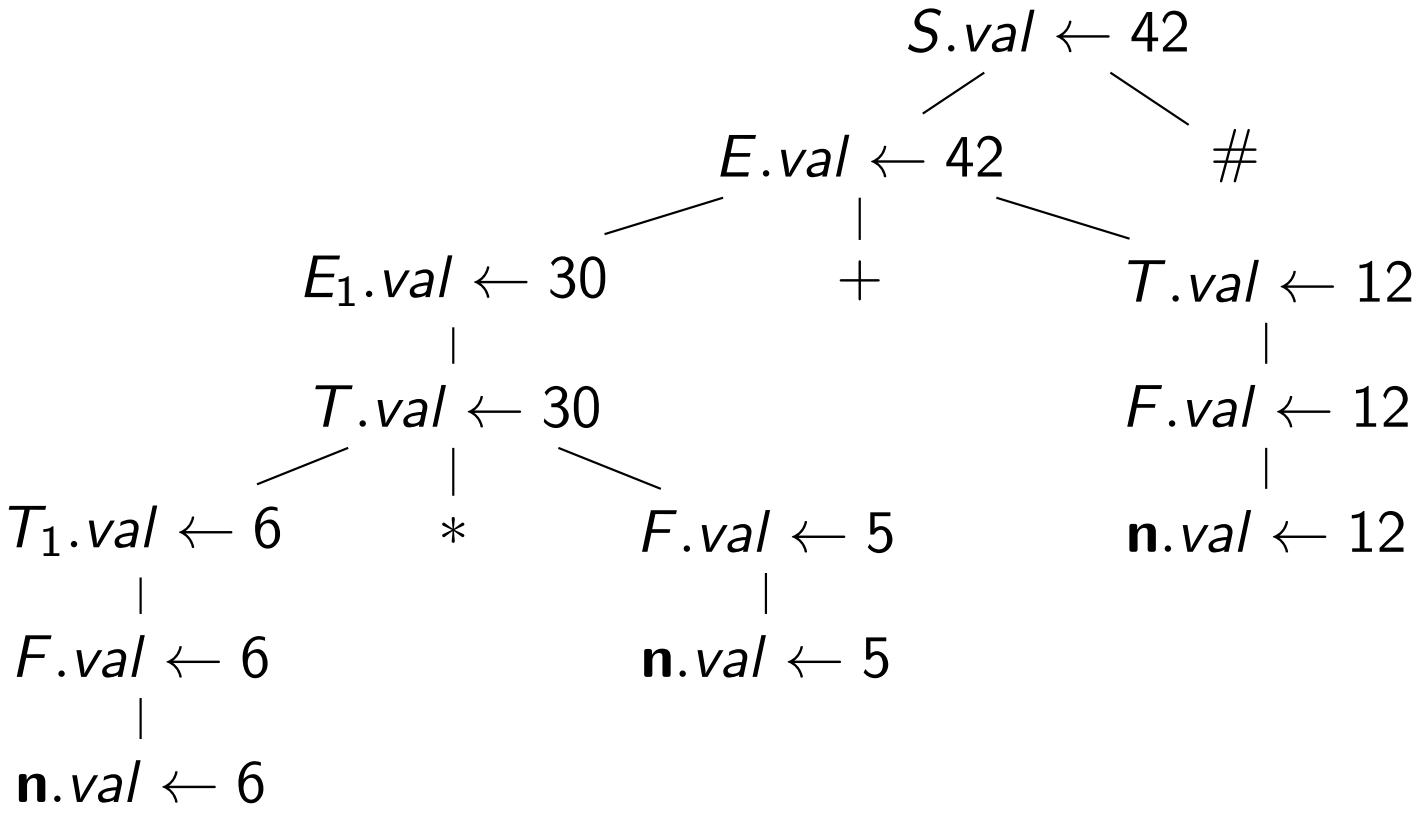
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

Obtenemos la respuesta



UNIVERSIDAD SIMÓN BOLÍVAR

Expresiones, de nuevo

Para reconocimiento descendente

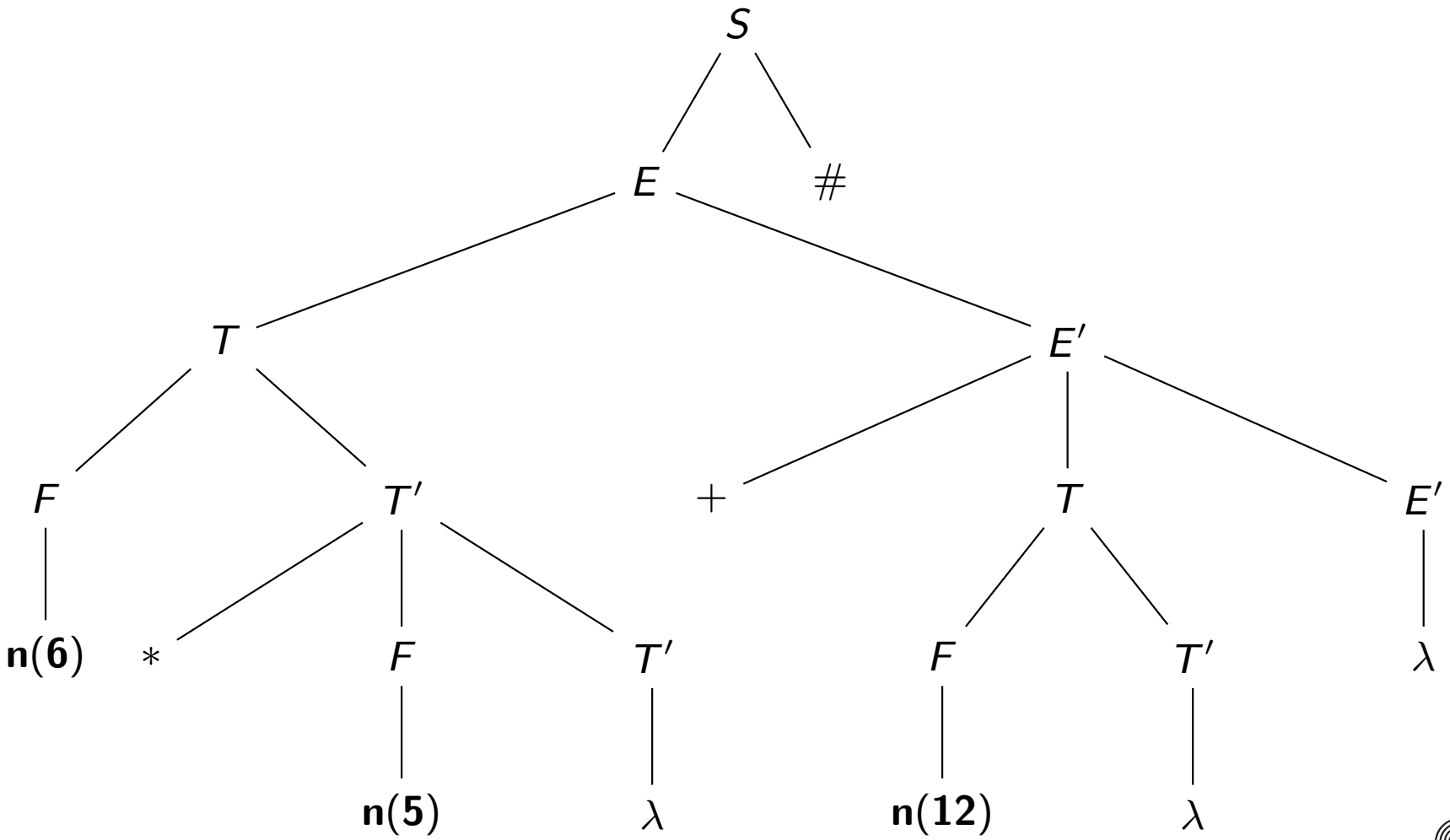
$$
\begin{array}{ll}
S \rightarrow E\# & \{S.val \leftarrow E.val\} \\
E \rightarrow TE' & \left\{\begin{array}{l} E'.in \leftarrow T.val \\ E.val \leftarrow E'.val \end{array}\right\} \\
E' \rightarrow +TE'_1 & \left\{\begin{array}{l} E'_1.in \leftarrow E'.in + T.val \\ E'.val \leftarrow E'_1.val \end{array}\right\} \\
\quad \rightarrow \lambda & \{E'.val \leftarrow E'.in\} \\
T \rightarrow FT' & \left\{\begin{array}{l} T'.in \leftarrow F.val \\ T.val \leftarrow T'.val \end{array}\right\} \\
T' \rightarrow *FT'_1 & \left\{\begin{array}{l} T'_1.in \leftarrow T'.in * F.val \\ T'.val \leftarrow T'_1.val \end{array}\right\} \\
\quad \rightarrow \lambda & \{T'.val \leftarrow T'.in\} \\
F \rightarrow (E) & \{F.val \leftarrow E.val\} \\
F \rightarrow \mathbf{n} & \{F.val \leftarrow \mathbf{n}.val\}
\end{array}
$$

- El atributo *val* es sintetizado – el atributo *in* es heredado.

UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

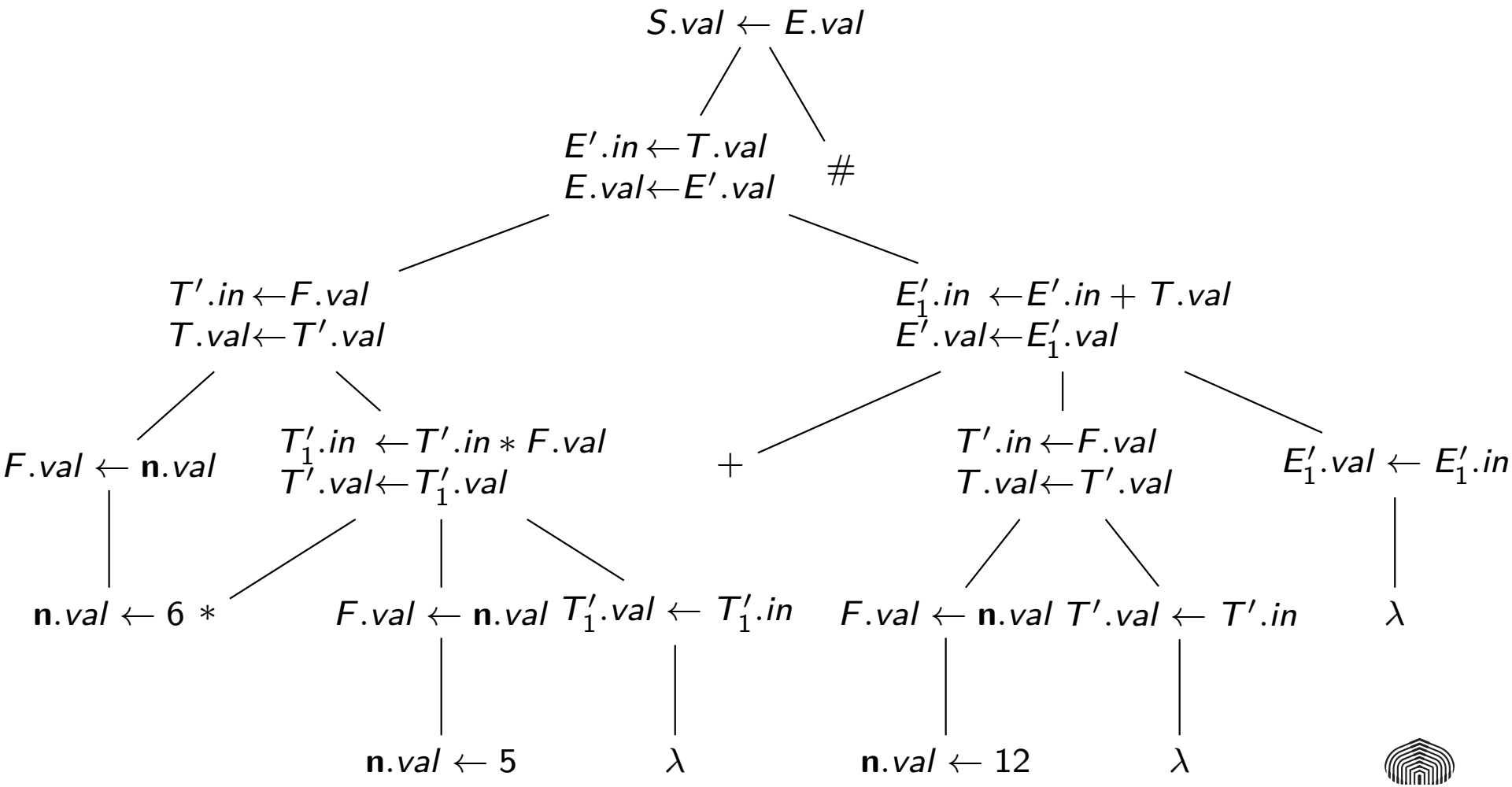
Paso 1 – La derivación para 6*5+12



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

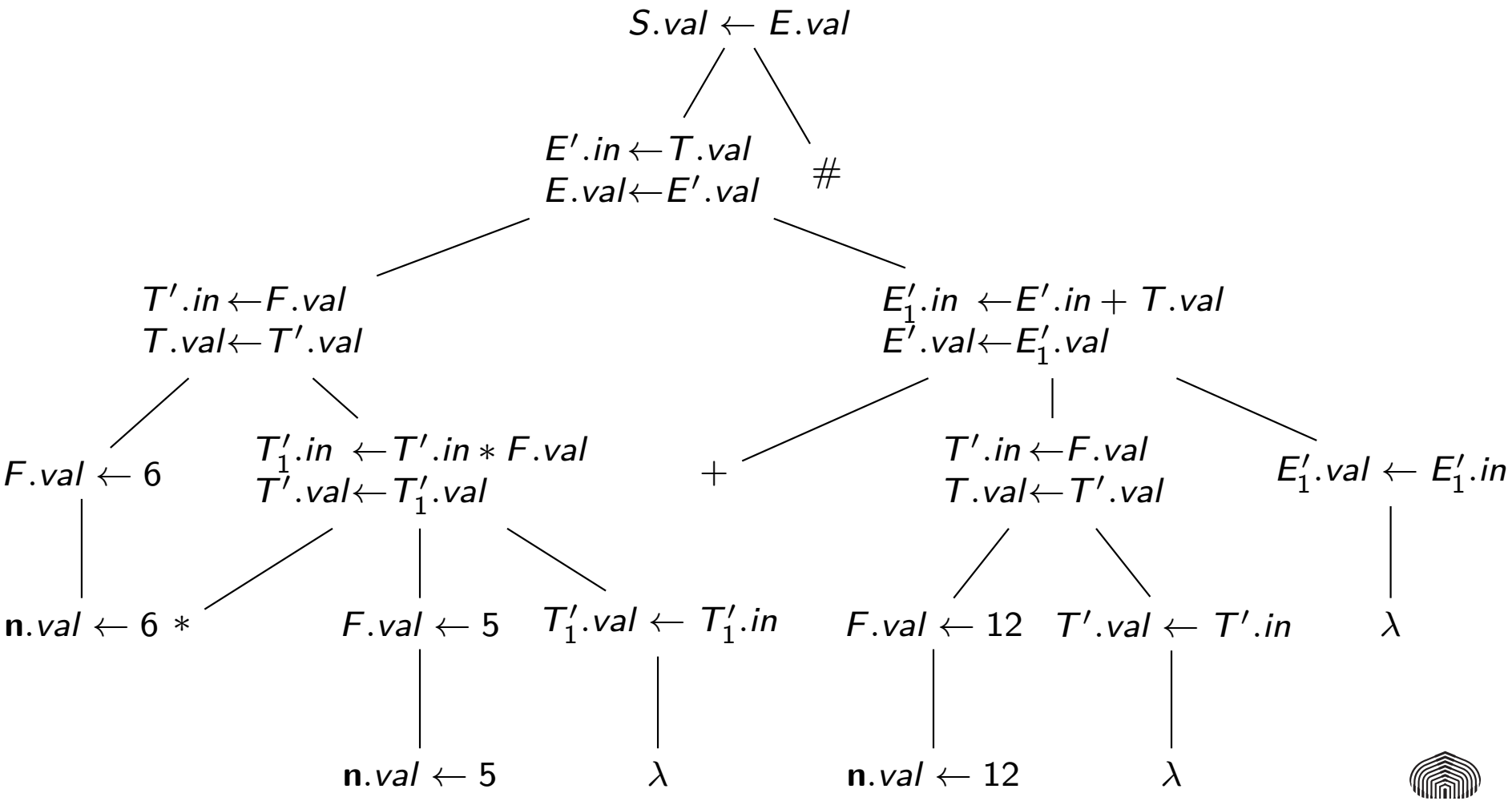
Paso 2 – Las ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

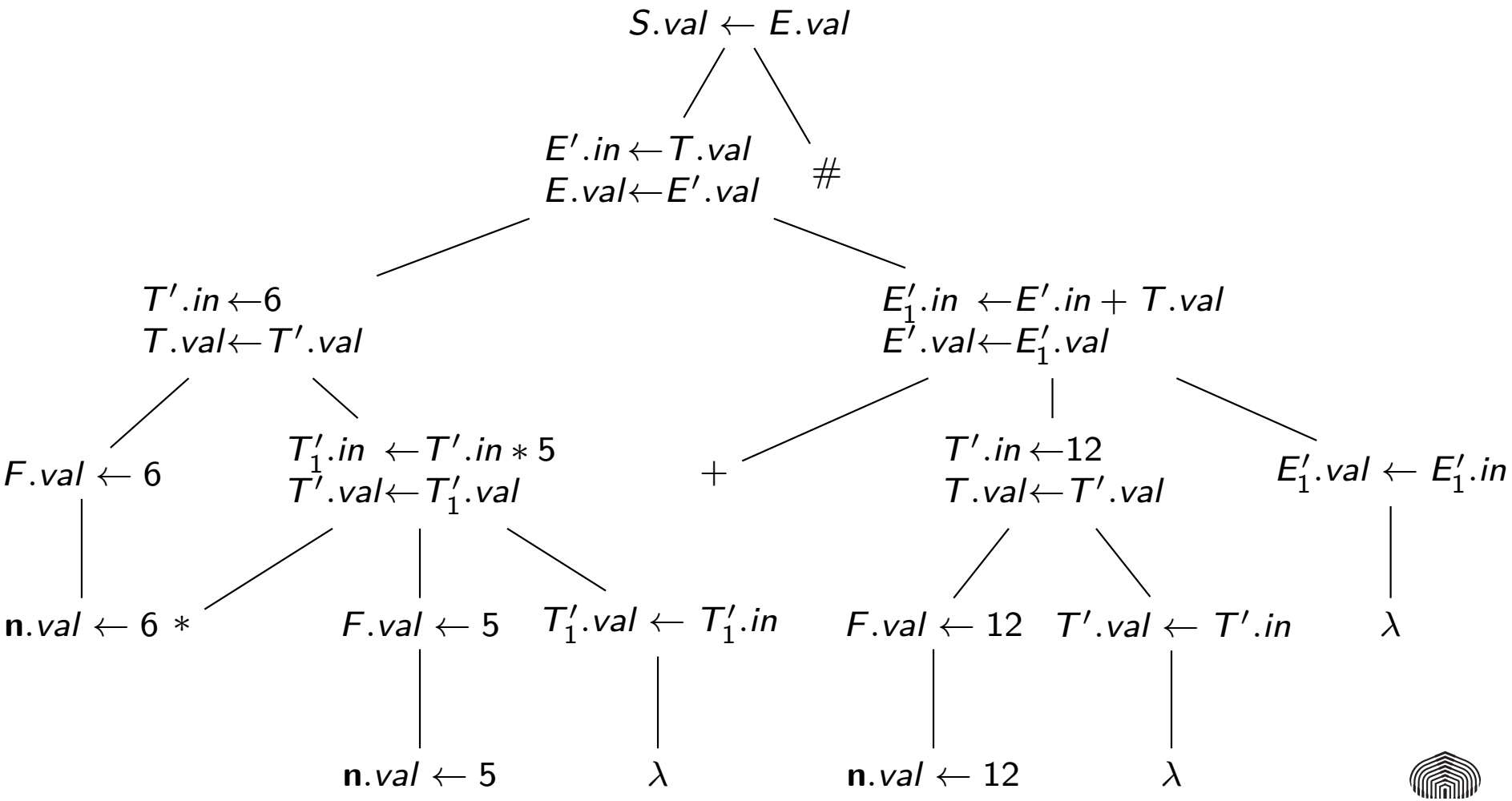
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

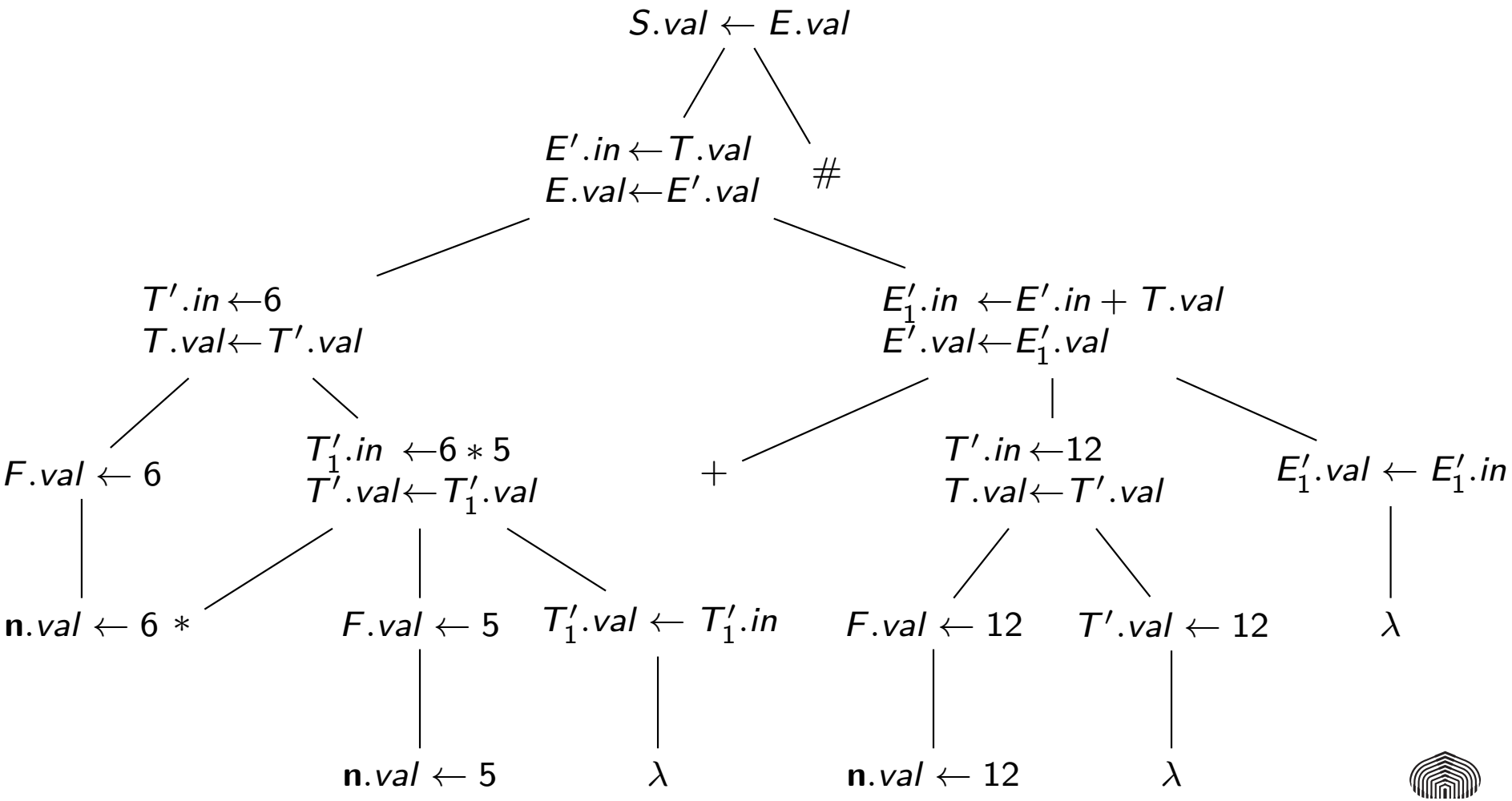
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

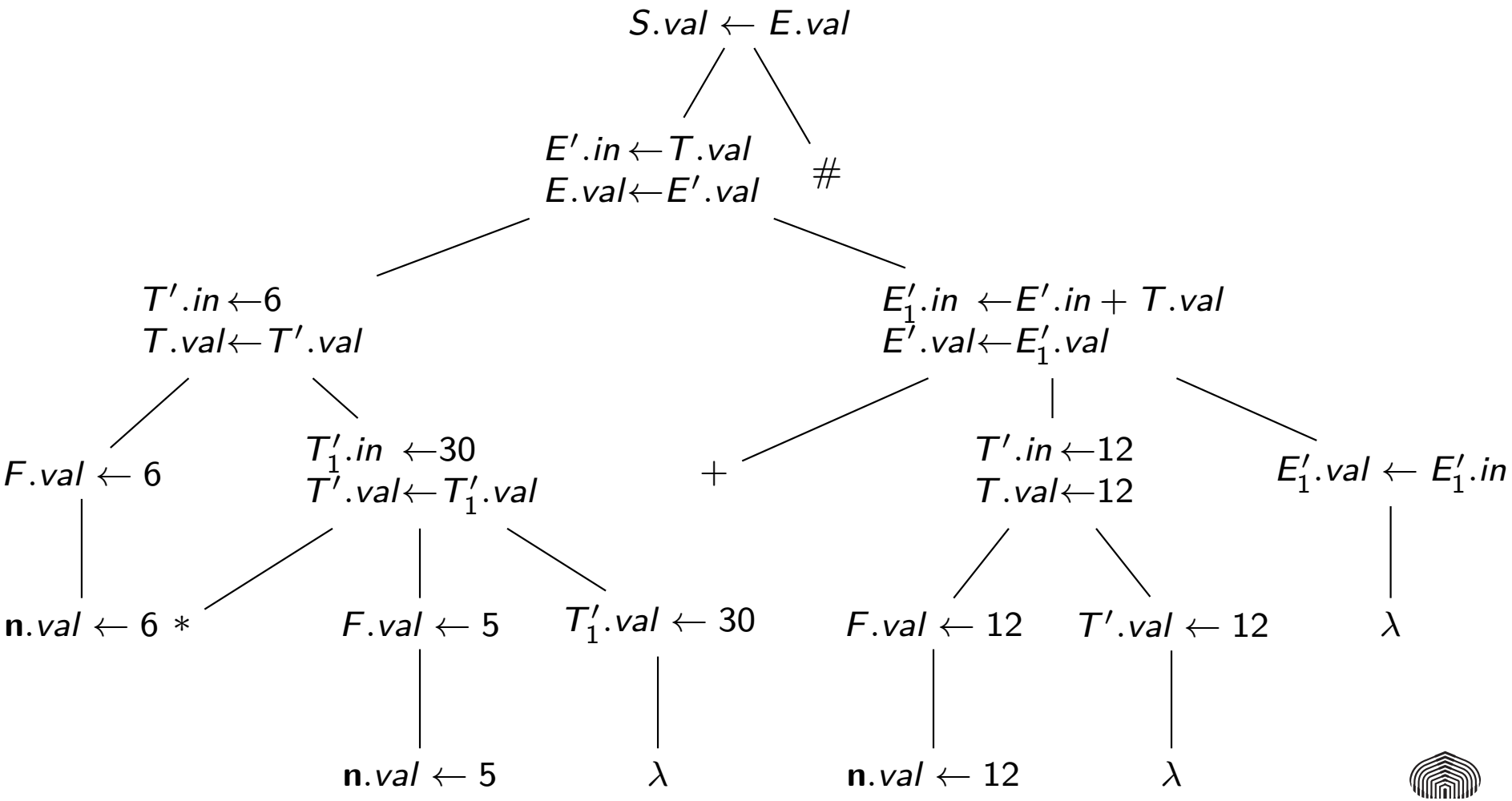
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

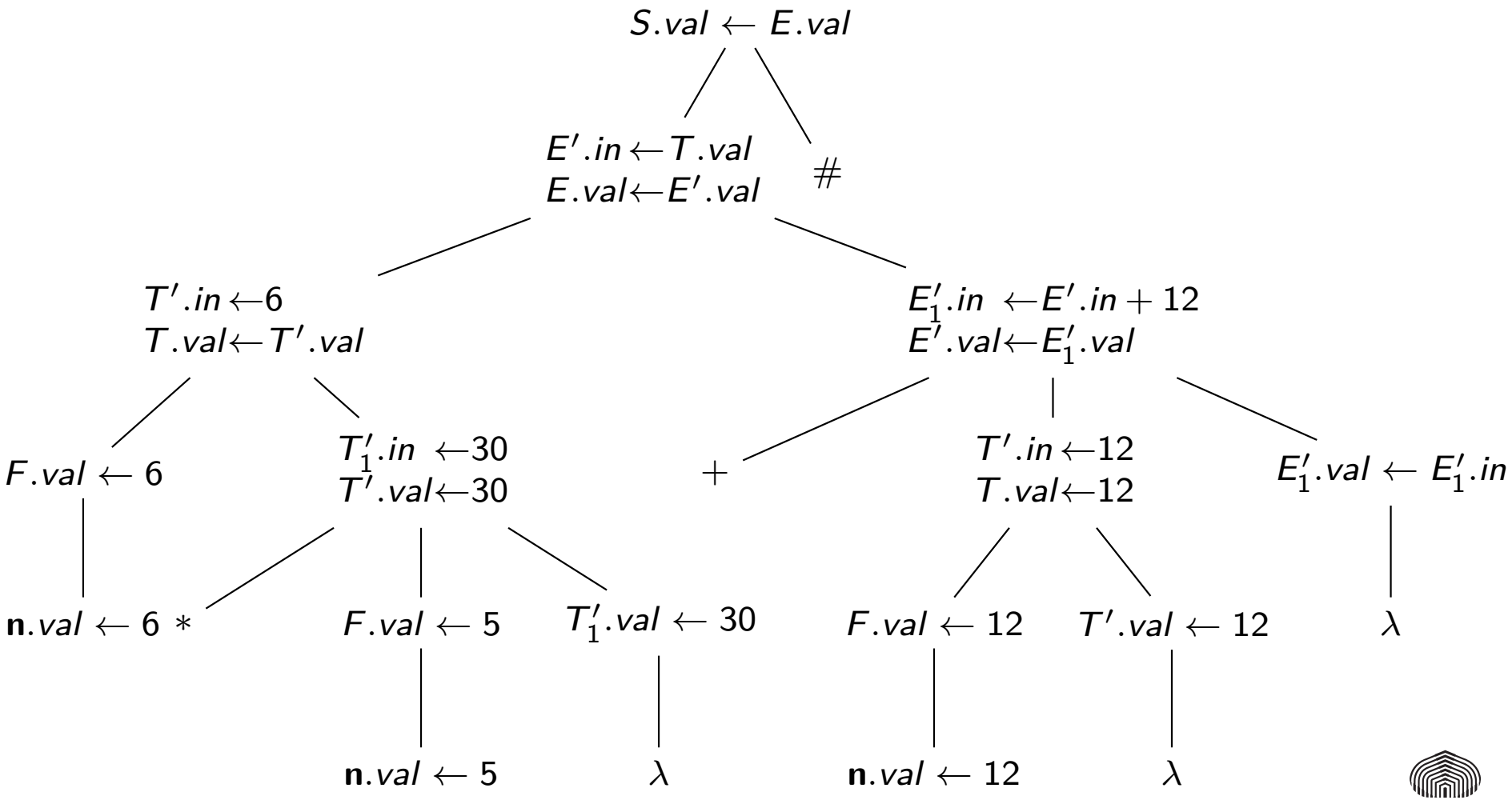
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

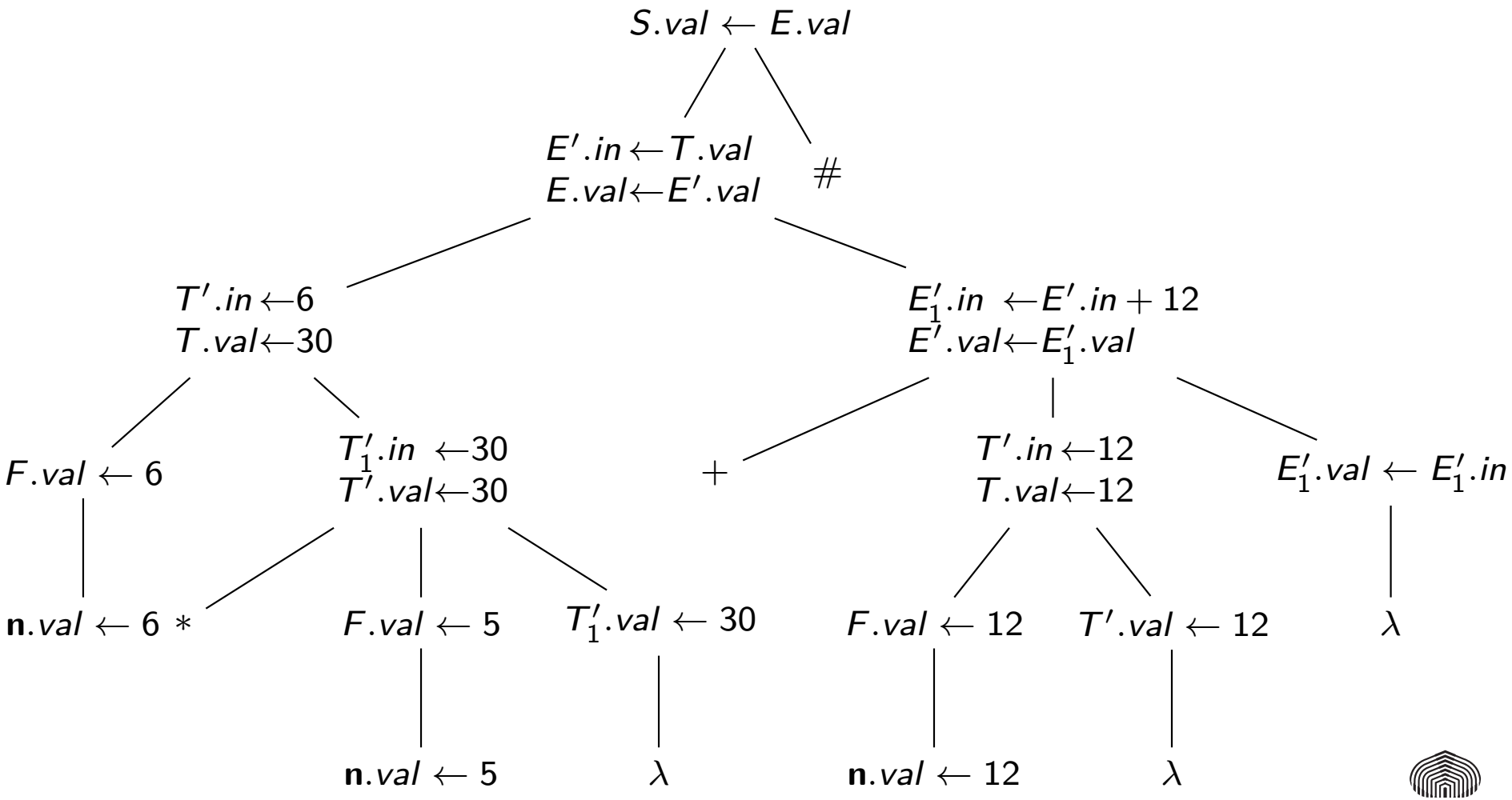
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

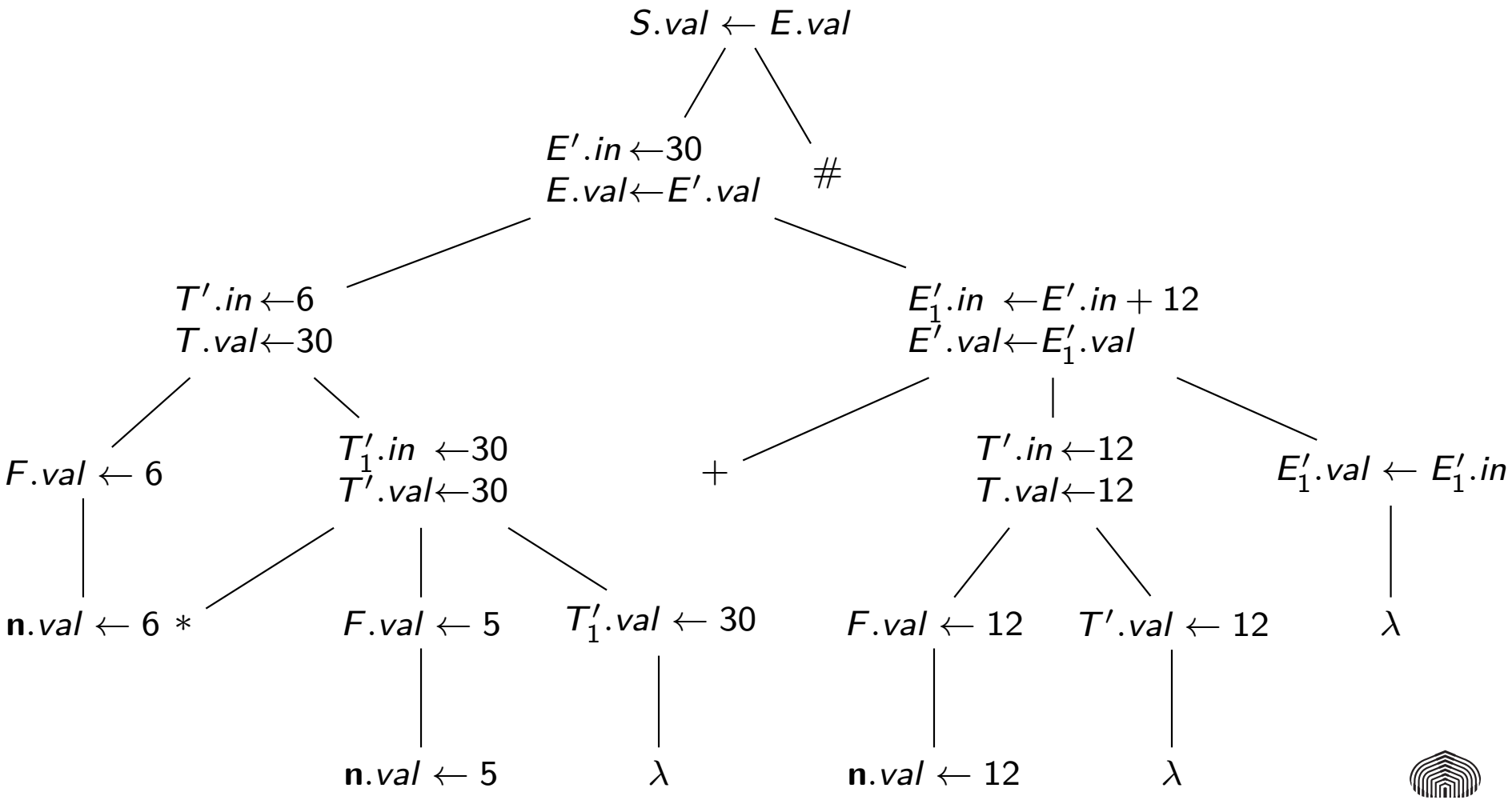
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

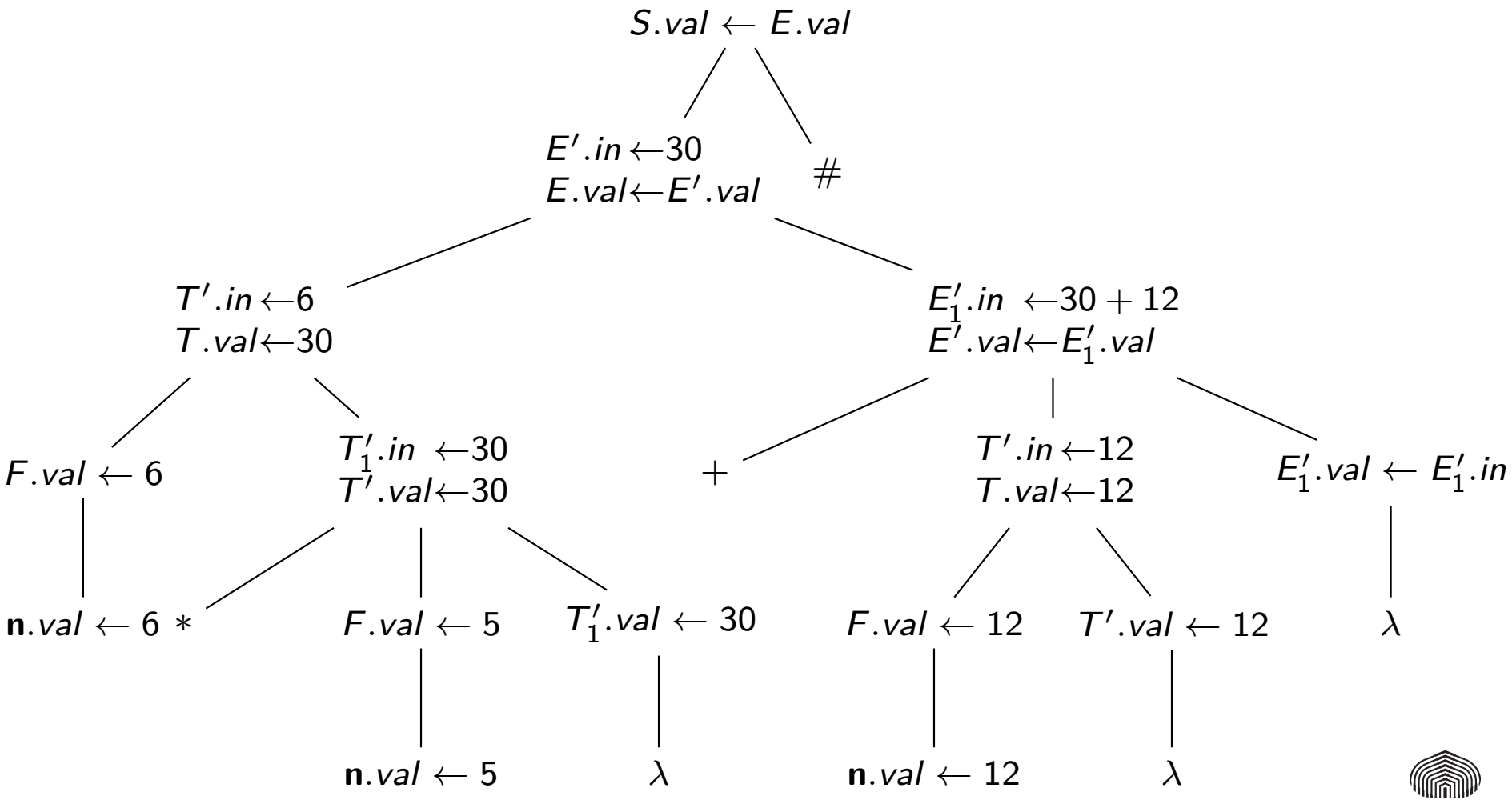
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

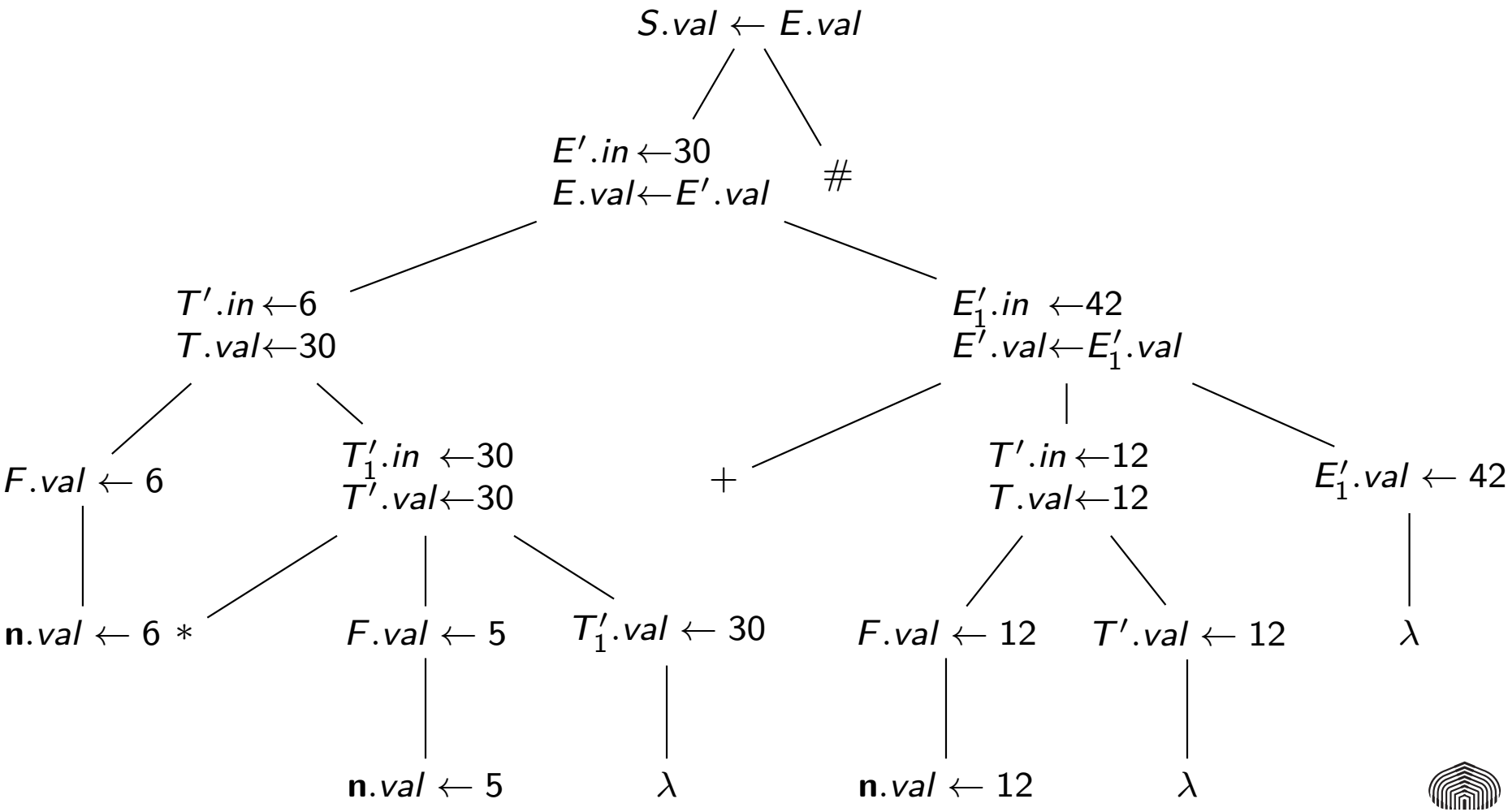
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

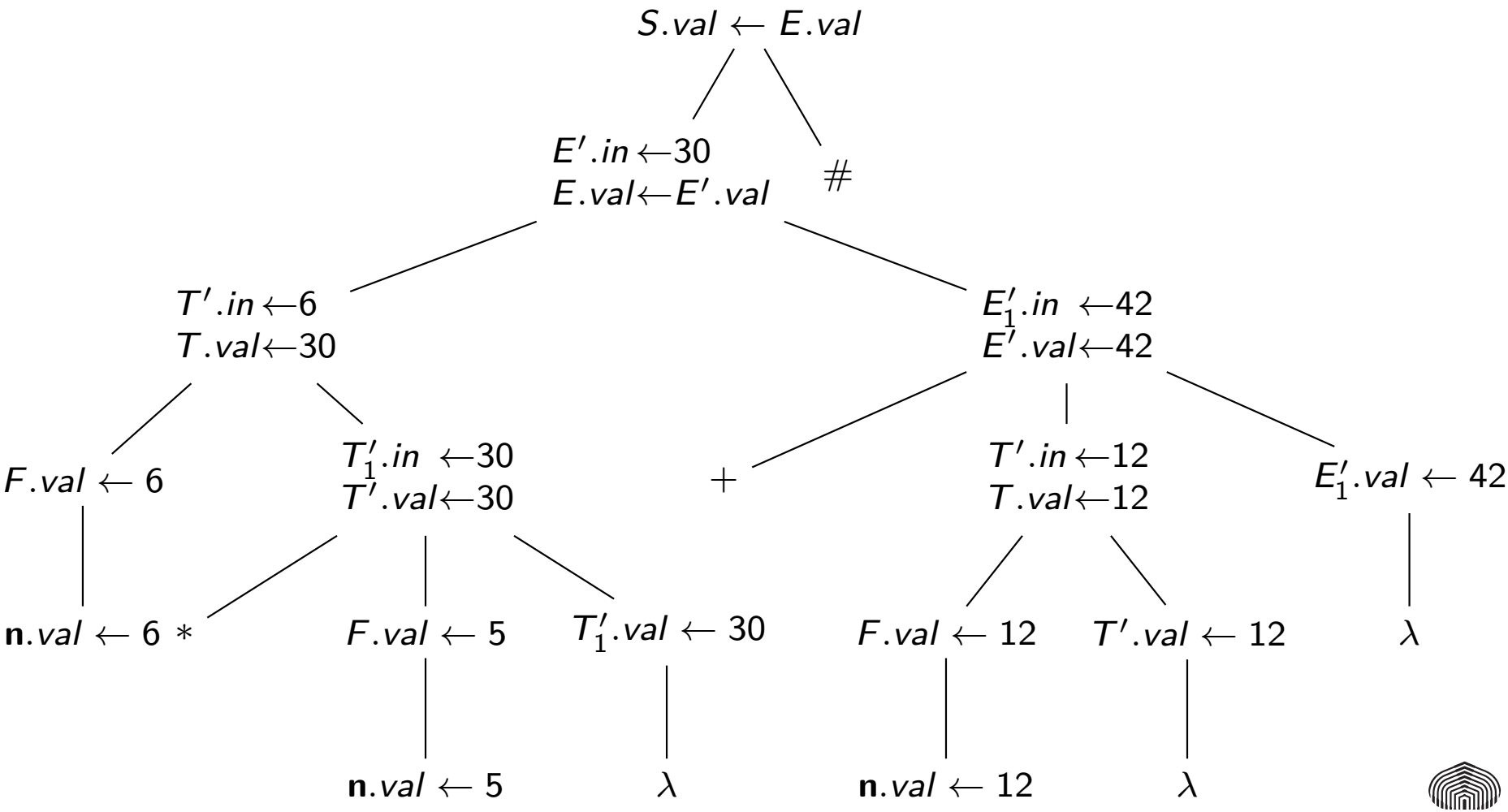
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

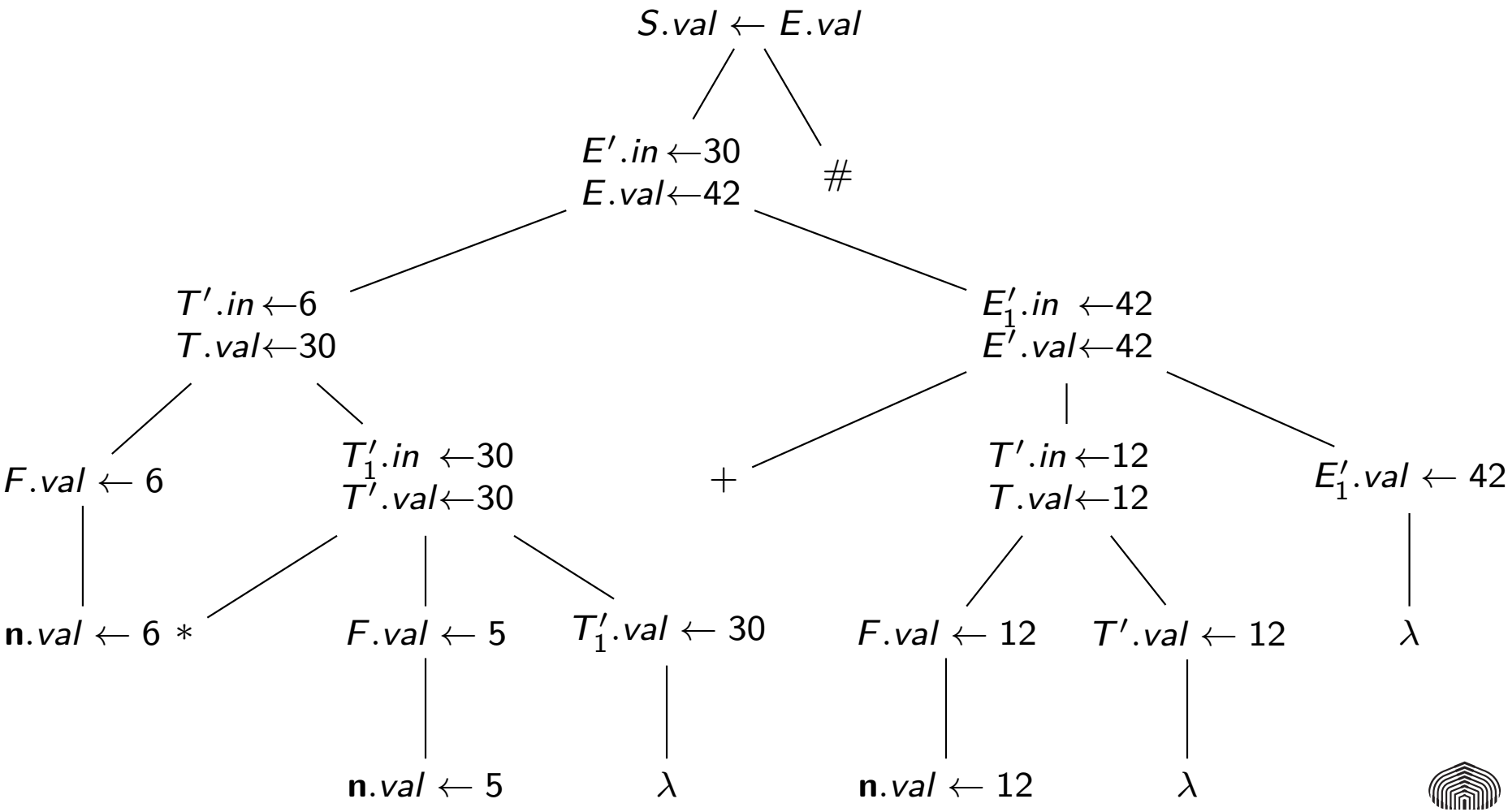
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

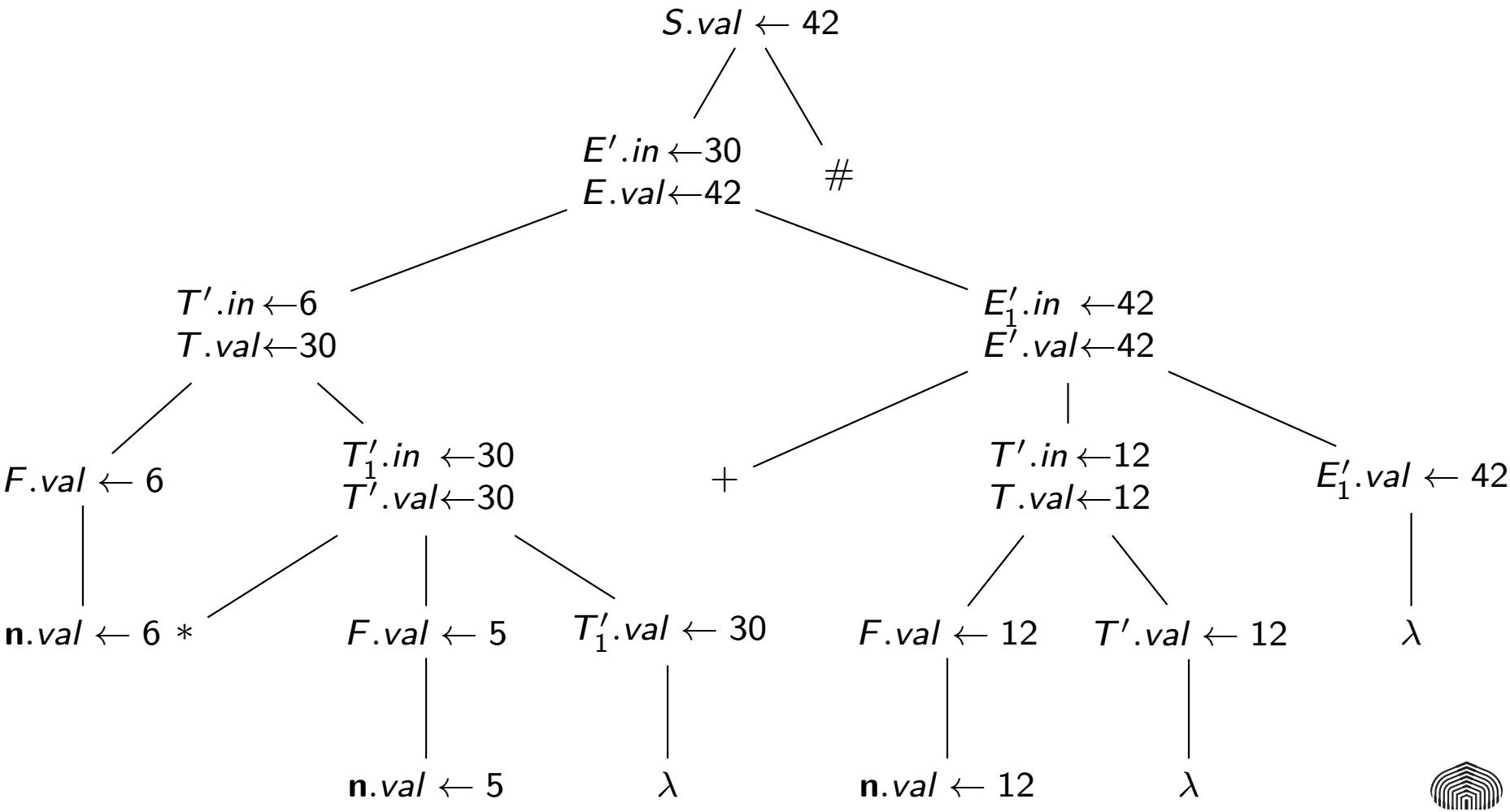
Paso 3 – Resolver ecuaciones



UNIVERSIDAD SIMÓN BOLÍVAR

Decorando el árbol

Obtenemos la respuesta



UNIVERSIDAD SIMÓN BOLÍVAR

Complejo, pero conveniente

- Incluir atributos heredados cambió el orden de recorrido del árbol.
  - En este ejemplo, solamente "bajaron".
  - En general, podrían moverse hacia hijos o hacia hermanos.
- Los atributos heredados son convenientes porque permiten expresar las dependencias en el contexto.
- Siempre es posible convertir una definición dirigida por sintaxis para usar **sólo** atributos sintetizados
  - Reescribiendo la gramática – quizás "deformándola".
  - Cambiando el tipo de los atributos.
  - Para algunas construcciones es más natural utilizar atributos heredados.

UNIVERSIDAD SIMÓN BOLÍVAR

Complejo, pero conveniente

- Incluir atributos heredados cambió el orden de recorrido del árbol.
  - En este ejemplo, solamente "bajaron".
  - En general, podrían moverse hacia hijos o hacia hermanos.
- Los atributos heredados son convenientes porque permiten expresar las dependencias en el contexto.
- Siempre es posible convertir una definición dirigida por sintaxis para usar **sólo** atributos sintetizados
  - Reescribiendo la gramática – quizás "deformándola".
  - Cambiando el tipo de los atributos.
  - Para algunas construcciones es más natural utilizar atributos heredados.

¿Cómo encontramos el orden de evaluación?

UNIVERSIDAD SIMÓN BOLÍVAR

Relación de Dependencia

Un **Grafo de Dependencia** permite representar el flujo de información entre los atributos de un árbol particular.

- Por cada nodo del árbol X que tenga un atributo $X.b$, definir un nodo en el grafo para $X.b$
- Teniendo

$$X.b \leftarrow f(c_1, c_2, \ldots, c_k)$$

entonces se establece una arista en el grafo con origen en el nodo que define a c_i y destino en el nodo $X.b$.

UNIVERSIDAD SIMÓN BOLÍVAR

Estableciendo el Orden de Evaluación

- El grafo caracteriza los órdenes de evaluación posibles.
- Si existe una secuencia de nodos

$$N_1, N_2, \ldots, N_k$$

tal que si hay una arista entre N_i y N_j entonces $i < j$, esa secuencia es un orden de evaluación posible.

  - Corresponde al **Orden Topológico** del grafo.
  - Si hay un ciclo en el grafo, es imposible obtener el orden.

UNIVERSIDAD SIMÓN BOLÍVAR

Estableciendo el Orden de Evaluación

- El grafo caracteriza los órdenes de evaluación posibles.
- Si existe una secuencia de nodos

$$N_1, N_2, \ldots, N_k$$

tal que si hay una arista entre N_i y N_j entonces $i < j$, esa secuencia es un orden de evaluación posible.
  - Corresponde al **Orden Topológico** del grafo.
  - Si hay un ciclo en el grafo, es imposible obtener el orden.
- Podemos usar DFS sobre un árbol **particular** – pero es muy costoso tratar de determinarlo para una SDD.

En la práctica, queremos SDD en los cuales se *garantice* la existencia de un orden.

UNIVERSIDAD SIMÓN BOLÍVAR

Definiciones S-Atribuidas

De abajo hacia arriba

- Una SDD que sólo usa atributos sintetizados se dice **S-atribuida**.
- Evaluar el cálculo de atributos desde las hojas hacia la raíz.
  - Recorrido *postorden*.
  - Los valores se propagan de abajo hacia arriba.
- Coinciden con la operación de un reconocedor ascendente.
  - Cada reducción corresponde con la evaluación de reglas.
  - El árbol se construye de las hojas hacia la raíz.

Las definiciones S-atribuidas son
las preferidas en la práctica

UNIVERSIDAD SIMÓN BOLÍVAR

Definiciones L-Atribuidas

De izquierda a derecha, y arriba hacia abajo

- Una SDD es **L-Atribuida** si todo atributo:
  - Es sintetizado.
  - Si se tiene $A \rightarrow X_1 X_2 \ldots X_n$ y $X_i.a$ es un atributo heredado, el cálculo solamente puede usar:
    - Atributos heredados asociados con A.
    - Atributos heredados o sintetizados asociados con cualesquiera $X_1, X_2, \ldots X_{i-1}$.
    - Atributos heredados de X_i siempre que no establezcan ciclos.
- Coinciden con la operación de un reconocedor descendente.
  - Expansiones usan resultados calculados a su izquierda en la producción.
  - Cuando se alcanza el final de la producción, se calculan resultados para el no terminal expandido.

UNIVERSIDAD SIMÓN BOLÍVAR

Controlar los Efectos de Borde

- Los efectos de borde son necesarios en la práctica
  - Para generar resultados visibles

$$S \rightarrow E\# \ \{\textbf{print}E.val\}$$

  - Para recopilar información – tabla de símbolos, tipos, ...

UNIVERSIDAD SIMÓN BOLÍVAR

Controlar los Efectos de Borde

- Los efectos de borde son necesarios en la práctica
  - Para generar resultados visibles

$$S \rightarrow E\# \ \{\textbf{print}E.val\}$$

  - Para recopilar información – tabla de símbolos, tipos, ...
- Permitiremos efectos de borde "controlados"
  - Efectos incidentales que no condicionan el orden de evaluación – si no estuvieran los efectos, la evaluación seguiría siendo igual.
  - Organizar el orden de evaluación de manera que cualquier otro orden de evaluación produzca el mismo resultado
    - El orden "preferido" agrega aristas implícitas.
    - El orden "preferido" agrega producciones inocuas.

UNIVERSIDAD SIMÓN BOLÍVAR

Controlar los Efectos de Borde

- Los efectos de borde son necesarios en la práctica
  - Para generar resultados visibles

$$S \to E\# \ \{\textbf{print} E.val\}$$

  - Para recopilar información – tabla de símbolos, tipos, ...
- Permitiremos efectos de borde "controlados"
  - Efectos incidentales que no condicionan el orden de evaluación – si no estuvieran los efectos, la evaluación seguiría siendo igual.
  - Organizar el orden de evaluación de manera que cualquier otro orden de evaluación produzca el mismo resultado
    - El orden "preferido" agrega aristas implícitas.
    - El orden "preferido" agrega producciones inocuas.
- Acciones con efectos de borde consideradas como generadoras de un atributo sintetizado que "no nos importa" propagar.

UNIVERSIDAD SIMÓN BOLÍVAR

Mutar o no mutar, esa es la cuestión

- Gramática de Atributos como modelo matemático.
  - Construir el sistema de ecuaciones.
  - Encontrar y usar **un** orden de evaluación válido.
- Esquemas de Traducción como implantación concreta.
  - Imponer **un** orden de evaluación *a priori* – izquierda a derecha.
  - Imponer **un** orden de dependencia *a priori* – S-Atribuido o L-Atribuido.
  - Permitir operaciones arbitrarias en las acciones semánticas, explotando sus efectos de borde.

UNIVERSIDAD SIMÓN BOLÍVAR

Gramática de Atributos

Definición L-Atribuida

$$
\begin{array}{ll}
D \rightarrow T\ L & \{L.in \leftarrow T.type\} \\
T \rightarrow \textbf{int} & \{T.type \leftarrow "integer"\} \\
T \rightarrow \textbf{float} & \{T.type \leftarrow "float"\} \\
L \rightarrow L_1\ , \textbf{id} & \{L_1.in \leftarrow L.in; addtype(\textbf{id}.lexema, L.in\} \\
L \rightarrow \textbf{id} & \{addtype(\textbf{id}.lexema, L.in)\}
\end{array}
$$

- Declaraciones al estilo C.
- Se sintetiza *type* en T – se hereda hacia L.
- Cada identificador de la lista se instala con su tipo.

UNIVERSIDAD SIMÓN BOLÍVAR

Esquema de Traducción

Definición L-Atribuida

$$
\begin{aligned}
D &\rightarrow T \ \{L.in \leftarrow T.type\} \ L \\
T &\rightarrow \textbf{int} \ \{T.type \leftarrow "integer"\} \\
T &\rightarrow \textbf{float} \ \{T.type \leftarrow "float"\} \\
L &\rightarrow \{L_1.in \leftarrow L.in\} \ L_1 \ , \textbf{id} \ \{addtype(\textbf{id}.lexema, L.in\} \\
L &\rightarrow \textbf{id} \ \{addtype(\textbf{id}.lexema, L.in)\}
\end{aligned}
$$

- Acciones en posiciones convenientes según sus efectos de borde.
- Posiciones seleccionadas pensando en el reconocedor descendente.

UNIVERSIDAD SIMÓN BOLÍVAR

Gramática de Atributos es el Esquema

Definición S-Atribuida

$$
\begin{array}{ll}
D \rightarrow T\ L & \{\forall s \in L.set \text{ hacer } addtype(s, T.type)\} \\
T \rightarrow \textbf{int} & \{T.type \leftarrow \text{"}integer\text{"}\} \\
T \rightarrow \textbf{float} & \{T.type \leftarrow \text{"}float\text{"}\} \\
L \rightarrow L_1\ , \textbf{id} & \{L.set \leftarrow L_1.set \cup \textbf{id}.lexema\} \\
L \rightarrow \textbf{id} & \{L.set \leftarrow \{\textbf{id}.lexema\}\}
\end{array}
$$

- Declaraciones al estilo C.
- Se sintetiza *type* en T – se sintetiza el conjunto de símbolos *set* en L.
- Una vez construido el conjunto, se instalan los identificadores.
- Posiciones seleccionadas pensando en el reconocedor descendente.

UNIVERSIDAD SIMÓN BOLÍVAR

Un ejemplo de reescritura gramatical

La SDD obvia resulta inconveniente

$D \rightarrow L : T$	$\{L.in \leftarrow T.type\}$
$T \rightarrow \textbf{int}$	$\{T.type \leftarrow "integer"\}$
$T \rightarrow \textbf{float}$	$\{T.type \leftarrow "float"\}$
$L \rightarrow L_1 \ , \textbf{id}$	$\{L_1.in \leftarrow L.in; addtype(\textbf{id}.lexema, L.in\}$
$L \rightarrow \textbf{id}$	$\{addtype(\textbf{id}.lexema, L.in)\}$

- Declaraciones al estilo Pascal.
- Se sintetiza *type* en T – se hereda hacia L en *in*.

No es S-Atribuida ni L-Atribuida

UNIVERSIDAD SIMÓN BOLÍVAR

Un ejemplo de reescritura gramatical

La SDD transformada también

$D \rightarrow \textbf{id}\ L$	$\{addtype(\textbf{id}.lexema, L.type)\}$
$T \rightarrow \textbf{int}$	$\{T.type \leftarrow "integer"\}$
$T \rightarrow \textbf{float}$	$\{T.type \leftarrow "float"\}$
$L \rightarrow ,\textbf{id}\ L_1$	$\{L.type \leftarrow L_1.type; addtype(\textbf{id}.lexema, L_1.type)\}$
$L \rightarrow :T$	$\{L.type \leftarrow T.type\}$

- Ahora es S-Atribuida ...
- ... y horrible.
- Recursiva por derecha – inconveniente para LR.

Es preferible "complicar" los atributos y su manejo
en lugar de "complicar" la gramática.

UNIVERSIDAD SIMÓN BOLÍVAR

Bibliografía

- [*Aho*]
  - Secciones 5.1 y 5.2
  - Ejercicios 5.1.1 a 5.1.3, 5.2.1 a 5.2.6
- [*Scott*]
  - Secciones 4.1 a 4.3
  - Ejercicios 4.1 a 4.5

UNIVERSIDAD SIMÓN BOLÍVAR