Traduzione

guidata dalla Sintassi

Definizioni guidate dalla sintassi (SDD: Syntax-Directed Definition)

Le tecniche di traduzione guidate dalle grammatiche libere possono essere applicate al type checking, alla generazione del codice intermedio o usate nell'implementazione di piccoli linguaggi per compiti particolari

- si associa informazione a un simbolo di una grammatica associando "**attributi**" al simbolo della grammatica che rappresenta il costrutto.
- in una **definizione guidata dalla sintassi** (Syntax-Directed Definition) i valori degli attributi sono calcolati da "**regole di valutazione**" associate alle produzioni della grammatica.

L'approccio più generale alla traduzione syntax-directed consiste nel:

- parsificare la stringa in input costruendo l'albero sintattico;
- visitare i nodi dell'albero per calcolare il valore degli attributi usando le regole semantiche.

La traduzione è il risultato della valutazione delle regole semantiche.

Definizioni guidate dalla sintassi

Esempio: Espressione aritmetica → valore numerico

Produzioni	Regole di valutazione
$E \rightarrow E_1 + T$	E.*val* = E_1.*val* + T.*val*
$E \rightarrow T$	E.*val* = T.*val*
$T \rightarrow T_1$ * **num**	T.*val* = T_1.*val* * **num**.*val*
$T \rightarrow$ **num**	T.*val* = **num**.*val*

- nelle prime due produzioni l'indice 1 in E_1 distingue le diverse occorrenze di E nelle produzioni stesse.
- E e T hanno un attributo *val*, che è un numero.
- **num.** *val* rappresenta il valore intero della stringa numerica riconosciuta dallo scanner.

Calcolo degli attributi

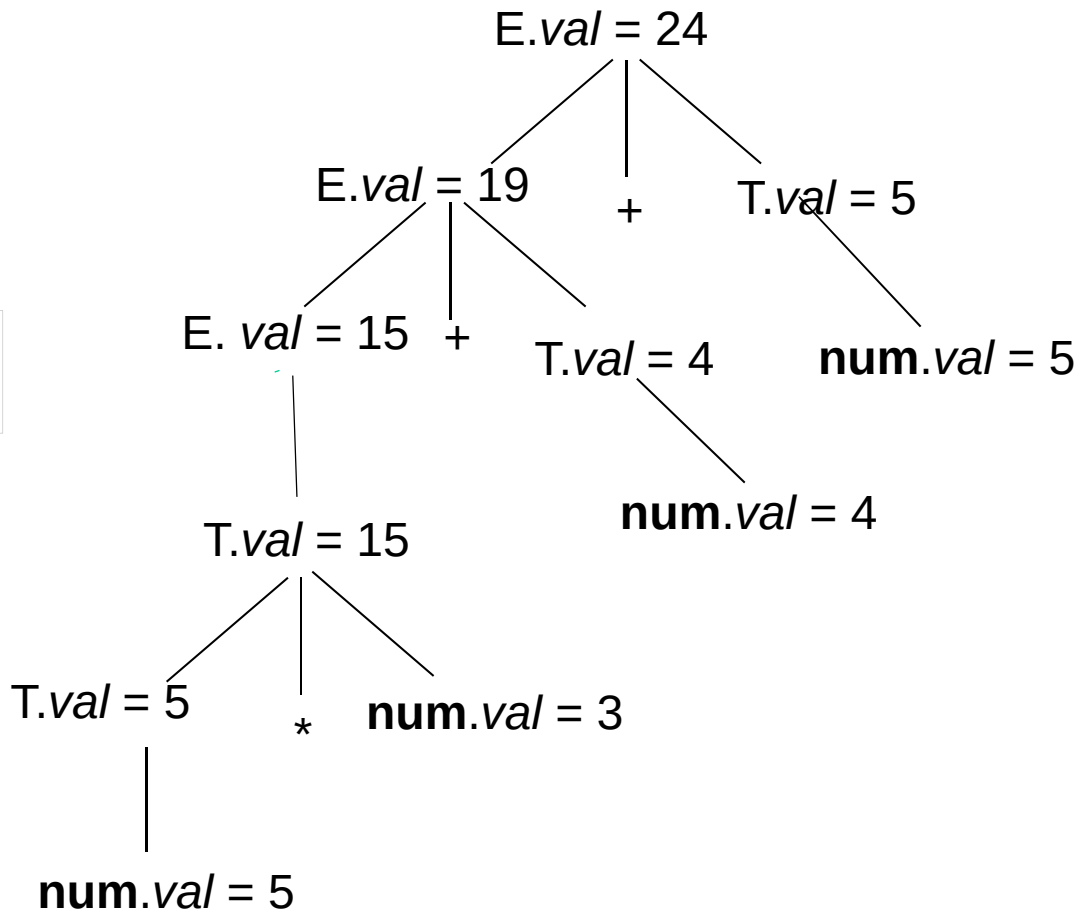
- La traduzione specificata da un SDD per una certa stringa è calcolata, *in linea di principio*, partendo dall'albero di parsificazione della stringa, usando le regole per valutare gli attributi in ogni nodo dell'albero.

- Un albero di parsificazione che mostra i valori degli attributi è chiamato ***albero di parsificazione annotato***.

Definizioni guidate dalla sintassi: valutazione degli attributi

Produzioni	Regole semantiche
$E \rightarrow E_1 + T$	E.*val* = E_1.*val* + T.*val*
$E \rightarrow T$	E.*val* = T.*val*
$T \rightarrow T_1$ * **num**	T.*val* = T_1.*val** **num**.*val*
$T \rightarrow$ **num**	T.*val* = **num**.*val*

Albero di parsificazione annotato per la stringa 5*3+4+5



Definizioni guidate dalla sintassi

Gli attributi possono essere di un tipo qualunque: numeri, tipi, riferimenti a tabelle, stringhe, sequenze di codice, . . .

Se **X** è un simbolo e ***a*** un suo attributo, usiamo **X.*a*** per denotare il valore di *a* in un particolare nodo dell'albero di parsificazione etichettato X.

Le definizioni guidate dalla sintassi, quando i valori degli attributi sono definiti solo in funzione di costanti e dei valori di altri attributi, vengono anche chiamate **grammatiche ad attributi**.

Gli **attributi per i simboli terminali** hanno i valori forniti dall'analizzatore lessicale. Nelle SDD non vi sono regole semantiche per calcolare i valori degli attributi per i terminali.

Attributi ereditati e sintetizzati

Per i **simboli non terminali** consideriamo due tipi di attributi:

- ***sintetizzati***: un attributo sintetizzato per una variabile A in un nodo *n* dell'albero di parsificazione è definito da una regola semantica associata alla produzione in *n* e il suo valore è calcolato solo in termini dei valori degli attributi nei nodi figli di *n* e in *n* stesso. (A è il simbolo a sinistra nella produzione).

- ***ereditati***: un attributo ereditato per una variabile A in un nodo *n* dell'albero di parsificazione è definito da una regola semantica associata alla produzione nel nodo padre di *n* e il suo valore è calcolato solo in termini dei valori degli attributi del padre di *n*, di *n* stesso e dei suoi fratelli.
  (A è un simbolo nel corpo della produzione, cioè al membro destro).

Definizioni guidate dalla sintassi

Esempio: Notazione infissa → Notazione postfissa

Per esempio la forma postfissa di 5*3+4+5 è: 5 3 * 4 + 5 +

Produzioni	Regole semantiche
$E \rightarrow E_1 + T$	E.*code* = E_1.*code* || T.*code* || '+'
$E \rightarrow T$	E.*code* = T.*code*
$T \rightarrow T_1$ * **num**	T.*code* = T_1.*code* || **num**.*lexval* || '*'
$T \rightarrow$ **num**	T.*code* = **num**.*lexval*

L'unico attributo *code* per le variabili E e T è *sintetizzato*, mentre il terminale **num** ha l'attributo *lexval*, che è questa volta la stringa fornita dall'analizzatore lessicale.

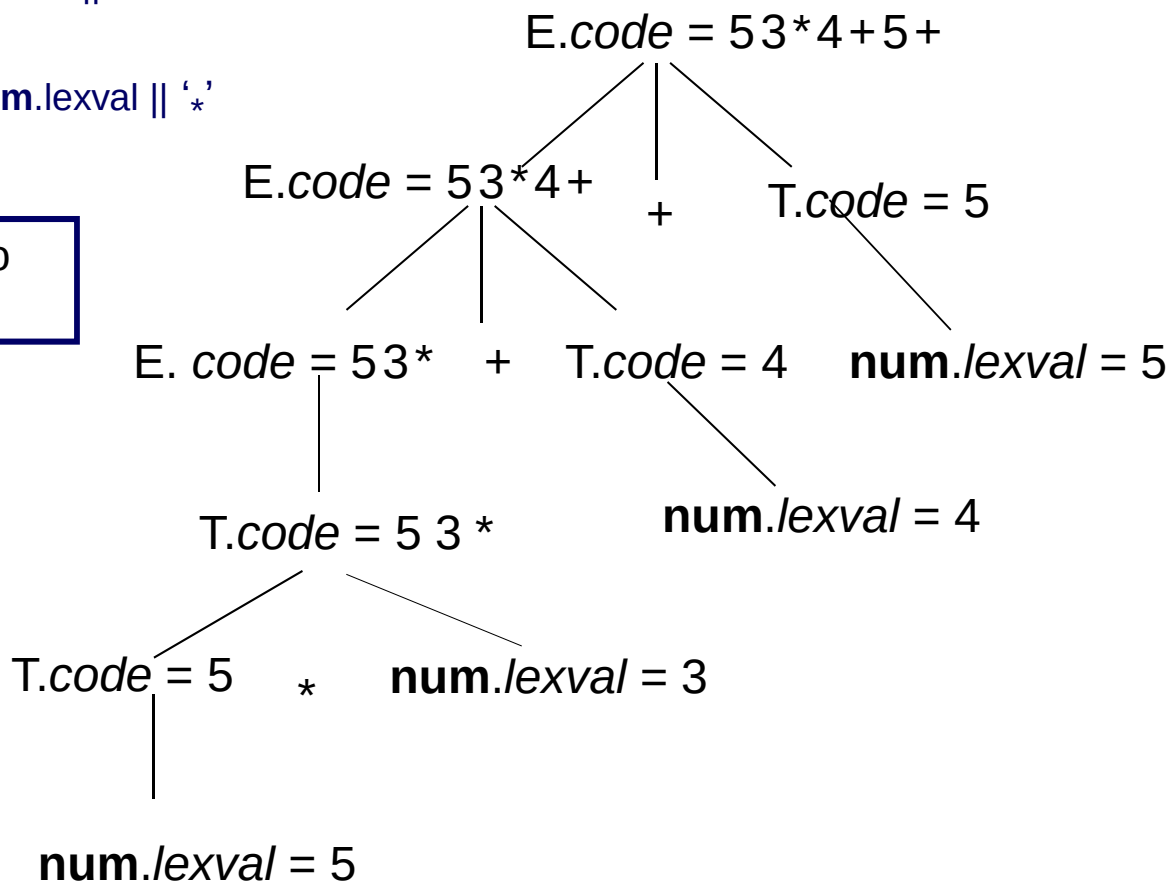
- L'operatore || rappresenta la concatenazione tra stringhe

Definizioni guidate dalla sintassi

Esempio: Notazione infissa → Notazione postfissa III

$E \rightarrow E_1 + T$	E.*code* = E_1.*code* || T.*code* || '+'
$E \rightarrow T$	E.*code* = T.*code*
$T \rightarrow T$***mun**	T.*code* = | T.*code* || **num**.lexval || '*'
$T \rightarrow$ **num**	T.*code* = **num**.*lexval*

Albero di parsificazione annotato per la stringa 5*3-4+5



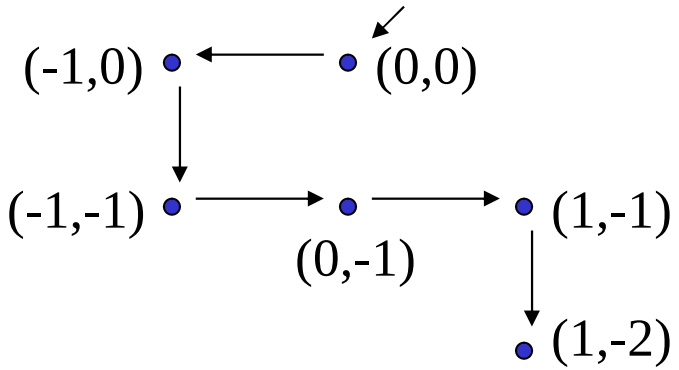
Definizioni guidate dalla sintassi

Esempio: Movimenti di un robot

Seq $\rightarrow$ Seq Istr | **begin**
Istr $\rightarrow$ **est** | **ovest** | **nord** | **sud**

begin ovest sud est est sud

Traduciamo la sequenza in una posizione relativa al punto di partenza:



Definizioni guidate dalla sintassi

Esempio: Movimenti di un robot

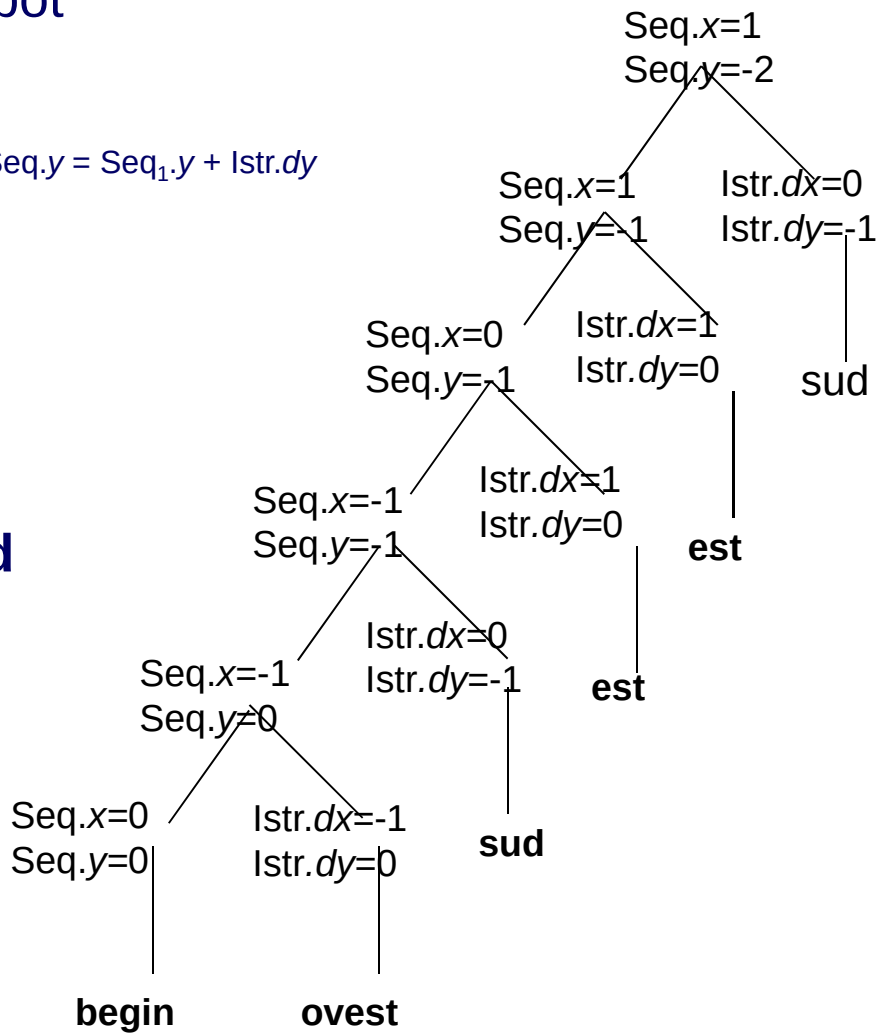
Produzioni	Regole semantiche
Seq $\rightarrow$ **begin**	Seq.*x* = 0 Seq.*y* = 0
Seq $\rightarrow$ Seq_1 Istr	Seq.*x* = Seq_1.*x* + Istr.*dx* Seq.*y* = Seq_1.*y* + Istr.*dy*
Istr $\rightarrow$ **est**	Istr.*dx* = 1, Istr.*dy* = 0
Istr $\rightarrow$ **ovest**	Istr.*dx* = -1, Istr.*dy* = 0
Istr $\rightarrow$ **nord**	Istr.*dx* = 0, Istr.*dy* = 1
Istr $\rightarrow$ **sud**	Istr.*dx* = 0, Istr.*dy* = -1

Definizioni guidate dalla sintassi

Esempio: Movimenti di un robot

Seq → **begin**	Seq.*x* = 0 ; Seq.*y* = 0
Seq → Seq_1 Istr	Seq.*x* = Seq_1.*x* + Istr.*dx* ; Seq.*y* = Seq_1.*y* + Istr.*dy*
Istr → **est**	Istr.*dx* = 1; Istr.*dy* = 0
Istr → **ovest**	Istr.*dx* = -1 ; Istr.*dy* = 0
Istr → **nord**	Istr.*dx* = 0 ; Istr.*dy* = 1
Istr → **sud**	Istr.*dx* = 0 ; Istr.*dy* = -1

begin ovest sud est est sud



Definizioni guidate dalla sintassi

E se volessimo una grammatica LL(1)? Eliminiamo la ricorsione sinistra (rinominando qualche variabile per migliorare la presentazione

Seq → **begin**
Seq → Seq_1 Istr

Seq → **begin** Seq'
Seq' → Istr Seq'
Seq' → ε

Mov→ **begin** Seq
Seq → Istr Seq
Seq → ε

Istr → **est**

Istr → **ovest**

Istr → **nord**

Istr → **sud**

Definizioni guidate dalla sintassi

Esempio: Movimenti di un robot II – con grammatica LL(1)

Produzioni	*Regole di traduzione*
Mov $\rightarrow$ **begin** Seq	Mov.*x* = Seq.fin*x* Mov.*y* = Seq.fin*y* *Seq.x =0* *Seq.y = 0*
Seq $\rightarrow$ instr Seq_1	Seq_1.*x* = Seq.*x* + Istr.*dx* Seq_1.*y* = Seq.*y* + Istr.*dy* Seq.fin*x* = Seq_1.fin*x* Seq.fin*y* = Seq_1.fin*y*
Seq $\rightarrow$ ε	Seq.fin*x* = Seq.*x* Seq.fin*y* = Seq.*y*
Istr $\rightarrow$ **est**	Istr.*dx* = 1, Istr.*dy* = 0
Istr $\rightarrow$ **ovest**	Istr.*dx* = -1, Istr.*dy* = 0
Istr $\rightarrow$ **nord**	Istr.*dx* = 0, Istr.*dy* = 1
Istr $\rightarrow$ **sud**	Istr.*dx* = 0, Istr.*dy* = -1

Definizioni guidate dalla sintassi

Esempio: Movimenti di un robot

La traduzione è la stessa ma si parte da una grammatica diversa (LL(1)).

Mov → **begin** Seq Mov.*x* = Seq.finx ; Mov.*y* = Seq.finy

Seq → Istr Seq_1 Seq_1.*x* = Seq.*x* + Istr.*dx* ; Seq_1.*y* = Seq.*y* + Istr.*dy*

Seq.finx = Seq_1.finx Seq.finy = Seq_1.finy

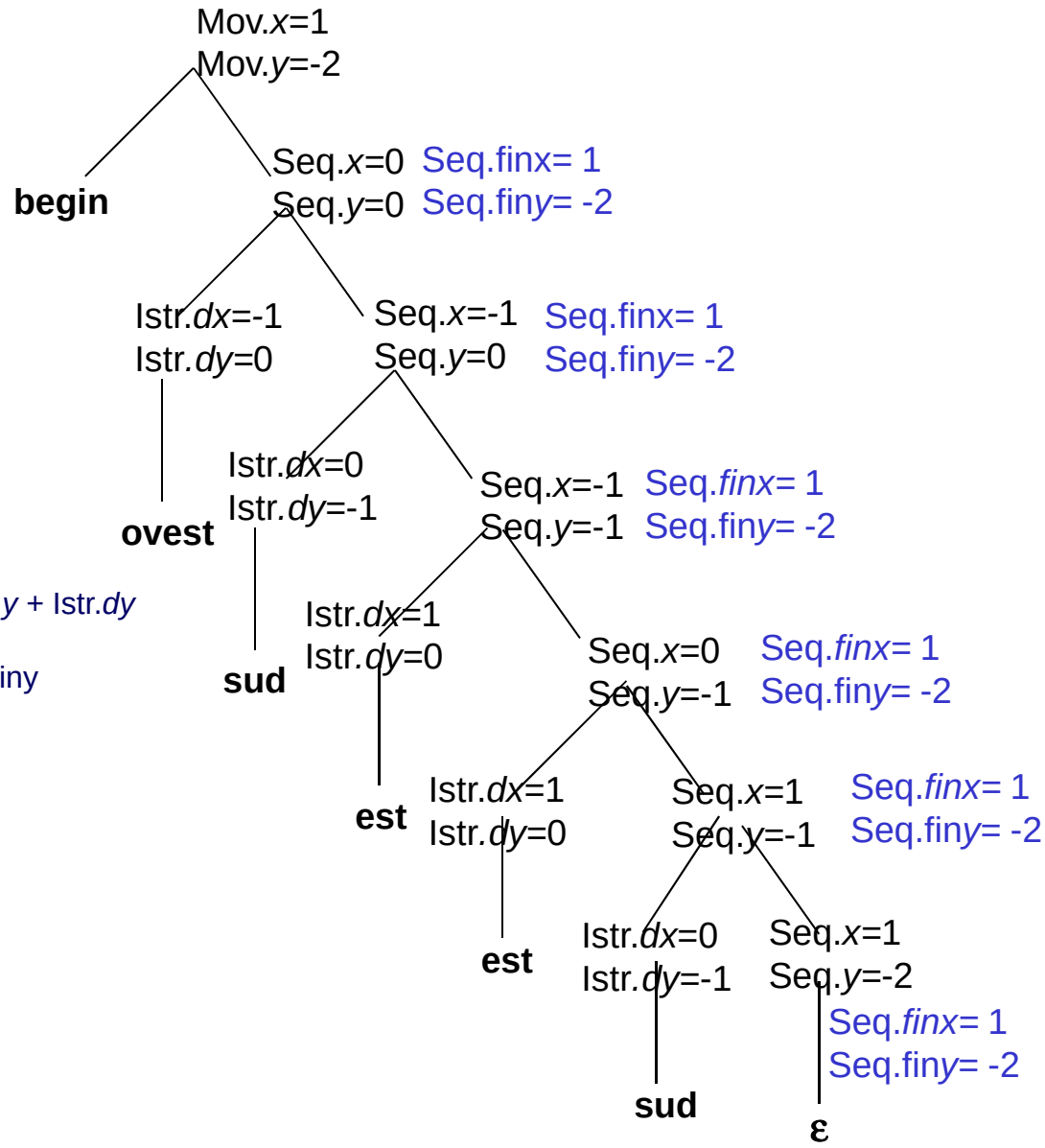
Seq → ε Seq.finx = Seq. x Seq.finy = Seq.y

Istr → **est** Istr.*dx* = 1; Istr.*dy* = 0

Istr → **ovest** Istr.*dx* = -1 ; Istr.*dy* = 0

Istr → **nord** Istr.*dx* = 0 ; Istr.*dy* = 1

Istr → **sud** Istr.*dx* = 0 ; Istr.*dy* = -1



begin ovest sud est est sud

Grafo delle dipendenze

Come costruire un albero annotato?
In quale ordine devono essere valutati gli attributi?

- prima di valutare un attributo in un nodo, si devono valutare gli attributi dai quali dipende il suo valore
- gli attributi sintetizzati possono essere valutati in ordine bottom-up
- negli SDD che hanno sia attributi sintetizzati, sia ereditati, non vi è garanzia che vi sia almeno un ordine in cui valutare gli attributi nei nodi perché potrebbero essere presenti regole “circolari” che rendono impossibile la valutazione.

Il problema di determinare se esiste una circolarità in qualche albero di parsificazione che una data SDD potrebbe dover tradurre è decidibile, ma ha complessità esponenziale in tempo.

Dato però un albero di parsificazione si può costruire un ***grafo delle dipendenze*** che mostri il flusso di informazione tra le istanze degli attributi. Tale grafo permette di scoprire se la valutazione è possibile e in tal caso trovare un ordine per la valutazione stessa.

Grafo delle dipendenze II

Costruzione del grafo delle dipendenze

Per ogni nodo *n* dell'albero di parsificazione etichettato X e
    per ogni attributo *a* del simbolo X della grammatica
        costruisci un nodo per X.*a*

Per ogni nodo *n* etichettato X in cui è applicata una produzione *p*
    se a *p* è associata una regola semantica che definisce l'attributo sintetizzato X.*c* in termini del valore Y.*b*, crea un arco da Y.*b* a X.*c*.

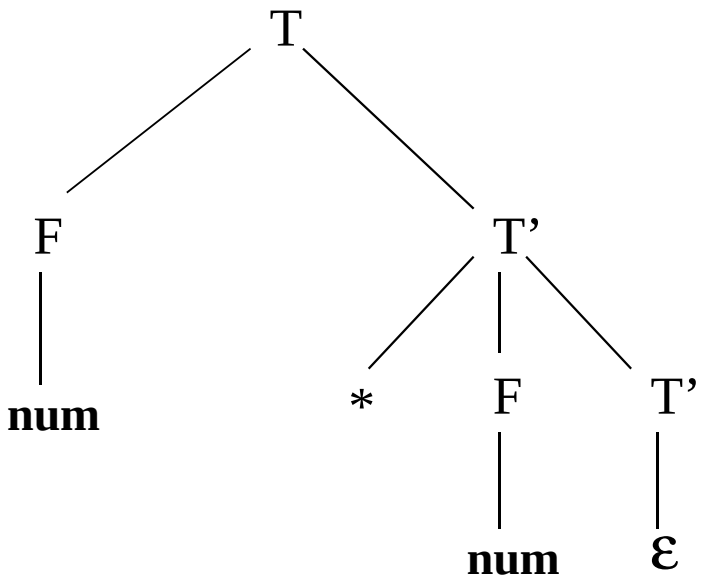
    se a *p* è associata una regola semantica che definisce l'attributo ereditato Z.*c* in termini del valore X.*a*, crea un arco da X.*a* a Z.*c*.

Se il grafo presenta un ciclo non è possibile valutare gli attributi. L'ordine di valutazione deve rispettare un ordinamento topologico dei vertici del grafo.

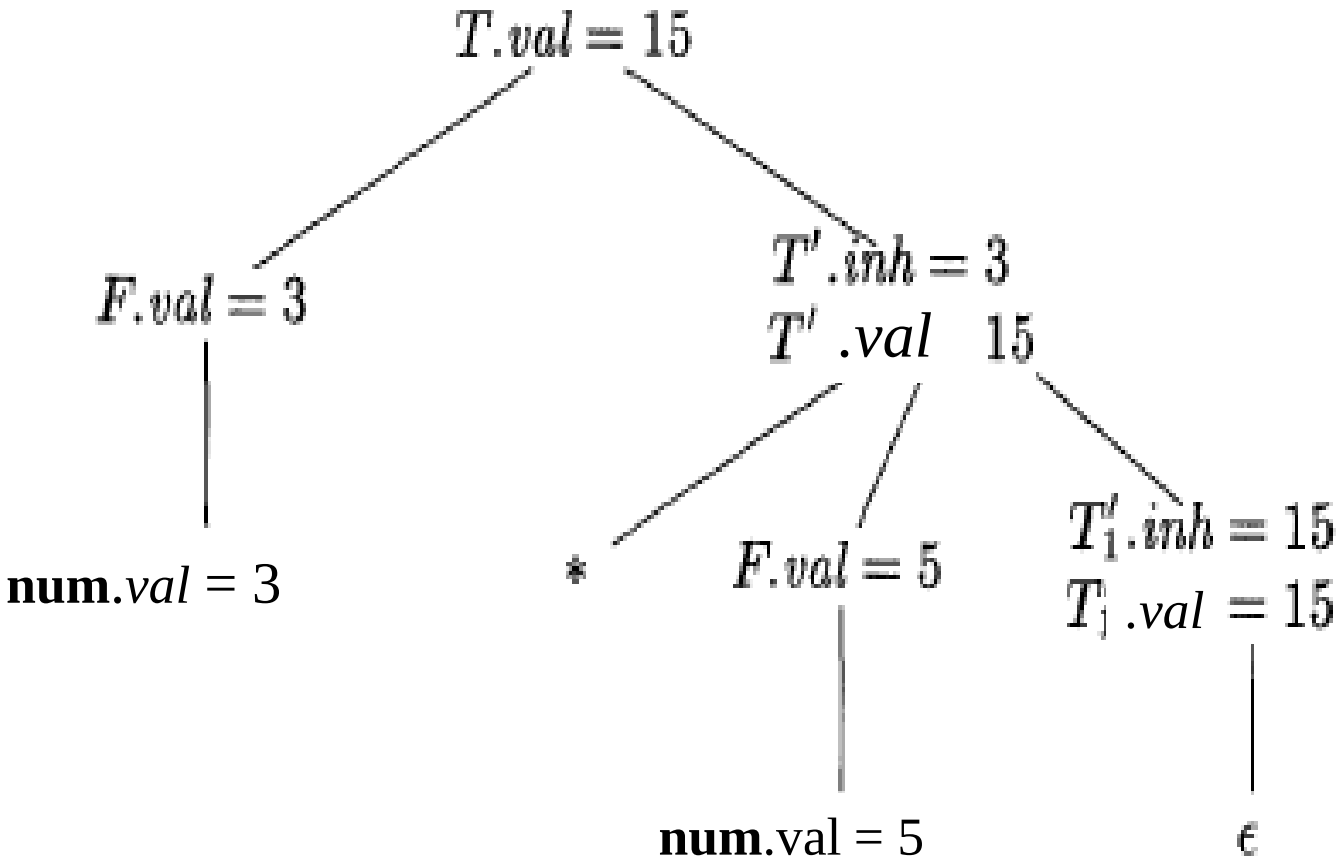
Esempio: espressioni aritmetiche semplificate

Produzioni	Regole semantiche
$T \rightarrow F\ T'$	$T'.inh = F.val$ $T.val = T'.s$
$T' \rightarrow * F\ T_1'$	$T_1'.inh = T'.inh \times F.val$ $T'.s = T_1'.s$
$T' \rightarrow \varepsilon$	$T'.s = T'.inh$
$F \rightarrow$ **num**	$F.val =$ **num**$.val$

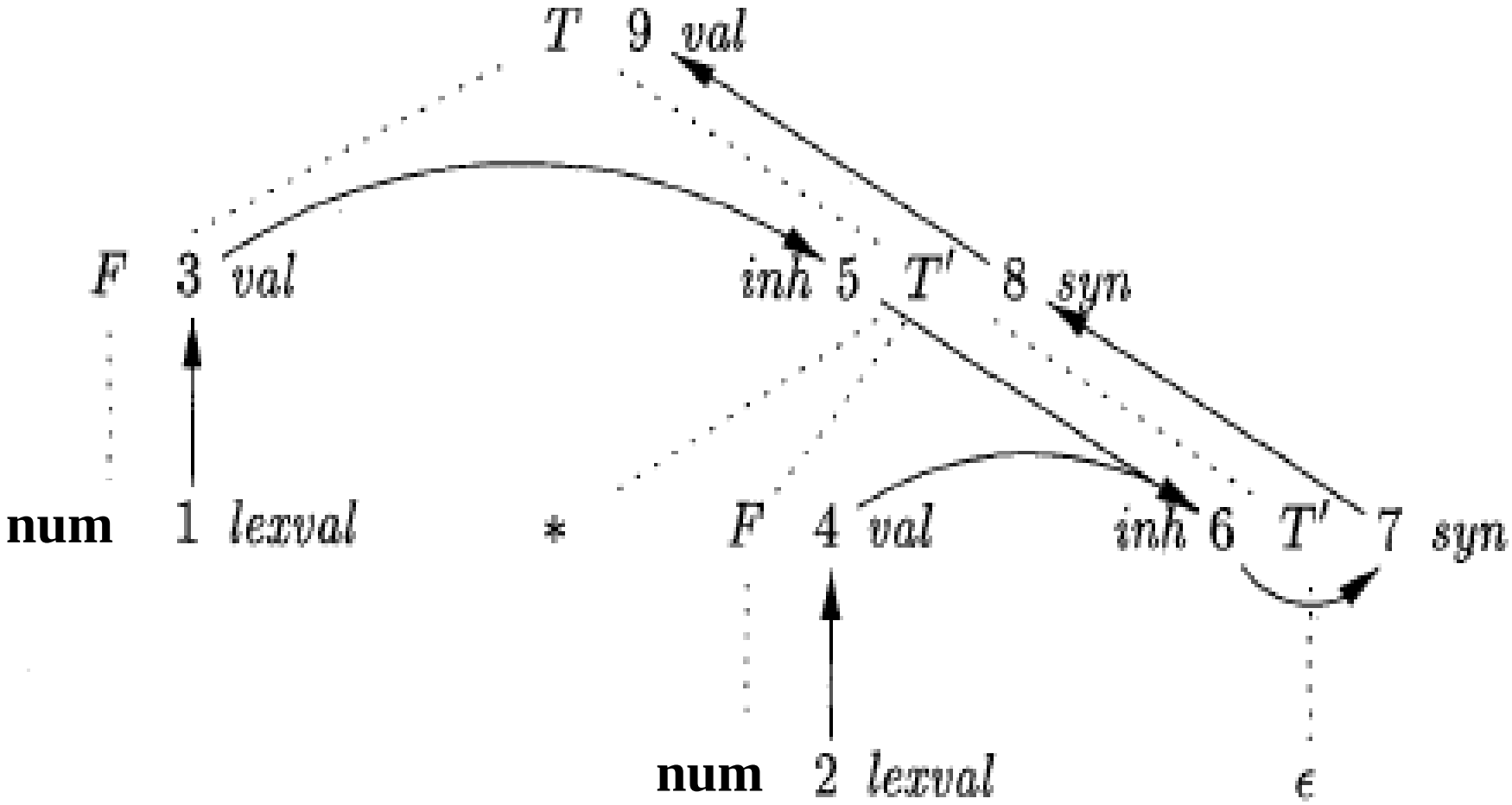
Albero di parsificazione della stringa **num** * **num**



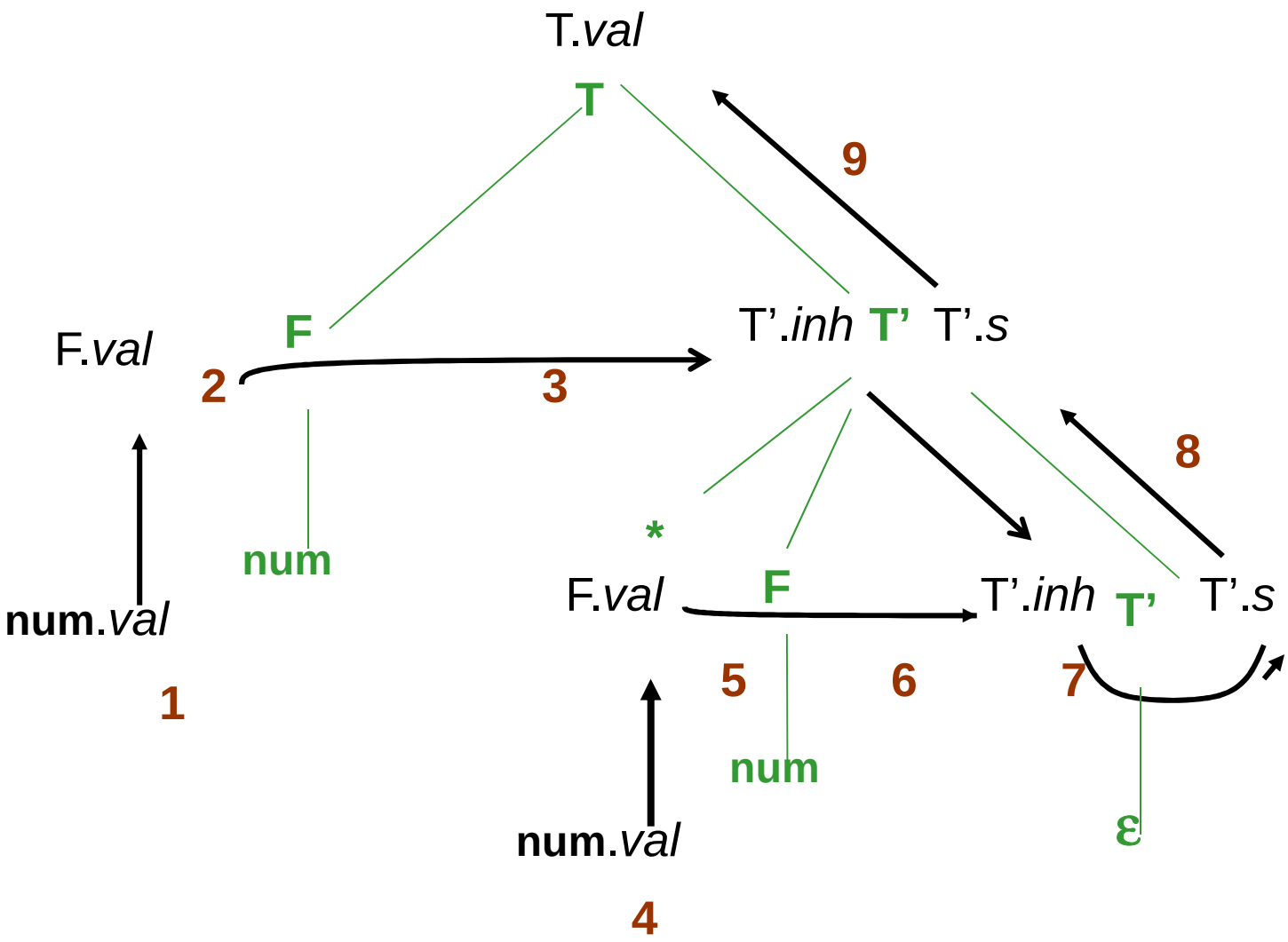
Esempio si albero annotato di 3*5

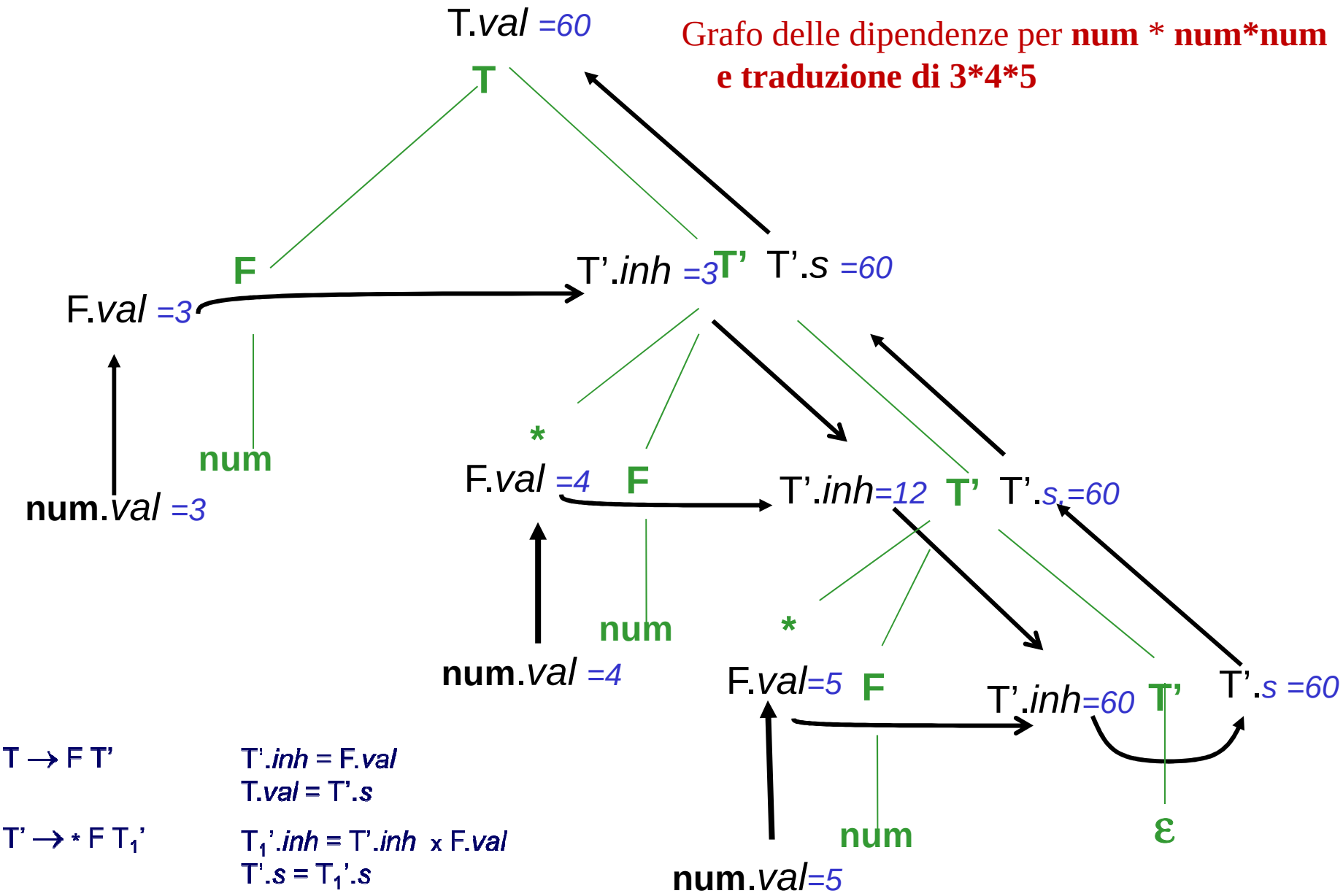


ordine di valutazione degli attributi



Grafo delle dipendenze per **num * num** e possibile ordine di valutazione





Grafo delle dipendenze per **num * num*num** **e traduzione di 3*4*5**

$T \rightarrow F\ T'$	$T'.inh = F.val$ $T.val = T'.s$
$T' \rightarrow * F\ T_1'$	$T_1'.inh = T'.inh \times F.val$ $T'.s = T_1'.s$
$T' \rightarrow \varepsilon$	$T'.s = T'.inh$
$F \rightarrow$ **num**	$F.val =$ **num**$.val$

Grafo delle dipendenze

Esempio: Lista delle differenze I

Input: stringhe formate da un numero, #, una sequenza di numeri separati da .
Prodotto della traduzione: la sequenza dei numeri ottenuti sottraendo il primo da tutti gli numeri, separati da '**,**'
Per esempio la traduzione di 4 # 6 ; 2 ; 8 è 2, -2, 4

Una grammatical per l'input è:

$C \rightarrow N\#L$

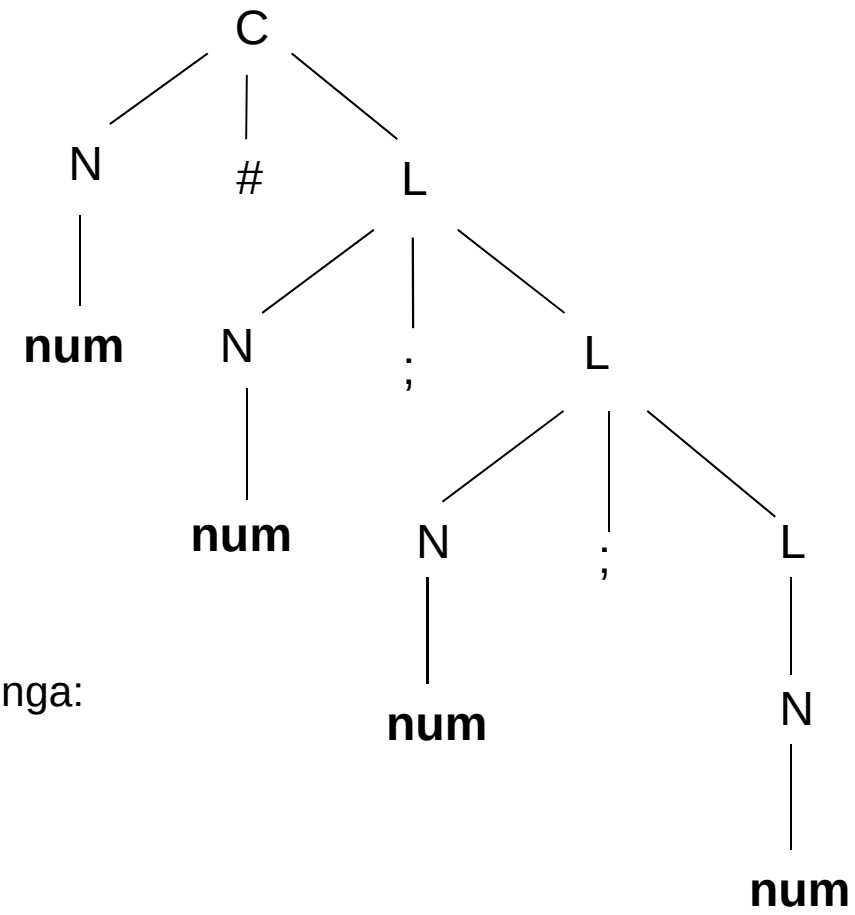
$L \rightarrow N;L_1$

$L \rightarrow N$

$N \rightarrow$ **num**

Esempio: lista delle differenze

Esempio di albero



Albero di parsificazione per la stringa:
num # **num**;**num**;**num**
Tokenizzazione **di 4 # 6 ; 2 ; 8**

- La sequenza si può costruire da L con un attributo sintetizzato (.*list*)
- L'informazione sul valore del primo elemento può essere passata con un attributi ereditato (.*elem*)

Grafo delle dipendenze

Esempio: Lista delle differenze I

Input: stringhe formate da un numero, #, una sequenza di numeri separati da .
Prodotto della traduzione: la sequenza dei numeri ottenuti sottraendo il primo da tutti gli numeri, separati da '**,**'
Per esempio la traduzione di 4 # 6 ; 2 ; 8 è 2, -2, 4

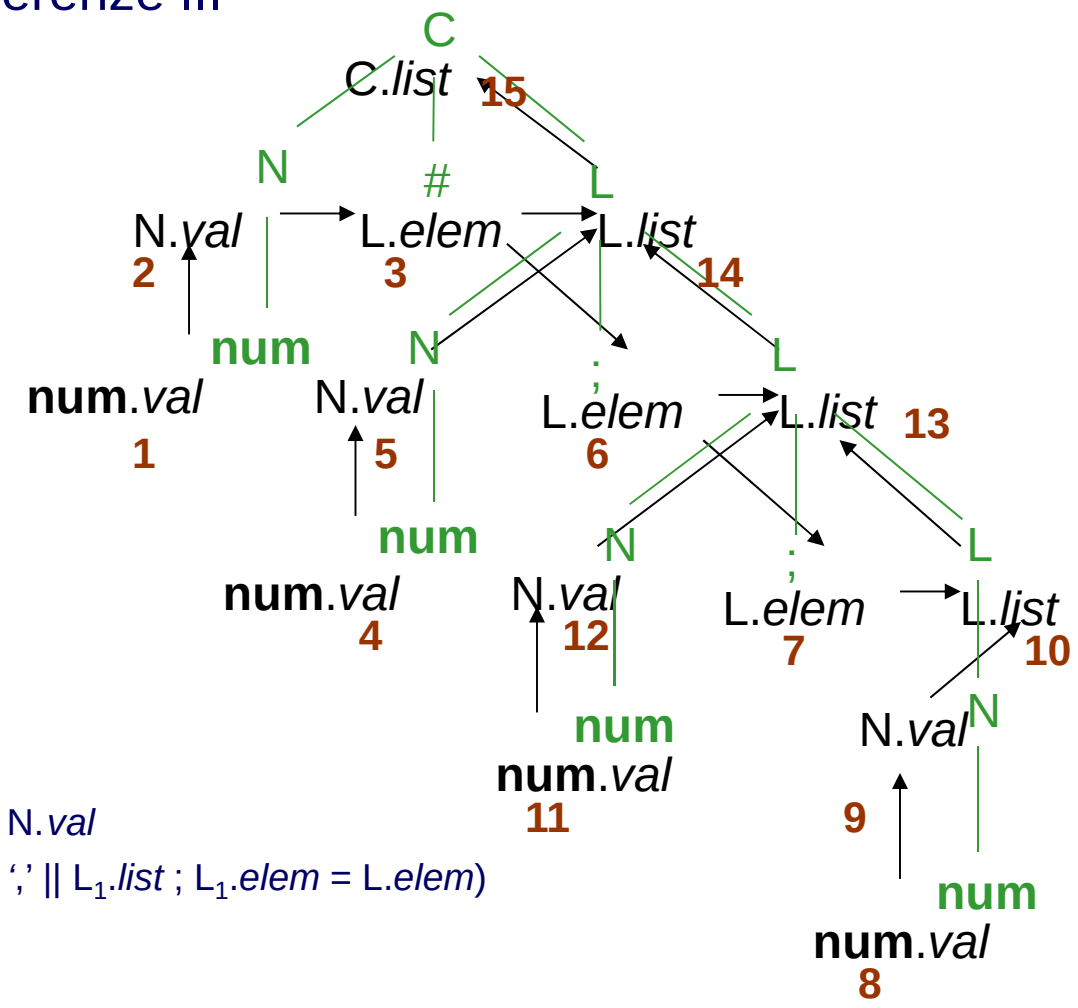
Produzioni	Regole semantiche
$C \rightarrow N\#L$	C.*list* = L.*list* ; L.*elem* = N.*val*
$L \rightarrow N;L_1$	L.*list* = N.*val* - L.*elem* || ' **,**' || L_1.*list* L_1.*elem* = L.*elem*)
$L \rightarrow N$	L.*list* = N.*val* - L.*elem*
$N \rightarrow$ **num**	N.*val* = **num**.*lexval*

list e *val*: sintetizzati
elem: ereditato

Grafo delle dipendenze

Esempio: Lista delle differenze III

Grafo delle dipendenzeper la stringa:
num # **num**;**num**;**num**



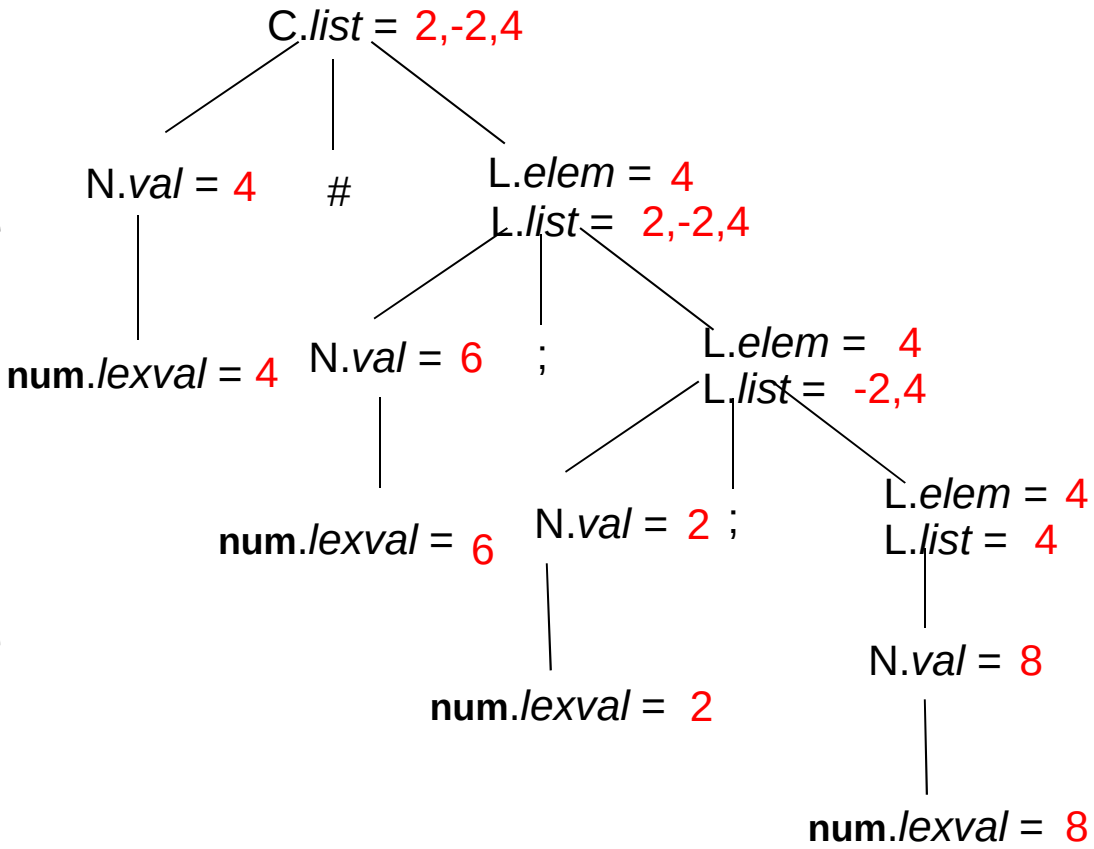
$C \rightarrow N\#L$	C.*list* = L.*list* ; L.*elem* = N.*val*
$L \rightarrow N;L_1$	L.*list* = N.*val* - L.*elem* || ‘,’ || L_1.*list* ; L_1.*elem* = L.*elem*)
$L \rightarrow N$	L.*list* = N.*val* - L.*elem*
$N \rightarrow$ **num**	N.*val* = **num**.*lexval*

Albero annotato con i valori degli attributi

Esempio: Lista delle differenze IV

$C \rightarrow N\#L$	C.*list* = L.*list* ; L.*elem* = N.*val*
$L \rightarrow N;L_1$	L.*list* = N.*val* - L.*elem* || ',' || L_1.*list* L_1.*elem* = L.*elem*)
$L \rightarrow N$	L.*list* = N.*val* - L.*elem*
$N \rightarrow$ **num**	N.*val* = **num**.*lexval*

Albero annotato per la stringa
4 # 6 ; 2 ; 8, fornita al parsificatore dall'analizzatore lessicale come
num # **num** ; **num** ; **num**



La traduzione di 4 # 6 ; 2 ; 8 è
2 , -2 , 4

Definizioni S-attribuite

Una SDD è **S-attribuita** se **tutti gli attributi sono sintetizzati**

Esempio:

$E \rightarrow E_1 + T$	E.*val* = E_1.*val* + T.*val*	{E.*val* = E_1.*val* + T.*val*}
$E \rightarrow E_1 - T$	E.*val* = E_1.*val* - T.*val*	{E.*val* = E_1.*val* - T.*val*}
$E \rightarrow T$	E.*val* = T.*val*	{E.*val* = T.*val*}
$T \rightarrow (E)$	T.*val* = E.*val*	{T.*val* = E.*val*}
$T \rightarrow$ **num**	T.*val* = num.*val*	{T.*val* = num.*val*}

Definizioni L-attribuite

Una SDD è **L-attribuita** se gli attributi sono
- **sintetizzati** oppure
- **ereditati** e soddisfano il seguente vincolo:

per ogni produzione $A \rightarrow X_1 X_2 \ldots X_n$, ogni attributo ereditato di X_j dipende solo da:

- attributi ereditati o sintetizzati dei simboli $X_1 X_2 \ldots X_{j-1}$ a sinistra di X_j nella produzione;
- attributi ereditati di A
- attributi ereditati o sintetizzati di X_j, purchè non vi siano cicli nel grafo delle dipendenze formati dagli attributi di questa occorrenza di X_j.

Nel grafo delle dipendenze gli archi tra gli attributi associati a una produzione possono andare da sinistra a destra, ma non da destra a sinistra (**L**eft to right).

Esempio di definizione L-attribuita

Produzioni	Regole semantiche
1. C $\rightarrow$ N#L	C.*list* = L.*list* ; L.*elem* = N.*val*
2. L $\rightarrow$ N;L_1	L.*list* = N.*val* - L.*elem* || ',' || L_1.*list* ; L_1.*elem* = L.*elem*)
3. L $\rightarrow$ N	L.*list* = N.*val* - L.*elem*
4. N $\rightarrow$ **num**	N.*val* = **num**.*val*

list e *val*: sintetizzati
elem: ereditato

- L'attributo ereditato di L (*elem*) nella produzione 1. dipende da quello sintetizzato (*val*) di N
-L'attributo sintetizzato (*list*) di L nelle produzione 2. e 3. dipende oltre che da quello sintetizzato di N (*val*) da quello ereditato di L (*elem*) ma non c'è circolarità immediata.

Traduzioni con ‘’side-effects”

Esempio: costruzione di alberi di sintassi ‘’astratta”
per espressioni aritmetiche.

Alcuni compilatori usano una forma di rappresentazione derivata direttamente dall’albero sintattico come rappresentazione intermedia. Si può usare una SDD per associare ad ogni stringa in input tale rappresentazione (spesso riferita come ‘’sintassi astratta”).

Funzioni usate:

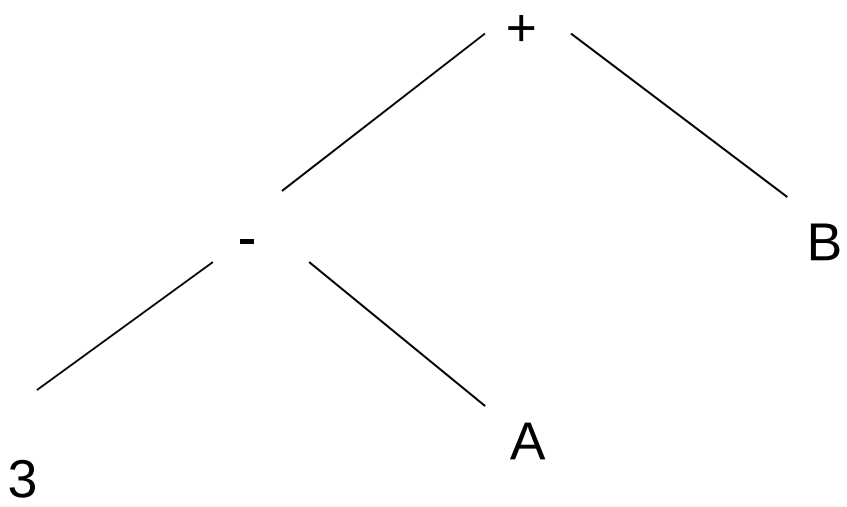
node (*op, left, right*): crea un nodo operatore con label *op* e i puntatori agli operandi, restituisce il puntatore al nodo creato.

leaf (**id**, *entry*): crea un nodo identificatore con label **id** e un puntatore alla entry per l’identificatore nella symbol-table, restituisce il puntatore al nodo creato.

leaf (**num**, *val*): crea un nodo numero con label **num** e il valore del numero, restituisce il puntatore al nodo creato.

Esempio

Rappresentandolo in forma semplificata, l'albero risultante dalla traduzione di **3-A+B** diventa:



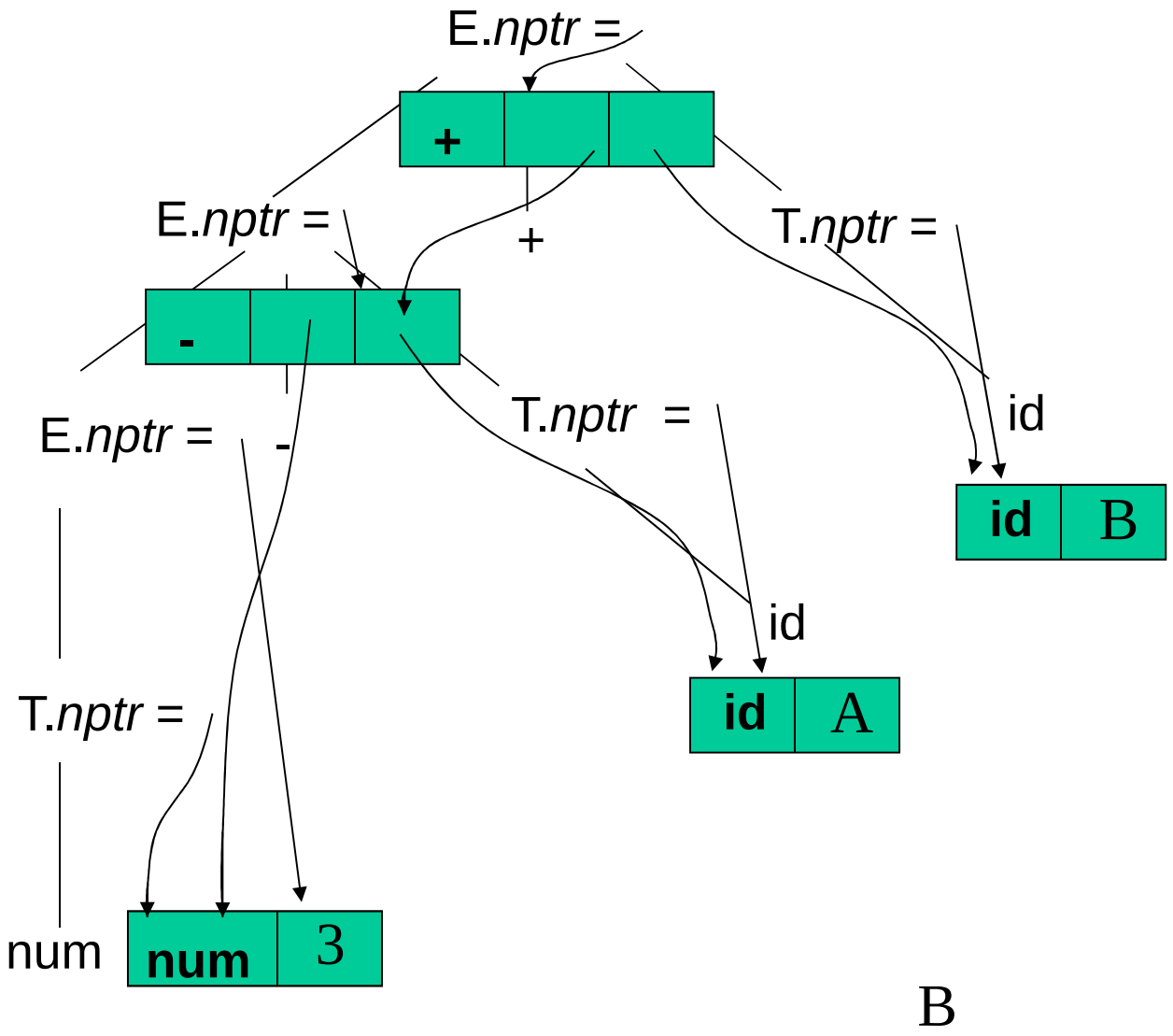
Traduzioni con ‘’side-effects”

Esempio: costruzione di alberi di sintassi ‘’astratta’’ per espressioni aritmetiche.

Produzioni	Regole semantiche
$E \rightarrow E_1 + T$	E.*nptr* = **new** *node* (‘+’, E_1.*nptr*, T.*nptr*)
$E \rightarrow E_1 - T$	E.*nptr* = **new** *node* (‘-’, E_1.*nptr*, T.*nptr*)
$E \rightarrow T$	E.*nptr* = T.*nptr*
$T \rightarrow (E)$	T.*nptr* = E.*nptr*
$T \rightarrow$ **id**	T.*nptr* = **new** *leaf* (**id**, **id**.*entry*)
$T \rightarrow$ **num**	T.*nptr* = **new** *leaf* (**num**, **num**.*val*)

Tutti gli attributi sono sintetizzati.

Esempio: traduzione di 3+ A +B (**num-id+id)**



Traduzioni con ‘’side-effects” dichiarazioni di tipo

Con side effect intendiamo la possibilità di agire su dati globali.

Il seguente SDD processa dichiarazioni ‘’alla C”. LA funzione addtype aggiunge l’informazione di tipo nella tabella dei simboli.

$D \rightarrow T\ L$	L.*type* = T.*type*
$T \rightarrow$ **int**	T.*type* = integer
$T \rightarrow$ **float**	T.*type* = floating
$L \rightarrow$ **id** , L_1	L_1.*type* = L.*type* *addtype*(**id**.*lexval*, L.*type*)
$L \rightarrow$ id	*addtype*(id.*lexval*, D.*type*)

Traduzioni con ‘’side-effects” dichiarazioni di tipo

Vediamo come funzona sull’input ’int a, b, c’: