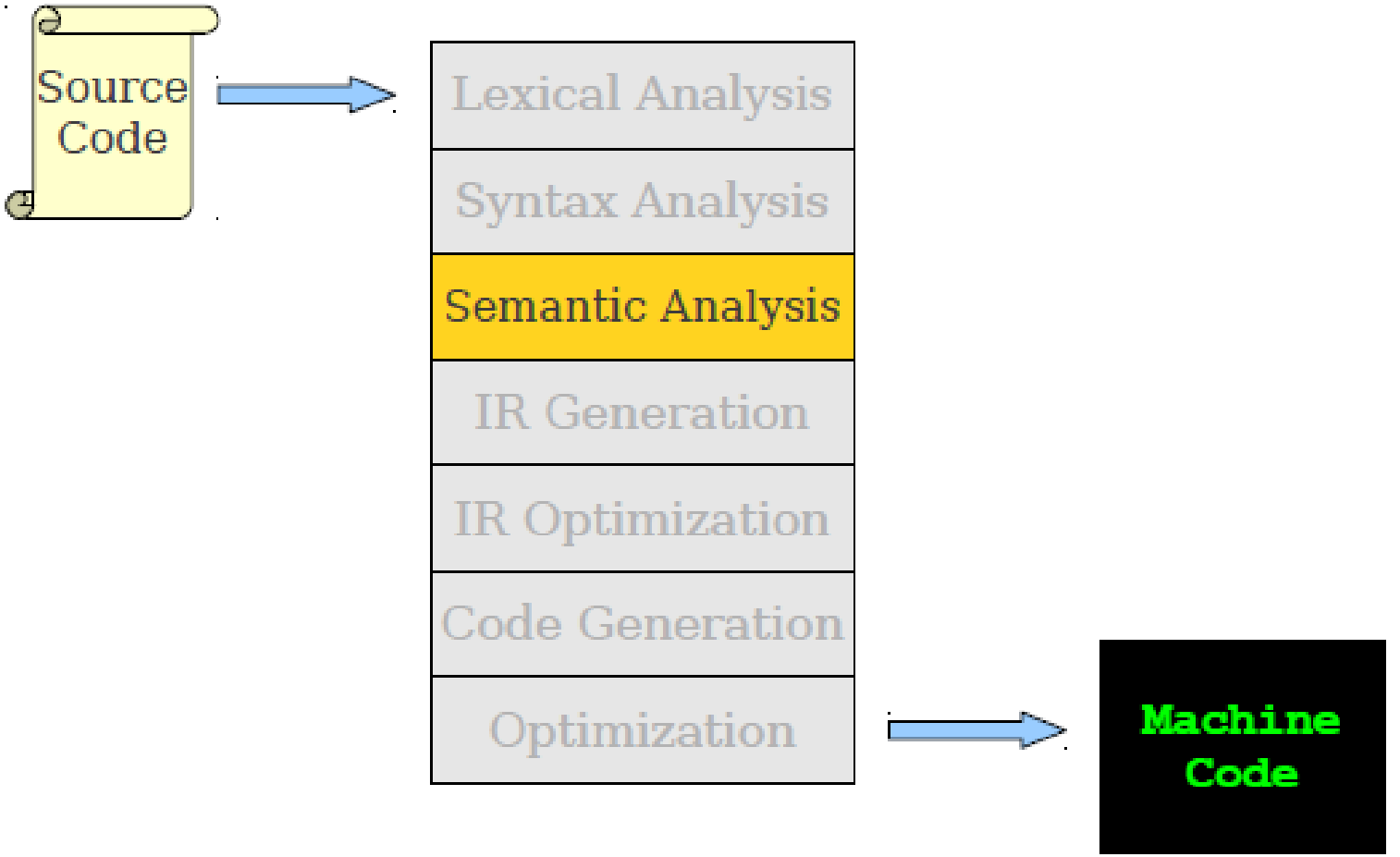
Analisi Semantica

Lunedì 10 Dicembre

Dove siamo arrivati?



Cos'è?

- L'analisi semantica interpreta il significato associato alla struttura sintattica e verifica che le regole di impiego del linguaggio siano soddisfatte.
- Assicura che il programma analizzato abbia un significato ben definito.

Obiettivi

- Verifica ciò che non può essere controllato nelle fasi precedenti, ovvero per esempio:
  - Le variabili sono state dichiarate prima di essere usate
  - Le espressioni hanno i tipi corretti
  - Una variabile o una classe non sia definita due volte
  - Variabili di tipo diverso siano diverse
  - Una classe implementi tutti i metodi
  - …
- Raccogliere le informazioni relative agli identificatori introdotti nella tabella dei simboli;
- Verificare la correttezza d'impiego degli identificatori e dei costrutti del linguaggio;
- Segnalare gli errori in modo chiaro e senza ridondanze.

Semantica statica o dinamica?

Semantica statica

- Indipendente dai dati su cui opera il programma sorgente:
- verificare che una variabile sia dichiarata una sola volta e che venga usata coerentemente al tipo dichiarato;
- rispetto delle regole che governano i tipi degli operandi nelle espressioni e negli assegnamenti (controllo sui tipi);
- rispetto delle regole di visibilità e univocità degli identificatori;
- correttezza delle strutture di controllo del linguaggio;
- rispetto delle regole di comunicazione fra i vari moduli (interni e/o esterni) che costituiscono il programma.
- verificare che le chiamate dei sottoprogrammi siano congruenti con le loro dichiarazioni.

Semantica dinamica

- dipendente dai dati su cui opera il programma sorgente:
- controllo sui cicli infiniti;
- controllo sulla dereferenziazione di puntatori NULL;
- controllo sui limiti degli array (per esempio, l'indice di un array non superi i limiti stabiliti dallla sua dichiarazione);
- un dato letto in input sia compatibile con il tipo della variabile a cui è destinato.

L'analizzatore semantico si occupa della semantica statica, mentre la semantica dinamica spetta all'interprete o al supporto esecutivo.

Esempio di semantica statica e dinamica

- Semantica statica: indipendente dai dati su cui opera il programma sorgente.

```
var  i : real;
    a : array [1..100] of integer;
     ……
     i:=3.5;
     a[i]:=3;
```

- Semantica dinamica: dipendente dai dati su cui opera il programma sorgente.

```
var  i : integer;
    a : array [1..100] of integer;
     ……
     read(i);
     a[i]:=3;
```

Analisi semantica statica

- L'analisi semantica statica, come quella lessicale e sintattica, consta di due fasi:
  - la descrizione delle analisi da eseguire
  - l'implementazione delle analisi mediante opportuni algoritmi.

Analisi semantica statica

- Nell'analisi sintattica esistono formalismi standard per descrivere la sintassi e i vari algoritmi di parsing per implementare la sintassi stessa.

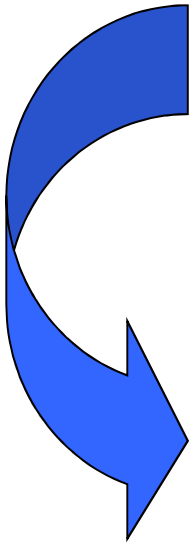
- Nell'analisi semantica la situazione non è così definita:
  - non esiste una metodologia standard per definire o descrivere la semantica statica di un linguaggio;
  - esiste un'enorme varietà di controlli semantici statici nei vari linguaggi.

Analisi semantica statica

- L’idea di base, è in ogni caso, quella di aumentare il lavoro del parser con azioni speciali di tipo semantico.

Metodi per la descrizione della semantica

- Esistono diversi approcci per la descrizione della semantica statica di un linguaggio.
  Uno tra questi si definisce attraverso



  - **Le grammatiche con attributi (attribute grammars)**

Grammatiche con attributi

- Sono grammatiche context-free in cui sono state aggiunte proprietà delle entità sintattiche del linguaggio (**attributi**) e regole di valutazione di tali proprietà (**regole semantiche o equazioni di attributi**).
- Una grammatica con attributi specifica quindi sia azioni sintattiche che semantiche.
- Le grammatiche con attributi sono utilizzabili in tutti i linguaggi di programmazione che obbediscono al principio della "SEMANTICA GUIDATA DALLA SINTASSI" che asserisce che la semantica non dipende dal contesto ma è strettamente legata alla sintassi.
- Ciò accade per tutti i moderni linguaggi di programmazione.

Attributi

- Un attributo è qualunque proprietà delle entità sintattiche di un linguaggio.
- Gli attributi possono variare molto rispetto al loro contenuto, alla loro complessità e principalmente in relazione al momento in cui essi sono calcolati.
- Gli algoritmi per l'implementazione dell'analisi semantica non sono chiaramente esprimibili come quelli di parsing.
- Esempi di attributi sono:
  - Il tipo di una variabile
  - Il valore di una espressione
  - La locazione di una variabile in memoria
  - Il codice oggetto di una procedura
  - ……

Attributi

- Gli attributi possono essere:
  - STATICI: calcolati al tempo di compilazione (i tipi di dati, il numero delle cifre significative…)
  - DINAMICI: calcolati al tempo di esecuzione (valore di una espressione, le locazioni di memoria di una struttura dati dinamica)
- Il calcolo di un attributo e l'associazione del suo valore ad un costrutto sintattico è detto **binding** (legame) dell'attributo.
- Il momento in cui questo legame avviene è detto **binding time**. Differenti attributi possono avere binding time diversi, o anche lo stesso attributo può avere differenti binding time che variano da linguaggio a linguaggio.
- Noi siamo interessati agli attributi statici.

Binding time

- In linguaggi dichiarativi come C e Pascal, il tipo di una variabile o di un'espressione è un attributo definito al tempo di compilazione. Il type checker (analizzatore semantico che calcola tale attributo per tutte le entità del linguaggio per le quali è definito e verifica che tali tipi siano conformi alle regole dei tipi del linguaggio) in C e in Pascal agisce durante la fase di compilazione, mentre in linguaggi come LISP o alcuni linguaggi ad oggetti tale processo avviene durante l'esecuzione.

- Il valore di un'espressione è generalmente calcolato al tempo di esecuzione. In alcuni casi però (3+4*5 per esempio) l'analizzatore semantico può scegliere di valutare l'espressione durante la compilazione.

- L'allocazione di una variabile può essere sia statica che dinamica e dipende dal linguaggio e dal tipo di variabile: in FORTRAN tutte le variabili sono statiche, in LISP sono tutte dinamiche mentre in C e Pascal possono essere sia statiche che dinamiche.

Grammatiche con attributi

- Sono grammatiche context-free in cui sono state aggiunte proprietà delle entità sintattiche del linguaggio (**attributi**) e regole di valutazione di tali proprietà (**regole semantiche o equazioni di attributi**).

- Una grammatica con attributi è quindi una terna **(G, A, R)** dove :
  - **G** è una grammatica context-free
  - **A** è l'insieme degli attributi associato ad ogni simbolo terminale e non
  - **R** è l'insieme di regole associate alle varie produzioni di G.

Rappresentazione di un attributo

- In queste ipotesi gli attributi sono associati direttamente ai simboli della grammatica (terminali e non).

- Se X è un simbolo sintattico e ‘a’ è un attributo associato ad X scriveremo:

  X.a

  per accedere al valore corrispondente.

Grammatiche con attributi e semantica guidata dalla sintassi

- Dato un insieme di attributi

  $$a_1, a_2, \ldots\ldots, a_k$$

  il principio della **semantica guidata dalla sintassi** afferma che per ogni produzione del tipo

  $$X_0 \rightarrow X_1 \ldots. X_n$$

  i valori degli attributi $X_i.a_j$ per ogni simbolo sintattico X_i sono legati ai valori degli attributi degli altri simboli presenti nella produzione.

  Questo legame è definito nella forma:

  $$X_i.a_j = f_{ij}\,(X_0.a_0, X_0.a_1,\ldots, X_0.a_k,\ldots, X_n.a_0, X_n.a_1,\ldots, X_n.a_k)$$

  e costituisce una **regola semantica (attribute equation)**

- Si definisce grammatica con attributi l'insieme di tali regole per ogni produzione del linguaggio.

Grammatiche con attributi

- Le grammatiche con attributi sono specificate mediante tabelle, in cui, accanto ad ogni produzione, sono elencate le regole semantiche associate.

regola grammaticale	regole semantiche
regola 1	equazione 1.1 equazione 1.2 equazione 1.3
regola n	equazione n.1 equazione n.2

Osservazioni

- Sono uno strumento molto potente per l'analisi semantica.
- Pur non essendo semplici da usare, le grammatiche con attributi sono meno complesse di quanto sembrano:
  - di solito gli attributi sono pochi
  - le regole semantiche dipendono raramente da tutti gli attributi
  - spesso gli attributi possono essere separati in sottoinsiemi e le regole semantiche possono essere scritte separatamente per ogni sottoinsieme
- I manuali dei linguaggi di programmazione non definiscono la grammatica con attributi, sicché il progettista del compilatore deve scriversela a mano.
- Nonostante ciò, è importante studiare le grammatiche con attributi perché consentono di definire analisi semantiche più semplici, concise, con meno errori, e che consentono una più semplice comprensione del codice.

Esempio 1

- Si consideri la seguente grammatica per esprimere un numero in binario:

  number → *number digit* | *digit*

  digit → 0 | 1

- Un attributo significativo potrebbe essere il suo valore in decimale.

number → number digit | digit

digit → 0 | 1

- Definiamo un attributo ***val*** per i simboli non-terminali ***number*** e ***digit*** (number.val e digit.val).

- Come diventa la grammatica con attibuto ***val*** ?

regola grammaticale	regole semantiche
$number_1 \rightarrow number_2\ digit$	$number_1.val = 2*number_2.val + digit.val$
$number \rightarrow digit$	$number.val = digit.val$
$digit \rightarrow 1$	$digit.val = 1$
$digit \rightarrow 0$	$digit.val = 0$

1. La regola **digit → 1** implica che digit ha il valore che la cifra stessa rappresenta, e quindi **digit.val = 1**
2. Analogamente **digit → 0** implica che **digit.val=0**
3. Se il numero è derivato usando la regola ***number → digit*** allora il suo valore è **number.val=digit.val**
4. Consideriamo che il numero sia derivato usando la regola

   number → number digit

   Riscriviamo la regola distinguendo le due occorrenze di number:

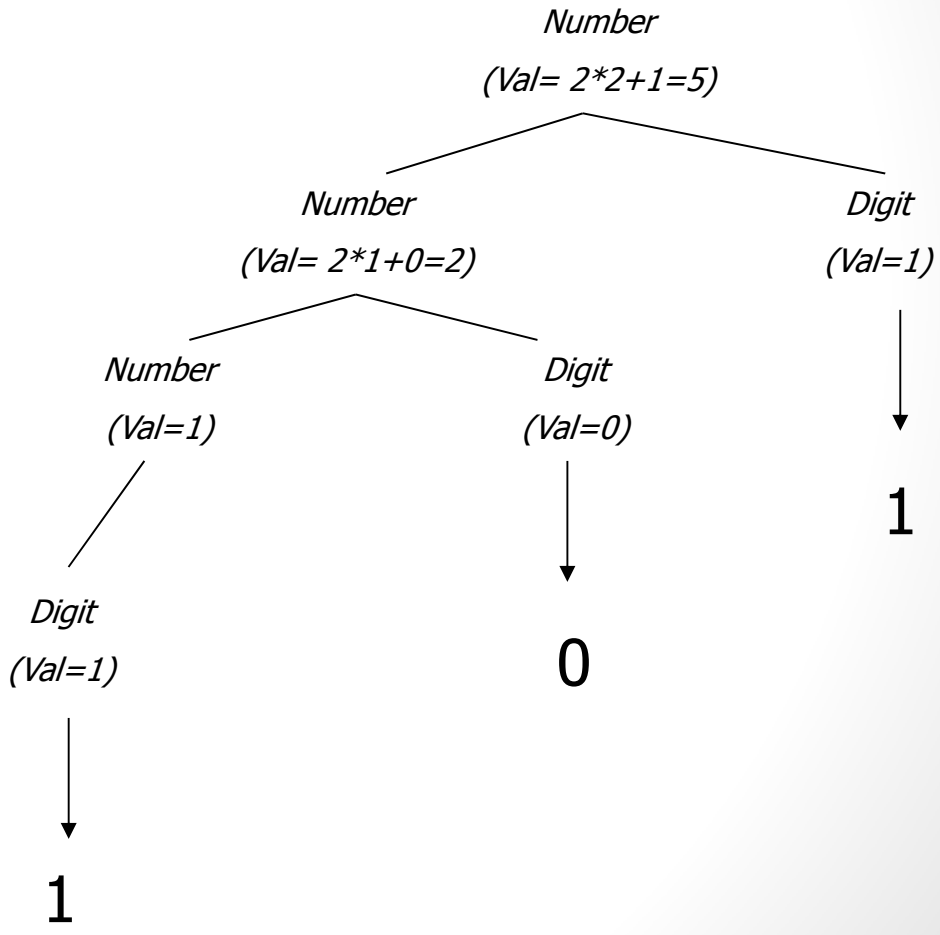
   $number_1$ → $number_2$ digit

   da cui si ottiene ***$number_1$ .val → 2 * $number_2$.val + digit.val***

Occorre notare la differenza tra la rappresentazione sintattica di digit e il suo contenuto semantico (valore). Nella regola **digit → 1** il simbolo 1 è un token mentre in **digit.val = 1** il simbolo 1 il valore numerico.

Albero sintattico decorato

- Il significato delle regole semantiche per una particolare stringa può essere descritto usando l'albero sintattico associato agli attributi (**albero sintattico decorato**).
- Per esempio descriviamo l'albero sintattico decorato per la stringa 101.
- Il calcolo dell'apposita regola semantica è indicato all'interno del nodo.
- E' importante osservare come avviene il calcolo degli attributi per avere un'idea di come possano funzionare gli algoritmi di calcolo degli attributi stessi.



Esempio 2

- Consideriamo la seguente grammatica per semplici espressioni aritmetiche:

exp → *exp* **+** *term* | *exp* **–** *term* | *term*

term → *term* * *factor* | *factor*

factor → **(***exp* **)** | ***number***

- L'attributo considerato è sempre *val* cioè il valore numerico di *exp* (*term* o *factor*)

exp → exp + term | exp – term | term

*term → term * factor | factor*

*factor → (exp) | **number***

- La grammatica con attributi corrispondente è:

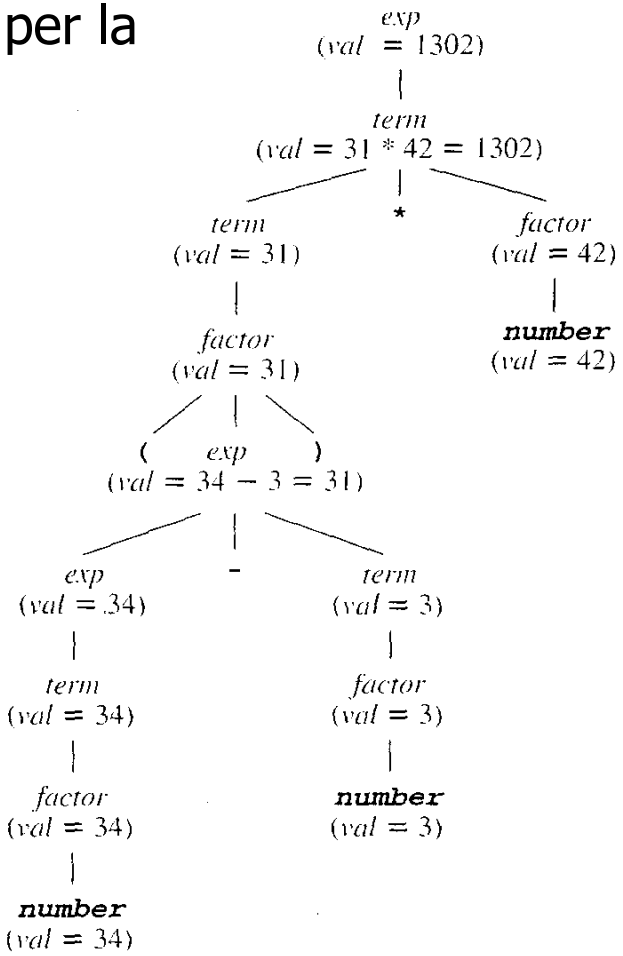
regola grammaticale	regole semantiche
$exp_1 \rightarrow exp_2 + term$	$exp_1.val = exp_2.val + term.val$
$exp_1 \rightarrow exp_2 - term$	$exp_1.val = exp_2.val - term.val$
$exp \rightarrow term$	$exp.val = term.val$
$term_1 \rightarrow term_2 * factor$	$term_1.val = term_2.val * factor.val$
$term \rightarrow factor$	$term.val = factor.val$
$factor \rightarrow (exp)$	$factor.val = exp.val$
$factor \rightarrow number$	$factor.val = number.val$

- ***number*** è considerato come simbolo terminale; ciò significa che sarà l'analizzatore lessicale, ad esempio, ad inizializzare opportunamente il campo **number.val**

- Alternativamente bisogna introdurre produzioni esplicite e corrispondenti regole semantiche (per esempio le regole dell'esempio precedente)

Albero sintattico decorato

Descriviamo l'albero sintattico decorato per la stringa (34-3)*42



Esempio 3

- Completiamo la grammatica dell'esempio 1 in modo da esprimere numeri in codice binario (b) e ternario (t):

based_num → *number base*

base → ***b*** | ***t***

number → *number digit* | *digit*

digit → 0 | 1 | 2

- Gli attributi sono:

based_num	:	val
base	:	val
number	:	val , base
digit	:	val , base

based_num → number base

*base → **b** | **t***

number → number digit | digit

digit → 0 | 1 | 2

- La grammatica con attributi è:

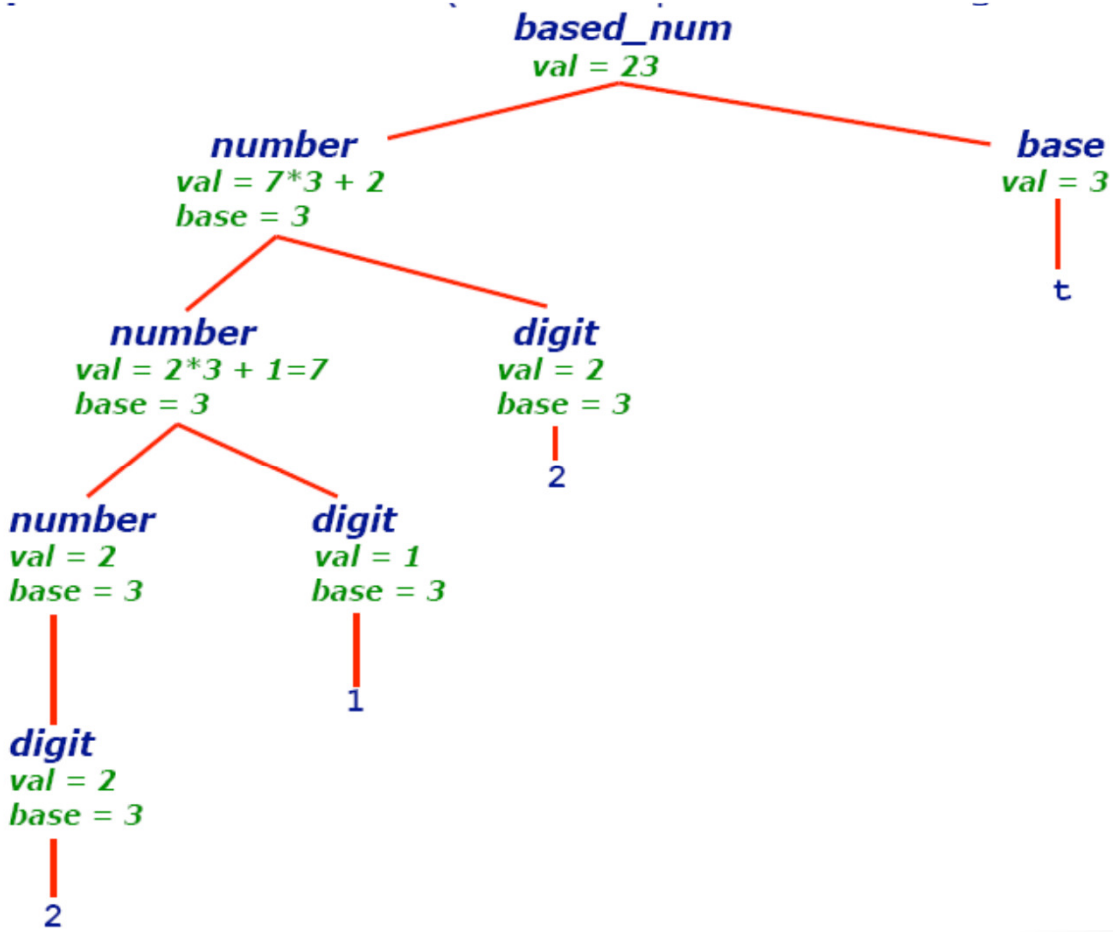
regola grammaticale	regole semantiche
based_num → number base	*based_num.val = number.val* *number.base = base.val*
base → b	*base.val = 2*
base → t	*base.val = 3*
$number_1 \rightarrow number_2\ digit$	$number_2.base = number_1.base$ $digit.base = number_1.base$ $number_1.val =$ `if` (($digit.val == error$) `or` ($number_2.val == error$)) `then` *error* `else` $number_2.base * number_2.val + digit.val$
number → digit	*number.val = digit.val* *digit.base = number.base*
digit → 2	*digit.val =* `if` (*digit.base == 2*) `then` error `else` *digit.val = 2*
digit → 1	*digit.val = 1*
digit → 0	*digit.val = 0*

Osservazioni

- Questa grammatica può generare stringhe corrette sintatticamente ma non semanticamente: ***21b*** (l'analizzatore semantico dovrebbe rilevare questo errore).
- Nelle regole semantiche è stato utilizzato il costrutto "if-then-else".
- Ciò è perfettamente lecito perché le regole semantiche saranno scritte in un linguaggio di programmazione (sarà quindi possibile invocare anche funzioni).

Albero sintattico decorato

- Albero sintattico decorato per la stringa 212t



Calcolo degli attributi

- Negli primi due esempi il calcolo degli attributo è avvenuto con una semplice visita dell'albero di parsing.
- In alcuni casi invece può servire completare l'analisi sintattica e la costruzione dell'albero di parsing per poi procedere con l'analisi semantica. Ciò implica che il compilatore deve effettuare più di una passata.

Problemi

- Come definire gli attributi?
- Come definire le regole semantiche?
- Come specificare l'ordine in cui calcolarli?
- Quali algoritmi utilizzare per calcolarli?

Definire e calcolare gli attributi

- La definizione degli attributi è relativamente semplice: basta inserire le opportune dichiarazioni nella parte iniziale di qualunque analizzatore sintattico.
- le parti destre delle regole devono essere espressioni effettivamente calcolabili al momento della derivazione della produzione. Questo vuol dire che gli attributi coinvolti devono essere già disponibili.

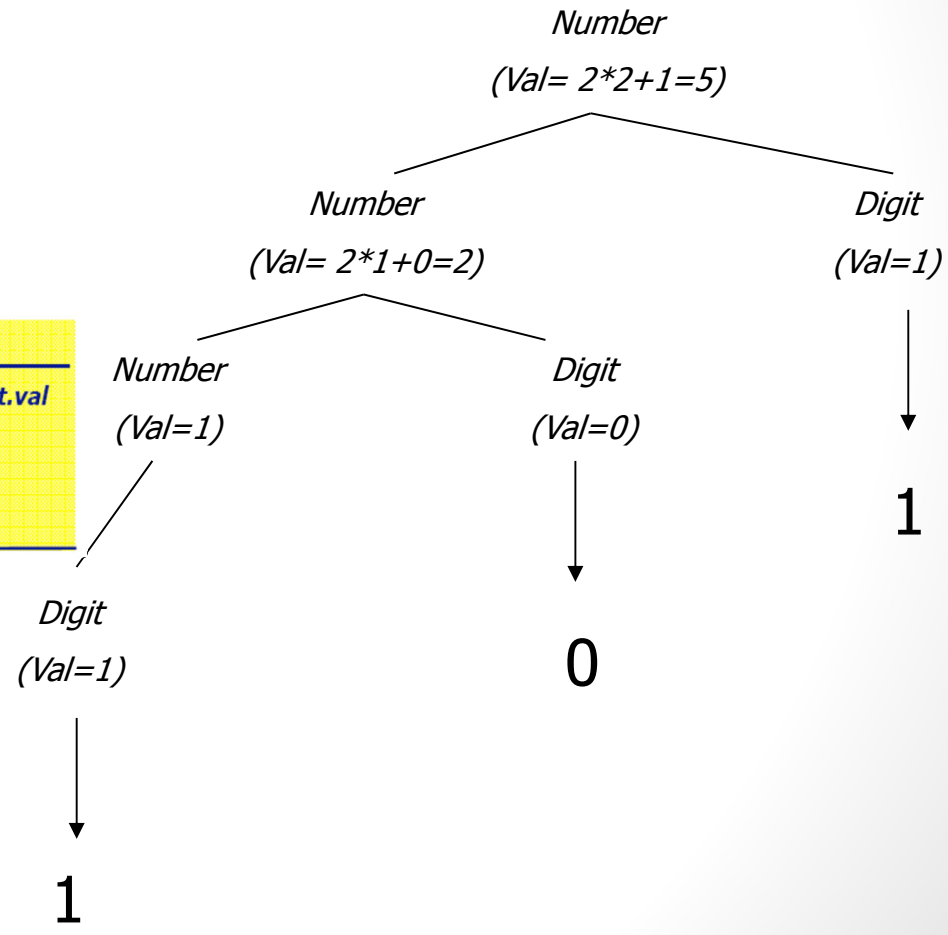
Calcolo degli attributi: in alcuni casi è semplice

- Dipendono dal tipo di visita dell'albero sintattico decorato.
- Esempio:

number → number digit | digit

digit → 0 | 1

regola grammaticale	regole semantiche
$number_1 \rightarrow number_2\ digit$	$number_1.val = 2*number_2.val + digit.val$
number → digit	*number.val = digit.val*
digit → 1	*digit.val = 1*
digit → 0	*digit.val = 0*



- In questo caso il calcolo del valore dell'attributo val si ottiene con una visita in ordine posticipato dell'albero decorato.

Dipendenze funzionali degli attributi

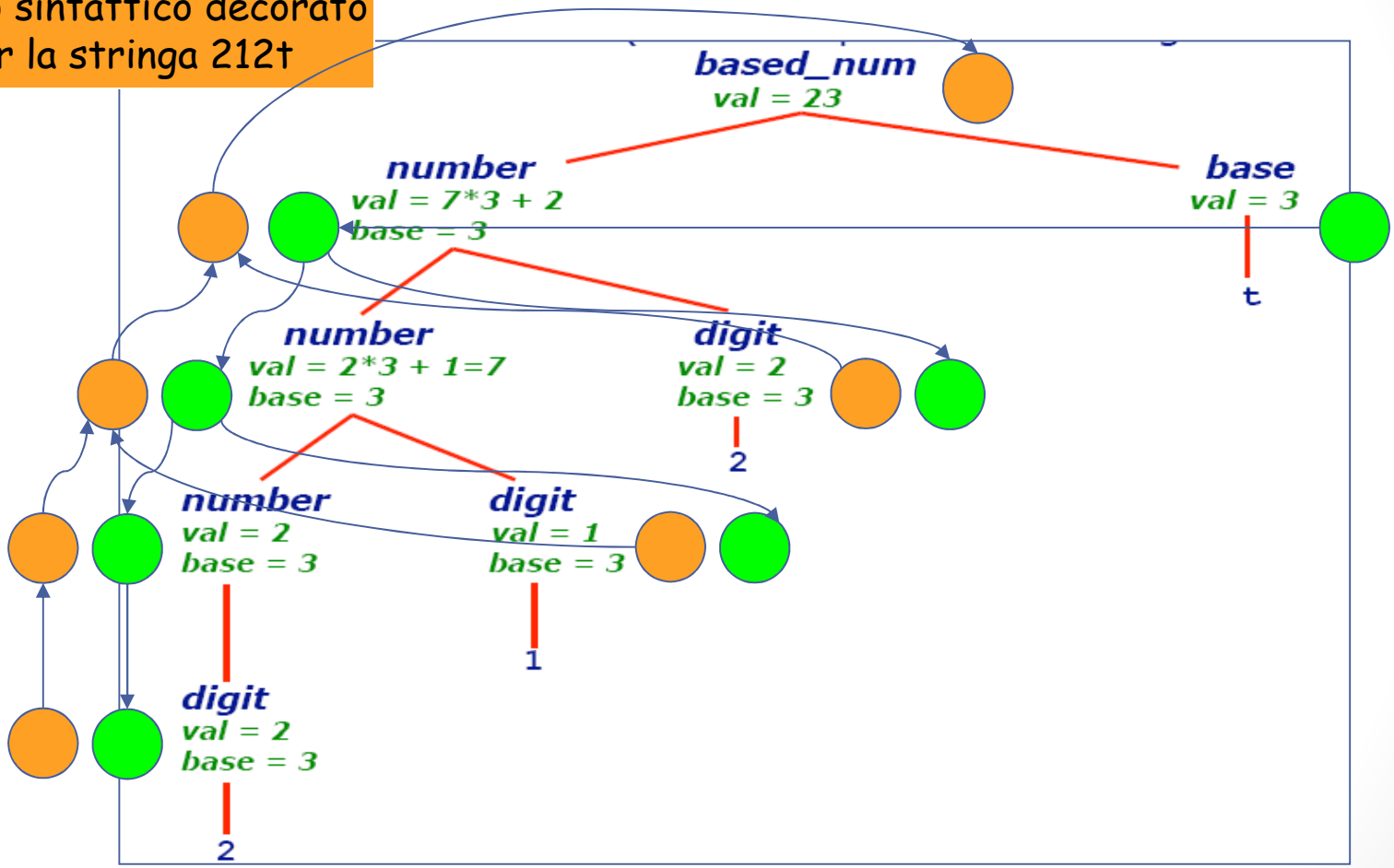
- Per capire come il diverso modo di visitare l'albero sintattico decorato può influire nella corretta valutazione degli attributi occorre introdurre il concetto di dipendenze funzionali degli attributi.
- Una regola semantica assegna un valore ad un attributo di un nodo dell'albero sintattico, in funzione dei valori di altri attributi del nodo stesso e dei nodi vicini (padre, fratelli e figli).
- L'attributo dipende dunque funzionalmente da altri attributi, i cui valori devono essere noti per consentire il calcolo.

Grafo delle dipendenze

- Data una grammatica con attributi, ad ogni regola è associato un grafo delle dipendenze.
- Per ogni nodo dell'albero di parsing etichettato con il simbolo X, il grafo delle dipendenze avrà un nodo per ogni attributo di X.
- Se una regola definisce il valore dell'attributo A.b in funzione di X.c, allora esisterà un arco da X.c a A.b.

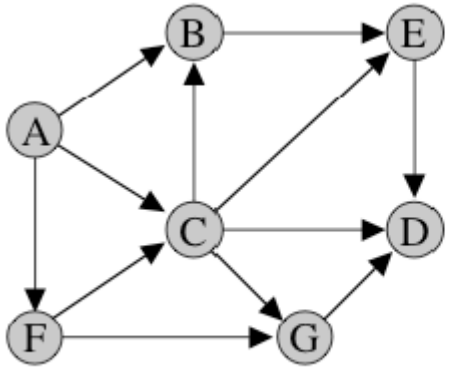
Per esempio…

Albero sintattico decorato per la stringa 212t

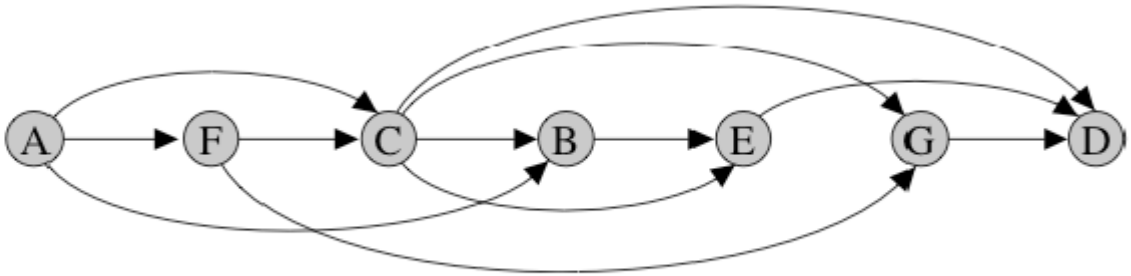


Algoritmo generale per il calcolo degli attributi

- L'algoritmo deve calcolare l'attributo di un nodo prima del calcolo dell'attributo del nodo successore; bisogna cioè trovare un ordinamento topologico del grafo.
- Il grafo deve essere aciclico.
- Il tempo di calcolo può diventare eccessivo.
- Cos'è: sequenza dei vertici in modo tale che se esiste un arco da u a v, allora u precede v nell'ordinamento.



(a)



(b)

(a) Esempio di grafo aciclico; (b) Ordinamento topologico del grafo (a)

Ordinamento topologico

Topological_Sort (G, L) **//L ritorna la sequenza dei nodi nell'ordinamento** **//topologico**

INIZIALIZZA (G) //colora di bianco tutti i nodi

for ogni u ∈ V **do**

if color [u] = white **then DFS-topologica** (G, u, L)

DFS-topologica (G, u, Pila)

color [u] ← gray

for ogni v ∈ ADJ [u] **do**

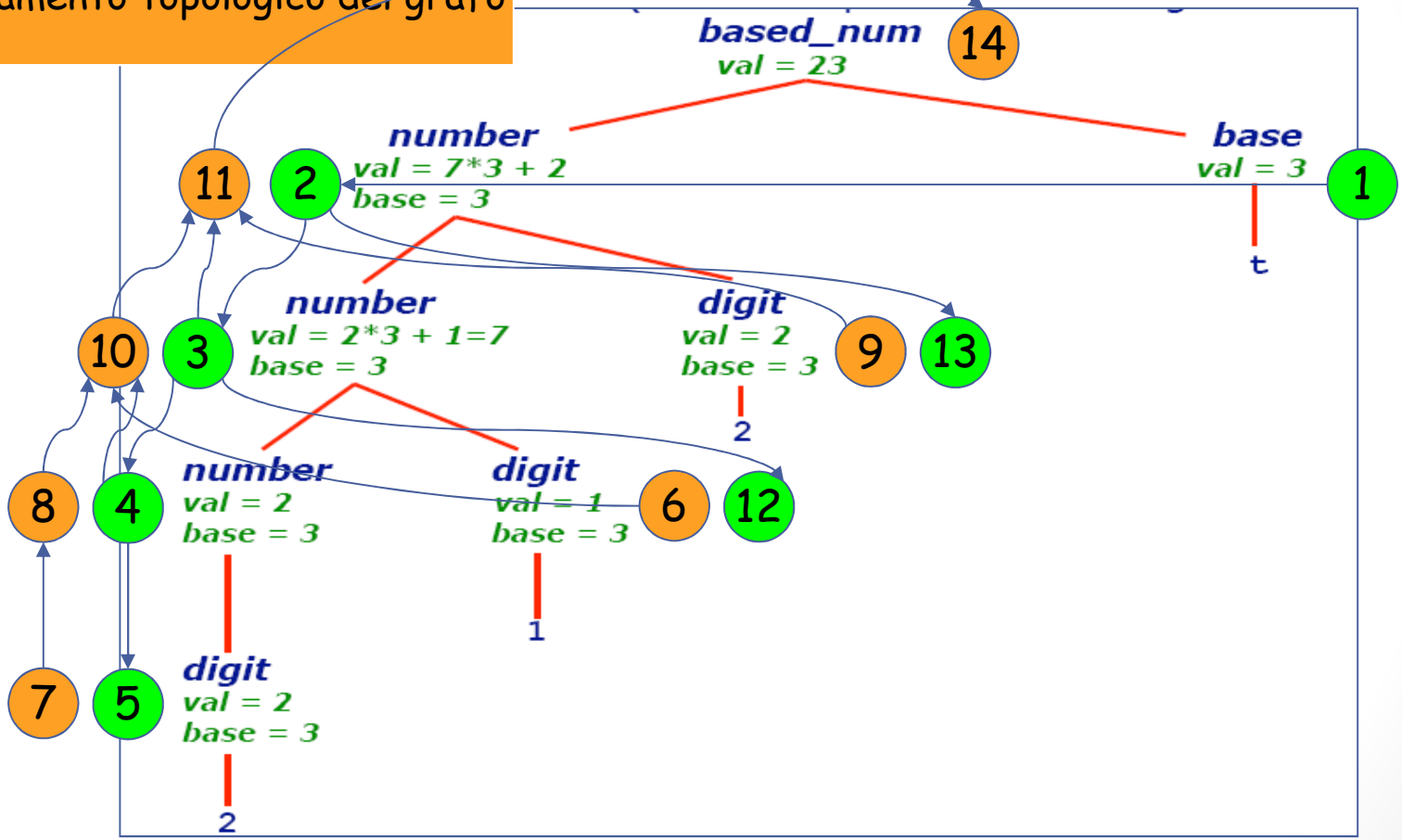
if color [v] = white **then DFS-topologica** (G, v, Pila)

color [u] ← black

InserimentoInTesta (u, **Pila**)

Per esempio…

Ordinamento topologico del grafo



Si restringe la classe delle grammatiche con attributi

- Esistono classi di grammatiche con attributi non circolari:
  - Attributi sintetizzati (synthesized attributes)
  - Attributi ereditati (inherited attributes)

Attributi sintetizzati

- Un attributo associato ad un nodo dell'albero sintattico si dice ***sintetizzato*** se il suo valore dipende solo dai valori degli attributi dei nodi figli.

formalmente: un attributo *a* è ***sintetizzato***, se data una regola $A \rightarrow X_1 \ldots X_n$, l'unica equazione con *a* nella parte sinistra ha la forma: $A.a = f(X_1.a_1, \ldots, X_1.a_k, \ldots, X_n.a_1, X_n.a_k)$

- Una grammatica che ha tutti attributi sintetizzati è detta grammatica ***puramente sintetizzata*** o grammatica con ***S-attributi***.

Attributi ereditati

- Un attributo associato ad un nodo dell'albero sintattico si dice ***ereditato*** se il suo valore dipende dai valori degli attributi del nodo padre e/o dei nodi fratelli.

- Gli attributi ereditati sono utili per esprimere le dipendenze di un costrutto di un linguaggio rispetto al suo contesto.

Algoritmi per il calcolo degli attributi

- Nel caso di grammatiche con ***S-attributi*** il calcolo del valore degli attributi si ottiene con una sola visita in ordine posticipato dell'albero sintattico decorato.

- Schematicamente ciò può essere rappresentato dal seguente pseudocodice:

```
procedure PostEval ( T: treenode );
begin
   for each child C of T do
      PostEval ( C );
   compute all synthesized attributes of T;
end;
```

Algoritmi per il calcolo degli attributi

- Nel caso di grammatiche con ***attributi ereditati*** non è chiaro l'algoritmo di visita dell'albero: infatti in questo caso bisogna ritardare l'applicazione delle regole semantiche fino al momento in cui le informazioni contestuali e il valore degli altri attributi lo consentono.

Grammatiche con L-attributi

- Si tratta di grammatiche in cui gli attributi di un nodo sono:
  - ***sintetizzati***
    oppure
  - ***ereditati*** che dipendono:
    - dagli attributi ereditati del nodo padre
      oppure
    - dagli attributi ereditati o sintetizzati dei nodi fratelli che lo precedono.
- In quest'ultimo caso il calcolo del valore degli attributi si ottiene con una sola visita anticipata sinistra dell'albero sintattico decorato.

Passate per il calcolo degli attributi

- Le strategie di calcolo degli attributi in una passata (discendente o ascendente) sono applicabili quindi quando le dipendenze fra gli attributi soddisfano le condizioni piuttosto restrittive viste in precedenza.
- In certi casi non è possibile ricondursi a tali condizioni per cui è necessario ricorrere a strategie più potenti come quelle a più passate.
- Ogni passata visita l'albero parzialmente decorato con i valori degli attributi calcolati dalle passate precedenti, e calcola quel (sotto)insieme degli attributi per cui gli insiemi delle dipendenze sono disponibili nell'albero.
- A seconda delle modalità di visita (ascendenti, discendenti, da sx a dx o da dx a sx) si possono trattare le diverse classi di grammatiche ad attributi.

Analisi sintattico-semantica

- Fino a questo momento abbiamo supposto che l'albero sintattico sia già costruito al momento dell'analisi semantica.
- Abbiamo separato di fatto l'analisi lessicale da quella semantica.
- In alcuni casi abbiamo visto che è possibile incorporare l'analisi semantica entro la procedura di analisi sintattica:
  - nelle grammatiche con S-attributi il calcolo degli attributi rispetta l'ordine di costruzione dell'albero da parte di un analizzatore sintattico ascendente
  - nelle grammatiche con L-attributi il calcolo degli attributi rispetta l'ordine di costruzione dell'albero da parte di un analizzatore sintattico discendente.