Corso di Laurea Magistrale in Ingegneria Informatica
A.A. 2013-2014

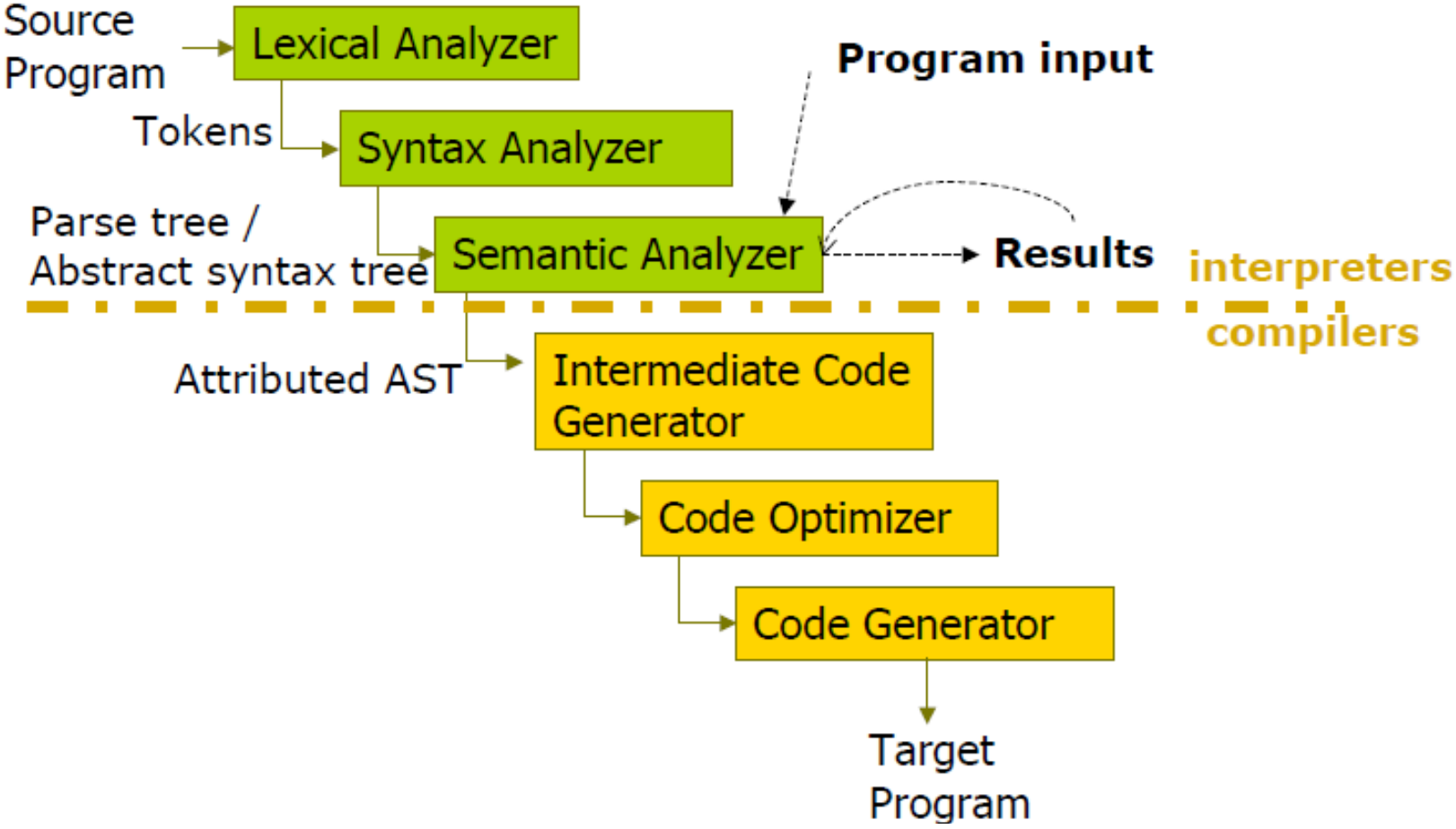
Linguaggi Formali e Compilatori

Semantica e traduzione guidata dalla sintassi (cenni)

Giacomo PISCITELLI

Compile-time semantic evaluation

Finora si è visto come il parser "traccia" le derivazioni di una sequenza di token, producendo via via il **parse tree** delle varie istruzioni del programma.



Abstract Syntax Trees

Il resto del compilatore, però, ha bisogno di conoscere di più dei costrutti del programma: in particolare le proprietà non esprimibili nel modello non contestuale. Di qui la necessità di costruire l'Abstract Syntax Tree (**AST**), cioè di:

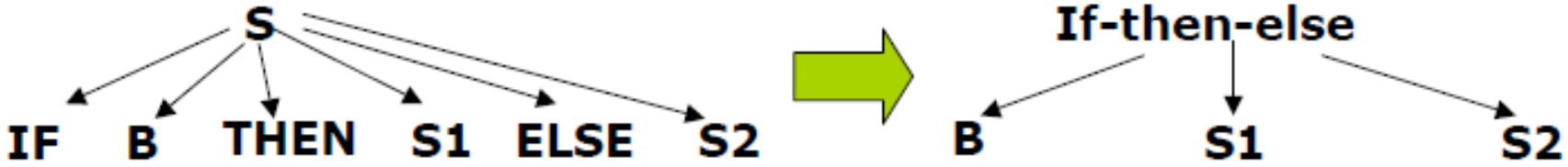
- ✓ associare, ai terminali e non terminali del parse tree, **attributi** (es. valore e tipo per le variabili e le espressioni) che consentano l'analisi e verifica di condizioni di natura "semantica";
- ✓ specificare le **regole** (o azioni) **semantiche** di verifica delle anzidette condizioni per ciascuna produzione della sintassi: le azioni indicano, là dove si usa una produzione, come valutare gli attributi del nodo non terminale in relazione a quelli dei nodi "figli".

Una **Syntax Directed Definition (SDD)** consiste nell'associare un set di attributi a ciascun simbolo della grammatica e un set di azioni semantiche a ciascuna produzione.

Abstract Syntax Trees

Gli Abstract Syntax Tree sono forme sintetiche di parse tree che consentono di rappresentare in maniera adeguata i costrutti linguistici di una grammatica

- ✓ Operatori e parole chiave non appaiono come simboli terminali
- ✓ Operatori e parole chiave, invece, definiscono il significato dei nodi "figli"



- ✓ Catene di singole produzioni possono essere collassate

Constructing an Abstract Syntax Trees

- Considerando la grammatica

  $E \rightarrow int \mid (E) \mid E + E$

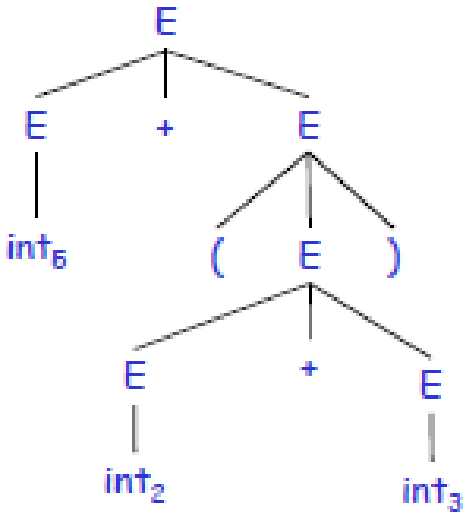
- e la stringa

  5 + (2 + 3)

- dopo l'analisi lessicale sarà stata ricavata una lista di token

  int5 '+' '(' int2 '+' int3 ')'

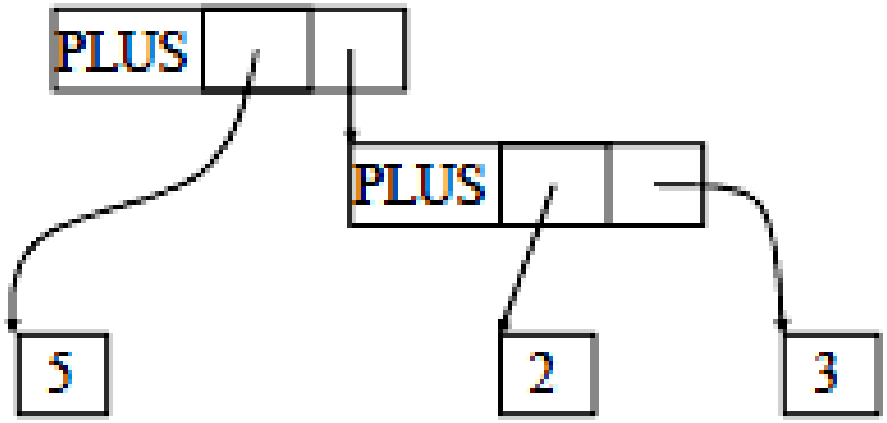
- e avremo costruito un parse tree basato su troppi simboli terminali . . . . .



- Traces the operation of the parser
- Does capture the nesting structure
- But too much info
  - Parentheses
  - Single-successor nodes

Constructing an Abstract Syntax Trees

- . . . . che può essere sostituito dal seguente AST, più compatto e facile da trattare . . . .



- che conserva la struttura di nesting del parse tree
- ma astrae dalla pura sintassi e consente di associare ad un nodo terminale anche altri attributi (oltre al semplice valore) mediante un record da inserire nella tavola dei simboli.

Defining a SDD from AST

Grammatica:

E ::= E + T | E – T | T
T ::= (E) | id | num

Uso di una SDD

Problema: costruire un AST per ogni espressione

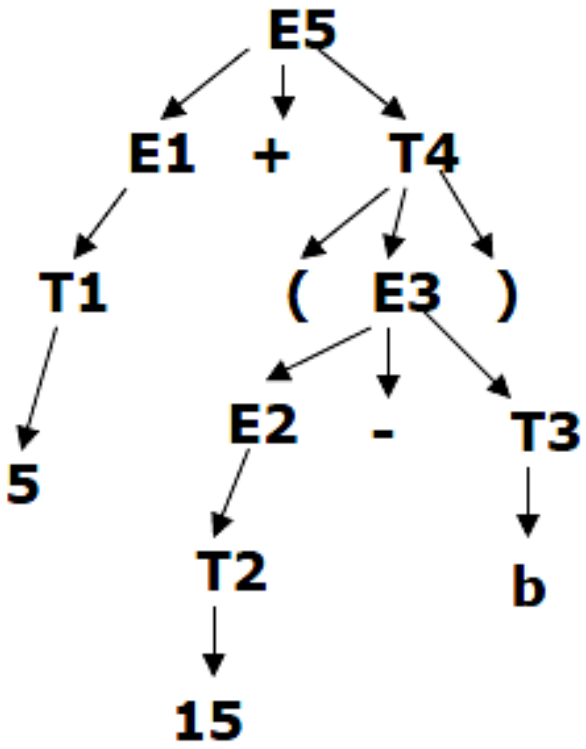
- Approccio grammatica ad attributi
  - ✓ Associare ogni non-terminale con un AST
  - ✓ Per ogni AST: un pointer ad ogni nodo nell'AST

    `E.nptr T.nptr`
- Definizioni: come determinare gli attributi?
  - ✓ Nel bottom-up: si fa uso di synthesized attribute
  - ✓ Se si conosce l'AST di ogni figlio, come determinare l'AST del genitore?

Example: constructing AST

Bottom-up parsing: si valuta un attributo ad ogni riduzione

Parse tree for 5+(15-b)



1. reduce 5 to T1 using T::=num:
   T1.nptr = leaf(5)
2. reduce T1 to E1 using E::=T:
   E1.nptr = T1.nptr = leaf(5)
3. reduce 15 to T2 using T::=num:
   T2.nptr=leaf(15)
4. reduce T2 to E2 using E::=T:
   E2.nptr=T2.nptr = leaf(15)
5. reduce b to T3 using T::=num:
   T3.nptr=leaf(b)
6. reduce E2-T3 to E3 using E::=E-T:
   E3.nptr=node('-',leaf(15),leaf(b))
7. reduce (E3) to T4 using T::=(E):
   T4.nptr=node('-',leaf(15),leaf(b))
8. reduce E1+T4 to E5 using E::=E+T:
   E5.nptr=node('+',leaf(5),
   node('-',leaf(15),leaf(b)))

Syntax Directed Translation scheme (SDT)

Specificata una SDD, è poi possible definire uno schema di traduzione guidato dalla sintassi (**SDT**), una notazione, cioè, per attaccare frammenti di programma (semantic actions) alle produzioni di una grammatica.

I frammenti di programma sono eseguiti quando, durante l'analisi, si fa uso della produzione. L'esecuzione di tali frammenti, nell'ordine indotto dall'analisi sintattica, produce come risultato la traduzione del programma.

La posizione in cui eseguire un'azione semantica è indicata racchiudendo il programma tra parentesi graffe e scrivendo il tutto nel corpo della produzione, come mostrato nel seguente esempio:

$$rest \rightarrow + \; term \; \{\text{print ('+')}\} \; rest1$$

Ogni SDT può essere realizzato prima costruendo un parse tree e poi eseguendo le azioni con attraversamento in pre-ordine sinistro.

Semantic Actions

- Poichè i simboli (non-terminali e terminali) della grammatica rappresentano costrutti di programmazione, gli attributi sono associati ai simboli. In ciò consiste quello che si chiama costruzione degli AST.
- Ogni simbolo della grammatica può avere attributi
  - Gli attributi dei simboli terminali (detti anche token lessicali) possono essere determinati dallo scanner.
- Ogni produzione può richiedere una "azione"
  - indicata nella seguente modalità: $X \rightarrow Y1 \ldots Yn \{ action \}$
  - che può far riferimento alla determinazione degli attributi dei simboli.

Analisi semantica e sintesi dell'output

La fase di analisi semantica dell'input dipende dalla semantica del linguaggio sorgente.

La fase di sintesi dell'output dipende sia dalla semantica del linguaggio sorgente che dalla semantica del linguaggio destinazione.

Data la complessità della definizione della semantica di un linguaggio, le fasi di analisi semantica dell'input e di sintesi dell'output **non** possono essere automatizzate, come invece avviene per le fasi di analisi lessicale e analisi sintattica.

Pertanto, i moduli che realizzano le fasi di analisi semantica e sintesi dell'output devono essere realizzati manualmente (non esistono strumenti di generazione automatica).

Sintesi dell'output

Approccio classico (due passate):

1. si sintetizza l'albero sintattico (struttura dati) nella fase di analisi sintattica
2. si utilizza l'albero sintattico per la sintesi dell'output

Approccio semplificato (una passata):

1. si sintetizza l'output direttamente durante l'analisi sintattica, senza costruire l'albero sintattico

L'approccio semplificato ad una passata funziona solo nei casi "semplici" di traduzione e, più in generale, di elaborazione dell'input. Invece, costruendo l'albero sintattico e visitandolo successivamente si può implementare un insieme molto più grande di funzioni di traduzione e di elaborazione

Semantica e tipi di regole

Se si considera un linguaggio assolutamente generale (e non soltanto un linguaggio di programmazione) le regole semantiche possono essere espresse:

- ✓ *a parole* → (poco precisa e ambigua)
- ✓ *mediante azioni* → **Semantica operazionale**
- ✓ *mediante funzioni matematiche* → **Semantica denotazionale**
- ✓ *mediante formule logiche* → **Semantica assiomatica**

Il particolare tipo di regole semantiche utilizzate nell'ambito dei compilatori è di tipo operazionale ed è noto con il nome di *grammatica ad attributi*, senza dubbio il più diffuso nel campo dei linguaggi di programmazione.

Traduzione guidata dalla sintassi

Sarebbe difficile progettare in modo globale un algoritmo di traduzione di un linguaggio complesso senza prima analizzare la frase sorgente nei suoi costituenti definiti dalla sintassi.

L'analisi sintattica riduce la complessità del problema del calcolo della traduzione, decomponendolo in sottoproblemi più semplici.

Per ogni costituente della frase sorgente il traduttore esegue le azioni appropriate per preparare la traduzione, azioni che consistono nel raccogliere certe informazioni (ad es. le tabelle dei simboli), nel fare dei controlli semantici, o nell'emettere certe parti della frase pozzo.

In altre parole il lavoro del traduttore è modularizzato secondo la struttura descritta dall'albero sintattico della frase. Perciò questo tipo di traduttori è detto ***guidato dalla sintassi***.

Grammatiche ad attributi

Si parla di **traduzione guidata dalla sintassi** nei casi in cui è possibile definire il processo di sintesi dell'output (traduzione) sulla base della struttura sintattica dell'input.

In altre parole, in questi casi l'operazione di traduzione è fortemente legata alla sintassi dell'input; pertanto la traduzione può essere definita in modo "parallelo" alla definizione della sintassi dell'input. In moltissime applicazioni informatiche la funzione di traduzione (o di elaborazione) è guidata dalla sintassi; a tal fine, è possibile estendere i formalismi di specifica della sintassi al fine di catturare alcuni aspetti "semantici".

In particolare, vengono definite e utilizzate estensioni del formalismo delle grammatiche non contestuali: le **grammatiche ad attributi**, che estendono le grammatiche non contestuali introducendo le nozioni di:

1. **attributi** associati ai simboli (terminali e non terminali) della grammatica
2. **azioni semantiche** e **regole semantiche**, che affiancano le regole sintattiche della grammatica

Tramite le azioni semantiche, è possibile specificare la fase di traduzione in parallelo alla specifica della sintassi del linguaggio sorgente.

Anche per traduzioni guidate dalla sintassi, si può avere la necessità di costruire l'albero sintattico (due passate distinte per analisi sintattica e sintesi dell'output) oppure no (una sola passata per analisi sintattica e sintesi dell'output)

Attributi, regole semantiche, azioni semantiche

Nelle grammatiche ad attributi:

- un **attributo** è una proprietà associata al simbolo, che può essere letta o assegnata dalle azioni semantiche (proprio come una variabile di un linguaggio di programmazione)
- ogni simbolo terminale o non terminale può avere uno o più attributi
- gli attributi sono usati per rappresentare un "significato" (tramite un valore numerico o una stringa o una struttura di dati, …) che viene associato ai simboli sintattici
- ad ogni regola di produzione (regola sintattica) può essere associata una regola semantica
- una **regola semantica** è una sequenza di azioni semantiche
- una **azione semantica** è un'azione in grado di assegnare valori agli attributi (e che quindi calcola tale "significato") e di avere altri effetti (proprio come una o più istruzioni di un linguaggio di programmazione)

Attributi e azioni semantiche

esempio:

- in una grammatica per espressioni aritmetiche sui numeri interi, possiamo associare un attributo di tipo intero ad ogni simbolo non terminale della grammatica
- possiamo poi definire delle azioni semantiche in modo tale che il valore di ogni attributo corrisponda al valore della sottoespressione associata al corrispondente simbolo non terminale
- pertanto il valore dell'attributo associato alla radice dell'albero sintattico è uguale al valore dell'intera espressione aritmetica

Attributi sintetizzati e attributi ereditati

Gli attributi si distinguono in:

- **attributi sintetizzati**: sono gli attributi il cui valore dipende solo dai valori degli attributi presenti nel sottoalbero del nodo dell'albero sintattico a cui sono associati
- **attributi ereditati**: sono gli attributi il cui valore dipende solo dai valori degli attributi presenti nei nodi predecessori e nei nodi fratelli del nodo dell'albero sintattico a cui sono associati

Gli attributi sintetizzati realizzano un flusso informativo **ascendente** nell'albero sintattico (dalle foglie verso la radice).

Gli attributi ereditati realizzano un flusso informativo **discendente** (dalla radice verso le foglie) e **laterale** (da sinistra verso destra e viceversa) nell'albero sintattico.

Valutazione degli attributi

L'esecuzione di una regola semantica avviene in parallelo all'applicazione della corrispondente regola sintattica nell'analisi sintattica.

Pertanto, l'ordine di applicazione delle regole semantiche è conseguenza dell'ordine di applicazione delle regole di produzione nell'analisi sintattica.

In particolare, dato un certo ordine di applicazione, le istruzioni che nelle regole semantiche effettuano la valutazione degli attributi potrebbero non essere eseguibili in modo corretto: ad esempio, può accadere che in una azione semantica si tenti di leggere il valore di un attributo non ancora assegnato. La valutazione può avere anche *side effects* (effetti collaterali) come la stampa di valori o l'aggiornamento di una variabile globale.

Questi problemi, noti come problema della valutazione degli attributi, sono fondamentali per il trattamento automatico delle grammatiche ad attributi.

In una definizione guidata dalla sintassi si assume che i simboli terminali abbiano solo attributi sintetizzati: i valori per questi attributi sono in genere forniti dall'analizzatore lessicale.

Il simbolo iniziale, se non diversamente specificato, non ha attributi ereditati.

S-attributi e L-attributi

Per risolvere il problema della valutazione degli attributi, occorre restringere i tipi di attributi che possono essere definiti.

In particolare, si definiscono due classi di attributi che garantiscono la corretta valutazione degli attributi:

1. **S-attributi (S-attributed)**: che coincide con la classe degli attributi sintetizzati. Quando una SDD è costruita con S-attributi, si possono valutare i suoi attributi in qualunque ordine bottom-up dei nodi del parse tree; è spesso particolarmente semplice valutare gli attributi effettuando un attraversamento in post ordine sinistro del parse tree, valutando gli attributi al nodo N quando l'attraversamento lascia N per l'ultima volta, cioè viene applicata la seguente funzione postorder alla radice del parse tree.

```
postorder (N)   { for (ciascun figlio C di N, da sinistra) postorder (C);
                  valuta gli attributi associati al nodo N }
```

2. **L-attributi (L-attributed)**: è una sottoclasse degli attributi ereditati, in cui si permette ad ogni attributo A di dipendere solo dal nodo padre o dai nodi fratelli a sinistra del nodo dell'albero sintattico a cui l'attributo A è associato.

Syntax Directed Translation Schemes (SDT)

Sia che la SDD abbia una grammatica sottostante LR-parsable e attributi sintetizzati, sia che la SDD abbia una grammatica sottostante LL-parsable e attribute ereditati, in entrambi i casi, le regole semantiche in una SDD possono essere convertite in una SDT con azioni che vengono eseguite al tempo giusto: durante il parsing, un'azione nel corpo di una produzione viene eseguita appena tutti i simboli della grammatica alla sinistra dell'azione sono stati incontrati (cfr. "*marker nonterminals*" in sect. 5.4 of the text).

La più semplice realizzazione di una SDD si ottiene quando si può condurre un'analisi bottom-up e la SDD ha S-attributi. In tal caso si può costruire una SDT in cui ciascuna azione abbia luogo al termine della produzione e venga eseguita contestualmente alla riduzione del corpo della parte iniziale della produzione. Le SDT in cui le azioni appaiono al termine destro del corpo delle produzioni vengono dette **SDT post-fisse**.

Making bottom-up every translation

Ogni traduzione realizzabile con un approccio top-down, può anche essere realizzata in modo bottom-up. Infatti, data una SDD con L-attributi e basata su una grammatica LL, si può adattare la grammatica in modo da poter ottenere la stessa SDD sulla grammatica modificata con un parsing LR (cfr. procedura all'inizio del par. 5.5.4 del testo).

Grammatiche ad attributi

Esempio: Il calcolo del valore decimale di un numero frazionario in base due (D. Knuth 1968)

L'esempio mostra il ruolo degli attributi ereditati, e la loro sostituibilità con gli attributi sintetizzati.
Il linguaggio sintatticamente è definito dall'espressione regolare

$$L = \{ 0,1\}^*. \{0, 1\}^*$$

da interpretare come un numero binario con il punto che separa la parte intera da quella frazionaria. Ad es. il significato di 1101.01 è il numero 13,25

Sintassi

1. num → str . str
2. str → str bit
3. str → bit
4. bit → 0
5. bit → 1

num è l'assioma, *str* definisce la stringa binaria della parte intera e frazionaria, che è composta di *bit*.

Grammatiche ad attributi

Attributo	Orientamento	Dominio	Classi sintattiche
f fattore di scala che dà il peso ad un bit	ereditato	intero	str, bit
l lunghezza della stringa	sintetizzato	intero	str
v valore del numero	sintetizzato	razionale	num

Grammatiche ad attributi (1 di 2)

1. num → str1 . str2	f of str1 := 0
	f of str2 := - l of str2
	v of num= v of str1 +v of str2
2. str0 → str1 bit	f of str1 := f of str0 +1
	f of bit := f of str
	l of str0 := l of str1 + 1
	v of str0 := v of str1 + v of bit
3. str → bit	f of bit:= f of str
	l of str := 1
	v of str := v of bit
4. bit → 0	v of bit := 0
5. bit →1	v of bit := 2 f of bit

Grammatiche ad attributi (2 di 2)

1. num → str1 . str2
   v of num= v of str1 +v of str2 * 2 l of str2
2. str0 → str1 bit
   v of str0 := 2* v of str1 +v of bit
   l of str0 := l of str1 + 1
3. str → bit
   v of str := v of bit
   l of str := 1
4. bit → 0
   v of bit := 0
5. bit → 1
   v of bit := 1