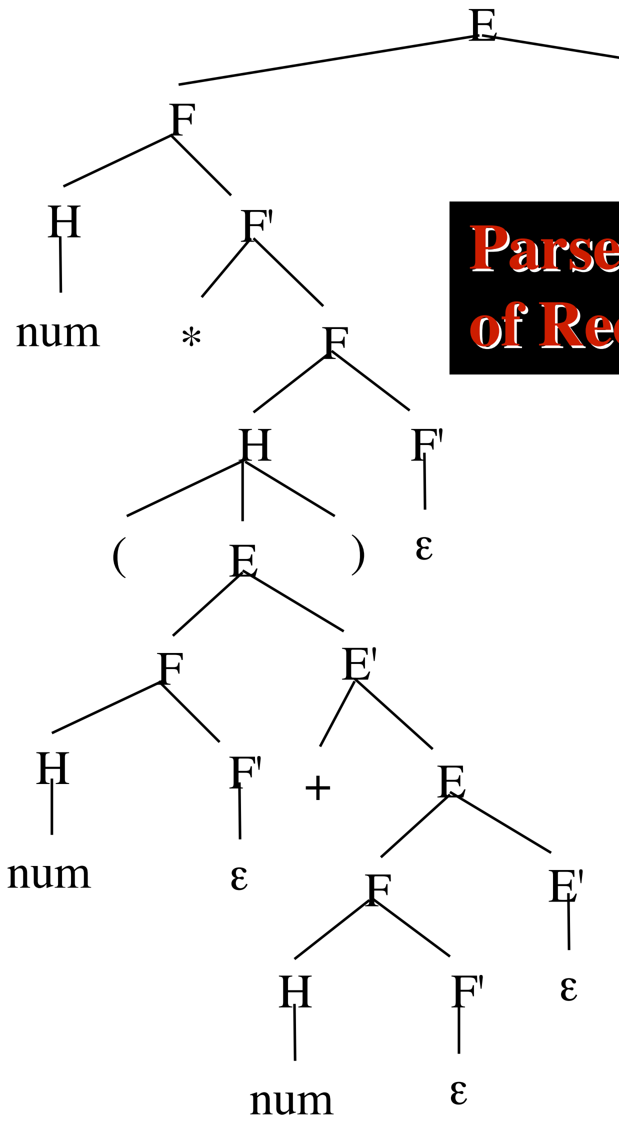
From Syntax Analyzers to Tree/Code Generators

- **Parse Trees**
- **Syntax Trees and Abstract trees**
- **How to extend Analyzers: Attribute Grammars**
- **Trees and Code as Grammar Attributes**

Parse Tree



Parse Tree relates the **Language Phrases** with **a way of Recognizing** them using a **Specific Grammar**

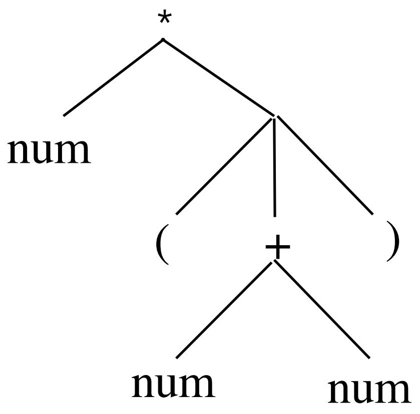
E::= E (+|*) E | num | (E)

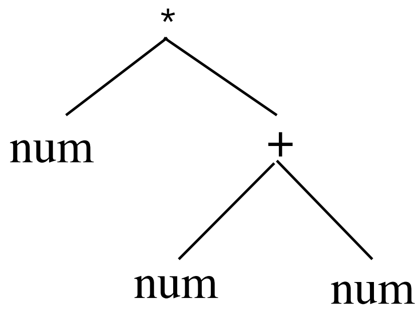
E::= F E'[0]
E'::= + E[1] ε [2]
F::= H F'[3]
F'::= * F [4] ε [5]
H::= num[6] (E) [7]

E::= F + E | F
F::= H * F | H
H::= num | (E)

Syntax and Abstract Trees

Sometimes, Parse Tree may be to **Useless** and **Expensive**





It relates **Terms** with the **Subterms in their concrete syntax** form

It relates **Terms** with the **Subterms ignoring any concrete aspect**

We need a general framework for Recognizing Phrases and Producing different kinds of information about them

Attribute Grammars

One **Methodology**,

One **Technique**,

One **Tool** for

Property Analyses (possibly Contextual)

and

Code Generation

Attribute Grammars Syntax - 1

Let $G = \{\Sigma, V, s, P\}$ be a Grammar

- $P \subseteq V \times E_{\Sigma+V}$
- $L(G) = I(G)$ -- see later on

Let $B \in \Sigma+V$ be a (grammatical symbol of G).

Let $\{\mathbf{a_i}\}$ be a family of symbols, called attribute symbols.

Attribute a_i of B be

- a syntactic entity denoted by $\mathbf{B.a_i}$
- with, as associated meaning, a value that depends from the Parse Trees in which the attribute is considered
- the value type, if any, depends from the Attribute Grammar, extending G

$\mathbf{Meta_{\{ai\}}}$ be a Programming Language extended with attributes, viewed as constant (final variables)

Actions $\{\alpha_i\} \subseteq \{a_i\} \times Meta_{\{ai\}}$ be assignments of the form $\mathbf{B.a_i} = e_\mathbf{i}$ where $e_\mathbf{i} \in \mathbf{Meta_{\{ai\}}}$

Let $G^A \equiv \{\Sigma, V, s, P^A, \{a_i\}\}$ be an atribute grammar extending G. Then:

- $G^A\!\downarrow \equiv \{\Sigma, V, s, P\} = G$, *con* $P = P^A\!\downarrow$
- $P^A \equiv P \times \{\alpha_\mathbf{i}\}$
- …

Attribute Grammars Syntax - 2

Let $G = \{\Sigma, V, s, P\}$ be a Grammar ….

…
Actions $\{\alpha\}$ be assignments of the form $\mathbf{B.a_i} = e_i$ where $e_i \in \mathbf{Meta}_{\{ai\}}$

Let $G^A \equiv \{\Sigma, V, s, P^A, \{a_i\}\}$ be an attribute grammar extending G. Then:

- $G^A\downarrow \equiv \{\Sigma, V, s, P\} = G$, *con* $P = P^A\downarrow$
- $P^A \equiv P \times \{\alpha\}$
- $\forall p \in P^A$, let $p \equiv B := \beta\ \{X1.a1 = e1, \ldots, Xk.ak = ek\}$
  - $\{X1, \ldots, Xn\} \subseteq Sym(\beta) + \{B\}$
  - $Sym(e1) + \ldots + Sym(ek) \subseteq Sym(\beta) + \{B\}$
  - no side effects (otherwise are called, SDD for Syntax Directed Definitio

where: $Sym(U) \subseteq \Sigma + V$ is the Set of grammatical symbols in U

Attribute Grammars
Syntax - Example

Let $G^A \equiv \{\Sigma, V, s, P^A, \{a_i\}\}$ be an attribute grammar extending G. Then:

- $G^A\downarrow \equiv \{\Sigma, V, s, P\} = G$, *con* $P=P^A\downarrow$
- $P^A \equiv P \times \{\alpha\}$
- $\forall p \in P^A$, let $p \equiv B:=\beta\ \{X1.a1=e1, \ldots, Xk.ak=ek\}$
  - $\{X1, \ldots, Xn\} \subseteq Sym(\beta)+\{B\}$
  - $Sym(e1)+\ldots+Sym(ek) \subseteq Sym(\beta)+\{B\}$
  - no side effects (otherwise are called, SDD for Syntax Directed Definition}

Where: $Sym(U) \subseteq \Sigma+V$ is the Set of grammatical symbols in U

E::= F E’ {E’.d=E.d+1}
E'_1::= + F E'_2 {E'_2.d=E'_1.d+1}
E’::= ε {ε.d=E’.d}

What about the type of attribute of d?
What about the value of E.d?

What does this grammar defines ?

Attribute Grammars
Semantics i.e. Meaning

$E ::= F\ E'\ \{E'.d = E.d+1\}$
$E'_1 ::= +\ F\ E'_2\ \{E'_2.d = E'_1.d+1\}$
$E' ::= \varepsilon\ \{\varepsilon.d = E'.d\}$

What does this grammar defines ?

An Attribute Grammar defines a set of *Parse Trees* whose *nodes* are extended with *attribute-value pairs* according to the Grammar and the Parse Tree

We need to define structures for:

- Labeled Trees
- Labeled Trees with Attributes

before giving semantics

Binary Relation: Tree, Labeled Tree

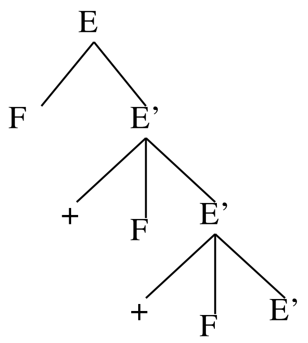
Trees (are binary relations):

- $T = <R, E \subseteq R^2>$
- $T = <R, E \subseteq R^2, L:R->U>$ -- *labeled on a set U*

where: R=*set of the nodes* E=*set of the (directed) edges* L= *Labeling Function on U*

Operators: root*: T --> R,* arity: *R --> N,* son*: T* x *N --> T*
Selectors: R(T)= R, E(T)=E, L(T)=L
Shortening: sons: $T \text{ --> } T^N$, L: $T^N \text{ --> } U^N$

sons(T)=son(T,1)…son(T,arity(root(T))) $L(T_1,\ldots,T_k)=L(root(T_1)\ldots L(root(T_k))$



T = <R(T),E(T),L(T)>, where:
R(T) = {0,1,2,3,4,5,6,7,8}
L(T) = {(0,E),(1,F),(2,E'),(3,+),(4,F),(5,E'),(6,+),(7,F),(8,E')}
E(T) = {(0,1),(0,2),(2,3),(2,4),(2,5),(5,6),(5,7),(5,8)}

Grammars and Parse Trees

- I(G) is the set of Parse Trees of G

- I(G) is obtained by =>*, using => on the tree set T(G).

Let: $G \equiv \langle \Sigma, V, s \in V, P \equiv \{x_i ::= x_{i,1} \ldots x_{i,ki} \mid i \leq |P|\} \rangle$

$T(G) \equiv \{T \equiv \langle R, E \subseteq R^2, L: R \rightarrow \Sigma \cup V \rangle \mid L(root(T)) = s$ and

$\quad$ if $(l,r) \in E$ then $(l,r) \in \{(l,r_1),\ldots,(l,r_h) \mid L(l)=x \;\&\; L(r_i)=x_i \;\&\; x::=x_1 \ldots x_h \in P\} \subseteq E\}$

$=> \equiv \{(T,U) \in T(G)^2 \mid$ either $arity(root(T))=0 \,\&\, L(root(T))=x=L(root(U)) \,\&\, L(sons(U))=x_1 \ldots x_k$

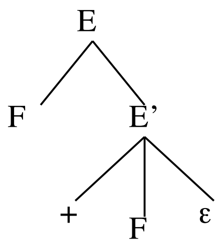
$\quad$ for some $x::=x_1 \ldots x_k \in P$, …

$\quad$ or $L(root(T))=L(root(U)) \,\&\, son(T,j)=son(U,j)$ $(\forall j \in 1..k \backslash i) \,\&$

$\quad$ $son(T,i) => son(U,i)$ for $1 \leq i \leq k \equiv arity(root(T))$ }

$I(G) \equiv \{T \mid T_s =>^* T \;\&\; arity(root(T_s))=0 \;\&\; L(root(T_s))=s \;\&\ldots$

Complete and Correct



E::= F E'
E'::= + F E'
E'::= ε

AST: Abstract Syntax Trees

Abstract Syntax Trees are the elements of:

One Binary, Anti-Symmetric, Relation ">" on Terms s.t:

$$f_k(t_1,\ldots,t_k) > t_i \qquad (\forall\, 1 \leq i \leq k \;\&\; f_k(t_1,\ldots,t_k), t_i \in Terms)$$

Equally: $> \equiv \{(f_k(t_1,\ldots,t_k),t_i) \mid 1 \leq i \leq k,\ f_k(t_1,\ldots,t_k),t_i \in T_\Sigma\}$

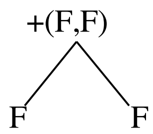
where: $T_\Sigma \equiv \{f_k(t_1,\ldots,t_k) \mid f_k \in \Sigma \;\&\; t_1,\ldots,t_k \in T_\Sigma\}$ --- *Terms on* Σ

Equally: $> \equiv \langle R,\ E \subseteq R^2, L{:}R \text{->} T_\Sigma \rangle$

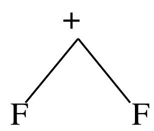
where: $E \equiv \{(l, r_1),\ldots,(l, r_k) \mid L(l) = Opt(f_k(t_1,\ldots,t_k)) \;\&\; L(r_i) = Opt(t_i)\ (\forall f_k(t_1,\ldots,t_k) \in T_\Sigma)\}$

$\Sigma = \{F_0, +_2\}$
$T_\Sigma ::= \{F\} \cup \{+(u,t) \mid u,t \in T_\Sigma\}$

$Opt: T_\Sigma \text{->} \Sigma$
$Opt(f_k(t_1,\ldots,t_k)) = f_k$



or



For a different label
-ing function L

Attribute Grammars Semantics

$T = \langle R, E \subseteq R^2, L: R \to U \rangle$ --- *U-labeled Trees*

$T^A = \langle T, L_{\{ai\}}: R(T) \to W_{\{ai\}} \rangle$ --- *U-labeled Trees attributated by values on $W_{\{ai\}}$*

Grammar: $G = \{\Sigma, V, s, P\}$

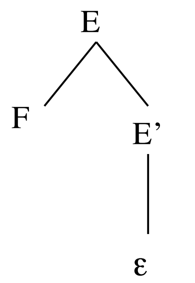
- $P \subseteq V \times E_{\Sigma+V}$
- $L(G) = I(G)$

Attribute Grammar: $G^A \equiv \{\Sigma, V, s, P^A, \{a_i\}\}$

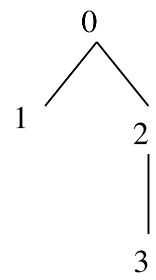
- $G^A\downarrow \equiv \{\Sigma, V, s, P\} = G$, *con $P = P^A\downarrow$*
- $P^A \equiv P \times \{\alpha\}$
- $L(G^A) \equiv \{\langle T, L_{\{ai\}} \rangle \mid T \in L(G^A\downarrow)\ \&\ \forall r_0 \in R(T):$
  - $sons(r_0) = r_1, \dots, r_n, (\forall i \in 0..n) L(r_i) = B_i,\ B_0 ::= B_1, \dots, B_n \{B_i.a_{ji} = e_{i,ji}\}^{i \in 0..n} \in P^A$
  - $L_{aj}(r_i) \equiv Sem_{Meta}(e_{i,j}\, [L_{ak}(r_h)/B_h.a_k]^{(h \neq i)\ or\ (k \neq j)})\}$
    - $E[V/x]$ means E where the value V is replacing x
    - $F^{c(i)}$ means $F[V_1/i] \dots F[V_n/i]$ for $c(i) = \{V_{1,\dots,}V_n\}$

Attribute Grammar Example

$E ::= F\ E' \ \{E'.d=E.d+1\}$
$E'_1 ::= +\ F\ E'_2 \ \{E'_2.d=E'_1.d+1\}$
$E' ::= \varepsilon \ \{\varepsilon.d=E'.d\}$



parse tree

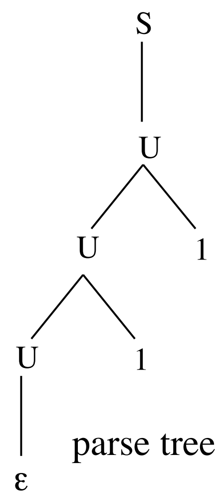


The effective T

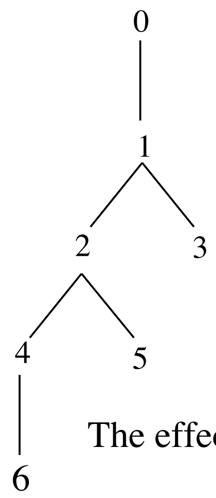
$$T^A=\{<R=\{0,1,2,3\}, E, L=\{(0,E),(1,F),(2,E'),(3,\varepsilon)\}>,$$
$$L_d=\{(0,Sem_{Meta}(\bot)), \ldots$$
$$(2,Sem_{Meta}((E.d+1)[L_d(0)/E.d])),$$
$$(3,Sem_{Meta}((E'.d)[L_d(2)/E'.d]))\}\}$$

Apply Semantics: Another Example [Aho-pag.316]

S::= U {U.count:=0}
$U ::= U_1$ 1 {U_1.count=U.count+1}
U::= ε {print(U.count}



parse tree



The effective tree

T^A={<R={0,1,2,3,4,5,6}, E, L={(0,S),(1,U),(2,U),(3,1),(4,U),(5,1),(6,ε)}>,

L_{count}={(0,Sem_{Meta}(⊥)), (1, Sem_{Meta}(0)), (2,Sem_{Meta}((U.count+1)[L_{count}(1)/U.count])),
(3,Sem_{Meta}(⊥)), (4,Sem_{Meta}((U.count+1)[L_{count}(2)/U.count])), (5,Sem_{Meta}(⊥)),
(6, Sem_{Meta}(print(U.count)[L_{count}(4)/U.count]))}

Sem_{Meta}(print(U.count)[L_{count}(4)/U.count]) = Sem_{Meta}(print(U.count+1)[L_{count}(2)/U.count])
= Sem_{Meta}(print(U.count+1+1)[L_{count}(1)/U.count])
= Sem_{Meta}(print(0+1+1))

Apply Attribute Grammar to Parse Trees: An Exercise

Let *mk-tree* be an arity-variant procedure that applies to a *label* and to n *trees* and results the tree rooted on a node having, as label, the first argument, and, as sons, the n trees, in the order of appearance from left. Say: 1) the type of the attributes; 2) the family $L_{\{ai\}}$; 3) the value of each attribute relating to 2+3.

E::= F E' [0]
E'::= + E [1]| ε [2]
F::= H F' [3]
F'::= * F [4]| ε [5]
H::= num [6]| (E) [7]

E::= F E' [0]	**E.tree:= mk-tree('E', F.tree, E'.tree), F.depth:=E.depth+1, E'.depth:=E.depth+1**
E'::= + E [1]	**E'.tree:= mk-tree('E'', mk-leaf('+'), E.tree), +.depth:=E'.depth+1,E.depth:=E'.depth+1**
E'::= ε [2]	**E'.tree:= mk-tree('E'', mk-leaf('ε')) ε,depth:=E'.depth+1**
F::= H F' [3]	**F.tree:= mk-tree('F', H.tree, F'.tree), H.depth:=F.depth+1, F'.depth:=F.depth+1**
F'::= * F [4]	**F'.tree:= mk-tree('F'', mk-leaf('*'), F.tree), *.depth:=F'.depth+1, F.depth:=F'.depth+1**
F'::= ε [5]	**F'.tree:= mk-tree('F'', mk-leaf('ε')) ε.depth:=E'.depth+1**
H::= num [6]	**H.tree:= mk-tree('H', mk-leaf(num)), num.depth:=H.depth+1**
H::= (E) [7]	**H.tree:= mk-tree('H', mk-leaf('('), E.tree, mk-leaf(')')), E.depth:= H.depth+1, (.depth:=H.depth+1,).depth:=H.depth+1,**