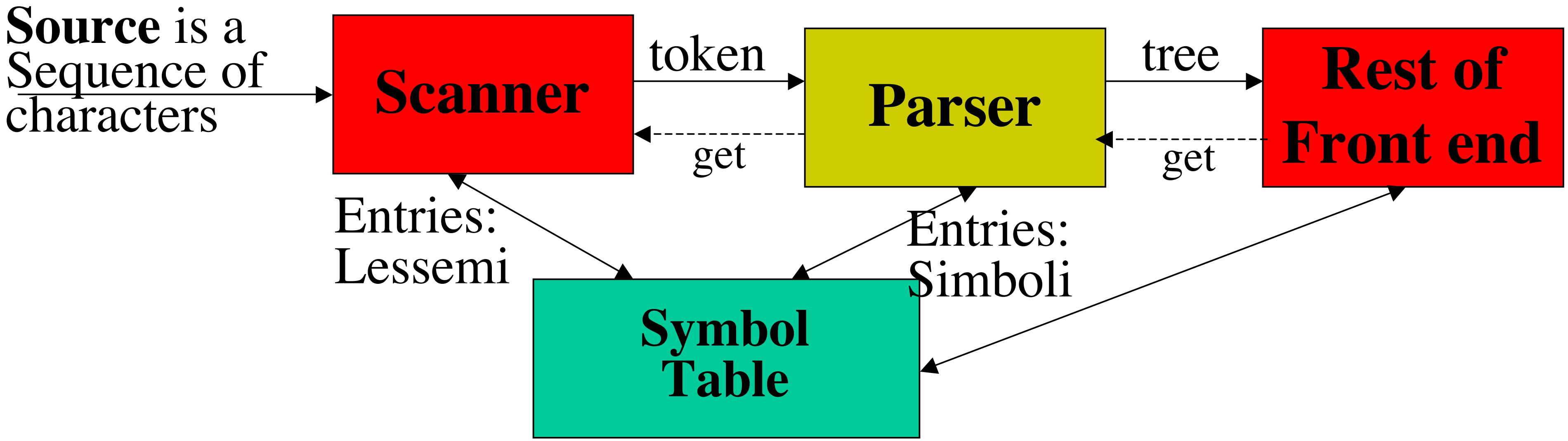
SYNTACTIC ANALYSIS

- **How to define Syntax;**
- **How to do Analysis;**
- **At what extent Analysis can be done in Linear Time;**
- **How to build (Linear) Parser Generators;**
- **Relationship between Analysis and Internal Representation (Tree Representation)**

One Pass Structure (Two phase pipeline)



Syntactic Analysis (Parser) is driven from
Semantics Analysis which is asking for visiting a subtree not built yet

How to define Syntax

Syntactic Analysis

- It scans sequences of tokens to check for *phrase structures* that belong to the Syntax of the Language
- Syntax, just like Lexics, is expressed by a Language: The Syntactic Language
- Syntactic Langs are much more complicate than Lexical ones

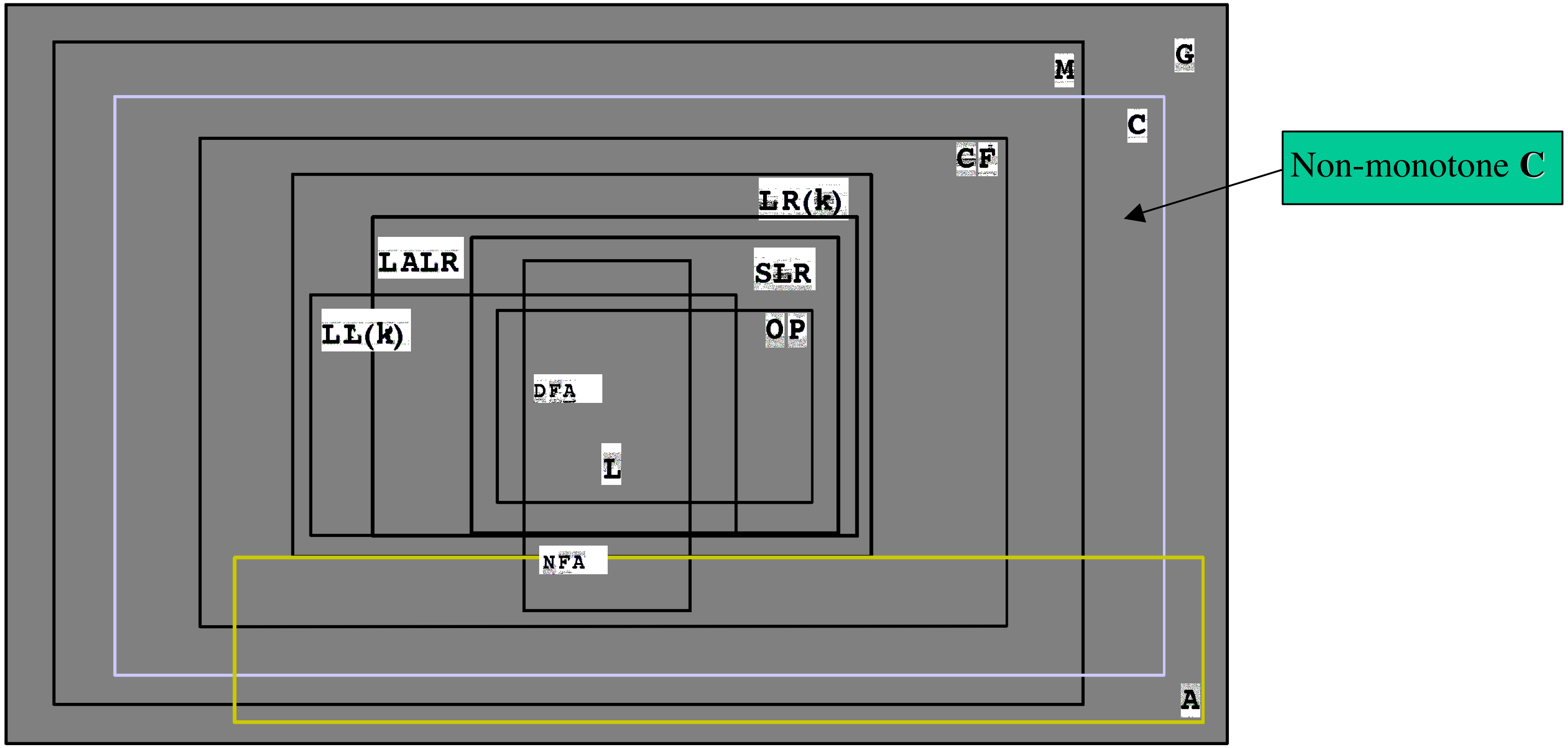
$\{u^n v^n \mid n \geq 0\}$ has been proved not be regular

but

num+(3-((id*id)+num)/id) $\in \{(u\alpha)^n (\alpha v)^n \mid n \geq 0, \alpha \in L, u=\text{“(”}, v=\text{“)”}\}$

Grammar Classification (Chomsky)

Grammars Inclusion

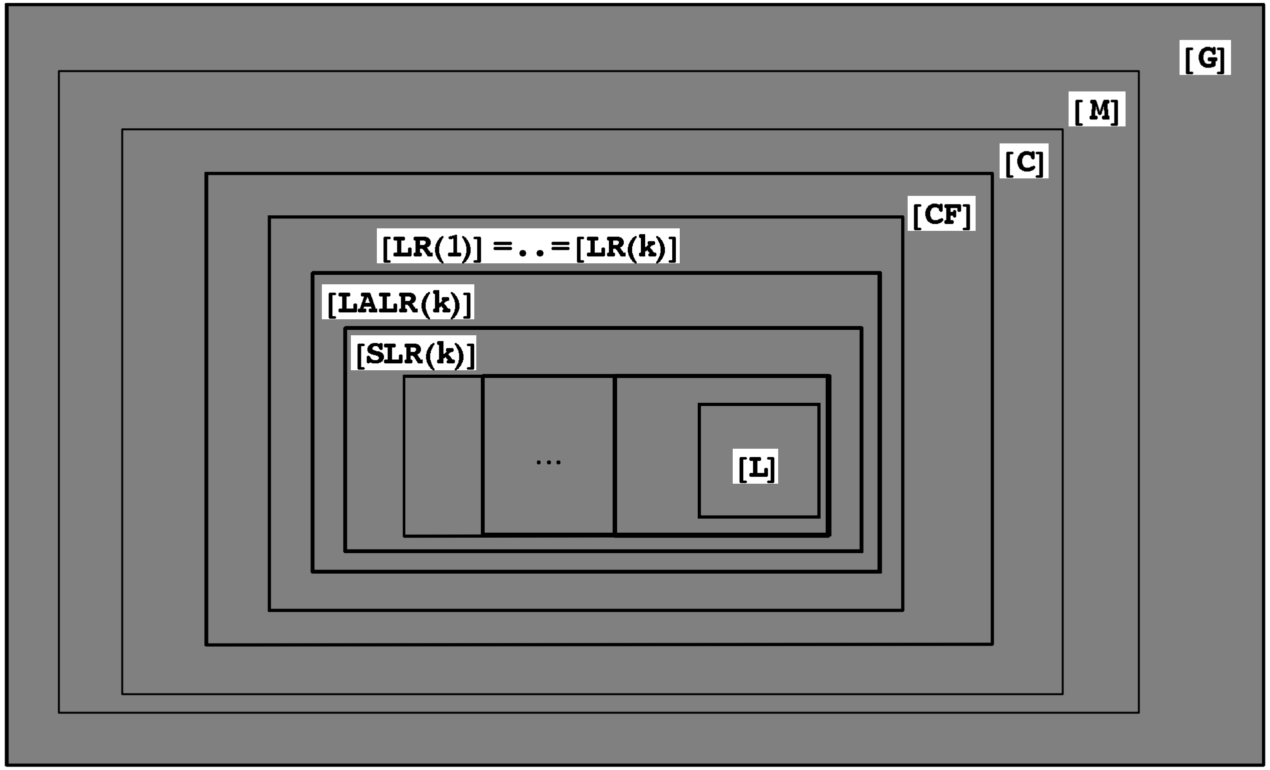


G=General [Recursive Enumerable but Non-Recursive -{$u^n v^{akermann(n)}$}]
A=Ambiguous
M=Monotone [Recursive Languages - {$u^n v^{n!}$}]
C=Contextual [{$u^n v^n z^n$}]
LR(k)=Context-Free [{$u^n v^n$}]
LALR(k)=Context-Free [{$u^n v^n$}]
SLR(k)=Simple Left-to-right rightmost reversed [Viable-Prefix; Bottom-Up/k symbols{$u^n v^n$}]
LL(K)=Leftmost-Left Left-to-right [Predictive; Top-Down/k symbols{$u^n v^n$}]
OP=Operator-Precedence [{$u^n v^n$}]
L=Linear [Recursive Grammars; Regular Languages]
DFA= -- [Regular Grammars/Expressions; Regular Languages]
NFS= -- [Regular Grammars/Expressions; Regular Languages]

Kinds of Grammar [Defined Language Features]

Language Classification

Languag e Inclusion



[G] = Recursively Enumarable Languages
[M] = Recursive Languages
[C] = Contextual Languages: $\{u^n v^n z^n \mid n \geq 0\}$
[CF] = Context-Free Languages: $\{u^n v^m z^k \mid n,m,k \geq 0$ and $(n=m$ or $m=k)\}$
[LR(k)] = LR/k symbols Languages: $\{u^m v^n \mid m > n \geq 0\}$
[LALR(k)] = LALR/k symbols Languages
[SLR(k)] = SLR/k symbols Languages
[LL(k)] = LL/k symbols Languages: $\{u^n v^n \mid n \geq 0\}$
[L] = Regular Languages: $\{u^n v^m \mid n \geq 0, m \geq 0\}$

Definitions: Derivation, SF

Let **G = <V,Σ,s∈V,P>**

Derivation is a binary relation $\Rightarrow_G$ su $(\Sigma \cup V)^* \times (\Sigma \cup V)^*$

$$\alpha A\beta \Rightarrow \alpha\gamma\beta \quad \textit{sse} \quad A::=\gamma \in P$$

Subscript, G, is omitted, in $\Rightarrow_G$, when the grammar G is clearly stated from the context

=>*: Transitive and Reflexive Closure of **=>**

- $\alpha \Rightarrow^* \alpha$
- if $\alpha_1 \Rightarrow \ldots \Rightarrow \alpha_n$ then $\alpha_1 \Rightarrow^* \alpha_n$

=>⁺: Transitive Closure of **=>**

if $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$ and $\alpha_1 \neq \alpha_2 \neq \ldots \neq \alpha_n$ allora $\alpha_1 \Rightarrow^+ \alpha_n$

Sentential form of G

SF={γ | s =>* γ}

L(G): Language Generated by a Grammar

Let **G = <V,Σ,s∈V,P>**

$L(G)=\{w \in \Sigma^* \mid s \Rightarrow^+ w\}$

(where $\Rightarrow$ is $\Rightarrow_G$)

Example: Let G below

p1: E::= E+E
p2: E::= E*E
p3: E::= id

Then

id+id∈L(E)

A proof (Lefmost Derivation):

E $\Rightarrow_{(p1)}$ E+E $\Rightarrow_{(p3)}$ id+E $\Rightarrow_{(p3)}$ id+id

A different proof (Rightmost Derivation):

E $\Rightarrow_{(p1)}$ E+E $\Rightarrow_{(p3)}$ E+id $\Rightarrow_{(p3)}$ id+id

Ambiguous Grammars are Bad Definitions for Lang. Syntax

Example: Let G below

p1: E::= E+E
p2: E::= E*E
p3: E::= id

Then

id+id*id∈L(E)

A proof (Lefmost Derivation):

$E \Rightarrow_{(p1)} E+E \Rightarrow_{(p3)} id+E \Rightarrow_{(p2)} id+E*E \Rightarrow_{(p3)} id+id*E \Rightarrow_{(p3)} id+id*id$

A different proof (another Leftmost Derivation):

$E \Rightarrow_{(p2)} E*E \Rightarrow_{(p1)} E+E*E \Rightarrow_{(p1)} id+E*E \Rightarrow_{(p1)} id+id*E \Rightarrow_{(p1)} id+id*id$

Different Leftmost (Rightmost) Derivations lead to different Parse Trees

Derivations (on the P-tree domain)

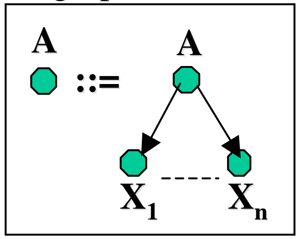
Let **G = <V,Σ,s∈V,P>**

$(\Sigma \cup V)_T^*$

Smallest set such that, $\forall a \in \Sigma \cup V$:

- **leaf:** $<[a,-]> \in (\Sigma \cup V)_T^*$
- $\forall <t1,\ldots,tn> \in (\Sigma \cup V)_T^*$
  - **Tree:** $<[a,<t1,\ldots,tn>]> \in (\Sigma \cup V)_T^*$
  - **Forest:** $<t1,\ldots,tn,u1,\ldots,um> \in (\Sigma \cup V)_T^*$, $\forall <u1,\ldots,um> \in (\Sigma \cup V)_T^*$

A graphical view



Productions on $(\Sigma \cup V)_T^*$

$A::=X_1 \ldots X_n \in P$ sse $<[A,-]>::= <[A,<<[X_1,-]>,\ldots,<[X_n,-]>>]> \in P_T$

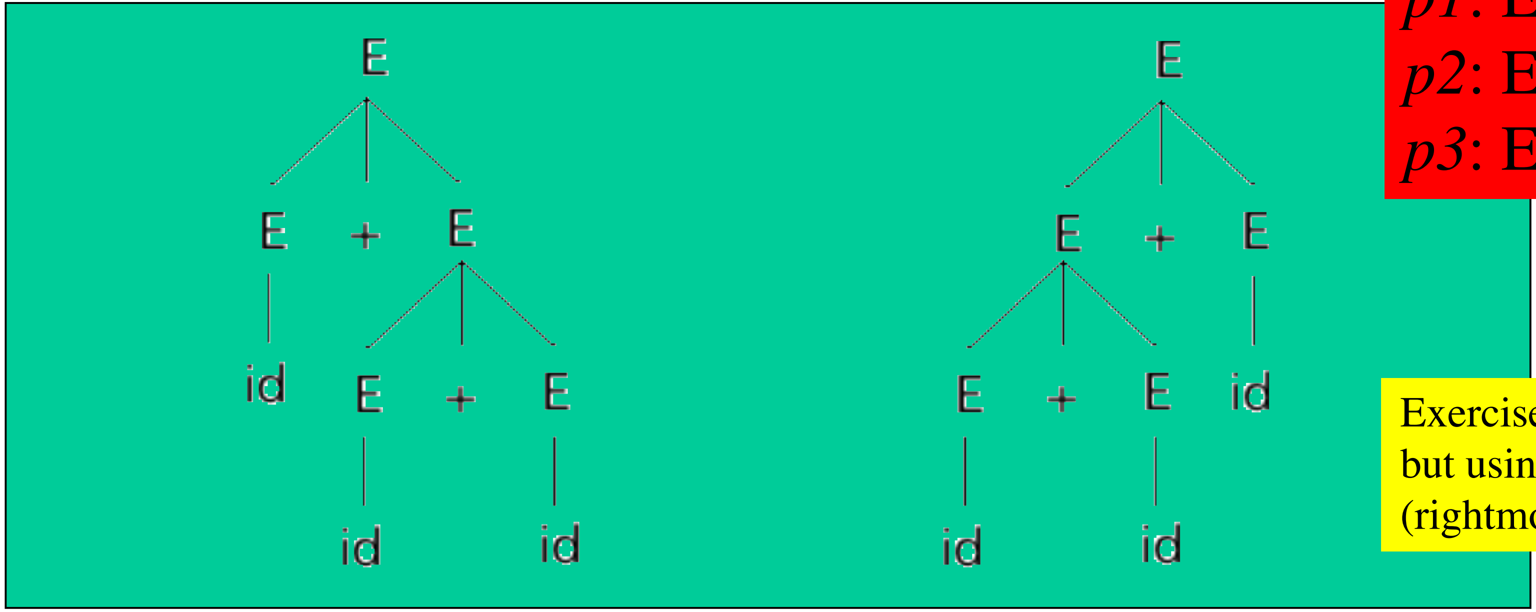
Relation => on $(\Sigma \cup V)_T^*$ x $(\Sigma \cup V)_T^*$

$\alpha A \beta \Rightarrow \alpha \gamma \beta$ sse $A::=\gamma \in P_T$

Ambiguous Grammars
A Graphical View

A different proof of ambiguity that uses: The *last trees* of two different Tree-derivations

p1: E::= E+E
p2: E::= E*E
p3: E::= id



Exercise: Show the same but using leftmost (rightmost) derivations

From Grammars to Languages

A methodology for finding L(G), given G

$A_0 ::= e_0$
$A_1 ::= e_1$
…
$A_n ::= e_n$

1) Partial ordering $\geq^G$ on non-Terminals

$\forall i \geq 0,\quad e_i \equiv f(A_{i_1},\ldots,A_{i_{n_i}})\quad$ then: $A_{i_1},\ldots,A_{i_{n_i}} \geq^G A_i$

Removal of Mutual Recursion, when possible

$A_j ::= g(A_{j_1},\ldots,A_i,\ldots,A_{j_{nj}})\quad$ con $A_{j_1},\ldots,A_{j_{nj}} \geq^G A_i \geq^G A_j$
$A_i ::= f(A_{i_1},\ldots,A_j,\ldots,A_{i_{n_i}})\quad$ con $A_{i_1},\ldots,A_{i_{n_i}} \geq^G A_i$

⇓

$A_j ::= g(A_{j_1},\ldots,A_i,\ldots,A_{j_{nj}})$
$A_i ::= f(A_{i_1},\ldots, g(A_{j_1},\ldots,A_i,\ldots,A_{j_{nj}}),\ldots,A_{i_{n_i}})$

Example

S::= u A B v | B u
A::= v u v | B
B::= u S v | u v u

B ≥G A ≥G S

S::= u A B v | B u
B::= u S v | u v u

S::= u A B v | B u
B::= u (u A B v | B u) v | u v u

S::= u A B v | B u
A::= v u v | B
B::= uu A B vv | u B uv | u v u

A::= v u v | B
B::= uu A B vv | u B uv | u v u

A::= v u v | B
B::= uu (v u v | B) B vv | u B uv | u v u

S::= u A B v | B u
A::= v u v | B
B::= uu vuv B vv | uu B B vv | u B uv | u v u

2) Productions as equations on languages $L(A_i)$

$\forall i \geq 0, \quad L(A_i) = L(e_i)$

- L(e) is an expression on $\mathbf{2}^{\Sigma^*}$ containing only:
  - X (finite products)
  - ∪ (possibly, denumerable unions)
- L(e) is continuos on $\mathbf{2}^{\Sigma^*}$

Whenever $L(A_i) = L(e_i)$ is recursive: $L(e_i) \equiv E(L(A_i))$

Recursive equations X=E(X), have to be solved in the variable $X \equiv L(A_i)$ on $\mathbf{2}^{\Sigma^*}$ using the **Tarski's (Fixpoint) Iteration** below:

$$X = \bigcup_{i \in \aleph} E(\bot)^i$$

$$E(\bot)^0 = E(X \leftarrow \bot)$$

$$E(\bot)^{k+1} = E(X \leftarrow E(\bot)^k)$$

What about a system of equations

A system of Recursive equations:

$$\{X_1=E_1(X_1,\ldots X_n),\ldots, X_n=E_n(X_1,\ldots X_n)\}$$

$X_i \equiv L(A_i)$ on $\mathbf{2^{\Sigma^*}}$ using the **Tarski's (Fixpoint) Iteration** below:

$$\mathbf{X_j = \cup_{i\in\aleph} E_j(\perp,\ldots,\perp)^i}$$

$$\mathbf{E_j(\perp,\ldots,\perp)^0 = E_j(X_1 \leftarrow \perp ,\ldots, X_n \leftarrow \perp)}$$

$$\mathbf{E_j(\perp ,\ldots,\perp)^{k+1} = E_j(X_1 \leftarrow E_1(\perp)^k ,\ldots, X_n \leftarrow E_n(\perp)^k)}$$

Example1: Tarski's Fixpoint Iteration

S::=u S | ε → X= {u}×X∪{λ}

E(X)

$E(X\leftarrow \bot)^0 = \{u\}\times\bot\cup\{\lambda\} = \bot\cup\{\lambda\} = \{\lambda\}$

$E(X\leftarrow \bot)^1 = \{u\}\times\{\lambda\}\cup\{\lambda\} = \{u, \lambda\}$

$E(X\leftarrow \bot)^2 = \{u\}\times\{u,\lambda\}\cup\{\lambda\} = \{uu, u, \lambda\}$

$E(X\leftarrow \bot)^3 = \{u\}\times \{uu, u, \lambda\} \cup\{\lambda\} = \{u^3,u^2,u, \lambda\}$

↓

$E(X\leftarrow \bot)^n = \{u^n,u^{n-1},\ldots,u, \lambda\}$

↓

$L(S)=\{u^n \mid n\in\aleph\}$ **= u***