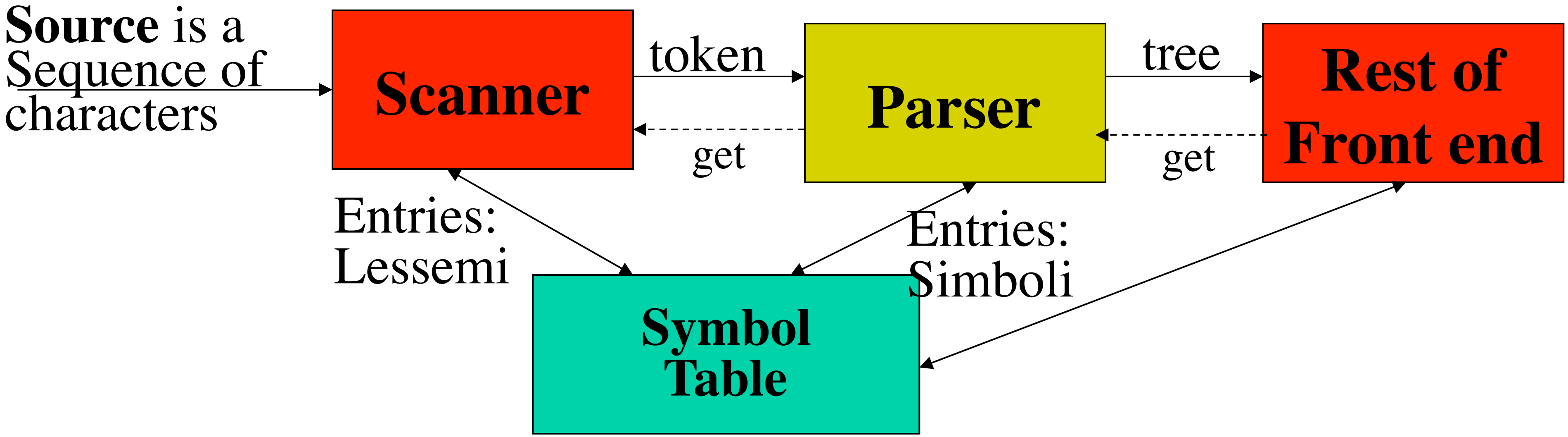
SYNTACTIC ANALYSIS

- 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 and Syntactic Languages
- Syntactic Languages and Grammars
- Classification of Grammars
- Classification of Languages
- Foundations: Derivation, Sentential Form, Ambiguity, Tarski' s Fixpoint Iteration

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 complicated than Lexical ones

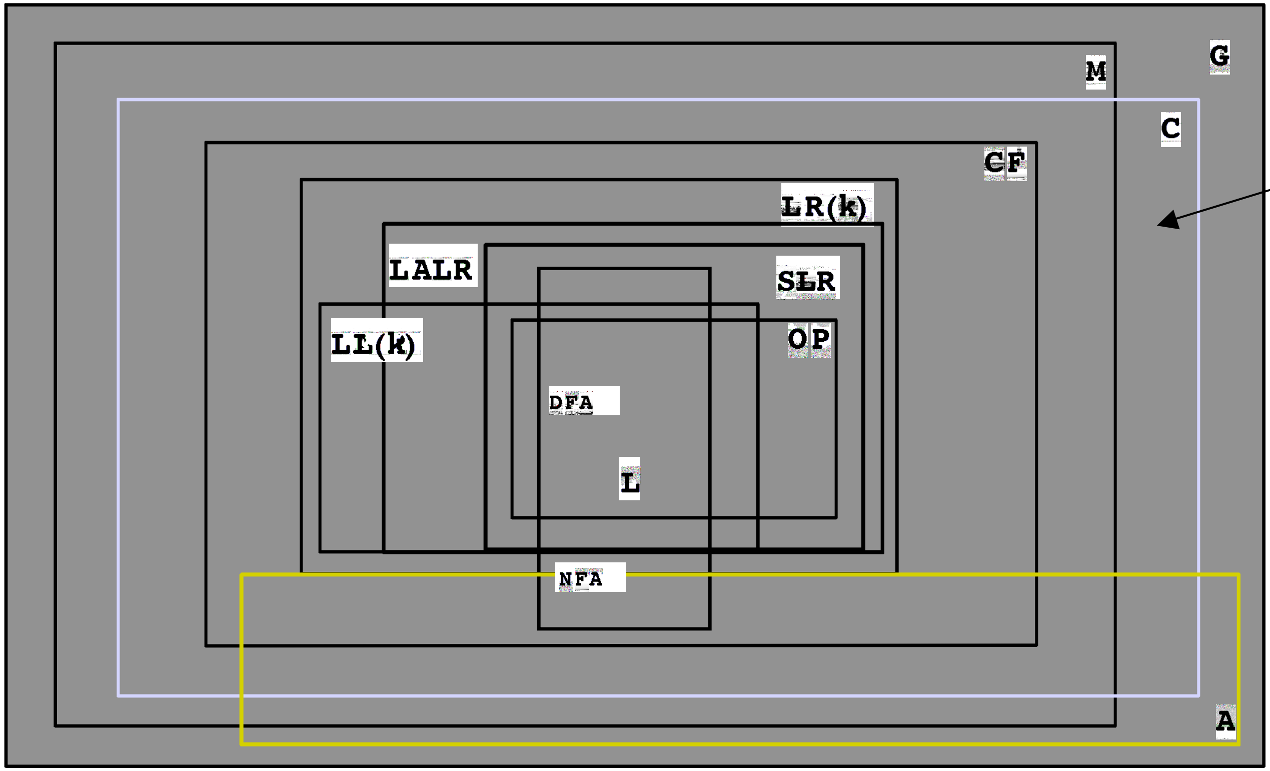
$\{u^n v^n \mid n \geq 0\}$ has been proved not be regular

but

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

Grammar Classification (Chomsky)

Grammars Inclusion



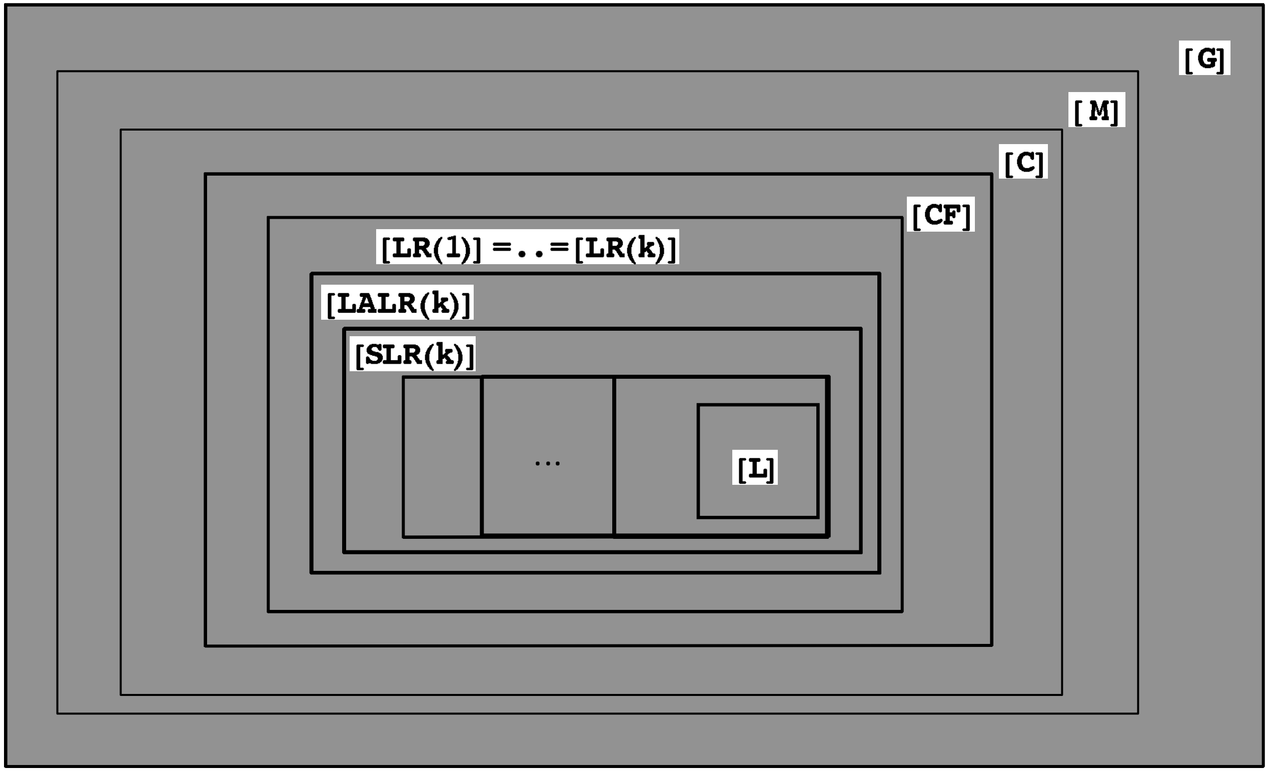
Non-monotone C

G=General [Recursive Enumerable but Non-Recursive -{$u^nv^{akermann(n)}$}]
A=**Ambiguous**
M=Monotone [Recursive Languages - {$u^nv^{n!}$}]
C=Contextual [{$u^nv^nz^n$}]
LR(k)=**Context-Free** [{u^nv^n}]
LALR(k)=**Context-Free** [{u^nv^n}]
SLR(k)=**Simple Left-to-right rightmost reversed** [Viable-Prefix; Bottom-Up/k symbols{u^nv^n}]
LL(K)=**Leftmost-Left Left-to-right** [Predictive; Top-Down/k symbols{u^nv^n}]
OP=Operator-Precedence [{u^nv^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: $\{u^n v^m z^k \mid n,m,k \geq 0 \text{ and } (n=m \text{ or } m=k)\}$
[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
$SF=\{\gamma \mid s \Rightarrow^* \gamma\}$

L(G): Language Generated by a Grammar

Let **G = <V,Σ,s∈V,P>**

L(G)={w ∈Σ* | s =>⁺ w}

(where **=>** is **=>$_G$**)

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 =>$_{(p1)}$ E+E =>$_{(p3)}$ id+E =>$_{(p3)}$ id+id

A different proof (Rightmost Derivation):
E =>$_{(p1)}$ E+E =>$_{(p3)}$ E+id =>$_{(p3)}$ 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 =>$_{(p1)}$ E+E =>$_{(p3)}$ id+E =>$_{(p2)}$ id+E*E =>$_{(p3)}$ id+id*E =>$_{(p3)}$ id+id*id

A different proof (another Leftmost Derivation):

E =>$_{(p2)}$ E*E =>$_{(p1)}$ E+E*E =>$_{(p1)}$ id+E*E =>$_{(p1)}$ id+id*E =>$_{(p1)}$ id+id*id

Different Leftmost (Rightmost) Derivations lead to different Parse Trees

Derivations (on the P-tree domain)

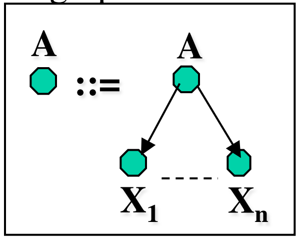
Let **G = <V,Σ,s∈V,P>**

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

Smallest set such that, ∀a∈ Σ∪V:

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

A graphical view



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

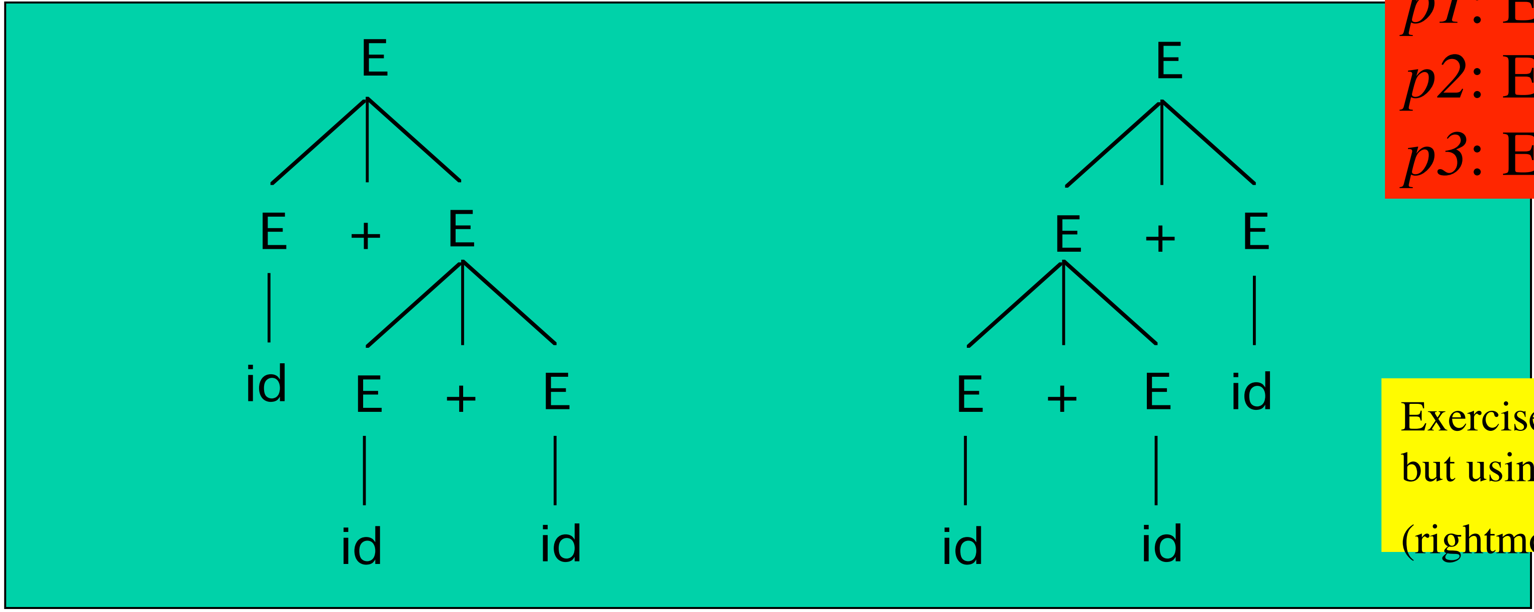
$A::=X_1 \ldots X_n$ ∈ P sse <[A,-]>::= <[A,<<[X_1,-]>,…,<[X_n,-]>>]> ∈ P_T

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

αAβ => αγβ sse A::=γ ∈ 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

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

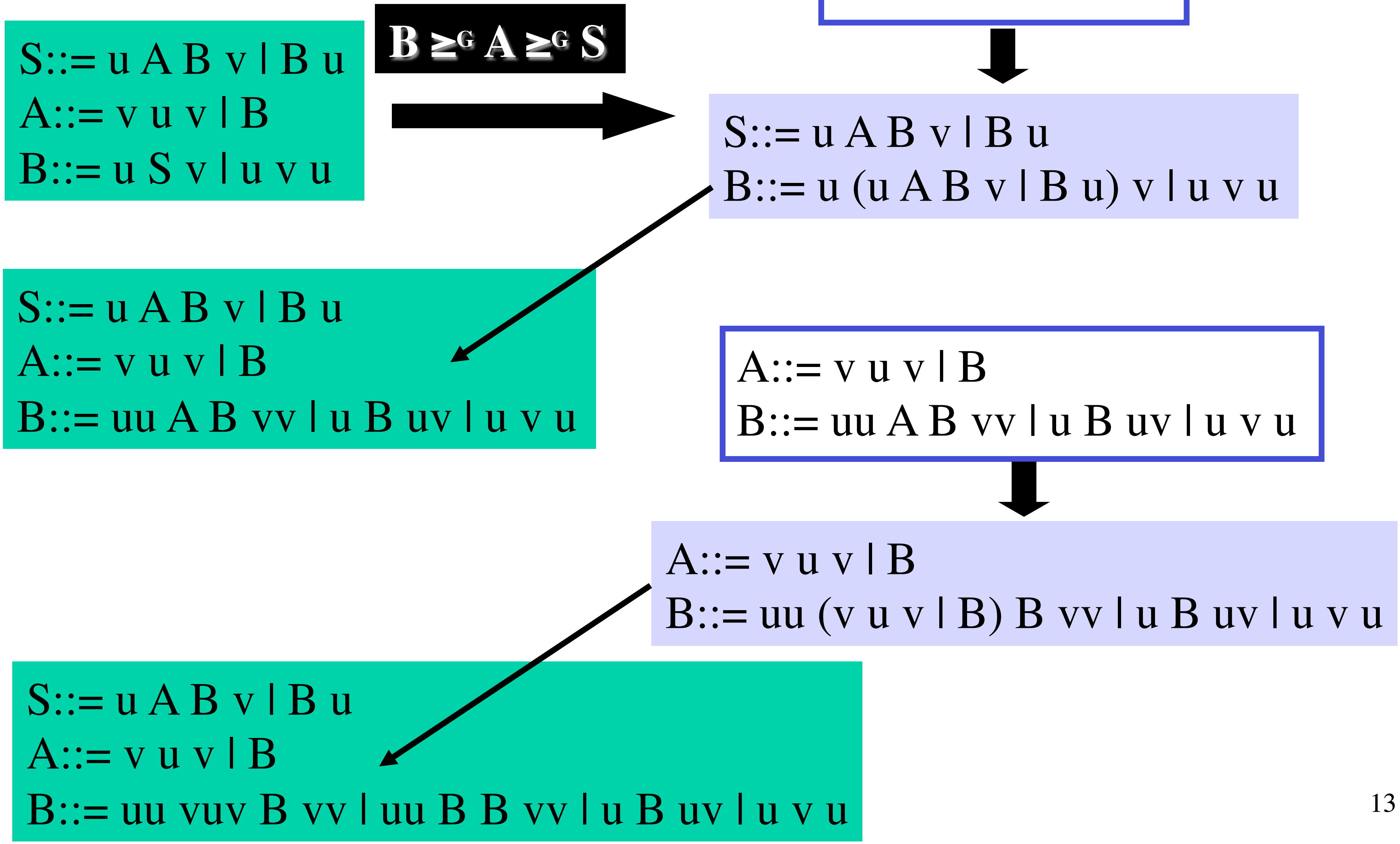
Removal of Mutual Recursion, when possible

$A_j ::= g(A_{j1},\ldots,A_i,\ldots,A_{jn_j})$ con $A_{j1},\ldots,A_{jn_j} \geq^G A_i \geq^G A_j$
$A_i ::= f(A_{i_1},\ldots,A_j,\ldots,A_{i_{n_i}})$ con $A_{i_1},\ldots,A_{i_{n_i}} \geq^G A_i$

$\Downarrow$

$A_j ::= g(A_{j1},\ldots,A_i,\ldots,A_{jn_j})$
$A_i ::= f(A_{i_1},\ldots, g(A_{j1},\ldots,A_i,\ldots,A_{jn_j}),\ldots,A_{i_{n_i}})$

Example



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

$B \geq^G A \geq^G S$

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:

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

$$E_j(\bot,\ldots,\bot)^0 = E_j(X_1 \leftarrow \bot ,\ldots,X_n \leftarrow \bot)$$

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

Example1: Tarski’ s Fixpoint Iteration

$$S::=u\ S \mid \varepsilon \quad\longrightarrow\quad X=\underbrace{\{u\}\times X\cup\{\lambda\}}_{\boxed{E(X)}}$$

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

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

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

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

$$\Downarrow$$

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

$$\Downarrow$$

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

Example2: Tarski’ s Fixpoint Iteration

$$S ::= u\ S\ v \mid z \longrightarrow X = \underbrace{\{u\}\times X \times \{v\} \cup \{z\}}_{\boxed{E(X)}}$$

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

$$E(X \leftarrow \bot)^{\mathbf{1}} = \{u\}\times\{z\}\times\{v\} \cup \{z\} = \{uzv, z\}$$

$$E(X \leftarrow \bot)^{\mathbf{2}} = \{u\}\times\{uzv, z\}\times\{v\} \cup\{z\} = \{u^{\mathbf{2}}zv^{\mathbf{2}}, uzv, z\}$$

$$\Downarrow$$

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

$$\Downarrow$$

$$L(S)=\{u^{\mathbf{n}}zv^{\mathbf{n}} \mid n\geq 0\}$$

Example3: Tarski’ s Fixpoint Iteration

A::= A+A
A::= A∗A
A::= id

$$\Rightarrow \quad X= \{x+y \mid x,y\in X\}\cup\{x*y \mid x,y\in X\}\cup\{id\}$$

$$\boxed{E(X)}$$

$$E(X\leftarrow \bot)^{\mathbf{0}} = \{x+y \mid x,y\in\bot\}\cup\{x*y \mid x,y\in\bot\}\cup\{id\}$$
$$= \bot\cup\bot\cup\{id\} = \{id\}$$

$$E(X\leftarrow \bot)^{\mathbf{1}} = \{x+y \mid x,y\in \{id\}\}\cup\{x*y \mid x,y\in \{id\}\}\cup\{id\}$$
$$= \{id+id\}\cup\{id*id\}\cup\{id\} = \{id,\ id+id,\ id*id\}$$

$$E(X\leftarrow \bot)^{\mathbf{2}} = \{x+y \mid x,y\in E(X\leftarrow \bot)^{\mathbf{1}}\}\cup\{x*y \mid x,y\in E(X\leftarrow \bot)^{\mathbf{1}}\}\cup\{id\}$$
$$= \{id,\ id+id,\ id*id, id+id+id,\ id+id*id,\ id*id+id,\ id*id*id,\ldots,\ id*id*id*id\}$$

$$\Downarrow$$

$$E(X\leftarrow \bot)^{\mathbf{n}} = \{id,\ id\ t^{k} \mid t\in\{+id,\ *id\},\ k \in[1..2^{n}\text{-}1]\}$$

$$\Downarrow$$

$$L(A)=\{id\ t^{\mathbf{n}} \mid t\in\{+id,\ *id\},\ n\in\aleph\}$$

How to do Syntactic Analysis

- Top-Down and leftmost derivation
- Bottom-Up and reversed roghtmost dervation

TOP-DOWN and BOTTOM-UP Parsers

Let **G=<V,Σ,s∈V,P>** be a (context free) grammar. Let **w** be a sequence of words in **Σ.**

- Analysis has to answer to the following question:

  is **w ∈L(G) or not** ?

- or, equivalently:

  is **s** =>***w or not** ?

- **Membership:** is this Decision Problem, computable?

  -- Yes. It is decidable for all classes of **Monotone Grammars**.

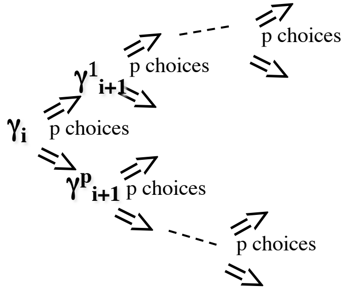
- The solution consists in defining a procedure (**The Parser Core**) able to construct a derivation $s \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_k \equiv w$, if one exists.

Construction of a Derivation

- The solution consists in defining a procedure (**The Parser Core**) able to construct a derivation $s \Rightarrow \gamma_1 \Rightarrow \ldots \Rightarrow \gamma_k \equiv w$, if one exists.

- The construction of a derivation could be done in a non-efficient way, and even worse, at a non-linear, up to exponential, complexity time (/space) cost.

Trying *p* optional productions at each γ_i leads to:

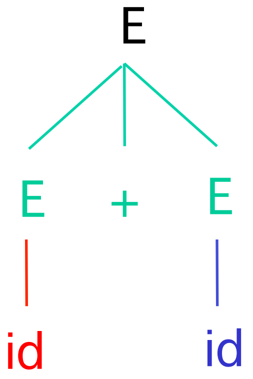


construction of (exponential) (O(p^n)) derivations to find the one right or to answer “no-accept”.

Top-Down

Simple for Handmade Constructions,
Few Grammars

$E \underset{l}{=>} E+E \underset{l}{=>} id + E \underset{l}{=>} id+id$



Leftmost non-terminal of Left-Sentential-Form

First Applicable Production

Failure: Backward to the last alternative

Step 1
Step 2
Step 3

Top-Down = Leftmost

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

LSF forms a Complete Base for Context-Free Grammars

G = <V,Σ,s∈V,P>

Left Sentential Form (of G): $\boldsymbol{LSF_G}$

$\alpha\beta\gamma \in LSF_G$ *iff* $s \ {}_{l}\!\!\Rightarrow^{+} \alpha\beta\gamma$

αAβ $_{l}$**=> αγβ** ***iff*** A::=γ ∈P & α ∈Σ*

Only $\boldsymbol{LSF_G}$

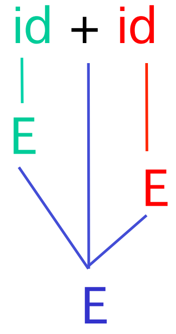
$$\begin{aligned} L(G) &= \{w \in \Sigma^* \mid s \Rightarrow^{+} w\} \\ &= \{w \in \Sigma^* \mid s \ {}_{l}\!\!\Rightarrow^{+} w\} \end{aligned}$$

Bottom-Up

More Complicated Techniques

Many More Grammars - Many More Languages

$\boxed{E \underset{r}{=>} E+E \underset{r}{=>} E+\mathrm{id} \underset{r}{=>} \mathrm{id}+\mathrm{id}}$



Looking for Handle

reduction

failure: backward for “true” Handle

Step 1
Step 2
Step 3

Bottom-Up = Rightmost Reversed

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

RSF forms a Complete Base for Context-Free Grammars

$G = <V,\Sigma,s\in V,P>$

Right Sentential Form (of G):RSF_G

$\alpha\beta\gamma \in RSF_G$ *iff* $s \; {}_r\!\!=>^+ \alpha\beta\gamma$

$\alpha A\beta \; {}_r\!\!=> \alpha\gamma\beta$ *iff* $A::=\gamma \in P$ & $\beta\in\Sigma^*$

Only RSF

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

$=\{w\in \Sigma^* \mid s \; {}_r\!\!=>^+ w\}$

$B::=\beta$ $\in P$ is **Handle** of **$\alpha\beta\gamma$** $\in RSF_G$

if and only if $\alpha B\gamma \in RSF_G$