

Pattern Recognition VIII

Structural & Syntactic PR

Michal Haindl

Faculty of Information Technology, KTI
Czech Technical University in Prague
Institute of Information Theory and Automation
Czech Academy of Sciences
Prague, Czechia



<http://ro.utia.cz/>

MI/MIE-ROZ 2023-2024



Outline

- 1 Structural and Syntactic PR
 - Notions
- 2 Syntactic PR
 - Relations
 - Formal Language Description
 - Grammars
 - Finite-State Languages and Finite-State Automata
 - Syntactic Analysis
 - Grammar Inference
- 3 Structural PR
 - Graph Matching



Structural and Syntactic PR

PR based on symbolic data structures (strings, trees, graphs, arrays ..)

PR made by parsing the pattern structure according to a given set of syntax rules

- objects described with **primitives** and relations
- recognition - comparison of symbolic object representation with predefined object models



Structural and Syntactic PR

- **structural** recognition - similarity measure between the input and number of prototype models



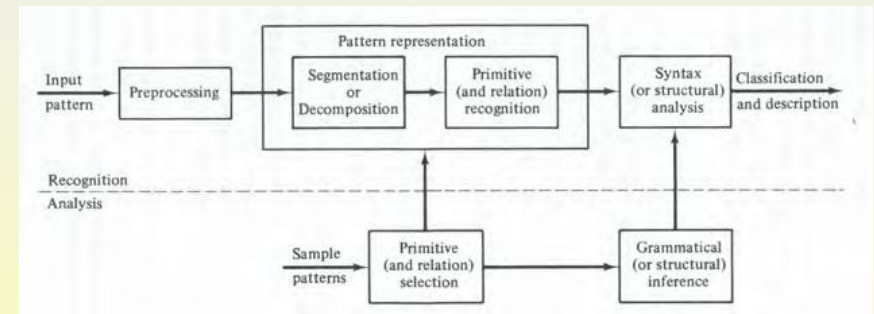
Structural and Syntactic PR

- **structural** recognition - similarity measure between the input and number of prototype models
- **syntactic** recognition (analysis) - parser checks an input whether it is generated from a grammar of a class

$\omega_i \sim G_i$
 assign X into class ω_i if $i = 1, \dots, K$
 $X \in \mathcal{L}(G_i)$



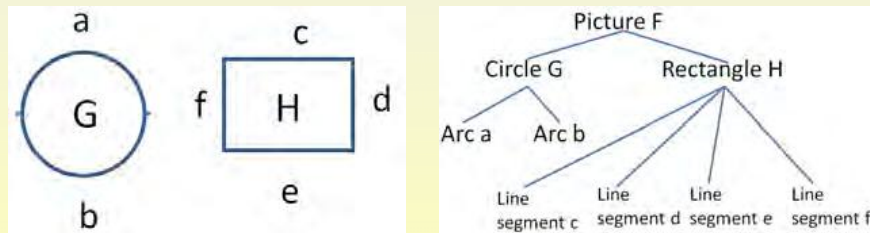
Syntactic & Structural Pattern Recognition System



Syntactic Representation

primitive - an elementary subpattern, cannot be further divided

tree - hierarchical pattern representation, subset of a relational graph

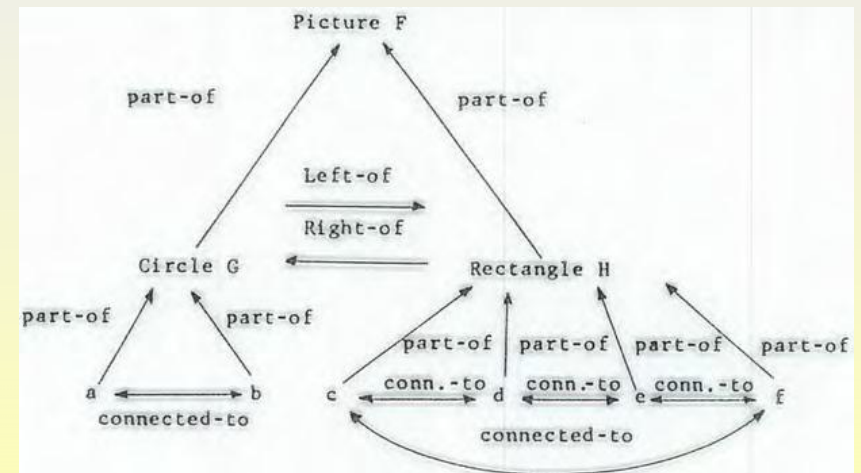


tree



Syntactic Representation 2

relational graph - consists from several related hierarchical trees

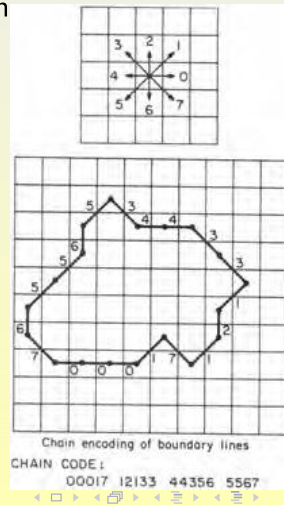


Notions

⊙ primitives $\cup \cap \prod \Pi$

- primitive - elementary pattern component
- compact but adequate description
- easily extractable,
e.g., chain code (Freeman)

← ↖ ↑ ↗ → ↘ ↓ ↙



Notions

⊙ primitives $\cup \cap \prod \Pi$

- primitive - elementary pattern component
- compact but adequate description,
- easily extractable,
e.g., chain code (Freeman)
- number of primitives should be small
- primitives must be able to represent all patterns
- primitives should be easily segmentable and recognizable
- primitives should correspond with significant natural elements of the object

⊙ alphabet - the set of all primitives

⊙ object representation - symbolic (strings, trees, graphs, arrays)

⊙ word - chain of primitives,

$X \vdash Y$ word Y directly derived from X

Notions

⊙ primitives $\cup \cap \prod \Pi$

- primitive - elementary pattern component
- compact but adequate description,
- easily extractable,
e.g., chain code (Freeman)
- number of primitives should be small
- primitives must be able to represent all patterns
- primitives should be easily segmentable and recognizable
- primitives should correspond with significant natural elements of the object

⊙ alphabet - the set of all primitives

⊙ object representation - symbolic (strings, trees, graphs, arrays)

⊙ word - chain of primitives,

$X \vdash Y$ word Y directly derived from X

Notions

⊙ primitives $\cup \cap \prod \Pi$

- primitive - elementary pattern component
- compact but adequate description,
- easily extractable,
e.g., chain code (Freeman)
- number of primitives should be small
- primitives must be able to represent all patterns
- primitives should be easily segmentable and recognizable
- primitives should correspond with significant natural elements of the object

⊙ alphabet - the set of all primitives

⊙ object representation - symbolic (strings, trees, graphs, arrays)

⊙ word - chain of primitives,

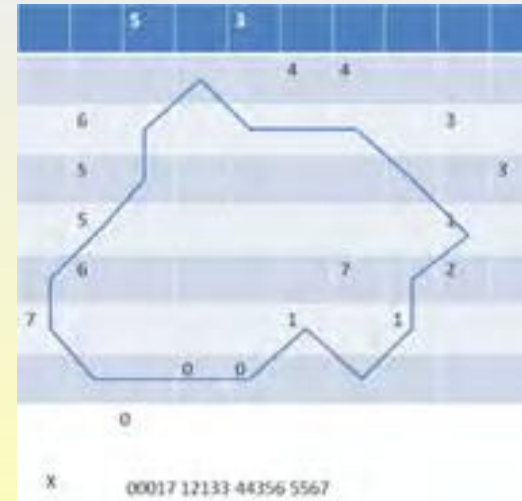
$X \vdash Y$ word Y directly derived from X

Structural PR - Template Matching

- 1 define primitives and their possible relations
- 2 construct a description in the form of string of primitives for each class prototype
- 3 recognition - for each object extract primitives, construct a descriptive string of primitives
- 4 based on the string of primitives classify an object into a class for which this string is the best match



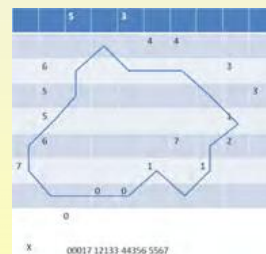
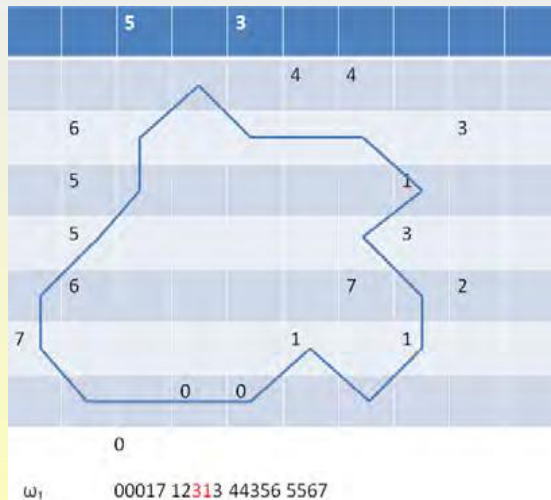
Structural PR - Template Matching



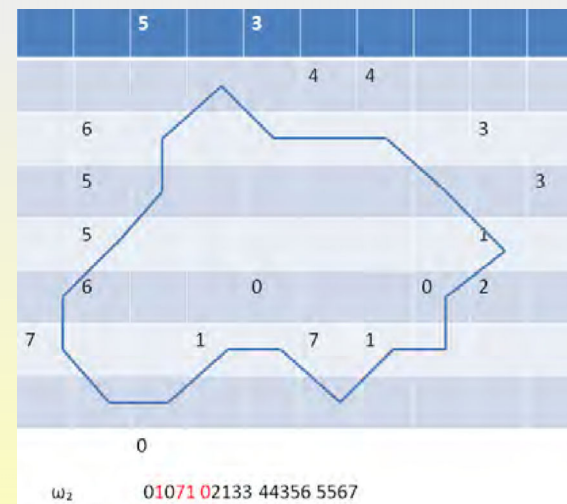
object



Structural PR - Template Matching



Structural PR - Template Matching

 $X \in \omega_1$

2 differences

(X - 4 differences from ω_2)

Syntactic Notions

- ⊙ grammar $G = \langle V_t, V_n, R, S \rangle$
 - G grammar - specifies the rules governing the composition of primitives into patterns
 - V_t a finite set of terminal symbols (alphabet)
 - V_n a finite set of nonterminal symbols $V_t \cap V_n = \emptyset$
 - $V = V_n \cup V_t$
 - V^* a set of all finite-length strings of symbols from V
 - R a finite set of substitution rules $R \subset V^* \times V^*$
 - S start symbol (initial shape) $S \in V_n$
- ⊙ relational structure - provides structural description of patterns
- ⊙ $\mathcal{L}(G)$ language - provides structural description of patterns, formal language set of words described by a grammar
language $\mathcal{L}(G)$ - finite $\times$ infinite a set of words,
- ⊖ learning - grammar inference



Syntactic Notions

- ⊙ grammar $G = \langle V_t, V_n, R, S \rangle$
 - G grammar - specifies the rules governing the composition of primitives into patterns
 - V_t a finite set of terminal symbols (alphabet)
 - V_n a finite set of nonterminal symbols $V_t \cap V_n = \emptyset$
 - $V = V_n \cup V_t$
 - V^* a set of all finite-length strings of symbols from V
 - R a finite set of substitution rules $R \subset V^* \times V^*$
 - S start symbol (initial shape) $S \in V_n$
- ⊙ relational structure - provides structural description of patterns
- ⊙ $\mathcal{L}(G)$ language - provides structural description of patterns, formal language set of words described by a grammar
language $\mathcal{L}(G)$ - finite $\times$ infinite a set of words,
- ⊖ learning - grammar inference



Syntactic Notions

- ⊙ grammar $G = \langle V_t, V_n, R, S \rangle$
 - G grammar - specifies the rules governing the composition of primitives into patterns
 - V_t a finite set of terminal symbols (alphabet)
 - V_n a finite set of nonterminal symbols $V_t \cap V_n = \emptyset$
 - $V = V_n \cup V_t$
 - V^* a set of all finite-length strings of symbols from V
 - R a finite set of substitution rules $R \subset V^* \times V^*$
 - S start symbol (initial shape) $S \in V_n$
- ⊙ relational structure - provides structural description of patterns
- ⊙ $\mathcal{L}(G)$ language - provides structural description of patterns, formal language set of words described by a grammar
language $\mathcal{L}(G)$ - finite $\times$ infinite a set of words,
- ⊖ learning - grammar inference



Grammar Example

The dog barks furiously.

rewriting rules:

$\langle \text{sentence} \rangle \rightarrow \langle \text{noun phrase} \rangle \langle \text{verb phrase} \rangle$
 $\langle \text{noun phrase} \rangle \rightarrow \langle \text{article} \rangle \langle \text{noun} \rangle$
 $\langle \text{verb phrase} \rangle \rightarrow \langle \text{verb} \rangle \langle \text{adverb} \rangle$
 $\langle \text{article} \rangle \rightarrow \text{the}$
 $\langle \text{noun} \rangle \rightarrow \text{dog}$
 $\langle \text{verb} \rangle \rightarrow \text{barks}$
 $\langle \text{adverb} \rangle \rightarrow \text{furiously}$

$V_n = \{ \langle \text{sentence} \rangle, \langle \text{noun phrase} \rangle, \langle \text{verb phrase} \rangle, \langle \text{article} \rangle, \langle \text{noun} \rangle, \langle \text{verb} \rangle, \langle \text{adverb} \rangle \}$
 $V_t = \{ \text{the, dog, barks, furiously} \}$
 $S = \langle \text{sentence} \rangle$



Grammar Example

The dog barks furiously.

rewriting rules:

$\langle \text{sentence} \rangle \rightarrow \langle \text{noun phrase} \rangle \langle \text{verb phrase} \rangle$
 $\langle \text{noun phrase} \rangle \rightarrow \langle \text{article} \rangle \langle \text{noun} \rangle$
 $\langle \text{verb phrase} \rangle \rightarrow \langle \text{verb} \rangle \langle \text{adverb} \rangle$
 $\langle \text{article} \rangle \rightarrow \text{the}$
 $\langle \text{noun} \rangle \rightarrow \text{dog}$
 $\langle \text{verb} \rangle \rightarrow \text{barks}$
 $\langle \text{adverb} \rangle \rightarrow \text{furiously}$

$V_n = \{ \langle \text{sentence} \rangle, \langle \text{noun phrase} \rangle, \langle \text{verb phrase} \rangle, \langle \text{article} \rangle, \langle \text{noun} \rangle, \langle \text{verb} \rangle, \langle \text{adverb} \rangle \}$

$V_t = \{ \text{the, dog, barks, furiously} \}$

$S = \langle \text{sentence} \rangle$



Grammar Example

The dog barks furiously.

rewriting rules:

$\langle \text{sentence} \rangle \rightarrow \langle \text{noun phrase} \rangle \langle \text{verb phrase} \rangle$
 $\langle \text{noun phrase} \rangle \rightarrow \langle \text{article} \rangle \langle \text{noun} \rangle$
 $\langle \text{verb phrase} \rangle \rightarrow \langle \text{verb} \rangle \langle \text{adverb} \rangle$
 $\langle \text{article} \rangle \rightarrow \text{the}$
 $\langle \text{noun} \rangle \rightarrow \text{dog}$
 $\langle \text{verb} \rangle \rightarrow \text{barks}$
 $\langle \text{adverb} \rangle \rightarrow \text{furiously}$

$V_n = \{ \langle \text{sentence} \rangle, \langle \text{noun phrase} \rangle, \langle \text{verb phrase} \rangle, \langle \text{article} \rangle, \langle \text{noun} \rangle, \langle \text{verb} \rangle, \langle \text{adverb} \rangle \}$

$V_t = \{ \text{the, dog, barks, furiously} \}$

$S = \langle \text{sentence} \rangle$



Syntactic PR

- 1 define primitives and their possible relations
- 2 construct a description grammar (hand analysis / **grammar inference**)
- 3 recognition - for each object extract primitives, recognize primitives classes, describe their mutual relations, construct a description word representing an object
- 4 based on the description word syntactic analysis classify an object into a class for which its grammar can generate the description word



Syntactic PR 2

grammar is a 4-tuple $G = (V_n, V_t, R, S)$

- V_n is a finite set of nonterminal symbols
- V_t is a set of terminal symbols (primitive classes)
- R is a finite subset of

$$V^* V_n V^* \times V^*$$

V^* denotes the set of all strings over $(V_n \cup V_t)$. An element of $(\alpha, \beta) \in R$ ($\alpha \rightarrow \beta$) is called a production

- S is a start symbol from V_n

Language $\mathcal{L}(G)$ generated by a grammar G is the set of sentences generated by G , i.e.

$$\mathcal{L}(G) = \{ \omega \mid \omega \in V_t^* \wedge S \Rightarrow^* \omega \}$$

where relation $\Rightarrow$ on V^* is defined: if $\alpha\beta\gamma \in V^*$ and $\beta \rightarrow \delta$ is a production (substitution) rule in R , then $\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$ and

$\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$.



Syntactic PR 2

grammar is a 4-tuple $G = (V_n, V_t, R, S)$

- V_n is a finite set of nonterminal symbols
- V_t is a set of terminal symbols (primitive classes)
- R is a finite subset of

$$V^* V_n V^* \times V^*$$

V^* denotes the set of all strings over $(V_n \cup V_t)$, An element of $(\alpha, \beta) \in R$ ($\alpha \rightarrow \beta$) is called a production

- S is a start symbol from V_n

Language $\mathcal{L}(G)$ generated by a grammar G is the set of sentences generated by G , i.e.

$$\mathcal{L}(G) = \{\omega \mid \omega \in V_t^* \wedge S \xRightarrow{*} \omega\}$$

where relation $\Rightarrow$ on V^* is defined: if $\alpha\beta\gamma \in V^*$ and $\beta \rightarrow \delta$ is a production (substitution) rule in R , then $\alpha\beta\gamma \Rightarrow \alpha\delta\gamma$ and

$\xRightarrow{*}$ denotes the reflexive and transitive closure of $\Rightarrow$



Relations

relation - an association between sets of information

A n-ary relation $\tilde{M}$ over the sets $M_1, \dots, M_n$ is a subset of their Cartesian product $\tilde{M} \subseteq M_1 \times \dots \times M_n$

unary ($n = 1$) relation $\tilde{M} \subset M$

Example M :

triangle	$\triangle$	square	$\square$
circ	$\bigcirc$	product	Π

binary ($n = 2$) relation $\smile \quad \tilde{M} \subset M \times M$

$\smile$ is specified by a subset M defined on the cartesian product
ordered pair of objects $\langle x, y \rangle \in \smile$



Relations

relation - an association between sets of information

A n-ary relation $\tilde{M}$ over the sets $M_1, \dots, M_n$ is a subset of their Cartesian product $\tilde{M} \subseteq M_1 \times \dots \times M_n$

unary ($n = 1$) relation $\tilde{M} \subset M$

Example M :

triangle	$\triangle$	square	$\square$
circ	$\bigcirc$	product	Π

binary ($n = 2$) relation $\smile \quad \tilde{M} \subset M \times M$

$\smile$ is specified by a subset M defined on the cartesian product
ordered pair of objects $\langle x, y \rangle \in \smile$



Relations 2

Properties of Binary Relations $x, y, z \in M$

- reflexive $x \smile x$
- symmetric $x \smile y \Rightarrow y \smile x$
- transitive $x \smile y \wedge y \smile z \rightarrow x \smile z$
- antisymmetric $x \smile y \wedge y \smile x \rightarrow x = y$
- irreflexive if $\langle x, x \rangle \notin \smile$, e.g., $<$

Examples:

- $\equiv$ identity $x \equiv y$
- $\sim$ equivalence - if is reflexive, symmetric, transitive
- $\approx$ compatibility - if is reflexive, symmetric, e.g.,
 $M = \{ball, bed, doll\}$
 $\smile = \{\langle x, y \rangle : x, y \in M \wedge$
 $x \smile y \text{ if both share some common letter}\}$



Relations 2

Properties of Binary Relations $x, y, z \in M$

- reflexive $x \sim x$
- symmetric $x \sim y \Rightarrow y \sim x$
- transitive $x \sim y \wedge y \sim z \rightarrow x \sim z$
- antisymmetric $x \sim y \wedge y \sim x \rightarrow x = y$
- irreflexive if $\langle x, x \rangle \notin \sim$, e.g., $<$

Examples:

- $\equiv$ identity $x \equiv y$
- $\sim$ equivalence - if is reflexive, symmetric, transitive
- $\approx$ compatibility - if is reflexive, symmetric, e.g.,
- $<$ $< = \{ \langle x, y \rangle : \langle x, y \rangle \in R \times R \wedge x < y \}$
- $- \bullet -$ $- \bullet - = \{ \langle x, y \rangle : x \text{ touches } y \}$
- $\rightarrow$ $\rightarrow = \{ \langle x, y \rangle : x \text{ is inside } y \}$



Relations 2

Properties of Binary Relations $x, y, z \in M$

- reflexive $x \sim x$
- symmetric $x \sim y \Rightarrow y \sim x$
- transitive $x \sim y \wedge y \sim z \rightarrow x \sim z$
- antisymmetric $x \sim y \wedge y \sim x \rightarrow x = y$
- irreflexive if $\langle x, x \rangle \notin \sim$, e.g., $<$

n -ary Relation $\sim \quad \tilde{M} \subseteq M_1 \times M_2 \times \dots \times M_n$



Relational Structure

relational structure $\langle M, F \rangle$
 M set of primitives
 $N = \{n_1, \dots, n_m\}$ set of relation names
 $F(n_i) = \sim$ function assigning a n -ary relation on $\tilde{M}^n \forall n_i$
 every n -ary relation can be specified as a set of binary relations

Example:

- $\sim$ ordering (relational structure)
 - 1 transitive
 - 2 irreflexive
 - 3 $\forall x, y, x \neq y : \langle x, y \rangle \in \sim \vee \langle y, x \rangle \in \sim$
 $(\langle x, y \rangle \in \sim \quad x \text{ is leftwards from } y)$
- language theory studies these sequences

- every word from selected alphabet is relational structure
- every relational structure can be transformed to a word from a alphabet



Relational Structure

relational structure $\langle M, F \rangle$
 M set of primitives
 $N = \{n_1, \dots, n_m\}$ set of relation names
 $F(n_i) = \sim$ function assigning a n -ary relation on $\tilde{M}^n \forall n_i$
 every n -ary relation can be specified as a set of binary relations

Example:

- $\sim$ ordering (relational structure)
 - 1 transitive
 - 2 irreflexive
 - 3 $\forall x, y, x \neq y : \langle x, y \rangle \in \sim \vee \langle y, x \rangle \in \sim$
 $(\langle x, y \rangle \in \sim \quad x \text{ is leftwards from } y)$
- language theory studies these sequences

- every word from selected alphabet is relational structure
- every relational structure can be transformed to a word from a alphabet



Formal Language Description

V a finite, non empty set - alphabet
 word X a finite sequence of V elements
 $\emptyset$ empty word
 $|X|$ word length (# of elements)
 $X \models Y$ word Y is derived from the word X
 V^* a set of all words above V
 $\mathcal{L} \subset V^*$ language

$$\mathcal{L}(G) = \{X \in V_t^* : S \models X\}$$

$G_i \equiv G_j$ equivalent grammars if $\mathcal{L}(G_i) = \mathcal{L}(G_j)$



Example - Formal Language Description

$$V = \{a, b\}$$

finite

$$\mathcal{L}_1 = \{a, aa, aba\}$$

$$\mathcal{L}_2 = \{\emptyset\}$$

infinite

$$\mathcal{L}_3 = \{\emptyset, ab, aabb, \dots, a^n b^n, \dots\}$$

$$\mathcal{L}_4 = \{X \in V^* \mid |X| \geq 2, \text{ word starts and ends with identical symbol } \}$$

- finite languages - possible specification through enumeration
 - $\oplus$ syntactic analysis comparison with all $\mathcal{L}$ words
- infinite languages, e.g., set description
 - $\mathcal{L}_3 = \{\emptyset, a^n b^n : n \geq 1\}$



Example - Formal Language Description

$$V = \{a, b\}$$

finite

$$\mathcal{L}_1 = \{a, aa, aba\}$$

$$\mathcal{L}_2 = \{\emptyset\}$$

infinite

$$\mathcal{L}_3 = \{\emptyset, ab, aabb, \dots, a^n b^n, \dots\}$$

$$\mathcal{L}_4 = \{X \in V^* \mid |X| \geq 2, \text{ word starts and ends with identical symbol } \}$$

- finite languages - possible specification through enumeration
- infinite languages, e.g., set description
 - $\mathcal{L}_3 = \{\emptyset, a^n b^n : n \geq 1\}$

Example $\mathcal{L}_3$:

$$V_n = \{\alpha\}, \quad V_t = \{a, b\}, \quad S = \{\alpha\}, \quad R : \begin{cases} \alpha \rightarrow \emptyset & \text{rule}_1 \\ \alpha \rightarrow a \alpha b & \text{rule}_2 \end{cases}$$



Texture Grammars

graph grammars - texture primitives represented by nodes, two nodes are connected by an arc if no other primitive is in some specified neighbourhood of both primitives $\Rightarrow$ undirected planar graph,

classification = graph recognition



Grammar Types - Noam Chomsky

0 Unrestricted (General) $G = \langle V_t, V_n, R, S \rangle$

- R no restrictions
- $\ominus$ analysis undecidable, difficult

1 Context-Sensitive

2 Context-Free



Grammar Types - Noam Chomsky

0 Unrestricted (General) $G = \langle V_t, V_n, R, S \rangle$

- R no restrictions
- $\ominus$ analysis undecidable, difficult

1 Context-Sensitive

- $R : \{X_1 \alpha X_2 \rightarrow X_1 Y X_2\}$
- $X_1, X_2, Y \in V^* \quad Y \neq \emptyset, \quad \alpha \in V_n$
- $l(\alpha) \leq l(Y)$

Example: $\mathcal{L} = \{0^n 1^n 0^n \mid n = 1, 2, \dots\}$

2 Context-Free

- $R : \{\alpha \rightarrow Y\} \quad Y \in V^* \setminus \emptyset \quad \alpha \in V_n$
 $l(\alpha) = 1$



Grammar Types - Noam Chomsky

0 Unrestricted (General) $G = \langle V_t, V_n, R, S \rangle$

- R no restrictions
- $\ominus$ analysis undecidable, difficult

1 Context-Sensitive

- $R : \{X_1 \alpha X_2 \rightarrow X_1 Y X_2\}$
- $X_1, X_2, Y \in V^* \quad Y \neq \emptyset, \quad \alpha \in V_n$
- $l(\alpha) \leq l(Y)$

Example: $\mathcal{L} = \{0^n 1^n 0^n \mid n = 1, 2, \dots\}$

2 Context-Free

- $R : \{\alpha \rightarrow Y\} \quad Y \in V^* \setminus \emptyset \quad \alpha \in V_n$
 $l(\alpha) = 1$



Grammar Types - Noam Chomsky

0 Unrestricted (General) $G = \langle V_t, V_n, R, S \rangle$

1 Context-Sensitive

2 Context-Free

Example: $G = \langle V_t, V_n, R, S \rangle$

$V_n = \{S, \alpha, \beta\} \quad V_t = \{a, b\}$

- $$R : \begin{cases} S \rightarrow a\beta & r_1 \\ S \rightarrow b\alpha & r_2 \\ \alpha \rightarrow aS & r_3 \\ \alpha \rightarrow b\alpha\alpha & r_4 \\ \alpha \rightarrow a & r_5 \\ \beta \rightarrow bS & r_6 \\ \beta \rightarrow a\beta\beta & r_7 \\ \beta \rightarrow b & r_8 \end{cases}$$

$S \xrightarrow{(2)} b\alpha \xrightarrow{(4)} bb\alpha\alpha \xrightarrow{(5)} bbaa \xrightarrow{(5)} bbaa, \text{ etc. } \mathcal{L} \dots \#a = \#b$



Grammar Types - Noam Chomsky

0 Unrestricted (General) $G = \langle V_t, V_n, R, S \rangle$

1 Context-Sensitive

2 Context-Free

3 Regular (Finite-State)

- $R : \begin{cases} \alpha \rightarrow a\beta & r_1 \\ \alpha \rightarrow b & r_2 \end{cases}$
 $\alpha, \beta \in V_n, \quad a, b \in V_t$
 - note α, β, a, b are single symbols

Example : $\mathcal{L} = \{0^m 1^n : m, n = 1, 2, \dots\}$



Grammar Types - Noam Chomsky

0 Unrestricted (General) $G = \langle V_t, V_n, R, S \rangle$

1 Context-Sensitive

2 Context-Free

3 Regular (Finite-State)

- $R : \begin{cases} \alpha \rightarrow a\beta & r_1 \\ \alpha \rightarrow b & r_2 \end{cases}$
 $\alpha, \beta \in V_n, \quad a, b \in V_t$
 - note α, β, a, b are single symbols

Example : $V_n = \{S, \alpha\} \quad V_t = \{a, b\}$

$R : \begin{cases} S \rightarrow a\alpha & r_1 \\ \alpha \rightarrow a\alpha & r_2 \\ \alpha \rightarrow b & r_3 \end{cases}$

$S \xrightarrow{(1)} a\alpha \xrightarrow{(2)} a a\alpha \xrightarrow{(2)} a a a\alpha \xrightarrow{(3)} a a a b$



Grammars

substitution rules usage

- deterministic
- stochastic $\langle V_t, V_n, R, p, S \rangle \quad \sum_i p_i(\alpha) = 1$
- fuzzy $\tilde{p} : R \rightarrow \langle 0; 1 \rangle \quad \sum_i \tilde{p}_i(\alpha) \neq 1$



Grammars

substitution rules usage

- deterministic
- stochastic $\langle V_t, V_n, R, p, S \rangle \quad \sum_i p_i(\alpha) = 1$

Example: $V_n = \{S, \alpha, \beta\} \quad V_t = \{a, b\}$

$R : \begin{cases} \alpha \xrightarrow{p(\cdot)} a S & r_1 \\ \alpha \xrightarrow{p(\cdot)} b \alpha \alpha & r_2 \\ \alpha \xrightarrow{p(\cdot)} a & r_3 \\ \beta \xrightarrow{p(\cdot)} b S & r_4 \\ \beta \xrightarrow{p(\cdot)} a \beta \beta & r_5 \\ \beta \xrightarrow{p(\cdot)} b & r_6 \end{cases}$

- consistent $\sum_{X \in \mathcal{L}(G)} P_G(X) = 1$
 $P_G(X) = \sum_{i=1}^k p_i \quad k \text{ no. of alternatives}$
 - General, Context-Sensitive, Context-Free, Regular



substitution rules usage

- deterministic
- stochastic $\langle V_t, V_n, R, p, S \rangle$ $\sum_i p_i(\alpha) = 1$
- fuzzy $\tilde{p} : R \rightarrow \langle 0; 1 \rangle$ $\sum_i \tilde{p}_i(\alpha) \neq 1$



substitution rules usage

- deterministic
- stochastic $\langle V_t, V_n, R, p, S \rangle$ $\sum_i p_i(\alpha) = 1$
- fuzzy $\tilde{p} : R \rightarrow \langle 0; 1 \rangle$ $\sum_i \tilde{p}_i(\alpha) \neq 1$

relations

- chain
- n-ary

substitution rules types

- General
- Context-Sensitive
- Context-Free
- Regular



Stochastic Finite-State Grammar - Example

$$G = \langle V_t, V_n, R, S \rangle \quad V_n = \{\alpha_1, \dots, \alpha_k\} \quad V_t = \{a_1, \dots, a_m\} \quad S = \alpha_1$$

$$R: \begin{array}{ll} \alpha_i \xrightarrow{p_{i,1,1}} a_1 \alpha_1 & \alpha_i \xrightarrow{p_{i,1}} a_1 \\ \alpha_i \xrightarrow{p_{i,1,2}} a_2 \alpha_1 & \vdots \\ \vdots & \alpha_i \xrightarrow{p_{i,j}} a_j \\ \alpha_i \xrightarrow{p_{i,k,l}} a_l \alpha_k & \vdots \\ \vdots & \alpha_i \xrightarrow{p_{i,m}} a_m \\ \alpha_i \xrightarrow{p_{i,k,m}} a_m \alpha_k & \end{array}$$

$$\sum_{j=1}^k \sum_{l=1}^m p_{i,j,l} + \sum_{l=1}^m p_{i,l} = 1$$



Stochastic Finite-State Grammar - Example

$$V_n = \{S, \alpha, \beta\} \quad V_t = \{0, 1\}$$

$$R: \begin{array}{ll} S \xrightarrow{1} 1\alpha & \beta \xrightarrow{0.3} 0 \\ \alpha \xrightarrow{0.8} 0\beta & \beta \xrightarrow{0.7} 1S \\ \alpha \xrightarrow{0.2} 1 & \end{array}$$

$$S \Rightarrow 1\alpha \Rightarrow 10\beta \Rightarrow 100 \quad p(100) = 1 \times 0.8 \times 0.3 = 0.24$$



Stochastic Finite-State Grammar - Example

$\mathcal{L}$:

word x	$p(x)$
11	0.2
100	0.24
$(101)^n 11$	$0.2 \times (0.56)^n$
$(101)^n 100$	$0.24 \times (0.56)^n$

$$\sum_{x \in \mathcal{L}} p(x) = 0.2 + 0.24 + \sum_{n=1}^{\infty} (0.2 + 0.24)(0.56)^n = 1$$



Non-Deterministic Grammars 2

- Stochastic (Probabilistic) Grammar (p-grammar) (G, F)
 - substitution rules are accompanied with corresponding probabilities, F is a function assigning probabilities to the production in P , usually unconditional, p-grammar can be of 0-3 type,
- Web Grammar (1D / 2D)

$$\alpha \rightarrow \beta, E$$

generated sentences are directed graphs with symbols at their vertices ("webs"), α, β are webs, E is an embedding of β , if $\text{card}(V_T) = 1$ then all points have the same label and the grammar is called the graph grammar,
 context-sensitive web - if $\exists a \in \alpha$ such that $\alpha - \{a\}$ is a subweb of β and all edges between points of the host web and points of $\alpha - \{a\}$ are in E
 context-free web - if $\text{card}(\alpha) = 1$



Non-Deterministic Grammars 2

- Stochastic (Probabilistic) Grammar (p-grammar) (G, F)
 - substitution rules are accompanied with corresponding probabilities, F is a function assigning probabilities to the production in P , usually unconditional, p-grammar can be of 0-3 type,
- Web Grammar (1D / 2D)

$$\alpha \rightarrow \beta, E$$

generated sentences are directed graphs with symbols at their vertices ("webs"), α, β are webs, E is an embedding of β , if $\text{card}(V_T) = 1$ then all points have the same label and the grammar is called the graph grammar,
 context-sensitive web - if $\exists a \in \alpha$ such that $\alpha - \{a\}$ is a subweb of β and all edges between points of the host web and points of $\alpha - \{a\}$ are in E
 context-free web - if $\text{card}(\alpha) = 1$



Non-Deterministic Grammars 3

finite-state (regular, G_3) languages - weaker than context-free (G_2)
 or context-sensitive (G_1) languages

simple primitives require more complex grammars and vice versa



Finite-State Languages and Finite-State Automata

Deterministic Finite-State Automaton

$A = \langle \Sigma, Q, \delta \rangle$

Deterministic Finite-State Automaton (enhanced)

$A = \langle \Sigma, Q, \delta, S, F \rangle$

Σ a finite set of input symbols (alphabet)

Q a finite set of states

$\delta: Q \times \Sigma \rightarrow Q$ mapping of $Q \times \Sigma$ to a next state

$S \in Q$ the initial state

$F \subseteq Q$ the set of final states $F = \{f\}$

Nondeterministic Finite-State Automaton (several possible outputs)

grammar G is equivalent with the automaton A if

$$\mathcal{L}(G) = \mathcal{L}(A)$$



Finite-State Languages and Finite-State Automata 2

⊙ $\forall$ (nondeterministic) regular $G_3 \exists$ the non deterministic finite-state automaton A

⊙ $\forall$ nondeterministic finite-state automata $A \exists$ (non deterministic) regular G_3

$G_3 \rightarrow A$

$G_3 = \langle V_t, V_n, R, S \rangle$

$\Sigma = V_t \quad Q = V_n \cup \{f\} \quad f \notin V_n \quad S_A = S_G \quad F = \{f\}$

if $S \rightarrow \emptyset \in R$ then $F = \{S, f\}$ else $F = \{f\}$

$\delta(\alpha, \beta \in V_n, x \in V_t)$

if $(\alpha \rightarrow x\beta) \in R$ then $\beta \in \delta(\alpha, x)$

if $(\alpha \rightarrow x) \in R$ then $f \in \delta(\alpha, x)$

if $(\delta \rightarrow \emptyset) \in R$ then $S_A = f$

similarly

a finite-state nondeterministic $A \rightarrow G_3$ (regular grammar)

context-free $G_2 \rightarrow$ pushdown automaton A



Syntactic Analysis, Syntactic Classifier

syntactic recognition

- matching - matching between the classified pattern and prototype patterns
 - ⊖ no structural information recovered
 - ⊕ very simple
- syntactic analysis
 - ⊕ complete pattern description
 - ⊖ complex

a finite language - SC can search all the words of the language

a regular grammar - SC a finite non-deterministic automaton

a context-free grammar - SC can use a stack automata



Syntactic Analysis, Syntactic Classifier 2

syntactic analysis (parsing)

- top-down approach
 - start symbol $S \rightarrow$ substitution rules until analysed word is generated
 - if a partial substitution creates a dead end $\rightarrow$ backtracking to the nearest higher tree level and picking another applicable rule or all possible combinations of substitution rules are applied in parallel
 - if the whole process is unsuccessful the grammar does not generate the pattern (does not belong to this class)
- ⊖ less efficient, too many incorrect branches generated
- tree pruning - prior knowledge consistency test to decrease number of generated branches



Syntactic Analysis - Context-Free Top-Down

assign X into class ω_i if $X \in \mathcal{L}(G_i)$

- 1 generate words X_i from the start symbol using G_i
- 2 if $\exists X_i$ that starts with identical terminal symbols with X cut common parts and further process remaining chains
- 3 if X_i starts with a non terminal symbols, use substitution rules
- 4 if remaining subchains X_i, X are empty accept G_i otherwise go to 1.



Example (SA Context-Free Top-Down)

$G_2: V_n = \{S, \alpha\} \quad V_t = \{a, b, c, +, \times\} \quad i \in \{a, b, c\}$
 $X = a + b \times c$ A sentence from G_2 ?

$$R: \begin{cases} S \rightarrow \alpha & r_1 \\ S \rightarrow \alpha + S & r_2 \\ \alpha \rightarrow i & r_3 \\ \alpha \rightarrow i \times \alpha & r_4 \end{cases}$$



Example (SA Context-Free Top-Down)

$G_2: V_n = \{S, \alpha\} \quad V_t = \{a, b, c, +, \times\} \quad i \in \{a, b, c\}$
 $X = a + b \times c$ A sentence from G_2 ?

$$R: \begin{cases} S \rightarrow \alpha & r_1 \\ S \rightarrow \alpha + S & r_2 \\ \alpha \rightarrow i & r_3 \\ \alpha \rightarrow i \times \alpha & r_4 \end{cases} \quad S \vdash \begin{cases} \alpha \\ \alpha + S \\ \alpha \vdash i \\ \alpha + S \vdash i + S & r_1 \quad (r_2 + r_3) \\ \alpha \vdash i \times \alpha \\ \alpha + S \vdash i \times \alpha + S \end{cases}$$



Example (SA Context-Free Top-Down)

$G_2: V_n = \{S, \alpha\} \quad V_t = \{a, b, c, +, \times\} \quad i \in \{a, b, c\}$
 $X = a + b \times c$ A sentence from G_2 ?

$$R: \begin{cases} S \rightarrow \alpha & r_1 \\ S \rightarrow \alpha + S & r_2 \\ \alpha \rightarrow i & r_3 \\ \alpha \rightarrow i \times \alpha & r_4 \end{cases} \quad S \vdash \begin{cases} \alpha \\ \alpha + S \\ \alpha \vdash i \\ \alpha + S \vdash i + S & r_1 \quad (r_2 + r_3) \\ \alpha \vdash i \times \alpha \\ \alpha + S \vdash i \times \alpha + S \end{cases}$$

$$b \times c: \quad S \vdash \alpha \vdash i \times \alpha \quad r_{II} \quad (r_1 + r_4)$$



Example (SA Context-Free Top-Down)

$G_2: V_n = \{S, \alpha\} \quad V_t = \{a, b, c, +, \times\} \quad i \in \{a, b, c\}$
 $X = a + b \times c$ A sentence from G_2 ?

$$R: \begin{cases} S \rightarrow \alpha & r_1 \\ S \rightarrow \alpha + S & r_2 \\ \alpha \rightarrow i & r_3 \\ \alpha \rightarrow i \times \alpha & r_4 \end{cases} \quad S \vdash \begin{cases} \alpha \\ \alpha + S \\ \alpha \vdash i \\ \alpha + S \vdash i + S & r_1 \quad (r_2 + r_3) \\ \alpha \vdash i \times \alpha \\ \alpha + S \vdash i \times \alpha + S \end{cases}$$

$$b \times c: \quad S \vdash \alpha \vdash i \times \alpha \quad r_{II} \quad (r_1 + r_4)$$

$$c: \quad \alpha \vdash i \quad r_{III} \quad \Rightarrow X \in G_2$$



Syntactic Analysis, Syntactic Classifier 3

- bottom-up approach
 starts with the analysed word, which is reduced applying the reverse substitution to the start symbol
 if the whole process is unsuccessful the grammar does not generate the pattern (does not belong to this class)



Syntactic Analysis, Syntactic Classifier 3

- bottom-up approach
 starts with the analysed word, which is reduced applying the reverse substitution to the start symbol
 if the whole process is unsuccessful the grammar does not generate the pattern (does not belong to this class)

Example:

$$X = a + b \times c \quad R: \begin{cases} S \leftarrow \alpha & r_1 \\ S \leftarrow \alpha + S & r_2 \\ \alpha \leftarrow i & r_3 \\ \alpha \leftarrow i \times \alpha & r_4 \end{cases}$$

$$a + b \times c \xrightarrow{3} \alpha + b \times c \xrightarrow{3} \alpha + b \times \alpha \xrightarrow{4} \alpha + \alpha \xrightarrow{1} \alpha + S \xrightarrow{2} S$$



Syntactic Analysis, Syntactic Classifier 3

- bottom-up approach
 starts with the analysed word, which is reduced applying the reverse substitution to the start symbol
 if the whole process is unsuccessful the grammar does not generate the pattern (does not belong to this class)
- relational structures matching
 relational structure of the analysed object is matched with RS's of grammar examples

Regular (stochastic)

⊕ easy using equivalent (stochastic) automaton



Error-Correcting Parsing

- 1 a grammar is expanded to cover all possible errors into its production (distorted patterns), for string grammars the substitution, deletion, and insertion errors are considered
- 2 the resulting covering grammar is used for parsing



Grammar Inference

grammar inference - grammar construction from training samples

given: training sets $T_i \quad i = 1, \dots, K$ contain words of $\mathcal{L}(G_i)$
 $T = \bigcup_{i=1}^K T_i$

task: $G_i (R)$ generating words T_i
 V_t known (sometimes counter-examples $V_t^* \setminus \mathcal{L}$)

stochastic $G_i \quad T_i = \{[X_1, P_1], \dots, [X_n, P_n]\}$
 G_i generates not only T_i but also similar words



Grammar Inference 2

- enumeration** detects a grammar from the finite set of possible grammars that can generate the whole training set
- ⊖ difficult to construct this set of grammars
 - ⊖ difficult to select one specific grammar
- induction** analysis of training set words and derivation of substitution rules



Grammar Inference 3

- ⊖ no general inference method, existing methods are more or less heuristic,
- ⊖ inference methods $G_i \rightarrow T_i$ only for regular grammars (3, no class generalization) unsuitable for PR
- ⊖ inferred grammars usually generates much larger language than the minimum language $\rightarrow$ too complex syntactic analysis,
- ⊖ increased descriptive power of language is paid with increased syntax analysis complexity,
- ⊖ training set is always finite and not sufficient to define an infinite language unambiguously,
- ⊖ a finite training set can be represented by an infinite set of languages.



Crespi-Reghizzi Method (Context-Free, Enumeration)

assumption: $T_i = \{[X_1; r_1, r_2, \dots, r_k], \dots, \}$
 (a substitution rule sequence known for every training word)

e.g., $X_1 = S \vdash a\alpha b \vdash aa\beta b \vdash aacb$ (2, context-free)
 $X_1 = [a[a[c]]b]$

Example:

$$\begin{aligned} X_1 &= [b[[[a]b]b]b] \\ X_2 &= [b[b[b[a]]b]b] \\ X_3 &= [b[b[[[a]b]b]b]b] \\ X_4 &= [b[b[a]]b] \end{aligned}$$



Crespi-Reghizzi Method (Context-Free, Enumeration)

assumption: $T_i = \{[X_1; r_1, r_2, \dots, r_k], \dots, \}$
 (a substitution rule sequence known for every training word)

e.g., $X_1 = S \vdash a\alpha b \vdash aa\beta b \vdash aacb$ (2, context-free)
 $X_1 = [a[a[c]]b]$

Example:

$$\begin{aligned} X_1 &= [b[[[a]b]b]b] \\ X_2 &= [b[b[b[a]]b]b] \\ X_3 &= [b[b[[[a]b]b]b]b] \\ X_4 &= [b[b[a]]b] \end{aligned}$$



Crespi-Reghizzi Method (Context-Free, Enumeration)

assumption: $T_i = \{[X_1; r_1, r_2, \dots, r_k], \dots, \}$
 (a substitution rule sequence known for every training word)

e.g., $X_1 = S \vdash a\alpha b \vdash aa\beta b \vdash aacb$ (2, context-free)
 $X_1 = [a[a[c]]b]$

Example:

$$\begin{aligned} X_1 &= [b[[[a]b]b]b] \\ X_2 &= [b[b[b[a]]b]b] \\ X_3 &= [b[b[[[a]b]b]b]b] \\ X_4 &= [b[b[a]]b] \end{aligned}$$



Crespi-Reghizzi Method - Example 2

- 1 G_1 generating $X_1 = [b[[[a]b]b]b]$
 - 1 find a subword Y being enclosed to the most brackets,
i.e., $Y = a$
 - 2 create the rule $\langle first_{symb}(Y) \alpha last_{symb}(Y) \rangle \rightarrow Y$,
i.e., $r_1 : \langle a \alpha a \rangle \rightarrow a$
 - 3 inverse substitution using (r_1)
 $X_1 = [b[\langle a \alpha a \rangle b]b]$

- 2 repeat steps in 1.
 $Y = \langle a \alpha a \rangle b$ $first_{symb}(Y) = a, last_{symb}(Y) = b$
 $r_2 : \langle a \alpha b \rangle \rightarrow \langle a \alpha a \rangle b$ $X_1 = [b[\langle a \alpha b \rangle b]b]$
 $r_3 : S \rightarrow \langle b \alpha b \rangle$
- 3 repeat steps 1., 2. for remaining grammars (X_2, X_3, X_4)



Crespi-Reghizzi Method - Example 2

1 G_1 generating $X_1 = [b[[[a]b]b]b]$

- 1 find a subword Y being enclosed to the most brackets,
i.e., $Y = a$
- 2 create the rule $\langle first_{symb}(Y) \alpha last_{symb}(Y) \rangle \rightarrow Y$,
i.e., $r_1 : \langle a \alpha a \rangle \rightarrow a$
- 3 inverse substitution using (r_1)

$$X_1 = [b[[\langle a \alpha a \rangle]b]b]$$

2 repeat steps in 1.

$$Y = \langle a \alpha a \rangle \quad first_{symb}(Y) = a, \quad last_{symb}(Y) = b$$

$$r_2 : \langle a \alpha b \rangle \rightarrow \langle a \alpha a \rangle b \quad X_1 = [b[\langle a \alpha b \rangle]b]$$

$$r_5 : S \rightarrow \langle b \alpha b \rangle$$

repeat steps 1., 2. for remaining grammars (X_2, X_3, X_4)



Crespi-Reghizzi Method - Example 2

1 G_1 generating $X_1 = [b[[[a]b]b]b]$

- 1 find a subword Y being enclosed to the most brackets,
i.e., $Y = a$
- 2 create the rule $\langle first_{symb}(Y) \alpha last_{symb}(Y) \rangle \rightarrow Y$,
i.e., $r_1 : \langle a \alpha a \rangle \rightarrow a$
- 3 inverse substitution using (r_1)

$$X_1 = [b[[\langle a \alpha a \rangle]b]b]$$

2 repeat steps in 1.

$$Y = \langle a \alpha a \rangle \quad first_{symb}(Y) = a, \quad last_{symb}(Y) = b$$

$$r_2 : \langle a \alpha b \rangle \rightarrow \langle a \alpha a \rangle b \quad X_1 = [b[\langle a \alpha b \rangle]b]$$

$$r_3 : \langle a \alpha b \rangle \rightarrow \langle a \alpha b \rangle b \quad X_1 = [b[\langle a \alpha b \rangle]b]$$

$$r_5 : S \rightarrow \langle b \alpha b \rangle$$

repeat steps 1., 2. for remaining grammars (X_2, X_3, X_4)



Crespi-Reghizzi Method - Example 2

1 G_1 generating $X_1 = [b[[[a]b]b]b]$

- 1 find a subword Y being enclosed to the most brackets,
i.e., $Y = a$
- 2 create the rule $\langle first_{symb}(Y) \alpha last_{symb}(Y) \rangle \rightarrow Y$,
i.e., $r_1 : \langle a \alpha a \rangle \rightarrow a$
- 3 inverse substitution using (r_1)

$$X_1 = [b[[\langle a \alpha a \rangle]b]b]$$

2 repeat steps in 1.

$$Y = \langle a \alpha a \rangle \quad first_{symb}(Y) = a, \quad last_{symb}(Y) = b$$

$$r_2 : \langle a \alpha b \rangle \rightarrow \langle a \alpha a \rangle b \quad X_1 = [b[\langle a \alpha b \rangle]b]$$

$$r_3 : \langle a \alpha b \rangle \rightarrow \langle a \alpha b \rangle b \quad X_1 = [b[\langle a \alpha b \rangle]b]$$

$$r_4 : \langle b \alpha b \rangle \rightarrow b \langle a \alpha b \rangle b \quad X_1 = \langle b \alpha b \rangle$$

$$r_5 : S \rightarrow \langle b \alpha b \rangle$$

3 repeat steps 1., 2. for remaining grammars (X_2, X_3, X_4)



Crespi-Reghizzi Method - Example 3

$$G_1 = \langle V_{t,1}, V_{n,1}, R_1, S \rangle$$

$$V_{t,1} = \{a, b\}, \quad V_{n,1} = \{S, \langle a \alpha a \rangle, \langle a \alpha b \rangle, \langle b \alpha b \rangle\}$$

$$R_1 : \begin{cases} \langle a \alpha a \rangle \rightarrow a & r_1 \\ \langle a \alpha b \rangle \rightarrow \langle a \alpha a \rangle b & r_2 \\ \langle a \alpha b \rangle \rightarrow \langle a \alpha b \rangle b & r_3 \\ \langle b \alpha b \rangle \rightarrow b \langle a \alpha b \rangle b & r_4 \\ S \rightarrow \langle b \alpha b \rangle & r_5 \end{cases}$$



Crespi-Reghizzi Method - Example 4

analogously G_2, G_3, G_4

$G = \langle V_t, V_n, R, S \rangle$ - combination of G_1, G_2, G_3, G_4

$V_t = \{a, b\},$

$V_n = \{S, \langle a \alpha a \rangle, \langle a \alpha b \rangle, \langle b \alpha b \rangle, \langle b \alpha a \rangle\}$

$$R : \begin{cases} \langle a \alpha a \rangle \rightarrow a & r_1 \\ \langle a \alpha b \rangle \rightarrow \langle a \alpha a \rangle b & r_2 \\ \langle a \alpha b \rangle \rightarrow \langle a \alpha b \rangle b & r_3 \\ \langle b \alpha b \rangle \rightarrow b \langle a \alpha b \rangle b & r_4 \\ S \rightarrow \langle b \alpha b \rangle & r_5 \\ \langle b \alpha b \rangle \rightarrow b \langle b \alpha b \rangle b & r_6 \\ \langle b \alpha b \rangle \rightarrow b \langle b \alpha a \rangle b & r_7 \\ \langle b \alpha a \rangle \rightarrow b \langle a \alpha a \rangle & r_8 \end{cases}$$

⊖ difficult for large T_i



Evans Method (Heuristic Induction)

1 select the most frequent T_i subwords

2 $\forall X_j \in T_i$ create $G_{i,j}$

3 $G_i = \bigcup_{j=1}^{card(T_i)} G_{i,j}$

reduction rules:

1 Eliminate all but one multiple occurrences of a rule.

2 Look for non terminal symbols which can be substituted by another non terminal symbol such that the grammar will contain multiple occurrences of some rules. Make the substitution and apply 1.

3 Look for such a couple $[\alpha, a]$ that adding the rule $\alpha \rightarrow a$ for some α, a the grammar will contain multiple occurrences of some rules. Apply 1.



Evans Method (Heuristic Induction)

1 select the most frequent T_i subwords

2 $\forall X_j \in T_i$ create $G_{i,j}$

3 $G_i = \bigcup_{j=1}^{card(T_i)} G_{i,j}$

reduction rules:

1 Eliminate all but one multiple occurrences of a rule.

2 Look for non terminal symbols which can be substituted by another non terminal symbol such that the grammar will contain multiple occurrences of some rules. Make the substitution and apply 1.

3 Look for such a couple $[\alpha, a]$ that adding the rule $\alpha \rightarrow a$ for some α, a the grammar will contain multiple occurrences of some rules. Apply 1.



Evans Method (Heuristic Induction)

1 select the most frequent T_i subwords

2 $\forall X_j \in T_i$ create $G_{i,j}$

3 $G_i = \bigcup_{j=1}^{card(T_i)} G_{i,j}$

reduction rules:

1 Eliminate all but one multiple occurrences of a rule.

2 Look for non terminal symbols which can be substituted by another non terminal symbol such that the grammar will contain multiple occurrences of some rules. Make the substitution and apply 1.

3 Look for such a couple $[\alpha, a]$ that adding the rule $\alpha \rightarrow a$ for some α, a the grammar will contain multiple occurrences of some rules. Apply 1.



Evans Method - Example

trainee set:

$X_1 = caaab$
 $X_2 = bbaab$
 $X_3 = caab$
 $X_4 = bbab$
 $X_5 = cab$
 $X_6 = bbb$
 $X_7 = cb$

the most frequent subwords ab, bb



Evans Method - Example 2

substitution rules:

r_1	r_2	r_3	r_4
$G_1: S \rightarrow c\alpha$	$\alpha \rightarrow a\beta$	$\beta \rightarrow a\gamma$	$\gamma \rightarrow ab$
$G_2: S \rightarrow bb\delta$	$\delta \rightarrow a\epsilon$	$\epsilon \rightarrow ab$	
$G_3: S \rightarrow c\varphi$	$\varphi \rightarrow a\psi$	$\psi \rightarrow ab$	
$G_4: S \rightarrow bb\chi$	$\chi \rightarrow ab$		
$G_5: S \rightarrow c\iota$	$\iota \rightarrow ab$		
$G_6: S \rightarrow bb\mu$	$\mu \rightarrow b$		
$G_7: S \rightarrow c\nu$	$\nu \rightarrow b$		



Evans Method - Example 2

substitution rules:

r_1	r_2	r_3	r_4
$G_1: S \rightarrow c\alpha$	$\alpha \rightarrow a\beta$	$\beta \rightarrow a\gamma$	$\gamma \rightarrow ab$
$G_2: S \rightarrow bb\delta$	$\delta \rightarrow a\epsilon$	$\epsilon \rightarrow ab$	
$G_3: S \rightarrow c\varphi$	$\varphi \rightarrow a\psi$	$\psi \rightarrow ab$	
$G_4: S \rightarrow bb\chi$	$\chi \rightarrow ab$		
$G_5: S \rightarrow c\iota$	$\iota \rightarrow ab$		
$G_6: S \rightarrow bb\mu$	$\mu \rightarrow b$		
$G_7: S \rightarrow c\nu$	$\nu \rightarrow b$		

reduction $(18 \rightarrow 13)$ $\gamma, \epsilon, \psi, \chi, \iota \rightarrow \gamma$ $\mu, \nu \rightarrow \mu$

$X_1: S \xrightarrow{(1)} c\alpha \xrightarrow{(2)} ca\beta \xrightarrow{(3)} caa\gamma \xrightarrow{(4)} caaab$



Evans Method - Example 2

substitution rules:

reduction $(18 \rightarrow 13)$

r_1	r_2	r_3	r_4
$G_1: S \rightarrow c\alpha$	$\alpha \rightarrow a\beta$	$\beta \rightarrow a\gamma$	$\gamma \rightarrow ab$
$G_2: S \rightarrow bb\delta$	$\delta \rightarrow a\epsilon$		
$G_3: S \rightarrow c\varphi$	$\varphi \rightarrow a\psi$		
$G_4: S \rightarrow bb\chi$			
$G_5: S \rightarrow c\iota$			
$G_6: S \rightarrow bb\mu$	$\mu \rightarrow b$		
$G_7: S \rightarrow c\nu$			



Evans Method - Example 3

G

1 reduction (18 → 13) $\gamma, \epsilon, \psi, \chi, \iota \rightarrow \gamma$ $\mu, \nu \rightarrow \mu$

$S \rightarrow c\alpha$	$S \rightarrow bb\delta$	$S \rightarrow bb\gamma$	$S \rightarrow c\gamma$
$S \rightarrow c\varphi$	$S \rightarrow c\mu$	$S \rightarrow bb\mu$	$\alpha \rightarrow a\beta$
$\beta \rightarrow a\gamma$	$\gamma \rightarrow ab$	$\delta \rightarrow a\gamma$	$\varphi \rightarrow a\gamma$
$\mu \rightarrow b$			

2 reduction (13 → 9) $\alpha, \mu, \varphi, \gamma \rightarrow \alpha$

3 reduction (9 → 5) $\delta, \beta \rightarrow \alpha$

4 selection $\alpha \rightarrow b$ according to 3. and replace b with α in the r_5 rule ($\rightarrow r_5 = r_3$)



Evans Method - Example 3

G

1 reduction (18 → 13) $\gamma, \epsilon, \psi, \chi, \iota \rightarrow \gamma$ $\mu, \nu \rightarrow \mu$

$S \rightarrow c\alpha$	$S \rightarrow bb\delta$	$S \rightarrow bb\gamma$	$S \rightarrow c\gamma$
$S \rightarrow c\varphi$	$S \rightarrow c\mu$	$S \rightarrow bb\mu$	$\alpha \rightarrow a\beta$
$\beta \rightarrow a\gamma$	$\gamma \rightarrow ab$	$\delta \rightarrow a\gamma$	$\varphi \rightarrow a\gamma$
$\mu \rightarrow b$			

2 reduction (13 → 9) $\alpha, \mu, \varphi, \gamma \rightarrow \alpha$

$S \rightarrow c\alpha$	$S \rightarrow bb\delta$	$S \rightarrow bb\gamma$	$S \rightarrow c\alpha$
$S \rightarrow c\alpha$	$S \rightarrow c\alpha$	$S \rightarrow bb\alpha$	$\alpha \rightarrow a\beta$
$\beta \rightarrow a\alpha$	$\alpha \rightarrow ab$	$\delta \rightarrow a\alpha$	$\alpha \rightarrow a\alpha$
$\alpha \rightarrow b$			

3 reduction (9 → 5) $\delta, \beta \rightarrow \alpha$

4 selection $\alpha \rightarrow b$ according to 3. and replace b with α in the r_5 rule ($\rightarrow r_5 = r_3$)



Evans Method - Example 3

G

1 reduction (18 → 13) $\gamma, \epsilon, \psi, \chi, \iota \rightarrow \gamma$ $\mu, \nu \rightarrow \mu$

$S \rightarrow c\alpha$	$S \rightarrow bb\delta$	$S \rightarrow bb\gamma$	$S \rightarrow c\gamma$
$S \rightarrow c\varphi$	$S \rightarrow c\mu$	$S \rightarrow bb\mu$	$\alpha \rightarrow a\beta$
$\beta \rightarrow a\gamma$	$\gamma \rightarrow ab$	$\delta \rightarrow a\gamma$	$\varphi \rightarrow a\gamma$
$\mu \rightarrow b$			

2 reduction (13 → 9) $\alpha, \mu, \varphi, \gamma \rightarrow \alpha$

$S \rightarrow c\alpha$	$S \rightarrow bb\delta$	$S \rightarrow bb\alpha$	$\alpha \rightarrow a\beta$
$\alpha \rightarrow b$	$\alpha \rightarrow ab$	$\alpha \rightarrow a\alpha$	$\beta \rightarrow a\alpha$
$\delta \rightarrow a\alpha$			

3 reduction (9 → 5) $\delta, \beta \rightarrow \alpha$

$S \rightarrow c\alpha$	$S \rightarrow bb\alpha$	$\alpha \rightarrow a\alpha$	$\alpha \rightarrow b$
$\alpha \rightarrow ab$			



Evans Method - Example 3

G

1 reduction (18 → 13) $\gamma, \epsilon, \psi, \chi, \iota \rightarrow \gamma$ $\mu, \nu \rightarrow \mu$

$S \rightarrow c\alpha$	$S \rightarrow bb\delta$	$S \rightarrow bb\gamma$	$S \rightarrow c\gamma$
$S \rightarrow c\varphi$	$S \rightarrow c\mu$	$S \rightarrow bb\mu$	$\alpha \rightarrow a\beta$
$\beta \rightarrow a\gamma$	$\gamma \rightarrow ab$	$\delta \rightarrow a\gamma$	$\varphi \rightarrow a\gamma$
$\mu \rightarrow b$			

2 reduction (13 → 9) $\alpha, \mu, \varphi, \gamma \rightarrow \alpha$

$S \rightarrow c\alpha$	$S \rightarrow bb\delta$	$S \rightarrow bb\alpha$	$\alpha \rightarrow a\beta$
$\alpha \rightarrow b$	$\alpha \rightarrow ab$	$\alpha \rightarrow a\alpha$	$\beta \rightarrow a\alpha$
$\delta \rightarrow a\alpha$			

3 reduction (9 → 5) $\delta, \beta \rightarrow \alpha$

$S \rightarrow c\alpha$	$S \rightarrow bb\alpha$	$\alpha \rightarrow a\alpha$	$\alpha \rightarrow b$
$\alpha \rightarrow ab$			



Evans Method - Example 4

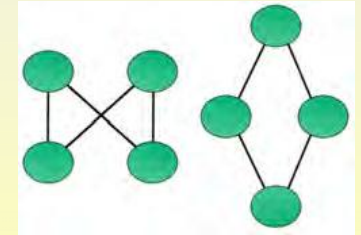
final grammar:

$$G \begin{cases} S \rightarrow c\alpha \\ S \rightarrow bb\alpha \\ \alpha \rightarrow a\alpha \\ \alpha \rightarrow b \end{cases}$$



Graph Matching

- graph isomorphism
an exact match of two graphs, given two graphs $G_i = (V_i, E_i)$, find one-to-one and onto mapping (an isomorphism) f between V_1, V_2 such that for each edge E_1 connecting any pair of nodes $v, v' \in V_1$, there is an edge in E_2 connecting $f(v), f(v')$; further if $f(v), f(v')$ are connected in G_2 , v, v' are connected in G_1 .



- subgraph isomorphism
finding an isomorphism between a graph G_1 and subgraphs of G_2 , more difficult problem

