AUTOMATA THEORY AND FORMAL LANGUAGES

UNIT 4: LANGUAGES AND FORMAL GRAMMARS

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

- Definition of a grammar (notation)
  - Sentential form. Sentence
  - Equivalent grammars
  - Sentences and handles
  - Recursion
- Chomsky Hierarchy
  - Regular grammars (Type 3, G3) and equivalences
- Parse trees
  - Ambiguity
- Context-free grammars languages(Type 2, G2)
  - Language generated by a Type-2 Grammar
  - Well-formed grammars
  - Chomsky. Normal Form. Greibach Normal Form

OUTLINE

- **Definition of a grammar (notation)**
  - **Sentential form. Sentence**
  - **Equivalent grammars**
  - **Sentences and handles**
  - **Recursion**
- Chomsky Hierarchy
  - Regular grammars (Type 3, G3) and equivalences
- Parse trees
  - Ambiguity
- Context-free grammars languages(Type 2, G2)
  - Language generated by a Type-2 Grammar
  - Well-formed grammars
  - Chomsky. Normal Form. Greibach Normal Form

Formal Grammars. Introduction

- We need a **better representation for languages**:
  - Valid for a finite or infinite number of elements.
  - Denote the structure of the words and the sentences in the language.
  - Valid for natural languages and programming languages.

Solution: GRAMMARS

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Introduction

- **< element > :** Symbols of the grammar.
- Notation for **possible choices**:

  < noun phrase>::= <noum group> <descriptor>
  <nominal syntagm>::= <noun group>

  < noun phrase >::= <noun group> <descriptor> | <noun group>

- **Values**:

  <noum> ::= John | Mary” | “house” | “student” | “thing”

- **Recursive rules**:
  - <objects> ::= <object> <objects>
  - Several steps: from <sentence> to <descriptor>

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Languages. Derivations rules

- **Productions, writing rules or derivation rules:**
  - Let Σ be an alphabet.
  - A production is an **ordered pair** (x,y) where $x, y \in \Sigma^*$
  - x = left side of the production
  - y = right side of the production
  - It is represented by: x::= y

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Languages. Derivations rules

- **Direct derivation:**
  - Let Σ be an alphabet and P a set of productions over Σ
  - Let v and w be two words over Σ, v, w $\in \Sigma^*$
  - We can say { "w is direct derivation of v", "v directly produces w", "w reduces directly to v" } $v \rightarrow w$

    If there are two words z, u $\in \Sigma^*$ that fulfill:

    $v = z \cdot x \cdot u$ $\quad$ $w = z \cdot y \cdot u$

    $(x ::= y) \in P$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Languages. Derivations rules

- **Direct derivation, examples:**
  - Let Σ be the Spanish alphabet of uppercase letters and ME::=BA is a production over Σ
    - CABALLO is a direct derivation of CAMELLO (CAMELLO directly generates CABALLO)
    - Using the production CA::=PE over Σ, PERA is a direct derivation of CARA

It is not really happens in most of the languages (specific roots)

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Languages. Derivations rules

- **Derivation of lenght n:**
  - Let Σ be an alphabet and P a set of productions over Σ
  - Let v and w be two words over Σ, v, w $\in \Sigma^*$
  - We can say { "w is a derivation of v" / "v produces w" / "w is reduced to v" } $v \overset{+}{\rightarrow} w$

If there are a finite sequence of words, $u_0, u_1, u_2, \ldots u_n$ (n>0) $\in \Sigma^*$ that fulfill

$v = u_0$

$u_0 \rightarrow u_1$

$u_1 \rightarrow u_2$

.....

$u_{n-1} \rightarrow u_n$

$w = u_n$

} **Derivation of lenght n**

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Languages. Derivations rules

- **Derivation, example:**
  - Given the alphabet Σ = {0,1,2,N,C} and the set of productions over this alphabet, P = {N::CN, N::=C, C::=0, C::=1, C::=2}
  - Indicate the lenght of the derivation

    $N \overset{+}{\rightarrow} 210$

COROLLARY: if $v \rightarrow w$ then $v \overset{+}{\rightarrow} w$ by means of a derivation with lenght = 1

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Languages. Derivations rules

- **Thue Relationship:**
  - Let Σ be an alphabet and P a set of productions over Σ
  - Let v and w be two words over Σ, v, w $\in \Sigma^*$
  - There is a Thue Relationship among v and w represented by **v $*\rightarrow$ w** if:
    - **v $+\rightarrow$ w**
    - **v = w**

    There is a derivation of lenght n or there are the same words
  - Properties:
    - Reflexion
    - Simetry (not in general)
    - Transitive

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definition

$G=\{\Sigma_T, \Sigma_N, S, P\}$

- Σ_T terminal symbols alphabet
- Σ_N nonterminal symbols alphabet
  - $\Sigma = \Sigma_T \cup \Sigma_N$ (alphabet or vocabulary of G)
  - $\Sigma_T \cap \Sigma_N = \varnothing$
- S axiom or start symbol ($S \in \Sigma_N$)
- P finite set of productions:
  - w → z
  - w →* z
  - w →+ z

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definition

Formal entity to specify (in a finite notation) the set of strings with symbols that make up a language.

- Σ_T → All the strings in the language represented by G (L (G)) are made with this alphabet symbols.
- Σ_N → set of auxiliary symbols introduced for the definition of G but not included in the strings of L (G)

S →Initial symbol from which the production rules P are applied.

P → set of production rules:

$$\mathbf{u::=v} \text{ where } u \in \Sigma^+ \text{ and } v \in \Sigma^* \; u = xAy$$

$$\text{where } x, y \in \Sigma^* \text{ and } A \in \Sigma_N$$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definition

- **Grammar**
  - Set of rules by which valid sentences in a language are constructed.
- **Nonterminal**
  - Grammar symbol that can be replaced to a sequence of symbols.
- **Terminal**
  - Actual word in the language (not further expansion is possible).
- **Production**
  - Grammar rule that defines how to replace/exchange symbols
- **Derivation**
  - A sequence of applications of the rules of a grammar that produces a finished string of terminals.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Backus Notation

- Formal notation to describe the syntax of a given language.
  - ::= meaning “is defined as”
  - | “or”
  - If u::=v and u::= w are two production rules included in P, then we can write u ::= v | w

- Nonterminal symbols are usually represented by uppercase letters.
- Terminal symbols by lowercase letters.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Backus Notation

Grammar not in Backus Notation:

G = ({0,1}, {N,C}, N,P)

P = {N::= NC,

N::=C,

C::= 0,

C ::=1}

Grammar in Backus Notation:

G = ({0,1}, {N,C}, N,P)

P = {N::= NC | C,

C::= 0 | 1}

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Sentential form**:
  - x is a sentential form if $S \rightarrow_* x$
- **Sentence**:
  - x is a sentence if it is a sentential form and $x \in \Sigma_T^*$
- **Language generated by a grammar *G*:**
  - $L(G)=\{x \text{ where } S \rightarrow_* x,\ x \in \Sigma_T^*\}$
- **Grammar equivalence**:
  - G_1 and G_2 are equivalent if $L(G_1)=L(G_2)$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Sentential form:**

Given a grammar G={Σ_T, Σ_N, S, P}

A word x $\in$ $(\Sigma_T \cup \Sigma_N)^*$ is a **sentential form** of G if:

$$\mathbf{S} * \rightarrow \mathbf{x}$$

Explanation:

- There is a Thue relation between the axiom and x:
  - x is the axiom.
  - Using the production rules P, it is possible to obtain x from a set of derivations beginning with *S.*

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Sentences or Words generated by the grammar:**

  Every sentential form x of a grammar that fulfills

  $$x \in \Sigma_T^*$$

Explanation:

- Words that are generated by the grammar.
- Sentential forms that only contain terminal symbols.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Language associated to a grammar:**

Given a grammar G={Σ_T, Σ_N, S, P}

The language generated/associated/described by G is

$L(G) = \{x \mid S \rightarrow^* x$ AND $x \in \Sigma_T^*\}$

Explanation:

- All the sentences generated by G.
- Motivation: Grammar Theory and Backus Representation

  →Notation to represent and describe languages.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Equivalent Grammars:**

Two grammars G1 and G2 are equivalent if they describe /generate the same language:

$$G1 \approx G2$$

If

$$L(G1) = L(G2)$$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Sentence:**

Given a grammar G={Σ_T, Σ_N, S, P}

v= xuy sentential form of G

u is a sentence of the sentential form **v** with regard a nonterminal symbol

U (U$\in \Sigma_N$) if:

$$\mathbf{S} \overset{*}{\rightarrow} \mathbf{xUy}$$

$$\mathbf{U} \overset{+}{\rightarrow} \mathbf{u}$$

Explanation:

- U can be replaced for u by means of one or several derivations in the sentential form xUy to generate the sentential form v, remaining x and y.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Simple Sentence:**

Given a grammar G={Σ_T, Σ_N, S, P}

v= xuy sentential form of G

u is a simple sentence if:

$\mathbf{S} \overset{*}{\rightarrow} \mathbf{xUy}$

$\mathbf{U} \rightarrow \mathbf{u}$ (direct derivation)

Explanation:

- U can be replaced for u by means of only one derivation.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Handle:**

  The handle of a sentential form v is the simple sentence of v situated at left.

COROLLARY:

- If the grammar is not ambiguous, then there is only one handle.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Recursion:**

A production rule is recursive if:

$$U \overset{+}{\rightarrow} x\,U\,y$$

- Left recursive production: $x = \lambda\ (U \overset{+}{\rightarrow} U\,y)$
- Right recursive production: $y = \lambda\ (U \overset{+}{\rightarrow} xU)$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Definitions

- **Recursion:**

Given a grammar G={Σ_T, Σ_N, S, P}

It is recursive in the nonterminal U $\in$ Σ_N if

$$\mathbf{U} \overset{+}{\longrightarrow} \mathbf{xUy}$$

- Left recursive grammar → x= λ
- Right recursive grammar → y= λ
- If a language is infinite, then the grammar that represents it must be recursive.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

- Definition of a grammar (notation)
  - Sentential form. Sentence
  - Equivalent grammars
  - Sentences and handles
  - Recursion
- **Chomsky Hierarchy**
  - **Regular grammars (Type 3, G3) and equivalences**
- Parse trees
  - Ambiguity
- Context-free grammars languages(Type 2, G2)
  - Language generated by a Type-2 Grammar
  - Well-formed grammars
  - Chomsky. Normal Form. Greibach Normal Form

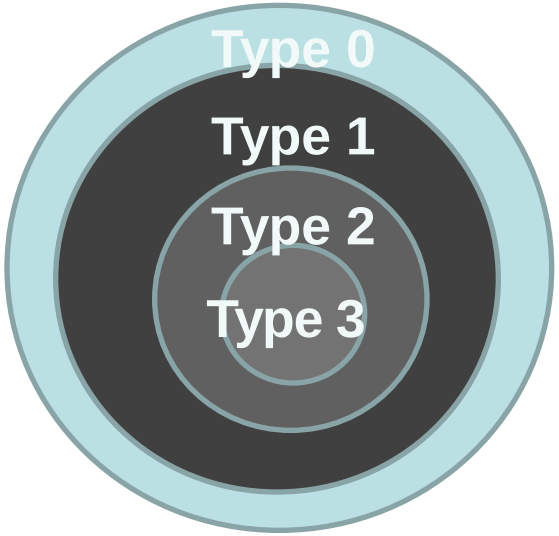
Formal Grammars. Chomsky Hierarchy

- **Type 0**: unrestricted grammars
  - u → v (u not null)
- **Type 1**: context-sensitive grammars
  - $P=\{xAy::=xvy \text{ where } x,y\in\Sigma_V{}^* \wedge A\in\Sigma_N \wedge v\in\Sigma_V{}^+\}$
  - uXw → uvw (v not null, X single nonterminal)
- **Type 2**: context-free grammars
  - $P=\{A::=v \text{ where } A\in\Sigma_N \wedge v\in\Sigma_V{}^+\}$
  - X → v (X single nonterminal)

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy

- **Type 3**: regular or linear grammars
  - X →a, X → aY, X → ε
  - linear on the left
    - $P = \{(A::=Ba) \cup (A::=a) \mid A,B \in \Sigma_N \wedge a \in \Sigma_T{}^*\}$
  - linear on the right
    - $P = \{(A::=aB) \cup (A::=a) \mid A,B \in \Sigma_N \wedge a \in \Sigma_T{}^*\}$



David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursively enumerable (Turing-recognizable)	Turing machine
Type-1	Context-sensitive	Context-sensitive	Linear-bounded
Type-2	Context-free	Context-free	Pushdown
Type-3	Regular	Regular	Finite

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G0)

- Languages defined by Type-0 grammars are accepted by Turing Machines: Recursively enumerable languages.

- More generic grammars (all possible grammars): unrestricted grammars or phrase-structured.

- Rules are of the form:
  - $u \rightarrow v$,
  - *where u and v are* arbitrary strings over a vocabulary *V and* $u \neq \lambda$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G0)

- Only restriction: $\lambda ::= v \notin P$
- Sentential form:

$$u = xAy,\ x,y \in \Sigma^*,\ A \in \Sigma_N$$

- Example:

$$G = \{(0,1), (S), S, P\}, \text{ where:}$$

$$P = \{(S \rightarrow 000S111), (0S1 \rightarrow 01)\}$$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G0)

- **Phrase-structure grammars:**
  - $(xAy::=xvy) \in P$, where:

    $x,y \in \Sigma^*$, $A \in \Sigma_N$, $v \in \Sigma^*$

- If $v=\lambda$, then:

  xAy ::= xy compressor rules

- Compressor Grammar: Contains at least a compressor rule.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G0)

- COROLLARY: : Every L(G0) can be described by means of a **Phrase-structure** G0 grammar.

G= ({a,b},{A,B.C},A,P)

P = {A::= aABC / abC

CB::= BC

bB::=bb

bC::=b}

G'= {a,b},{A,B.C,X,Y},A,P')

P' = {A::= aABC / abC

CB::=XB, XB::=XY

XY::=BY, BY::=BC

bB::=bb

bC::=b}

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G1)

- Languages defined by Type-1 grammars are accepted by linear-bounded automata: Context-sensitive languages.

- Syntax of some natural languages (Germanic).

- Rules are of the form:
  - *xAy ::= xvy*

  where:
  - $x,y \in \Sigma^*$, $A \in \Sigma_N$
  - $v \in \Sigma^+$ Compressor rules are not allowed.
  - *Exception:* $S \rightarrow \lambda$ can be included in P.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G1)

- Example:

  G= {(0,1,2,3,4,5,6,7,8,9,), (<number>, <digit>),<number>, P},

  P={(<number> → <digit><number>, <number> → <digit>,

  <digit>→0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9)}

- $\lambda \in L(G1)$ if $(S::=\lambda) \in P$
- Grammar that is not G1:

  G= ({a,b}, {S}, S, P)

  P= {S::= aaaaSbbbb, aSb::= ab}

- Grammar that is G1:

  G= ({a,b}, {S}, S, P)

  P= {S::= aaaaSbbbb, aSb::= abb}

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G1)

- **NO decrease property in G1:**

  Strings obtained using the derivations of a G1 are not decreasing length, that it is to say:

  $$\mathbf{u} \rightarrow \mathbf{v} \Rightarrow |\mathbf{v}| \geq |\mathbf{u}|$$

  the length of the right side in the production is greater than or equal to the length of the left side

  - Demonstration:

    $\alpha A \beta \rightarrow \alpha \gamma \beta$

    where $\gamma \in (\Sigma_T \cup \Sigma_{NT})^+$ due to the definition of G1 (no compressor rules)

    that it is to say, $\gamma \neq \lambda$ always, and then $|\gamma| \geq 1$

    Given that $|A| = 1$ at least, it is demonstrated.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G2)

- Languages defined by Type-2 grammars are accepted by push-down automata: context-free languages.
- Natural languages are almost entirely definable by type-2 tree structures.
- Rules are of the form:
  - *A := v (only one NT symbol on the left)*

  where
  - $A \in \Sigma_N$ (A::= λ can appear)
  - $v \in \Sigma^*$
- $\forall$ L(G2) $\exists$ L(G2') without rules A::= λ ($A \neq S$)

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G2)

- r $\in$ P have **only one nonterminal in the left side.**
- **(S::= λ) $\in$P** and **(A::= λ) $\notin$P**

  (algorithm for cleaning rules that are not generative)
- Example:

$$\mathbf{G= \{(a,b), (S,A),S, P\}}$$

$$P=\{(S \rightarrow aS,\ S \rightarrow aA,\ A \rightarrow bA,\ A \rightarrow b)\}$$

- **Context-Free:** A can be changed for v in any context.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G2)

- The syntax of most programming languages is context-free (or very close to it).
- EBNF / ALGOL 60…
- Due to memory constraints, long-range relations are limited.
- Common strategy: a relaxed CF parser that accepts a superset of the language, invalid constructs are filtered.
- Alternate grammars proposed: indexed, recording, affix, attribute, van Wijngaarden (VW).

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G3)

- Languages defined by Type-3 grammars are accepted by finite state automata: regular languages.
- Most syntax of some informal spoken dialog.
- $\forall$ L(G3) $\exists$ L(G3') without rules A::= λ (A $\neq$ S)
- *Two types:*
  - **left-linear** or left regular grammars.
  - **right-linear** or right regular grammars.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G3)

- **left-linear** or left regular grammars:
  - **A::= a**
  - **A::= Va**
  - **S::= λ**
- **right-linear** or right regular grammars
  - **A::= a**
  - **A::= aV**
  - **S::= λ**
- Where
  - $a \in \Sigma_T$
  - $A,V \in \Sigma_N$ S is the axiom

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy (G3)

- $r \in P$ have only one nonterminal symbol in the left side and the right side begins with a terminal symbol followed or not by nonterminal symbols (**right-linear)**

- Example:

$$G= \{(a,b), (S,A),S, P\},$$

$$P=\{(S \rightarrow aS,\ S \rightarrow aA,\ A \rightarrow bA,\ A \rightarrow b)\}$$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Chomsky Hierarchy

- A language is called type i (i= 0, 1, 2, 3) if there is a grammar G type i that fulfills L= L(Gi)
- Hierarchy: inclusion relationship.

$$G3 \subset G2 \subset G1 \subset G0$$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

EQUIVALENT GRAMMARS

1. ∀ G3 right-linear grammar with rules A::= aS

    ∃ A right-linear grammar G3' equivalent without rules A::= aS

2. ∀ G3 right-linear

    ∃ a G3' left-linear equivalent grammar.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Regular grammars (Equivalences)

1. ∀ G3 right-linear grammar with rules A::= aS

∃ An equivalent right-linear G3' grammar without rules A::= aS

Transformation Procedure:

1. Incorporation of a new symbol in the alphabet Σ_N
2. ∀ S::= x, where $x \in \Sigma^+$, a rule B::= x is added.
3. The rules A::= aS are transformed into A::= aB rules.
4. Rules S::= λ are unaffected by this algorithm.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

2. ∀ G3 right linear ∃ an equivalent G3' left linear

Transformation Procedure

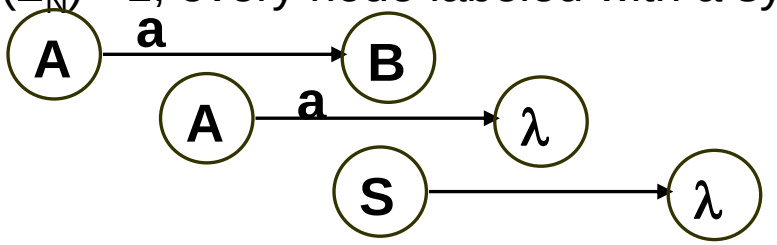
1. **Creation of a graph:**

   a) Number of nodes= C(Σ_N) +1, every node labeled with a symbol in Σ_N and another with λ

   b) every A::=aB ∈P

   c) every A::=a ∈P

   d) if S::= λ ∈P

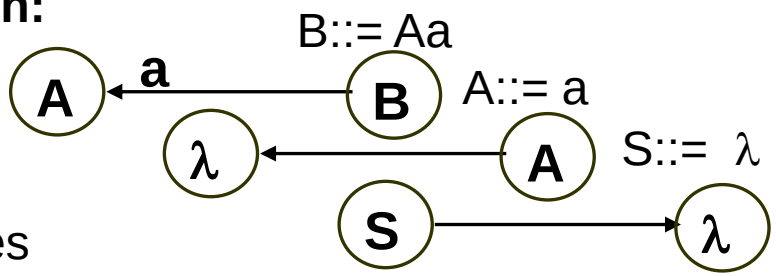


2. **Transformation of the previous graph:**

   a) Exchange labels of S and λ

   b) invert the direction of the arcs

   c) Undo the path and generate the new rules



⇒ interpret the graph to obtain the equivalent left-linear G3.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

- Definition of a grammar (notation)
  - Sentential form. Sentence
  - Equivalent grammars
  - Sentences and handles
  - Recursion
- Chomsky Hierarchy
  - Regular grammars (Type 3, G3) and equivalences
- **Parse trees**
  - **Ambiguity**
- Context-free grammars languages(Type 2, G2)
  - Language generated by a Type-2 Grammar
  - Well-formed grammars
  - Chomsky. Normal Form. Greibach Normal Form

Formal Grammars. Representations

- We can represent the application of rules to derive a sentence in two ways:
  - A derivation.
  - A parse tree.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

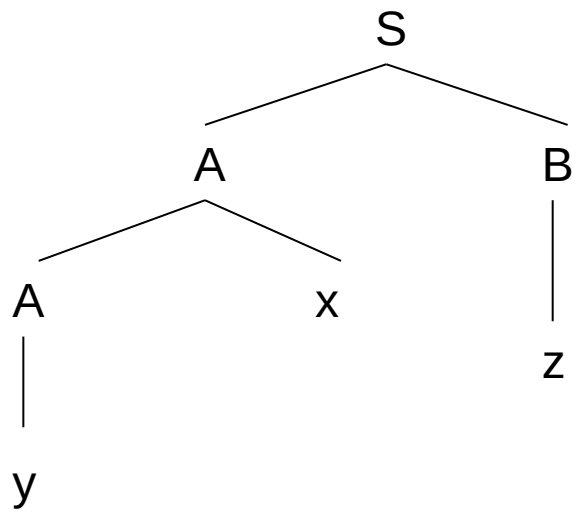
Formal Grammars. Representations (Derivation)

- A derivation is a sequence of applications of the rules of a grammar that produces a word (a string of terminals).
- Leftmost derivation:
  - the leftmost nonterminal symbol is always replaced when the rules are applied.
- Rightmost derivation:
  - the rightmost nonterminal symbol is always replaced when the rules are applied.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Representations (Parse tree)

- It shows how each symbol derives from other symbols in a hierarchical manner. It does not encode the order the productions are applied.



David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

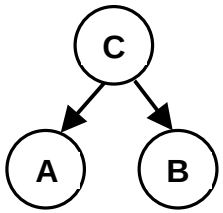
Formal Grammars. Representations (Parse tree)

- The derivations of the grammars G type 1, 2 and 3 can be represented by a hierarchical structure called **parse tree.**
- It represents productions used for the generation of a word, that it is say, the structure of the words according to G.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Representations (Parse tree)

- It is an ordered and labeled tree that is build:
  - The root is denoted by the axiom of G.
  - A direct derivation is represented by a set of branches coming out of a given node (left side of the P).
  - By the application of a rule → a symbol on the left side is replaced by a word *u* on the right side. For each symbol in u, a branch is drawn from a nonterminal to this symbol: e.g. Given a rule u=AB and the production P: C→u



The leftmost symbol in a P, it is also leftmost situated in the tree.

Formal Grammars. Representations (Parse tree)

- **In a G1, the tree must also retain the context information:**
- For each branch:
  - the starting node is called **parent** of the end node.
  - an end node is called **child** of the parent node.
  - two child nodes of the same parent are called **brothers**.
  - a node is an **ascendant** of another if it is its parent node or an ascendant of its parent node.
  - a node is a **descendant** of another if it is its child or a descendant of its child nodes.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Representations (Parse tree)

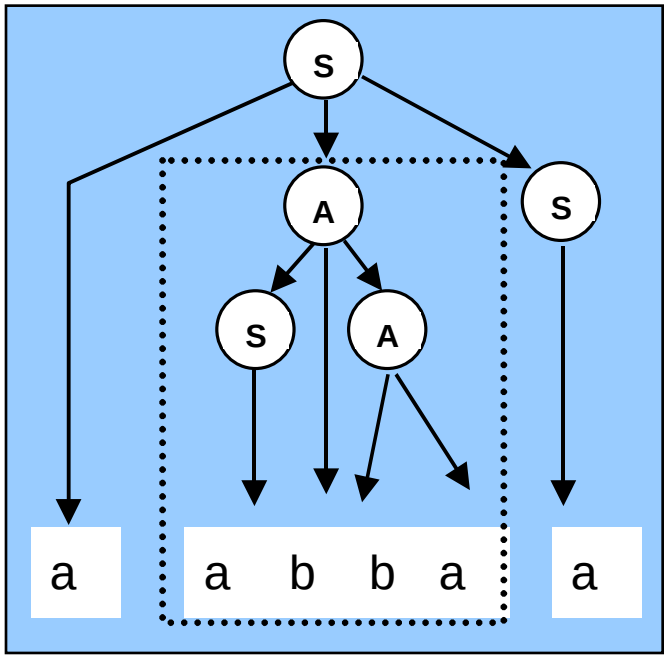
- In the process of construction of the tree, the end nodes of each successive step, read from left to right, give us the **sentential form** obtained by the derivation represented by the tree.

- The set of the leaves of the tree (nodes denoted by terminal symbols or λ) read from left to right give us the **sentence** generated by the derivation.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Representations (Parse tree)

- **Theorem:** the leaves nodes of a subtree, read from left to right, form a sentence with regard the nonterminal symbol root of the subtree.



abba is a sentence of the sentential form *aabbaa* with regard the symbol A

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Ambiguity

Concepts related to the derivation tree:

- If a sentence can be obtained from a G by means of two or more parse trees, the sentence is ambiguous.
- A G is ambiguous if it contains at least one ambiguous sentence.
- Even if a G is ambiguous, the language that it describes might be not ambiguous [Floyd 1962] $\Rightarrow$ it is possible to find an equivalent grammar G that it is not ambiguous.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Ambiguity

Concepts related to the derivation tree:

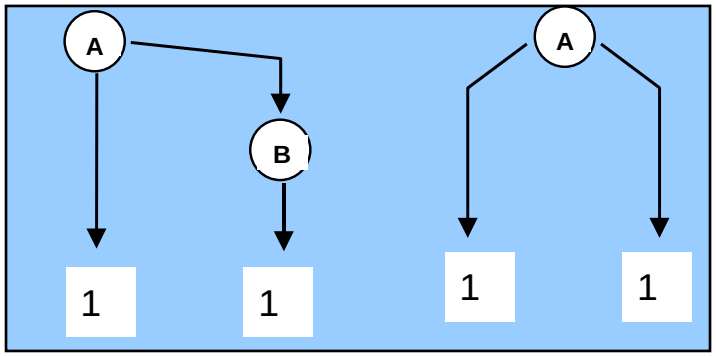
- There are languages for which it is not possible to find no ambiguous grammars $\Rightarrow$ Inherent ambiguous languages [Gross 1964]
- The property of ambiguity is undecidible. It is only possible to find sufficient conditions to ensure that a G is not ambiguous.
- Undecidible: there is not an algorithm that accepts a G and ensures in a finite time whether a G is ambiguous or not.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Ambiguity

- From the previous definitions, there are 3 levels of ambiguity:
  - **Sentence:** a sentence is ambiguous if it can be obtained by means of two or more different derivation trees.

e.g.: G = ({1}, {A,B}, A, {A::= 1B / 11, B::= 1})



  - **Grammar**: it is ambiguous if it contains at least an ambiguous sentence, e.g.: the previous G.

  - **Language:** a language L is ambiguous if there is an ambiguous grammar that generates it, e.g.: L = {11} is ambiguous.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Ambiguity

- **Inherent Ambiguous Languages:** languages for which it is not possible to find a not ambiguous G $\Rightarrow$ example L = $\{\{a^n b^m c^m d^n\} \cup \{a^n b^n c^m d^m\} / m,n \geq 1\}$

  example: L = {11} is not inherent ambiguous.

  G' = ({1}, {A}, A, {A..=11})

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

OUTLINE

- Definition of a grammar (notation)
  - Sentential form. Sentence
  - Equivalent grammars
  - Sentences and handles
  - Recursion
- Chomsky Hierarchy
  - Regular grammars (Type 3, G3) and equivalences
- Parse trees
  - Ambiguity
- **Context-free grammars languages(Type 2, G2)**
  - **Language generated by a Type-2 Grammar**
  - **Well-formed grammars**
  - **Chomsky. Normal Form. Greibach Normal Form**

Formal Grammars. Context-Independent Grammars

- Type 2 grammars in Chomsky Hierarchy.
- Importance:

  They are used in the definition of programming languages and the compilation processes.

Compiler
- analysis phase → grammar G
  - valid words (Finite Automata)
  - valid structure (Push-Down Automata)
- synthesis phase → Translator

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Context-Independent Grammars

- Languages generated by Chomsky Hierarchy type 2 grammars are called:
  - Context-Independent or Context-Free languages.
  - Represented by L(G2).
  - There are algorithms that recognize if a L (G2) is empty, finite or infinite.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Context-Independent Grammars

- Language, empty or not?:

  Let G2, m = C(Σ_{NT}), **L(G2) $\neq \phi$** if $\exists$ x $\in$ L(G2) that can be generated with a derivation tree in which all the paths have a length $\leq$ m

  All the derivation trees with paths $\leq$ m = C(Σ_{NT}) by means of the algorithm:

  a) Set of trees with length 0 (a tree having S as a root and without branches).

  b) From the set of trees of length n, we generate the set with length n+1 < m +1 by using the initial set and applying a production that does not duplicate a nonterminal in the considered path.

  c) The step b) is recursively applied until any tree with length $\leq$ m can not be obtained. Given that m and the number of productions P is finite $\Rightarrow$ the algorithm ends.

  L(G2) = ϕ if none of the trees generates a sentence

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Context-Independent Grammars

- Example of L(G2) = ϕ

  G = ({a,b}, {A,B,C,S}, S, P)

  P = {

  S ::= aB / aA

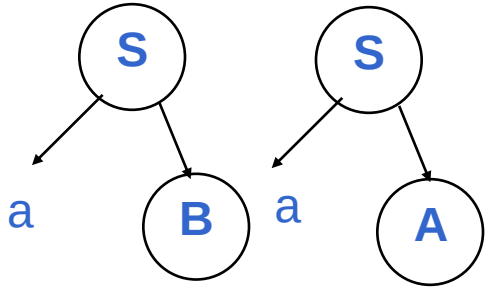
  A ::= B / abB

  B ::= bC}

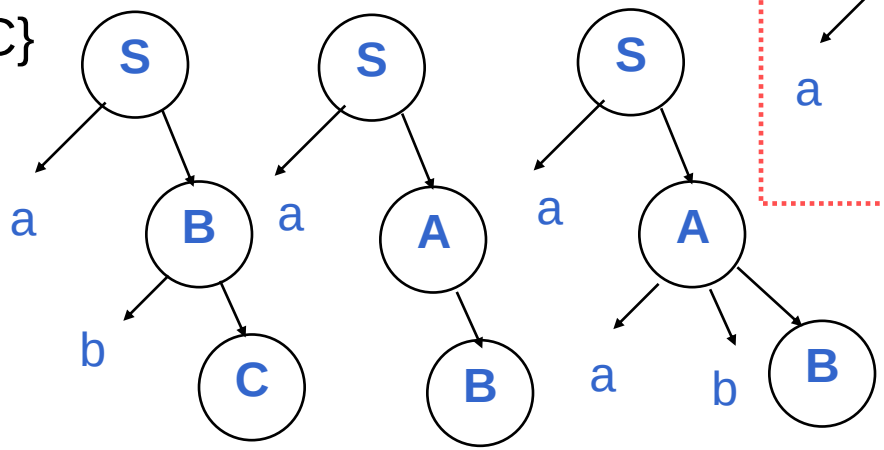
$m=C(\Sigma_{NT}) = 4$

1. m=0

2. m=1



3. m=2



4. m=3

Sentences are not generated and there are nonterminals already obtained

Empty language

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Context-Independent Grammars

- If L(G2) is nonempty, evaluate if **L(G2) = ∝**

  A graph is built whose nodes are labeled with the symbols of (Σ_{NT}) by means of the algorithm:

  a) if ∃ a production A::= α B β, an arc from A to B is created, where A,B $\in \Sigma_{NT}$ and $\alpha,\beta \in \Sigma^*$
  b) If there are not cycles in the graph, then L(G2) = finite
  c) L(G2) = ∝ if there are cycles that are accessible from the axiom and correspond to derivations A $\rightarrow$+ α A β, where /α/+ /β/ >0 (they can not be λ simultaneously).

L(G2) ≠ ∝ if there are not cycles in the graph.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Formal Grammars. Context-Independent Grammars

- Example of **L(G2) =** $\propto$

  G = ({a,b,c}, {A,B,C,S}, S, P)

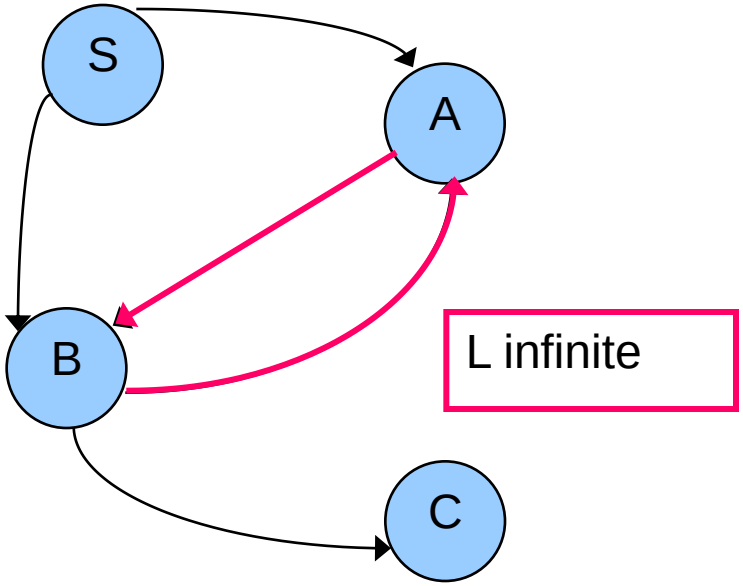
  P = {

  S ::= aB / aA

  A ::= abB

  B ::= bC / aA

  C ::= c }



David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Well-formed Grammars

- Transformation of a given G into another whose production rules are in a well-formed format without imperfections:
  1. **Possible imperfections:**
     1. Unnecesary Rules.
     2. Unreachable Symbols.
     3. Useless Rules.
  2. **Elimination of Not Generating symbols.**
  3. **Elimination of Not Generating rules.**
  4. **Elimination of Redenomination rules (Unit Productions).**

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

1. **Possible imperfections**
   1. **Unnecesary Rules**: rules **A::= A $\in$P** are unnecesary and make the grammar to become ambiguous $\Rightarrow$ eliminate.
   2. **Unreachable Symbols**: Given U::= x $\in$P, where U$\in\Sigma_N \neq$ S and does not appear in the right side of any other production rule, then U is unreachable.

      Every symbol U$\in\Sigma_N$ not unreachable fulfills S $\rightarrow$ * xUy.

      Elimination of unreachable symbols:

      a) boolean matrix (Alfonseca, Chapter 3).

      b) Graph that is identical to L(G2) = $\propto$. Unreachable symbols are those not reached from the axiom.

      Eliminate those symbols and the rules that contain them.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

1. **Possible imperfections**

2. **Unreachable symbols: example:**

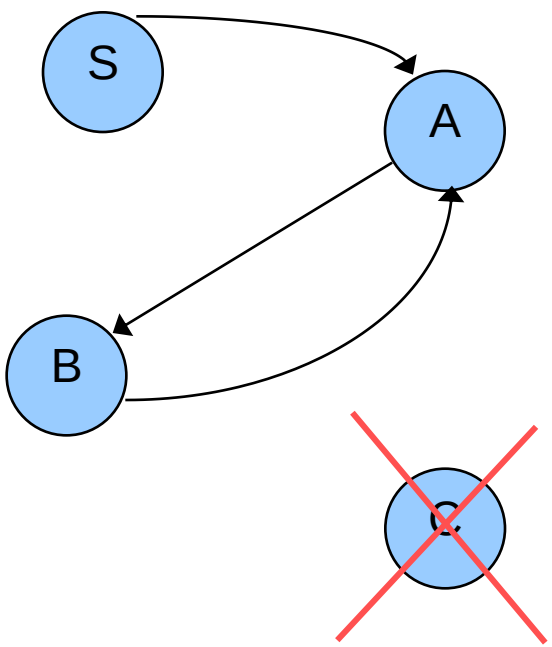
G = ({a,b,c}, {S,A,B,~~C~~}, S, P),

where P = {S::= aA

A ::= Bc

B::= bA

~~C::= c~~}



David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

1. **Possible imperfections**
   3. **Useless Rules**: are those that do not contribute to the words formation x $\in \Sigma_T^*$.

      Every not useless symbol fulfills U $\rightarrow$+t, and t $\in \Sigma_T^*$

      Algorithm:

      a) Annotate nonterminal symbols for which there is not a rule U ::= x where x $\in \Sigma^*$ (it is a string of T or λ, or in successive derivations it contains annotated nonterminal symbols)

      b) If every nonterminal is annotated $\Rightarrow$ there are not useless symbols (END).

      c) If a nonterminal symbol was annotated last time in a), then return to a).

      d) Every A $\in \Sigma_{NT}$ not annotated is useless.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

1. Possible imperfections

3. Useless Rules

example: G= ({e,f}, {S,A,B,C,D}, S, P)

where P= {

S::= Be

A::= Ae /e

B::= Ce / Af

C::= Cf

D::= f }

First iteration:

D::= f y A::= e

Second iteration:

B::= Af

Third iteration:

S::= Be

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

1. Possible imperfections

3. Useless Rules

example: G= ({e,f}, {S,A,B,~~C~~,D}, S, P)

where P= {

S::= Be

A::= Ae /e

B::= ~~C~~e / Af

C::= Cf

D::= f }

First iteration:

D::= f y A::= e

Second iteration

B::= Af

Third iteration:

S::= Be

Nonterminals not annotated: C → this symbol and the rules that contain it can be eliminated.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

2. **Elimination of Not Generating symbols:**

Given G2 = (Σ_T, Σ_N, S, P), $\forall \mathbf{A} \in \mathbf{\Sigma_N}$ we will build the grammar G(A), where A is the axiom. If **L(G(A))= $\phi \Rightarrow$ A is a Not Generating symbol** and can be eliminated, as well as all the rules containing it, obtaining another equivalent G2.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

3. **Elimination of Not Generating rules:** they are A::= λ (A$\neq$S)

   - Si $\lambda \in$**L(G)**: then we add S::= λ **and** $\forall$ A$\in\Sigma_{Nt}$ (A::= λ A$\neq$S) and

     $\forall$ rule de G with the structure B::=xAy, we add a P' rule with

     the structure B::= xy, except the case x=y=λ

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

4. Elimination of Redenomination rules: they are rules type A::=B

Algorithm:

a) The equivalent G' contains all the rules except:

A::= B

b) $\forall A \in \Sigma_{NT} \mid A \rightarrow^* B$ in G and $\forall$ (B::= x) $\in$P where $x \notin \Sigma_{NT} \Rightarrow$

P' = P + {A::= x}

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms

- There are possible notations for G2 grammars:
  - They define the valid structures for the production rules.
- We are going to study:
  - Chomsky Normal Form (CNF).
  - Greibach Normal Form (GNF).

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (CNF)

- Chomsky Normal Form (CNF).
  - Every nonempty context-free grammar without λ has a grammar *G* in which all productions are in one of the following two simple forms:
    - A → BC
    - A → a
  - Properties of *G*: no λ-productions, unit productions or useless symbols.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (CNF)

- Chomsky Normal Form (CNF).
  - Algorithm:

    a) *A* → *a* productions that are also in CNF.

    b) Arrange that all bodies of length 2 or more consist only of nonterminal.

    c) Break bodies of length 3 or more into a set of productions, each with a body consisting of two nonterminal.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (CNF)

- Chomsky Normal Form (CNF).
  - Algorithm:

    Step b):

    - For every terminal *a* that appears in a body of length 2 or more → create a new nonterminal *A* with only one production (*A*→*a*).
    - Use *A* instead of a everywhere *a* appears in a body of length 2 or more.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (CNF)

- Chomsky Normal Form (CNF).
  - Algorithm:

    Step c):

    - We break the productions

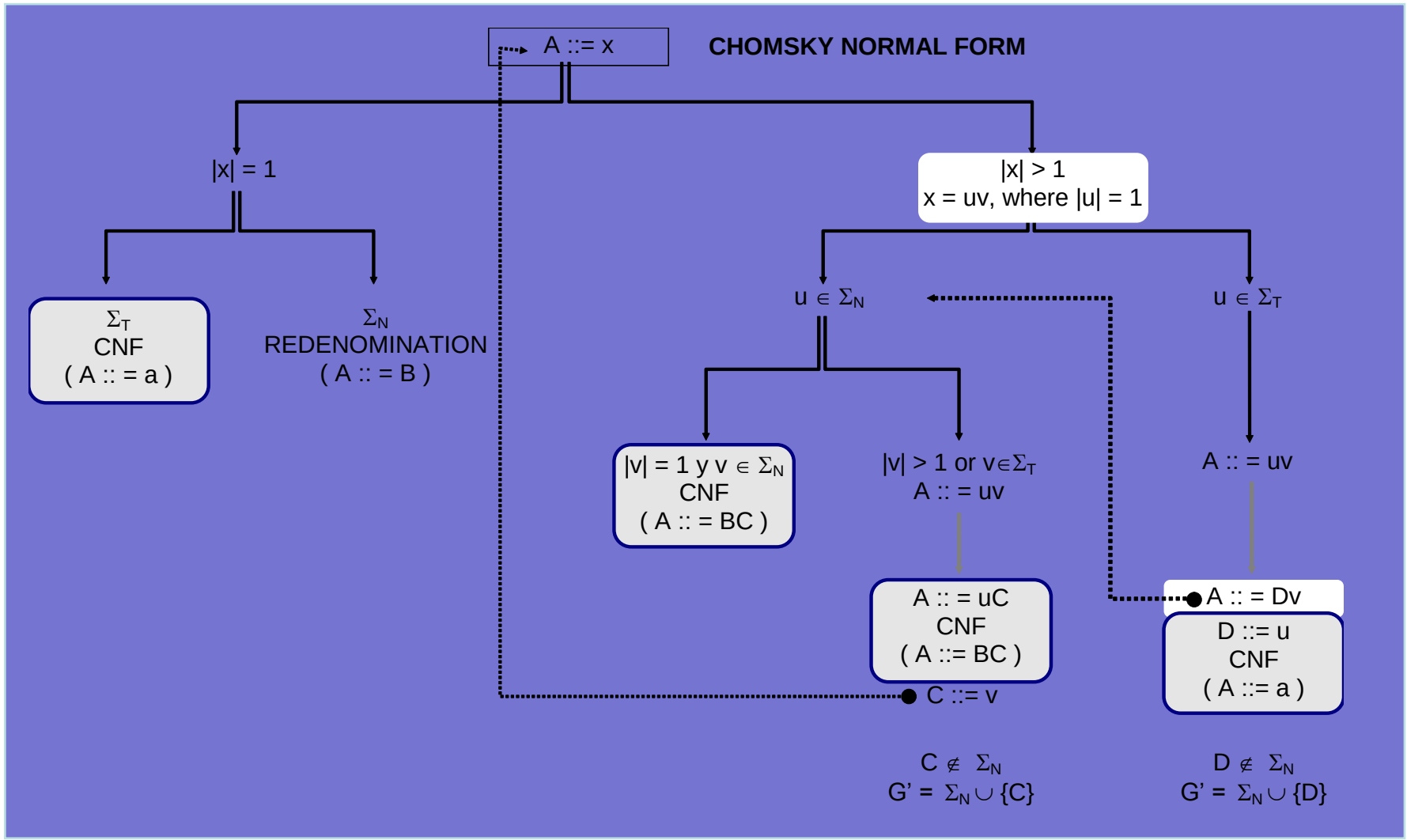
      $$A \rightarrow B_1B_2\cdots B_k,\ k \geq 3$$

      into a group of k-1 productions with two nonterminal in each body by introducing k-2 nonterminal $C_1, C_2, \cdots, C_{k-2}$

      $$A \rightarrow B_1C_1,\ C_1 \rightarrow B_2C_2, \cdots, C_{k-3} \rightarrow B_{k-2}C_{k-2},\ C_{k-2} \rightarrow B_{k-1}B_k$$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (CNF)



David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

- **GNF** is a very interesting notation for several techniques for syntax recognition. Every rule has in the right side an initial terminal symbol optionally followed by one or more nonterminal symbols.
- THEOREM: Every **context-free L without** λ can be generated by a G2 in which all the rules have the structure:
  - $\mathbf{A \rightarrow a\alpha}$ where $A \in \Sigma_{NT}$ $a \in \Sigma_T$ $\alpha \in \Sigma_{NT}^*$
  - If $\lambda \in L$, then it is necessary to define **S::= λ**
- THEOREM: Every G2 can be reduced to another equivalent G2 without left-recursive rules.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

- **GNF:** to tranform a G2 grammar into its equivalent Greibach normal form:

  1. Eliminate left recursion.
  2. Use the algorithm to transformate to GNF, verifying in every step that left recursive rules are not created, using step 1 to eliminate them if they are created.

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

1. Eliminate left recursion: only one step (several steps):

$G = (\{\alpha,\beta\}, \{A\}, A, P)$, where $P = \{A ::= A\ \alpha / \beta\}$

$A \rightarrow \beta$

$A \rightarrow A\ \alpha \rightarrow \beta\ \alpha$

$A \rightarrow A\ \alpha \rightarrow A\ \alpha\ \alpha \rightarrow \beta\ \alpha\ \alpha$

$A \rightarrow A\ \alpha \rightarrow A\ \alpha\ \alpha \rightarrow \ldots \rightarrow \beta\ \ldots\ldots\ \alpha\ \alpha$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

- Eliminating left recursion (one production)

$P=(A::=A\cdot\alpha|\beta)$

1. $\Sigma_N = \Sigma_N \cup \{A'\}$
2. $P'=\{A::= \beta \cdot A' , A'::= \alpha \cdot A' \mid \lambda\}$

- Example:
  - $P=\{ E ::= E * T \mid T\}$
  - Solution: $P' :=\{E ::= T E', E'::= * T E' \mid \lambda\}$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

- Eliminating the inmediate left recursion

$P=(A::=A\cdot\alpha_1|A\cdot\alpha_2|\ldots|A\cdot\alpha_n|\beta_1|\beta_2|\ldots|\beta_m|)$

1. $\Sigma_N = \Sigma_N \cup \{A'\}$
2. $P' = P \cup \{A::= \beta_1 \cdot A' | \beta_2 \cdot A' | \ldots | \beta_m \cdot A' | ,$
   $A'::= \alpha_1 \cdot A' | \alpha_2 \cdot A' | \alpha_n \cdot A' | \lambda\}$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

- Example:
  - $P : E ::= E * T \mid E + T \mid T$
- Solution:
  - $P' : E ::= T\ E'$

    $E' ::= * T\ E' \mid + T\ E' \mid \lambda$

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

- Recursion:
  - P: $A ::= B\ a$
    $B ::= A\ b \mid \lambda$
- Algorithm

1. Arrange the nonterminals in some order: $A_1, A_2, .., A_n$
2. For i=1 to n
   3. For j=1 to n
      replace each production of the form: $A_i \rightarrow A_j \cdot \gamma$
      by $A_i \rightarrow \delta_1 \cdot \gamma \mid \delta_2 \cdot \gamma \mid ... \mid \delta_k \cdot \gamma$
      where $A_j \rightarrow \delta_1 \mid \delta_2 \mid ... \mid \delta_k$ are all the current A_j productions
   4. Eliminate the inmediate left recursion in A_i

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

1. Eliminate left recursion: Summary

Given a grammar G = ({α_1, α_2, β_1, β_2}, {A}, A, P),

Where P = {A::= A α_1/ Aα_2 / β_1/ β_2}

The final result is:

A::= β_1/ β_2/ β_1X/ β_2X

X::= α_1/ α_2 /α_1 X/ α_2X

David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

2. Transformation of a well-formed G2 without left-recursion to GNF:

2.1 Define a partial relation in Σ_{NT}

$\Sigma_{NT} = \{A_1, A_2, ..., A_n\}$ based in: if $A_i \rightarrow A_j\ \alpha$, then A_i precedes A_j.

If there are "contradictory" rules, use on of them for the order and look at the rest to select the most convenient rule.

2.2 Classify the rules in three possible groups:

Group 1: $A_i \rightarrow a\ \alpha$,where a $\in \Sigma_T$ and $\alpha \in \Sigma^*$

Group 2: $A_i \rightarrow A_j\ \alpha$ where Ai precedes A_j in the set of orderly Σ_{NT}

Group 3: $A_k \rightarrow A_i\ \alpha$ where Ai precedes A_k in the set of orderly Σ_{NT}

2.3 The rules are transformed group 3 → group 2 → group 1: FNG

$A_k \rightarrow A_i\ \alpha$ substitute **A_i** for the right side of every rule that contains **Ai** as left side.

Repeat the same process with group 2 rules.

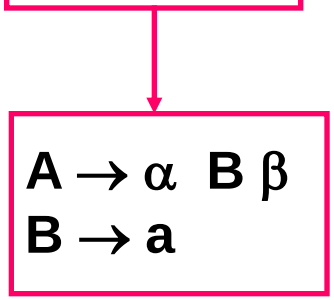
David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es

Context-Independent Grammars. Normal Forms (GNF)

2. Transformation of a well-formed G2 without left-recursion to GNF:

   2. 4. If all the rules are Group 1, the G is in GNF **and we have only to delete terminal symbols that do not appear on the right-hand header.**

$A \rightarrow \alpha\ a\ \beta$ **where** $a \in \Sigma_T$ **and** $\alpha \neq \lambda$



David Griol Barres - Computer Science Department – UC3M - dgriol@inf.uc3m.es