Formal Languages, Automata and Compilers

Lecture 1

2024-25

Formal Languages, Automata and Compilers

Lecture 1

Formal Languages, Automata and Compilers

- O. Captarencu: oana.captarencu@info.uaic.ro
  https://edu.info.uaic.ro/limbaje-formale-automate-si-compilatoare/
- A. Moruz:alex.moruz@info.uaic.ro

Evaluation

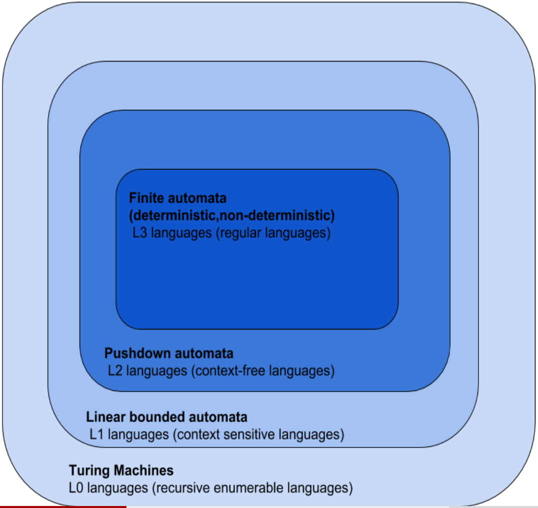
- 7 seminars, 6 laboratories;
- AS =seminar activity (max 10 points);
- AL = laboratory activtiy (max 10 points);
- T = written test during the examination session (a grade from 1 la 10) Final score:

  P = 2.5 * AS + 2.5 * AL + 5 * T
- Minimal conditions for passing the exam: $(AS + AL) \geq 10$, AS $\geq$ 4, AL $\geq$ 4, $T \geq 5$, $P \geq 50$;
- The final grade will be established according to the Gauss distribution

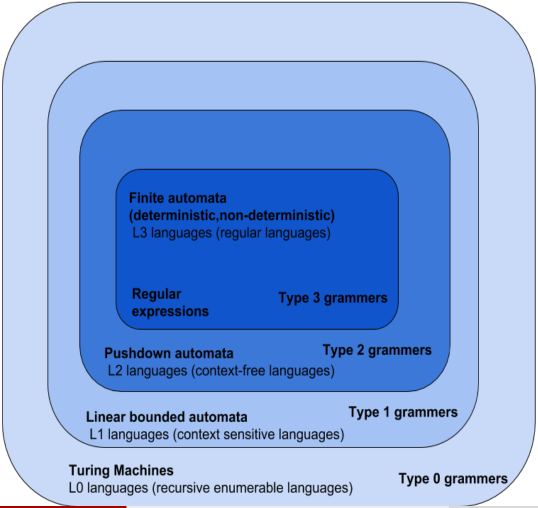
Evaluation

- SA = seminar activity (max 10 points):
  - the activity during the seminars - problem solving (max 2 points) + bonuses
  - written test (max 8 points)
- LA = laboratory activity:
  - project

Course Content (Part 1): Formal Languages and Automata



Course Content (Part 1): Formal Languages and Automata



Course Content (Part 1): Formal Languages and Automata

- Languages and grammars
- Regular languages; regular grammars, automata , regular expressions
- Context-free languages; context-free grammars, pushdown automata

Formal languages and automata: applications

- grammars
  - compilers : the syntax of programming languages
  - describe specific input for applications
  - describe the structure of XML documents (DTD)
  - Artificial Intelligence: in NLP (natural language processing)
- automata
  - compilers: lexical analysis
  - text processors: identification of specific patterns
  - modelling and verification of software and hardware systems
  - modelling network communication protocols
  - modelling of computer network protocols
  - AI: robotics
- regular expressions
  - describe and validate input in various applications
  - identification of patterns in text
  - tools in operating systems (grep, sed, awk)

Course Content (part II)

- Programming languages: design and implementation
- Lexical analysis
- Syntactic analysis
- Translation to intermediary code

Bibliography (selections)

1. A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman: Compilers: Principles, Techniques, and Tools. Boston: Addison-Wesley, 2007
2. Gh. Grigoras. Constructia compilatoarelor - Algoritmi fundamentali, Ed. Universitatii Al. I. ”Cuza Iasi”, ISBN 973-703-084-2, 274 pg., 2005
3. Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. (2006). Introduction to Automata Theory, Languages, and Computation (3rd ed.). Addison-Wesley
4. J. Toader - Limbaje formale şi automate, Editura Matrix Rom, Bucuresti, 1999.
5. J. Toader, S. Andrei - Limbaje formale şi teoria automatelor. Teorie şi practică, Editura Universitatii ”Al. I. Cuza”, Iasi, 2002.

Formal Languages, Automata and Compilers
Lecture 1

Basic notions

- Alphabet: (V) a finite set of symbols

Basic notions

- Alphabet: (V) a finite set of symbols
- Word: a finite range/succession of symbols
  - the empty word is denoted by ϵ (or λ).

Basic notions

- Alphabet: (V) a finite set of symbols
- Word: a finite range/succession of symbols
  - the empty word is denoted by ϵ (or λ).
- The length of word u: the number of symbols. Notation: $|u|$. $|\epsilon| = 0$

Basic notions

- Alphabet: (V) a finite set of symbols
- Word: a finite range/succession of symbols
  - the empty word is denoted by ϵ (or λ).
- The length of word u: the number of symbols. Notation: $|u|$. $|\epsilon| = 0$
- V^* - the set of all the words over alphabet V, including ϵ.

  $\{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

Basic notions

- Alphabet: (V) a finite set of symbols
- Word: a finite range/succession of symbols
  - the empty word is denoted by ϵ (or λ).
- The length of word u: the number of symbols. Notation: $|u|$. $|\epsilon| = 0$
- V^* - the set of all the words over alphabet V, including ϵ.
  $\{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$
- V^+ - the set of all the non-empty words over alphabet V
  $\{0, 1\}^+ = \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

Operations with words

- Concatenation of two words x, y: $x \cdot y$

  $x = 0100$, $y = 100$, $x \cdot y = 0100100$

  $x = 000$, $y = \epsilon$, $x \cdot y = 000$

Operations with words

- Concatenation of two words x, y: $x \cdot y$

  $x = 0100$, $y = 100$, $x \cdot y = 0100100$

  $x = 000$, $y = \epsilon$, $x \cdot y = 000$

- Concatenation is an associative operation

Operations with words

- Concatenation of two words x, y: $x \cdot y$

  $x = 0100$, $y = 100$, $x \cdot y = 0100100$

  $x = 000$, $y = \epsilon$, $x \cdot y = 000$

- Concatenation is an associative operation

- $(V^*, \cdot)$ is a monoid (ϵ is the identity element), the free monoid generated by V.

Languages

- Let V be an alphabet. A subset $L \subseteq V^*$ is a formal language over alphabet V if L has a finite (mathematical) description.
- A description can be:

Languages

- Let V be an alphabet. A subset $L \subseteq V^*$ is a formal language over alphabet V if L has a finite (mathematical) description.
- A description can be:
  - informal:
    - the set of words over alphabet $\{0,1\}$ which contain an even number of 0.
    - $L = \{x \in V^+ : |x| \textit{ is even}\}$.
    - $\{a^n b^n | n \in N\}$.

Languages

- Let V be an alphabet. A subset $L \subseteq V^*$ is a formal language over alphabet V if L has a finite (mathematical) description.
- A description can be:
  - informal:
    - the set of words over alphabet $\{0, 1\}$ which contain an even number of 0.
    - $L = \{x \in V^+ : |x| \text{ is even}\}$.
    - $\{a^n b^n | n \in N\}$.
  - formal (mathematical):
    - an inductive description
    - a generative description (using grammars)
    - a description using recognizers (finite automata, pushdown automata, etc.)
    - an algebraic description (regular expressions)

Language Operations

- Set operations (union, intersection)
- Product of languages: $L_1 \cdot L_2 = \{u \cdot v | u \in L_1, v \in L_2\}$

  Example:

  $L_1 = \{a^n, n \geq 1\}$, $L_2 = \{b^n, n \geq 1\}$

  $L_1 \cdot L_2 = \{a^n b^m, n \geq 1, m \geq 1\}$

- Iteration (Kleene product): $L^* = \bigcup_{n \geq 0} L^n$, where:
  - $L^0 = \{\epsilon\}$
  - $L^{n+1} = L^n \cdot L$

  Example:

  $L = \{a\}$, $L^0 = \{\epsilon\}, L^1 = L, L^2 = \{aa\}, \ldots, L^n = \{a^n\}$

  $L^* = \{a^n, n \geq 0\}$

Formal Languages, Automata and Compilers

Lecture 1

Grammars

Definition 1

A grammar is a system $G = (N, T, S, P)$*, where:*

- N *and* T *are disjoint alphabets:*
    - N *is the set of non-terminals*
    - T *is the set of terminals*
- $S \in N$ *is the start symbol (initial non-terminal)*
- P *is a finite set of rules (productions):* $x \to y$*, where* $x, y \in (N \cup T)^*$ *and* x *contains at least a non-terminal.*

Derivation

Definition 2

Let $G = (N, T, S, P)$ be a grammar $u, v \in (N \cup T)^*$.

v is directly derived (in one step) from u by application of rule $x \rightarrow y$, (written as $u \Rightarrow v$), if $\exists p, q \in (N \cup T)^*$ such that $u = pxq$ and $v = pyq$.

Derivation

Definition 2

Let $G = (N, T, S, P)$ be a grammar $u, v \in (N \cup T)^*$.
v is directly derived (in one step) from u by application of rule $x \to y$, (written as $u \Rightarrow v$), if $\exists p, q \in (N \cup T)^*$ such that $u = pxq$ and $v = pyq$.

- If $u_1 \Rightarrow u_2 \ldots \Rightarrow u_n, n > 1$, we say that u_n is derived from u_1 in grammar G and write: $u_1 \Rightarrow^+ u_n$.

Derivation

Definition 2

Let $G = (N, T, S, P)$ be a grammar $u, v \in (N \cup T)^*$.

v is directly derived (in one step) from u by application of rule $x \to y$, (written as $u \Rightarrow v$), if $\exists p, q \in (N \cup T)^*$ such that $u = pxq$ and $v = pyq$.

- If $u_1 \Rightarrow u_2 \ldots \Rightarrow u_n, n > 1$, we say that u_n is derived from u_1 in grammar G and write: $u_1 \Rightarrow^+ u_n$.
- We write $u \Rightarrow^* v$ if $u \Rightarrow^+ v$ or $u = v$.

Generated Language

Definition 3

The language generated by grammar G is:

$$L(G) = \{w \in T^* | S \Rightarrow^+ w\}$$

Generated Language

Definition 3

The language generated by grammar G is:

$$L(G) = \{w \in T^* | S \Rightarrow^+ w\}$$

Definition 4

Two grammars G_1 and G_2 are equivalent if $L(G_1) = L(G_2)$.

Example

- $G = (N, T, S, P)$, $N = \{S, S_1, X\}$, $T = \{a, b, c\}$, P consists of:
  1. $S \rightarrow abc$
  2. $S \rightarrow aS_1Xc$
  3. $S_1 \rightarrow abc$
  4. $cX \rightarrow Xc$
  5. $bX \rightarrow bb$
- $L(G) = \{abc, a^2b^2c^2\}$
- What is the equivalent grammar with only one non-terminal?

Example

- $L = \{a^n b^n | n \geq 1\}$
- Inductive definition:
  - $ab \in L$
  - if $X \in L$, then $aXb \in L$
  - No other word belongs to L

Example

- $L = \{a^n b^n | n \geq 1\}$
- Inductive definition:
  - $ab \in L$
  - if $X \in L$, then $aXb \in L$
  - No other word belongs to L
- Generative definition:
  - $G = (\{X\}, \{a, b\}, X, P)$, where $P = \{X \to aXb, X \to ab\}$
  - Derivation of the word $a^3 b^3$ from the start symbol:

    $X \Rightarrow aXb \Rightarrow a(aXb)b \Rightarrow aa(ab)bb$

Example

- $L = \{a^n b^n c^n | n \geq 1\}$
- $= (N, T, S, P)$, $N = \{S, X\}$, $T = \{a, b, c\}$, P contains the rules:
  1. $S \rightarrow abc$
  2. $S \rightarrow aSXc$
  3. $cX \rightarrow Xc$
  4. $bX \rightarrow bb$
- Derivation of word $a^3 b^3 c^3$:

  $S \Rightarrow^{(2)} a\underline{S}Xc \Rightarrow^{(2)} aa\underline{S}XcXc \Rightarrow^{(1)} aaab\underline{cX}cXc \Rightarrow^{(3)}$
  $aaa\underline{bX}ccXc \Rightarrow^{(4)} aaabbc\underline{cX}c \Rightarrow^{(3)} aaabb\underline{cX}cc \Rightarrow^{(3)}$
  $aaab\underline{bX}ccc \Rightarrow^{(4)} aaabbbccc = a^3 b^3 c^3$

Formal Languages, Automata and Compilers
Lecture 1

Chomsky Hierarchy

1. **Type 0 grammars (unrestricted grammars)**

   There are no restrictions on the rules;

Chomsky Hierarchy

1. **Type 0 grammars (unrestricted grammars)**

   There are no restrictions on the rules;

2. **Type 1 grammars (context-sensitive grammars)**

   rules of the form $pxq \to pyq$ where $x \in N$, $y \neq \epsilon$, $p, q \in (N \cup T)^*$, $S \to \epsilon$ (if this rule exists, S must not appear on the right side of the rules)

Chomsky Hierarchy

1. **Type 0 grammars (unrestricted grammars)**
   There are no restrictions on the rules;
2. **Type 1 grammars (context-sensitive grammars)**
   rules of the form $pxq \rightarrow pyq$ where $x \in N$, $y \neq \epsilon$, $p, q \in (N \cup T)^*$, $S \rightarrow \epsilon$ (if this rule exists, S must not appear on the right side of the rules)
3. **Type 2 grammars (context-free grammars)**
   rules of the form $A \rightarrow y$ where $A \in N$ and $y \in (N \cup T)^*$;

Chomsky Hierarchy

1. **Type 0 grammars (unrestricted grammars)**
   There are no restrictions on the rules;
2. **Type 1 grammars (context-sensitive grammars)**
   rules of the form $pxq \rightarrow pyq$ where $x \in N$, $y \neq \epsilon$, $p, q \in (N \cup T)^*$,
   $S \rightarrow \epsilon$ (if this rule exists, S must not appear on the right side of the rules)
3. **Type 2 grammars (context-free grammars)**
   rules of the form $A \rightarrow y$ where $A \in N$ and $y \in (N \cup T)^*$;
4. **Type 3 grammars (regular)**
   rules of the form $A \rightarrow u$ or $A \rightarrow uB$ where $A, B \in N$ and $u \in T^*$.

Examples

Type 1: $pxq \to pyq$ where $x \in N$, $y \neq \epsilon$, $p, q \in (N \cup T)^*$, $S \to \epsilon$

- $G = (N, T, S, P)$, $N = \{S, A, B\}$, $T = \{a, b, c\}$, P:

  (1)$S \to aaAc$

  (2)$aAc \to aAbBc$

  (3)$bB \to bBc$

  (4)$Bc \to Abc$

  (5)$A \to a$

  Type 1 grammar

- $G = (N, T, S, P)$, $N = \{S, X\}$, $T = \{a, b, c\}$, P:

  (1)$S \to abc$

  (2)$S \to aSXc$

  (3)$cX \to Xc$ (it is not a type 1 rule!, the grammar is a type 0 grammar)

  (4)$bX \to bb$

Examples

Type 2: $A \rightarrow y$ where $A \in N$ and $y \in (N \cup T)^*$

Type 3: $A \rightarrow u$ or $A \rightarrow uB$ where $A, B \in N$ and $u \in T^*$.

- G:

  $(1) x \rightarrow ax$

  $(1) x \rightarrow xb$

  $(2) x \rightarrow \epsilon$

Examples

Type 2: $A \rightarrow y$ where $A \in N$ and $y \in (N \cup T)^*$

Type 3: $A \rightarrow u$ or $A \rightarrow uB$ where $A, B \in N$ and $u \in T^*$.

- G:

  $(1) x \rightarrow ax$

  $(1) x \rightarrow xb$

  $(2) x \rightarrow \epsilon$

  (Type 2)

- G:

  $(1) x \rightarrow ax$

  $(2) x \rightarrow bx$

  $(3) x \rightarrow \epsilon$

Examples

Type 2: $A \to y$ where $A \in N$ and $y \in (N \cup T)^*$

Type 3: $A \to u$ or $A \to uB$ where $A, B \in N$ and $u \in T^*$.

- G:

  $(1) x \to ax$

  $(1) x \to xb$

  $(2) x \to \epsilon$

  (Type 2)

- G:

  $(1) x \to ax$

  $(2) x \to bx$

  $(3) x \to \epsilon$

  (Type 3)

Classification of languages

- A language L is of type j if there exists a grammar G of type j such that $L(G) = L$, where $j \in \{0, 1, 2, 3\}$.
- $\mathcal{L}_j$ denotes the set of all languages of type j, where $j \in \{0, 1, 2, 3\}$.
- It holds that: $\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$
- The inclusions are strict:
  - any language of type $j + 1$ is also of type $j \in \{0, 1, 2\}$
  - there exists languages of type j that are not of type $j + 1$, $j \in \{0, 1, 2\}$

Formal Languages, Automata and Compilers

Lecture 1

- A grammar $G = (N, T, S, P)$ is a type 3 grammar if all the rules are of the form: $A \to u$ or $A \to uB$ where $A, B \in N$ and $u \in T^*$.
- Type 3 grammars are also called regular grammars;
- Example: Let
  $G = (\{A, B\}, \{a, b\}, A, \{A \to aA, A \to B, B \to bB, B \to \epsilon\})$

- A grammar $G = (N, T, S, P)$ is a type 3 grammar if all the rules are of the form: $A \to u$ or $A \to uB$ where $A, B \in N$ and $u \in T^*$.
- Type 3 grammars are also called regular grammars;
- Example: Let
  $G = (\{A, B\}, \{a, b\}, A, \{A \to aA, A \to B, B \to bB, B \to \epsilon\})$
  $L(G) = \{a^n b^m, n, m \geq 0\}$

Examples

- $G = (\{D\}, \{0, 1, ..., 9\}, D, P)$
  where P is:
  $D \rightarrow 0D|1D|2D|\ldots|9D$
  $D \rightarrow 0|1|\ldots|9$
- $G = (\{A, B\}, \{l, d\}, A, P)$ where P is:
  $A \rightarrow lB,\; B \rightarrow lB|dB|\epsilon$ (l = latter, d = figure)

Examples

- $G = (\{D\}, \{0, 1, ..., 9\}, D, P)$
  where P is:
  $D \rightarrow 0D|1D|2D|\ldots|9D$
  $D \rightarrow 0|1|\ldots|9$
- $G = (\{A, B\}, \{l, d\}, A, P)$ where P is:
  $A \rightarrow lB,\ \ B \rightarrow lB|dB|\epsilon$ (l = latter, d = figure)
  $L(G)$: the set of identifiers

The Normal Form for Regular Grammars

- A regular grammar G is in normal form if all the rules are of the form $A \rightarrow a$ or $A \rightarrow aB$, where $a \in T$, and, if necessary $S \rightarrow \epsilon$ (in this case S does not appear in the right side of the rules).
- For any type 3 grammar there exists an equivalent grammar in normal form.

The Normal Form for Regular Grammars

- The equivalent grammar in normal form can be obtained as follows:
  - Remove (replace) the rules of the form $A \to B$ (unit rules) and $A \to \epsilon$ (ϵ-rules), except, if necessary, $S \to \epsilon$.
  - Any rule $A \to a_1 a_2 \ldots a_n$ is replaced by: $A \to a_1 B_1$, $B_1 \to a_2 B_2, \ldots$, $B_{n-2} \to a_{n-1} B_{n-1}$, $B_{n-1} \to a_n$, $n > 1$, $B_1, \ldots, B_{n-1}$ are new non-terminals.
  - Any rule $A \to a_1 a_2 \ldots a_n B$ is replaced by: $A \to a_1 B_1$, $B_1 \to a_2 B_2, \ldots$, $B_{n-2} \to a_{n-1} B_{n-1}$, $B_{n-1} \to a_n B$, $n > 1$, $B_1, \ldots, B_{n-1}$ are new non-terminals
  - The new grammar generates the same language

Formal Languages, Automata and Compilers
Lecture 1

Let L, L_1, L_2 be regular languages.

Then, the following languages are also regular languages:

- $L_1 \cup L_2$
- $L_1 \cdot L_2$
- L^*
- $L_1 \cap L_2$
- $L_1 \setminus L_2$

Closure under union

Let L, L_1, L_2 be regular languages.

Let $G_1 = (N_1, T_1, S_1, P_1)$ and $G_2 = (N_2, T_2, S_2, P_2)$ be type 3 grammars $L_1 = L(G_1)$, $L_2 = L(G_2)$.

Assume $N_1 \cap N2 = \emptyset$.

Closure under union: it can be proved that $L_1 \cup L_2 \in \mathcal{L}_3$:

Grammar $G = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, S, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\})$ is of type 3 and generates $L_1 \cup L_2$

Closure under product

Let L_1, L_2 be regular languages.

Let $G_1 = (N_1, T_1, S_1, P_1)$ and $G_2 = (N_2, T_2, S_2, P_2)$ be type 3 languages with $L_1 = L(G_1)$, $L_2 = L(G_2)$.

Assume $N_1 \cap N2 = \emptyset$.

Grammar $G = (N_1 \cup N_2, T_1 \cup T_2, S_1, P)$ where P contains:

- the rules of the form $A \to uB$ from P_1 (where $B \in N_1$)
- rules of the form $A \to uS_2$ for every rule of the form $A \to u$ from P_1 (with $u \in T_1^*$)
- all the rules from P_2

is of type 3 and generates the language $L_1 L_2$.

Example

$L = \{uc^n, u \in \{a, b\}^+, n \geq 2\}$

$L = L_1 \cdot L_2$, where: $L_1 = \{a, b\}^+$, $L_2 = \{c^n, n \geq 2\}$

$G1$:

1. $S_1 \rightarrow aS_1$
2. $S_1 \rightarrow bS_1$
3. $S_1 \rightarrow a$
4. $S_1 \rightarrow b$

$G2$:

1. $S_2 \rightarrow cS_2$
2. $S_2 \rightarrow cc$

$G = (\{S_1, S_2\}, \{a, b, c\}, S_1, P)$,

P :

1. $S_1 \rightarrow aS_1$
2. $S_1 \rightarrow bS_1$
3. $S_1 \rightarrow aS_2$
4. $S_1 \rightarrow bS_2$
5. $S_2 \rightarrow cS_2$
6. $S_2 \rightarrow cc$

Closure under iteration

Let L be a regular language

Let $G = (N, T, S, P)$ of type 3, which generates L ($L = L(G)$).

Assume S does not appear in the right side of any rule

Grammar $G' = (N, T, S, P')$ where P' contains

- the rules $A \to uB$ from P (where $B \in N$)
- rules $A \to uS$, for any rule $A \to u$ from P (where $u \in T^*$), different from $S \to \epsilon$
- the rule $S \to \epsilon$

is of type 3 and generates L^*

Example

$L = \{a^{n_1}b^{m_1}a^{n_2}b^{m_2}\dots a^{n_k}b^{m_k}, n_i, m_i \geq 1 \forall i \in \{1, k\}, k \geq 0\}$

$L = \{a^n b^m, n \geq 1, m \geq 1\}^*$

G :

1. $S \rightarrow x$
2. $x \rightarrow ax$
3. $x \rightarrow ay$
4. $y \rightarrow by$
5. $y \rightarrow b$

G' :

1. $S \rightarrow x$
2. $x \rightarrow ax$
3. $x \rightarrow ay$
4. $y \rightarrow by$
5. $y \rightarrow bS$
6. $S \rightarrow \epsilon$

Closure under intersection

Let L_1, L_2 be regular languages.

Let $G_1 = (N_1, T_1, S_1, P_1)$ and $G_2 = (N_2, T_2, S_2, P_2)$ type 3 grammars, **in normal form**, such that $L_1 = L(G_1)$, $L_2 = L(G_2)$.

Grammar $G = (N_1 \times N_2, T_1 \cap T_2, (S_1, S_2), P)$, with P:

- $(S_1, S_2) \rightarrow \epsilon$, if $S_1 \rightarrow \epsilon \in P_1$ and $S_2 \rightarrow \epsilon \in P_2$
- $(A_1, B_1) \rightarrow a(A_2, B_2)$, if $A_1 \rightarrow aA_2 \in P_1$ and $B_1 \rightarrow aB_2 \in P_2$
- $(A_1, A_2) \rightarrow a$, if $A_1 \rightarrow a \in P_1$ and $A_2 \rightarrow a \in P_2$

is a type 3 grammar and generates $L_1 \cap L_2$

Example

$L(G1) = \{w \in \{0,1\}^*,$ w contains at least a symbol '0'$\}$,

$L(G2) = \{w \in \{0,1\}^*,$ w ends with '1'$\}$

$L(G) = \{w \in \{0,1\}^*,$ w contains at least a symbol '0' and ends with '1'$\}$

$G1$:

1. $S_1 \rightarrow 1S_1$
2. $S_1 \rightarrow 0A$
3. $S_1 \rightarrow 0$
4. $A \rightarrow 1A$
5. $A \rightarrow 0A$
6. $A \rightarrow 1$
7. $A \rightarrow 0$

$G2$:

1. $S_2 \rightarrow 0S_2$
2. $S_2 \rightarrow 1S_2$
3. $S_2 \rightarrow 1$

G

1. $(S_1, S2) \rightarrow 1(S_1, S_2)$
2. $(A, S_2) \rightarrow 1(A, S_2)$
3. $(S_1, S_2) \rightarrow 0(A, S_2)$
4. $(A, S_2) \rightarrow 0(A, S_2)$
5. $(A, S_2) \rightarrow 1$

Example

$L(G1) = \{w \in \{0,1\}^*,$ w contains at least a symbol '0'$\}$,

$L(G2) = \{w \in \{0,1\}^*,$ w ends with '1'$\}$

$L(G) = \{w \in \{0,1\}^*,$ w contains at least a symbol '0' and ends with '1'$\}$

$G1$:

1. $S_1 \rightarrow 1S_1$
2. $S_1 \rightarrow 0A$
3. $S_1 \rightarrow 0$
4. $A \rightarrow 1A$
5. $A \rightarrow 0A$
6. $A \rightarrow 1$
7. $A \rightarrow 0$

$G2$:

1. $S_2 \rightarrow 0S_2$
2. $S_2 \rightarrow 1S_2$
3. $S_2 \rightarrow 1$

G

1. $S \rightarrow 1S$
2. $X \rightarrow 1X$
3. $S \rightarrow 0X$
4. $X \rightarrow 0X$
5. $X \rightarrow 1$