مفردات مادة نظرية الحوسبة
Theory of Computation

الهدف من المقرر :

- الإلمام بالمفاهيم الأساسية في نظرية الأتومات.
- التمييز بين الأنواع المختلفة للغات و الآلات التي تتعرف عليها كل لغة.
- التعرف على الأتومات و التعابير المنتظمة كنماذج رياضية تساعد في تعريف لغات البرمجة و اللغات الصورية.
- القدرة على استخدام مفاهيم نظرية الأتومات كأدوات أساسية في بناء العديد من الأنظمة المعلوماتية كالمترجمات.

الفصل الأول: مفاهيم أساسية في نظرية الأتومات Basic Concepts of Automata Theory
- الأبجديات Alphabets
- السلاسل Strings
- اللغات Languages
- العمليات على السلاسل Operations on Strings

الفصل الثاني:الأتومات المنتهي Finite Automata
- تعريف الأتومات Definition of Automata
- الأتومات المنتهي الحتمي Deterministic Finite Automata(DFA)
- الأتومات المنتهي اللاحتمي Non-Deterministic Finite Automata(NFA)
- التكافؤ بين الأتومات المنتهي الحتمي و الأتومات المنتهي اللاحتمي
  Equivalence of DFA and NFA
- الأتومات المنتهي اللاحتمي مع ε تحرك
  Non- Deterministic Finite Automata with ε Transition (ε-NFA)
- التكافؤ بين الأتومات المنتهي اللاحتمي مع ε تحرك و الأتومات المنتهي اللاحتمي
  Equivalence of NFA and ε-NFA

الفصل الثالث: التعابير المنتظمة و اللغات Regular Expressions and Languages
- التعابير المنتظمة Regular Expressions
- الأتومات المننتهي و التعابير المنتظمة Finite Automata and Regular Expressions
- خواص اللغات المنتظمة Properties of Regular Languages
- توطئة الضخ Pumping Lemma
- الأتومات المنتهي الحتمي الأصغري Minimal Finite Automata

الفصل الرابع: القواعد خارج السياق Context-Free Grammars
- شجرة الاشتقاق Derivation Tree
- اختصار القواعد خارج السياق Simplification of CFG
- صيغة تشومسكي المعيارية Chomsky Normal Form

الفصل الخامس: الأتومات بمكدس Push Down Automata
- تعريف الأتومات بمكدس Definition of PDA
- العلاقة بين الأتومات بمكدس و اللغات خارج السياق
  Relationship Between PDA and Context Free Languages
- خواص اللغات خارج السياق Properties of Context Free Languages

الفصل السادس: آلة تورينغ Turing Machine
- تعريف آلة تورينغ Definition of Turing Machine

أ. م. د. ريما القمحة

نظرية الأوتومات واللغات الصورية

Automata theory and formal languages

تعتبر هذه المادة في مقدمة المواد التي يجب أن تُدرس عند تعلم لغات الحاسوب وتسمى (نظرية الحوسبة) أو (نظرية الأوتومات) وتدخل بشكل أساسي بمفاهيم عديدة من ضمنها التعرف على اللغات الطبيعية وإنشاء المترجمات وتدخل أيضاً في مجال معالجة الدارات الالكترونية. وتعتبر نظرية الحوسبة هي أحد فروع علم الحاسوب التي تدرس فيما إذا كانت مسألة معينة قابلة للحل حاسوبياً وذلك باستخدام نماذج مختلفة.

وسندرس في هذه المادة المفاهيم التالية:

1) التعابير المنتظمة (Regular Expressions) :

وهي إحدى المفاهيم التي تغطيها هذه المادة وهي شائعة الاستخدام ولها تطبيقات عديدة وهي عبارة عن سلسلة نصّية تستخدم في وصف العديد من الأنماط الشائعة والتعرف عليها **مثل:** عنوان البريد الالكتروني وعناوين مواقع الانترنت، كما تستخدم في معالجة النصوص وفي صناعة المترجمات (compilers) حيث قامت ويكيبيديا (Wikipedia) بتصحيح أكثر من 250 ألف خطأ املائي في مقالاتها باستخدام تطبيق يعتمد على التعابير المنتظمة حيث يكون التعبير المنتظم هام للتعرف على أنماط معينة من السلاسل.

*مثال للتوضيح:

لجعل الآلة تتعرف على بريد الكتروني من نص ما، نستخدم التعابير المنتظمة للبريد الالكتروني ولتكن بالشكل:

$$(\text{charcter})^{+}(\text{Digits, charcter})^{*}@\,(\text{charcter})^{+}.(\text{charcter})^{+}$$

محارف   محارف أو أرقام   محارف   محارف

ترميز: +: تعني أنه يمكن تكرار المحرف مرة أو أكثر من مرة (أي محرف واحد على الأقل) .

*: تعني أنه يمكن تكرار المحرف صفر مرة أو أكثر (أي ولا محرف أو اكثر).

وهو يعتبر تعبير منتظم للبريد الالكتروني.

2) الفواعد النحوية خارج السياق (Context free languages):

اختصاراً CFL ، وخارج السياق تعني أن الكلمة فيها لها معنى واحد فقط اينما وكيفما وردت في الكلام .

*مثال عليها: لغات البرمجة. وتستخدم هذه القواعد بشكل أساسي في توصيف جميع صيغ اللغات البرمجية وكذلك اللغات الطبيعية وتستخدم أيضاً في البرامج التي تقوم بترجمة النصوص ومعالجة اللغات الطبيعية حيثُ تعتبر اللغات الطبيعية هي لغات بسياق ((يختلف معنى الكلمة حسب موقعها في الجملة)).

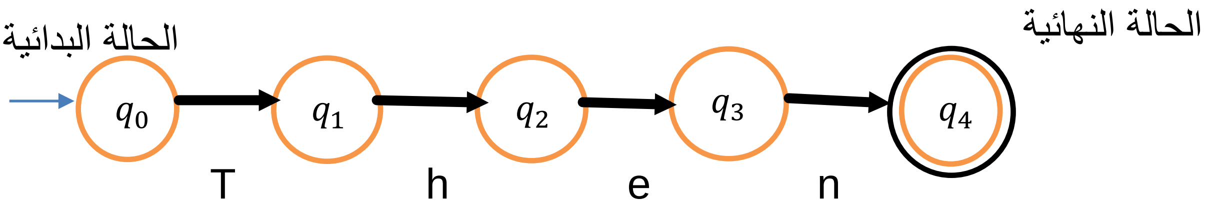
3) الأوتومات المنتهي (Finite Automate):

هو نموذج رياضي على شكل بيان (Graph) أو جدول (Table) ويحتوي على عدد منتهي من الحالات (States) ومن هنا جاءت تسمية هذا النموذج بالمنتهي(لأن عدد حالاته منته).

وهذه الحالات تتغير كرد على المدخلات (Inputs) والقواعد التي تخبر بكيفية تبدل الحالة كرد على المدخل وتسمى الانتقالات (Transitions).

*مثال للتوضيح:

للتعرف على كلمة (Then) من قبل الآلة(الأوتوم) يجب أولاً التعرف على حرف T كحالة بدء ومن ثم التعرف على حرف h ومن التعرف على حرف e و كحالة نهائية يجب التعرف على الحرف n. ونعّبر عن ذلك بالرسم:



q_i: هي اسماء الحالات. ((أوتومات للتعرف على كلمة Then)) .

بما أننا استطعنا تشكيل أوتو فيمكننا تحويله لبرنامج .

سبب دراسة الأوتومات :

- يستخدم في تصميم الدارات الالكترونية وبروتوكولات الاتصال ويستخدم أيضاً في برامج معالجة النصوص للبحث عن كلمات في ملف أو عبر الويب .
- تلعب أيضاً الأوتومات دوراً هاماً في صناعة المترجمات (Compilers) على مستوى المفردات كمحلل لفظي (Lexical Analyzer).
- تكون وظيفة الأوتومات هو التعرف على لغة ما.
- أثناء إيجاد حلول لمسائل حقيقية غالباً ما نواجه مسائل غير قابلة للحوسبة (تسمى Np) وقد نواجه مسائل صعبة الحل.
- يوجد آلة تسمى آلة تورينغ(Turing machine):

كُل ما يُحل بآلة تورينغ يكون قابل للحساب ويمكن إيجاد خوارزمية له و كُل مسألة لا تُحل بآلة تورينغ تكون غير من الممكن حوسبتها(لا يوجد لها حل حاسوبي).

وبالتالي نجد أن هذه المادة تعطيك الأدوات لتحديد فيما اذا كانت المسألة غير قابلة للحوسبة فتتجنبها أو إذا كانت صعبة الحل فتقوم بإيجاد مقاربة للحل .

و تسمى هذه المادة " الأوتومات واللغات الصورية " ونعني باللغات الصورية أي تشمل جميع اللغات الموجودة في العالم بما في ذلك اللغات الطبيعية كاللغة العربية والانكليزية أو اللغات البرمجية.

*ملاحظة: مفرد كلمة : (Automata) هو: (Automaton) لكن يكون استخدام المفرد قليل.

تعريف اللغات الصورية: هي اللغات التي يمكن تعريفها بواسطة عدد من القواعد.

مفاهيم أساسية:

◈ تعريف الرمز ($Symbol$):

هو كائن غير قابل للتجزئة ، مثل الحروف الإنكليزية $\{a, b, c, \dots, z\}$ والأرقام العربية $\{1,2,3, \dots ,9\}$ والحروف العربية $\{$أ, ب , ... , ي$\}$.

ملاحظة:
الأرقام العربية هي (1,2,3, ... ,9) بينما الأرقام (١,٢,٣,...,٩) تدعى الأرقام الهندية

ملاحظة (أو مثال) :

ab ليست رمزاً لأنه قابل للتجزئة إلى جزأين هما a و b .

◈ تعريف الأبجدية ($Alphabet$):

هي مجموعة منتهية وغير خالية من الرموز وتتميز بأنه لا يمكن توليد أي رمز منها بواسطة بقية الرموز ونرمز عادةً للأبجدية بالرمز Σ (سيغما).

أمثلة :

– إن $\{\ \} = \phi$ ليست أبجدية لأنها خالية (لا تحقق التعريف) .

– إن الأمثلة التالية هي أبجديات:

* ابجدية من حرفين : $\Sigma = \{a, b\}$.
* ابجدية اللغة الانكليزية (احرفها الصغيرة فقط) : $\Sigma = \{a, b, \dots , z\}$.
* أبجدية نظام العد الثنائي $\Sigma = \{0,1\}$.
* أبجدية اللغة العربية $\Sigma = \{$أ, ب , ... , ي$\}$.
* أبجدية نظام العد العشري $\Sigma = \{1,2,3, \dots ,9\}$.

- إن مجموعة الأعداد الطبيعية $\mathbb{N}$ ليست أبجدية لأنها مجموعة غير منتهية.

- هل المجموعة $\Sigma = \{0,1,01\}$ أبجدية ؟؟

الجواب: ليست أبجدية لأنها تحوي عنصر قابل للتجزئة (01) وبالتالي هو ليس رمزاً.

◈ تعريف السلسلة ($String$):

هي تسلسل(تعاقب/ تتالي) عدد منتهي من الرموز المأخوذة من أبجدية ما دون وجود فراغات بينها.

مثال :

لتكن لدينا الأبجدية $\Sigma = \{a, b\}$ عندئذ يكون:

aba هي سلسلة مشكلة من الأبجدية Σ و bababbbaa هي سلسلة أيضاً من نفس الأبجدية.

- نرمز للسلاسل عادةً برموز صغيرة ($x, y, z, w, r, v, \ldots$)

مثال :

نرمز للسلسة $\boldsymbol{aa}$ بالرمز $\boldsymbol{x}$: $x = aa$ ، نرمز للسلسلة bbb بالرمز w : $w = bbb$.

- قد تكون السلسلة مكونة من رمز واحد فقط مثل: $v = a$ ، $r = b$

◈ **<u>طول السلسلة</u>** <u>$(length\ of\ String)$:</u>

هو عدد الرموز المشكلة للسلسلة ونرمز لطول السلسلة w بـ $|w|$.

مثال (١) :

إذا كانت لدينا السلسلة $v = 0111$ هي سلسلة مشكلة من الأبجدية $\Sigma = \{0,1\}$ عندئذٍ يكون $|v| = 4$ لان عدد الرموز المكونة للسلسلة تساوي 4.

مثال (٢):

لتكن لدينا الأبجدية $\Sigma = \{2,7,8\}$ ولتكن السلسلة $x = 77827$ سلسلة مولدة من الأبجدية Σ فيكون طولها $|x| = 5$.

◈ **<u>السلسلة الفارغة</u>** <u>$(Empty\ String)$:</u>

هي سلسلة لا تحوي أي رموز وطولها يساوي الصفر ونرمز لها بـ (ε) وبالتالي $|\varepsilon| = 0$ وتفيدنا بالانتقال من حالة إلى حالة .

مثال للتوضيح: $A \xrightarrow{\varepsilon} B$

السلسلة الفارغة ε هنا للانتقال من الحالة A إلى الحالة B دون قراءة أي رمز .

أما في الحالة: $A \xrightarrow{0} B$

يجب قراءة الصفر للانتقال من الحالة A إلى الحالة B .

ملاحظة: ε **هي سلسلة معرفة على أي أبجدية**

* العمليات على السلاسل *

<u>التعاقب بين السلاسل</u> <u>$(Concatenation)$:</u>

إن تعاقب سلسلتين x و y هو سلسلة مشكلة من توضع رموز السلسلة الأولى x متبوعة مباشرةً برموز السلسلة الثانية y ونرمز لعملية التعاقب أحياناً بـ $(.)$

مثال : لتكن لدينا الأبجدية $\Sigma = \{a, b\}$ ولتكن السلسلة $x = aba$ والسلسلة $y = bbb$ فيكون:

$x.y = xy = ababbb$ (**نضع رموز** x **ثم رموز** y)

$y.x = yx = bbbaba$ (**نضع رموز** y **ثم رموز** x)

ملاحظات

❶ تعاقب السلاسل هي عملية غير تبديلية أي: $xy \neq yx$

❷ السلسلة الفارغة ε هي عنصر حيادي بالنسبة لعملية تعاقب السلاسل أي:

$$x\varepsilon = \varepsilon x = x$$
$$aba\,.\,\varepsilon = \varepsilon.\,aba = aba$$

❸ تعاقب السلاسل عملية تجميعية: $x.(y.z) = (x.y).z$

بادئة سلسلة $(Prefix)$:

تكون السلسلة u هي بادئة السلسلة v إذا وجدت سلسلة w بحيث يتحقق $v = uw$.

مثال :

لتكن السلسلة $v = ababbbaa$ عندئذٍ تكون مجموعة بادئات هذه السلسلة:

$$\{\varepsilon, a, ab, aba, abab, ababb, ababbb, ababbba, ababbbaa\}$$

الشرح :

أول عنصر في مجموعة البادئات دوماً يكون السلسلة الفارغة εثم نضع الرمز الأول من السلسلة a ثم الرمز الأول والثاني من السلسلة ab وهكذا .. مع المحافظة على ترتيب الرموز في السلسلة.

لاحقة سلسلة $(Suffix)$:

تكون السلسلة u هي لاحقة السلسلة v إذا وجدت سلسلة w بحيث يتحقق $v = wu$.

مثال :

لتكن السلسلة $v = ababbbaa$ عندئذٍ تكون مجموعة لاحقات هذه السلسلة :

$$\{\varepsilon, a, aa, baa, bbaa, bbbaa, abbbaa, babbbaa, ababbbaa\}$$

الشرح :

أول عنصر في مجموعة اللاحقات دوماً يكون السلسلة الفارغة εثم نضع الرمز الأخير من السلسلة a ثم الرمز الأخير والذي يسبقه aa ومن ثم آخر ثلاثة رموز في السلسلة baaوهكذا .. مع المحافظة على ترتيب الرموز في السلسلة.

السلسلة الجزئية $(Sub\ String)$:

تكون السلسلة u سلسلة جزئية من السلسلة v إذا كانت كل رموز السلسلة u موجودة في السلسلة v مع المحافظة على ترتيب الرموز.

مثال : لتكن لدينا السلسلة $v = ababbbaa$ ، إن كل من السلاسل التالية هي سلاسل جزئية من السلسلة v : $babbb\,, bab\,, abb\,, ab\,, baa\,, \ldots$

- هل السلسلة $aabb$ جزئية من السلسلة v السابقة ؟؟

الجواب: ليست سلسلة جزئية لأنها لم تحافظ على ترتيب الرموز في السلسلة.

ملاحظة: البادئات واللاحقات هي سلاسل جزئية

قوة السلسلة $(Power\ Of\ String)$:

قوة سلسلة w من الدرجة n هي عبارة عن تعاقب هذه السلسلة n مرة.

$$w^n = \underbrace{w.w.w\ \dots .w}_{n\ \text{مرة}}$$

مثال :

- لتكن $x = abb = ab^2$

$$x^1 = abb$$
$$x^2 = xx = abbabb$$
$$x^3 = xx^2 = x^2x = xxx = abbabbabb$$

- ولتكن $y = a^3b^4a$

$$y^1 = a^3b^4a$$
$$y^2 = yy = a^3b^4aa^3b^4a = a^3b^4a^4b^4a = aaabbbbaaaabbbba$$
$$y^3 = yy^2 = y^2y = a^3b^4a^4b^4a^4b^4a$$

قوة أبجدية $(Power\ Of\ Alphabet)$:

هي مجموعة السلاسل المولدة من الأبجدية Σ ذات الطول n ونرمز لها بـ Σ^n .

مثال : لتكن $\Sigma = \{0,1\}$ أبجدية فيكون:

$$\Sigma^0 = \{\varepsilon\}$$
$$\Sigma^1 = \{0,1\} = \Sigma$$
$$\Sigma^2 = \Sigma.\Sigma = \{0,1\}\{0,1\} = \{00,01,10,11\}$$
$$\Sigma^3 = \Sigma\Sigma\Sigma = \Sigma^2\Sigma = \Sigma\Sigma^2 = \{0,1\}\{00,01,10,11\}$$
$$\Longrightarrow \Sigma^3 = \{000,001,010,011,100,101,110,111\}$$

نعرف Σ^* على أنها مجموعة كل السلاسل التي يمكن توليدها من رموز الأبجدية Σ وهي مجموعة غير منتهية أي: $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \dots$

مـثـا ل : لتكن $\Sigma = \{0,1\}$ أبجدية فتكون مجموعة كل السلاسل التي يمكن توليدها من الأبجدية هي:

$$\Sigma^* = \{\varepsilon, 0,1,00,01,10,11,000,010,100, \dots\}$$

هي مجموعة غير منتهية وبالتالي أياً كانت x سلسلة مكونة من الرموز 0 و 1 فإنها تنتمي إلى Σ^*

ملاحظة

أحيانا يلزمنا تعريف كل السلاسل المولدة من الأبجدية Σ ماعدا السلسلة الفارغة عندئذٍ نرمز لها بـ Σ^+ وهي مجموعة كل السلاسل المولدة على Σ ماعدا السلسلة التي تولد ε والتي تكون Σ^0 وعليه فإن:

$$\Sigma^+ = \Sigma^* - \Sigma^0 = \Sigma^* - \{\varepsilon\}$$
$$\{\varepsilon\} = \Sigma^* - \Sigma^+ \quad \text{أو} \quad \Sigma^+ \cup \{\varepsilon\} = \Sigma^*$$

اللغة $(Languages)$:

هي مجموعة السلاسل المختارة من المجموعة Σ^* والمولدة من الأبجدية Σ ونرمز لها بـ (L) .

أي أنه إذا كانت Σأبجدية وكانت (L) لغة مولدة من Σ فإن $L \subseteq \Sigma^*$ أي أن اللغة المعرفة على أبجدية ما لا تحوي بالضرورة كل السلاسل المولدة من هذه الأبجدية.

مثال :

إذا كانت $\Sigma = \{a, b\}$ أبجدية عندئذٍ Σ^* هي مجموعة كل السلاسل المولدة من الأبجدية Σ ولنعرف اللغة $L \subseteq \Sigma^*$ والتي هي لغة مولدة من الأبجدية Σوكل كلماتها تحوي التعاقب aa عندئذٍ:

$$L = \{aa, baa, aab, \dots\dots\dots , aaaabbaab, \dots\dots\}$$

وهي مجموعة غير منتهية أيضا.

ملاحظة: قد تكون اللغة منتهية وقد تكون غير منتهية

مثال :

لتكن لدينا الأبجدية $\Sigma = \{a, b, c\}$

١) ما هي اللغة المكونة من كل الكلمات المولدة من Σ والتي تحوي c مرتين على الأقل ؟

(نريد جميع السلاسل التي تحوي c مرتين أو أكثر ولا يهمنا الترتيب(او التعاقب))

$$L_1 = \left\{ cc, acc, abcc, cacb, cbc, ccc, \underbrace{\dots\dots}_{\text{تعني غير منتهية}} \right\}$$

نلاحظ أنه من السلاسل التي لاتنتمي: $\varepsilon \notin L_1$ و $bbabb \notin L_1$ و $abc \notin L_1$

بالتالي نجد أنه ليس كل السلاسل المولدة من الأبجدية Σ تنتمي للغة .

٢) ماهي اللغة المكونة من كل الكلمات المولدة من الأبجدية Σ والتي تحوي c مرتين فقط ؟

$$L_2 = \{cc, acc, abcc, cacb, \dots\dots\}$$

نلاحظ أن $L_2 \subseteq L_1$ وأن: $\varepsilon \notin L_2$ و $cccc \notin L_2$ و $acbcbac \notin L_2$ وهكذا

٣) لتكن L_3 هي اللغة المكونة من كل الكلمات المولدة من الأبجدية Σ والتي تبدأ بـ a فتكون:

$$L_3 = \{a, aba, abcc, accb, \dots\dots\}$$

نلاحظ أنه من السلاسل التي لاتنتمي: $\varepsilon \notin L_3$ و $babcbb \notin L_3$ و $cabcc \notin L_3$

العمليات على اللغات $(Set\ Operations)$

⌘**اتحاد لغتين** $(Union)$:

اتحاد لغتين L_1 و L_2 يمثل بالشكل التالي $L_1 \cup L_2$ وهو عبارة عن اللغة التي تحوي جميع السلاسل الموجودة في L_1 أو L_2 أو كلاهما:

$$L_1 \cup L_2 = \{x;\ x \in L_1\ or\ \ x \in L_2\}$$

⌘**تقاطع لغتين** (Intersection):

تقاطع لغتين L_1 و L_2 يمثل بالشكل التالي $L_1 \cap L_2$ وهو عبارة عن اللغة التي تحوي جميع السلاسل الموجودة في L_1 و L_2 معاً:

$$L_1 \cap L_2 = \{x;\ x \in L_1\ and\ \ x \in L_2\}$$

⌘**فرق لغتين** (Difference):

فرق لغتين L_1 و L_2 يمثل بالشكل التالي $L_1 - L_2$ وهو عبارة عن اللغة التي تحوي السلاسل الموجودة في L_1 وغير موجودة في L_2 :

$$L_1 - L_2 = \{x;\ x \in L_1\ and\ \ x \notin L_2\}$$

⌘**متمم لغة** (Complement):

متمم اللغة L المعرفة على الأبجدية Σ هي اللغة التي تحوي السلاسل الموجودة في Σ^* وغير موجودة في L ونرمز لمتمم اللغة L بـ $\bar{L}$:

$$\bar{L} = \{x;\ x \in \Sigma^*\ and\ \ x \notin L\} = \Sigma^* - L$$

⌘**تعاقب لغتين** (Concatenation):

تعاقب لغتين L_1 و L_2 هي اللغة المكونة من السلاسل المشكلة من تعاقب سلاسل L_1 متبوعة بسلاسل L_2 :

$$L_1.L_2 = \{xy\,; x \in L_1\,, y \in L_2\}$$

⌘**إغلاق لغة** (Kleen Star , Closure):

إذا كانت L لغة فإن إغلاق اللغة L نرمز له بـ L^* وهي اللغة المشكلة من مجموعة كل السلاسل الممكنة الناتجة عن تعاقب السلاسل في L :

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots\dots$$

فيكون:

$$L^+ = L^* - \{\varepsilon\}$$
$$L^0 = \{\varepsilon\}\,, L^1 = L$$
$$L^2 = L.L\,, L^3 = L^2.L = L.L^2 = L.L.L \quad , \quad \dots\dots$$

مثال : لتكن لدينا اللغتين L_1 و L_2 المولدتين من الأبجدية$\{a,b\}$ $\Sigma =$ حيث:

$$L_1 = \{a, ab, aaaa\}$$
$$L_2 = \{bb, ab\}$$

فيكون التقاطع:

$$L_1 \cap L_2 = \{ab\}$$

والاتحاد:

$$L_1 \cup L_2 = \{a, ab, aaaa, bb\}$$

والفرق:

$$L_1 - L_2 = \{a, aaaa\}$$
$$L_2 - L_1 = \{bb\}$$

ومتمم L_1 :

$$\overline{L_1} = \Sigma^* - \{a, ab, aaaa\} = \{\varepsilon, aa, aaa, bbb, abb, \dots\}$$

ومتمم L_2 :

$$\overline{L_2} = \Sigma^* - \{bb, ab\} = \{\varepsilon, a, aa, aba, \dots\}$$

تعاقب $L_1.L_2$:

$$L_1.L_2 = \{abb, aab, abbb, abab, aaaabb, aaaaab\}$$

تعاقب $L_2.L_1$:

$$L_2.L_1 = \{bba, bbab, bbaaaa, aba, abab, abaaaa\}$$

إغلاق L_2 (أي إيجاد L_2^*) :

$$L_2^0 = \{\varepsilon\}\,, \qquad\qquad L_2^1 = L_2 = \{bb, ab\}$$
$$L_2^2 = L_2.L_2 = \{bb, ab\}\{bb, ab\} = \{bbbb, bbab, abbb, abab\}$$

$$L_2^3 = L_2 . L_2^2 = \{bb, ab\}\{bbbb, bbab, abbb, abab\}$$
$$L_2^3 = \{b^6, b^4ab, b^2ab^3, b^2abab, ab^5, ab^3ab, abab^3, ababab\}$$
$$\vdots$$
$$L_2^* = \{\varepsilon, bb, ab, bbbb, bbab, abbb, \dots \dots\}$$
$$L_2^* = L_2^0 \cup L_2^1 \cup L_2^2 \cup \dots \dots$$

ملاحظة: السلسلة $ababab$ يمكن كتابتها بالشكل $(ab)^3$

ونلاحظ أن : $(ab)^3 \neq a^3b^3$

خاصة: $\phi^0 = \phi^* = \{\varepsilon\}$