IBB University
College of Sciences
DEPARTMENT OF
COMPUTERES & INFORMATION TECHNOLOGY

CS 351: Computational Theory

L 1: INTRODUCTION

Instructor: D. KHALID M. AL-KAHSAH
Phone: 00967-739391930 (*WhatsApp*)
Email: kkahsah@gmail.com

Computational Theory: Introduction

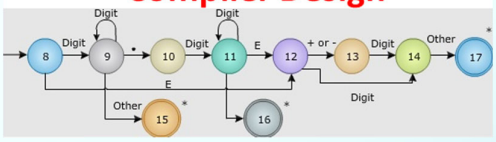
- This field of research was started by mathematicians in the 1930's, when they were trying to understand the meaning of a "computation".
- بدأ هذا المجال البحثي على يد علماء الرياضيات في ثلاثينيات القرن العشرين، عندما كانوا يحاولون فهم معنى "الحساب".
- A central question asked was whether all mathematical problems can be solved in a systematic way.
- و كان السؤال المركزي الذي تم طرحه هو ما إذا كان من الممكن حل جميع المشاكل الرياضية بطريقة منهجية.
- The research that started in those days led to computers as we know them today.
- إن الأبحاث التي بدأت في تلك الأيام أدت إلى ظهور أجهزة الكمبيوتر كما نعرفها اليوم.

Computational Theory: Introduction

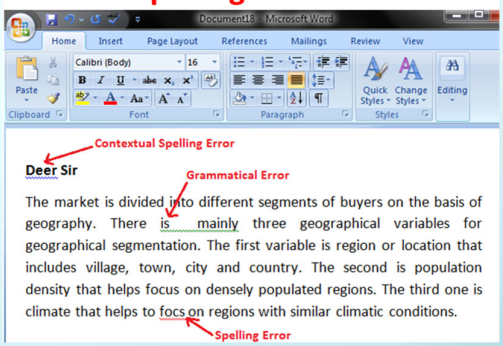
Practical applications

Vending Machine

Compiler Design



Spelling Checker



Computational Theory: Introduction

Three fundamental questions

- What can be computed?
  - (Computing models and computability)

- ما الذي يمكن حسابه؟
  - (نماذج الحوسبة وقابلية الحساب)

- What can be computed efficiently?
  - (Complexity)

- ما الذي يمكن حسابه بكفاءة؟
  - (التعقيد (الكفاءة))

- How can we build practical computing devices and systems?
  - (Architectures and Systems)

- كيف يمكننا بناء أجهزة او نظم عملية للحوسبة.
  - (المعمارية و النظم)

Objective

- Make a theory out of the idea of computation.
- بناء نظرية من فكرة الحساب.

What is “computation”?

- Processing of information based on a finite set of operations or rules.
- معالجة المعلومات بناءً على مجموعة منتهية من العمليات أو القواعد.

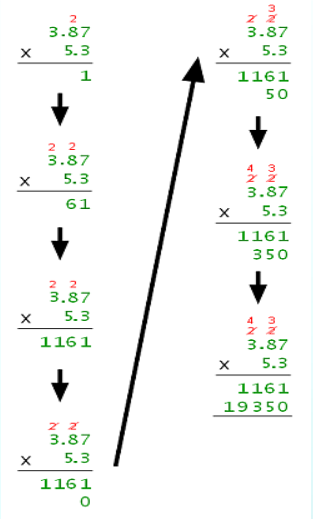
Computational Theory: Introduction

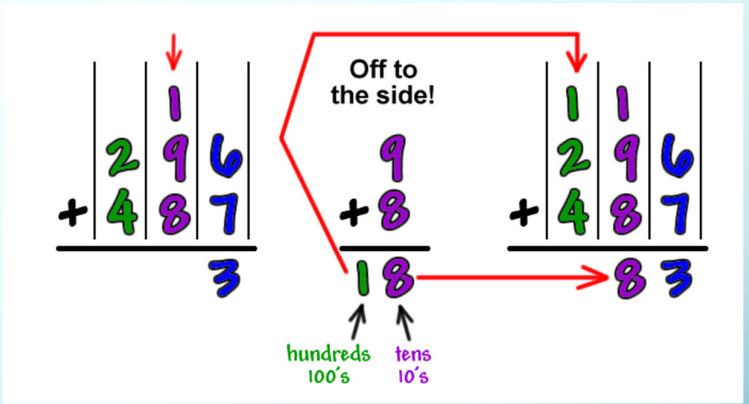
Computation means

- Solving problems through the mechanical, preprogrammed execution of a series of small, unambiguous steps.
- حل المشاكل عن طريق تنفيذ سلسلة من الخطوات البسيطة, غير المبهمة (الواضحة) ميكانيكياً او برمجياً.
  - Paper + Pencil Arithmetic
  - Abacus (المعداد)
  - Calculator w/moving parts (Babbage wheels, Mark I)
  - Ruler & compass geometry constructions
  - Digital Computers

Computational Theory: Introduction

Paper + Pencil Arithmetic





Computational Theory: Introduction

What do we want in a "theory"?

- **Generality**
  - Technology-independent.
  - Abstraction: ignores inessential details.

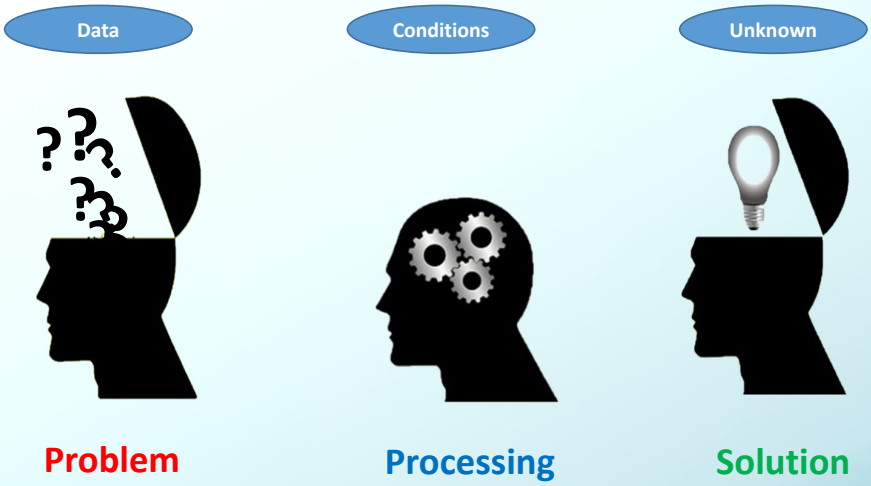
- **عمومية (بشكل عام)**
  - مستقل عن التكنولوجيا.
  - التجريد: يتجاهل التفاصيل غير الضرورية.

- **Precision**
  - Mathematical, formal.
  - Can prove theorems about computation, both positive (what can be computed) and negative (what cannot be computed).

- **دقة (بشكل دقيق)**
  - رياضي، رسمي.
  - القدرة على إثبات النظريات المتعلقة بالحساب، سواء كانت إيجابية (ما يمكن حسابه) أو سلبية (ما لا يمكن حسابه).

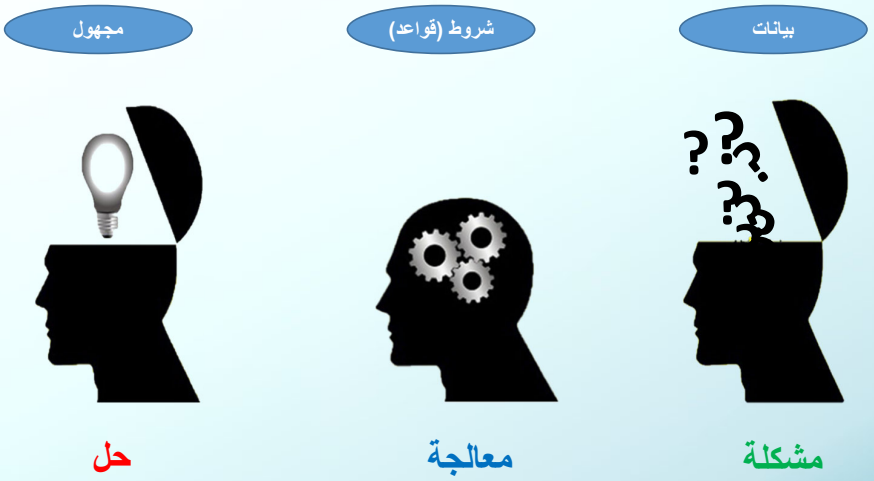
Computational Theory: Introduction

Problem



- To solve a problem means to find a way of determining the **unknowns** from the given **data** such that the **conditions** of the problem are satisfied.

مشكلة



❖ حل المشكلة يعني إيجاد طريقة لتحديد المجهول من البيانات المقدمة بحيث يتم تلبية شروط المشكلة.

Representing Information

❖ **Symbol:** The symbol is the smallest building block in the theory of computation and can be any letter, number or even pictograms.

✧ **For example**: a, b, 0, 1, #, (

❖ **رمز:** الرمز هو أصغر كتلة بناء في نظرية الحساب ويمكن أن يكون أي حرف أو رقم أو حتى صورة توضيحية.

✧ **مثال:** a, b, 0, 1, #, (

Representing Information

❖ **Alphabet:** From the symbols we can form an alphabet represented by the sigma sign (Σ). The alphabet is nothing more than a collection of symbols (**finite set**).

✧ **For example**: Σ = {a, b}, Σ = {0, 1}

❖ **أبجدية:** يمكننا تكوين أبجدية من الرموز و يرمز لها بالعلامة (Σ) (سيجما).
الأبجدية ليست أكثر من مجموعة من الرموز (**مجموعة منتهية**).

✧ **مثال:** Σ = {a, b}, Σ = {0, 1}

Representing Information

❖ **String:** A string is a sequence of symbols from the alphabet.

✧ **For example**: Σ = {a, b}

- **Possible strings are:**
  - String "a" of length 1.
  - String "aa" of length 2.
  - String "aab" of length 3.

❖ **السلسلة:** السلسلة عبارة عن سلسلة (تسلسل) من الرموز من الأبجدية..

✧ **مثال:** Σ = {a, b}

- **السلاسل المحتملة هي:**
  - **سلسلة "a" بطول 1.**
  - **سلسلة "aa" بطول 2.**
  - **سلسلة "aab" بطول 3.**

Representing Information

❖ **Language:** In general a language is a collection of words, but in the theory of computation a language is a collection of strings..

✧ Inputs (& outputs) of computations are **strings**.

❖ **اللغة:** بشكل عام، اللغة هي مجموعة من الكلمات، ولكن في نظرية الحساب، اللغة هي مجموعة من السلاسل الرمزية (سلاسل من الرموز).

✧ مدخلات (& مخرجات) العمليات الحسابية عبارة عن **سلاسل**.

Representing Information

❖ **Examples:**

✧ **Question**: How many strings of length **2** are possible over alphabet **{a,b}**?

✧ **Answer**: **4**, The strings are: **aa**, **ab**, **ba**, **bb**

✧ **Question**: How many strings of length **2** and starting with "a" are possible over alphabet **{a,b}**?

✧ **Answer**: **2**, The strings are: **aa**, **ab**

✧ **Question**: How many strings of length **n** are possible over alphabet **{a,b}**?

✧ **Answer**: 2^n, **Note**: There are **2** symbols in the alphabet.

✧ **Question**: How many strings of length **n** are possible over alphabet **{a,b,c,d}**?

✧ **Answer**: $|\Sigma|^n = 4^n$, **Note**: There are **4** symbols in the alphabet.

Computational Problems

- A single question that has infinitely many different instances

- سؤال واحد يحتوي على عدد لا نهائي من الحالات المختلفة

  - given a string x, does it have an even number of a's?
    - Example: aa, aaaa, ababaa, …
  - given a string x, does it have more a's than b's?
    - Example: aaab, ababaaaa, …

Examples of computational problems:

- On numbers على الاعداد

  - ADDITION: given two numbers x, y, compute x + y.
  - الجمع: اذا كان لدينا عددين x, y, احسب x + y.

  - PRIMALITY: given a number x, is x prime?
  - الأولية: اذا كان لدينا عدد x, هل x عدد اولي؟

Examples of computational problems:

❖About computer programs حول برامج الحاسوب

✧SYNTACTICALLY CORRECT C PROGRAM:

- Given a string of ASCII symbols, does it follow the syntax rules for the C programming language?

✧التصحيح النحوي لبرنامج بلغة C :

- إذا كان لدينا سلسة من رموز ASCII, هل تتبع قواعد بناء الجملة في لغة البرمجة C ؟

Examples of computational problems:

❖About computer programs حول برامج الحاسوب

✧HALTING PROBLEM:

- Given a computer program (say in C), can it ever get stuck in an infinite loop?

✧مشكلة التوقف:

- إذا كان لدينا برنامج حاسوب (مثلاً في لغة C), هل يمكن أن يعلق في حلقة غير منتهية ؟

Characteristics of computational problems

- **Discreteness** الانفصال
  - **State of the system** can be represented as a finite amount of information.
  - يمكن تمثيل **حالة النظام** بكمية محدودة (منتهية) من المعلومات.
- **Abstraction** التجريد
  - **Irrelevant details** can be ignored.
  - يمكن تجاهل **التفاصيل غير المهمة** .
- **Generality** العمومية
  - A single mathematical model applies to many devices.
  - نموذج رياضي واحد ينطبق على العديد من الأجهزة.

Example

- Is there an integer that if you multiply it by 2 and add 3 to it gives its square value?
- هل يوجد عدد صحيح إذا تم ضربه في 2 و اضيف اليه 3 تكون النتيجة مربع ذلك العدد؟
  - Is there an integer like x where $2x + 3 = x^2$?
  - هل يوجد عدد صحيح, مثلاً x , بحيث $2x + 3 = x^2$ ؟
  - Let $x \in N$
    - Is $2x + 3 = x^2$?

What is Computing? ما هو الحساب؟

- **First Definition:** التعريف الأول:
  - "Computing a yes/no answer"
  - حساب (إيجاد) إجابة نعم/لا.
    - means: "Determining if a string is in a language"
    - يعني: تحديد اذا كان الخيط الرمزي (السلسلة) في اللغة.

Theory of computation نظرية الحوسبة

- Is the branch that deals with how efficiently problems can be solved on a model of computation, using an algorithm.
- هو الفرع الذي يتعامل مع مدى كفاءة حل المشكلات على نموذج الحوسبة باستخدام خوارزمية.

Computational Theory: Introduction

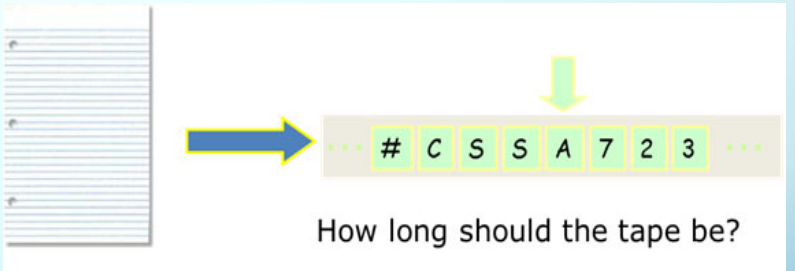
- A model of computation: is an abstract device used to perform computation.
- نموذج الحوسبة: هو جهاز مجرد (تجريدي) يستخدم لإجراء الحوسبة.

- A model of computation: is the definition of the set of allowable operations used in computation and their respective costs.
- نموذج الحوسبة: هو تعريف لمجموعة من العمليات الممكنة (المسموح بها) المستخدمة في الحوسبة و ما يخص تكلفتها.

Computational Theory: Introduction

Modeling Pencil and Paper

- Computing is normally done by writing certain symbols on paper.
- يتم إجراء الحوسبة عادة عن طريق كتابة رموز معينة على الورق.
- We may suppose this paper is divided into squares like a child's arithmetic book.
- يمكننا أن نفترض أن هذه الورقة مقسمة إلى مربعات مثل كتاب الحساب للأطفال.



Three Basic Concepts ثلاثة مفاهيم أساسية

1. Languages. اللغات
2. Grammars. القواعد
3. Automata. الأتمتة

Language Concepts مفاهيم اللغة

- An **alphabet**, denoted by $\sum$, is a finite, nonempty set of symbols.
- الابجدية, يرمز لها بالرمز $\sum$ (سيجما), هي مجموعة منتهية غير فارغة من الرموز.
- A **string** is a finite sequence of symbols from the alphabet.
- السلسلة, هي تسلسل منتهي من رموز الابجدية.
- **Language**: In general a language is a collection of words, but in the theory of computation a language is a collection of strings.
- اللغة: بشكل عام، اللغة هي مجموعة من الكلمات، ولكن في نظرية الحساب، اللغة هي مجموعة من السلاسل الرمزية.

For Example $\Sigma = \{a,b\}$

- L_1 = set of all strings of length 2.
  - $L_1 = \{aa, ab, ba, bb\}$
  - L_1 is called a finite language or a finite set.
- L_2 = set of all strings of length 3.
  - $L_2 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$
  - L_2 is called a finite language or a finite set.
- L_3 = set of all strings where each string start with an "a".
  - $L_3 = \{a, aa, ab, aaa, aab, aba, abb, ...\}$
  - L_3 is called an infinite language or an infinite set.

Note:
A language which can be formed over an alphabet can be finite or infinite.

ملاحظة:
اللغة التي يمكن تشكيلها على الأبجدية يمكن أن تكون منتهية أو غير منتهية.

Powers of an alphabet, Σ^k, قوى الابجدية

- Is the set of strings of length k with symbols from Σ.
- هي مجموعة السلاسل الرمزية بطول k من الابجدية Σ .

- Alphabet: $\Sigma = \{0, 1\}$
  - $\Sigma^0 = \{\ \}$
  - $\Sigma^1 = \{0, 1\}$
  - $\Sigma^2 = \{00,01, 10, 11\}$

Examples

- **Alphabet: $\Sigma = \{a, b\}$**
  - Σ^0 = Set of all strings over Σ of length 0
    - $\Sigma^0 = \{\varepsilon\}$
    - ε (epsilon) is a special symbol which represents an empty string "" with length 0.
    - $|\varepsilon| = 0$
    - $|\Sigma^0| = 0$
  - Σ^1 = Set of all strings over Σ of length 1
    - $\Sigma^1 = \{a,b\}$
    - $|\Sigma^1| = 2$

Examples

- **Alphabet: $\Sigma = \{a, b\}$**
  - Σ^2 = Set of all strings over Σ of length 2
    - $\Sigma^2 = \Sigma\ \Sigma = \{a,b\}\ \{a,b\} = \{aa,ab,ba,bb\}$
    - $|\Sigma^2| = 4$
  - Σ^3 = Set of all strings over Σ of length 3
    - $\Sigma^3 = \Sigma\ \Sigma\ \Sigma = \{a,b\}\ \{a,b\}\ \{a,b\} = \{aaa,aab,aba,abb,baa,bab,bba,bbb\}$
    - $|\Sigma^3| = 8$

Computational Theory: Introduction

- The set of all strings over Σ is denoted Σ^*.
- مجموعة جميع السلاسل الرمزية في الابجدية Σ يرمز لها Σ^* .

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots \Sigma^n \ \ldots$$

- for example: $\Sigma = \{a, b\}$

$$\Sigma^* = \{\varepsilon\} \cup \{a,b\} \cup \{aa,ab,ba,bb\} \cup \ldots$$

* means: 0, 1, 2, …, n

- Σ^* = Set of all strings over Σ of ALL lengths.
- Σ^* = مجموعة جميع السلاسل على الابجدية Σ من جميع الاطوال.
- Σ^* represents an infinite set.
- Σ^* تمثل مجموعة غير منتهية.
- Σ^* can be seen as the parent of all languages.
- Σ^* يمكن اعتبارها (النظر اليها) على انها الاب لجميع اللغات.

Computational Theory: Introduction

- A language L over an alphabet Σ is a subset of Σ^*.
  - That is, $L \subset \Sigma^*$.
- اللغة L على الابجدية Σ هي مجموعة جزئية من Σ^* .
  - هذا يعني, $L \subset \Sigma^*$.

examples:

- The set of legal English words.
- The set of legal C programs.
- The set of strings consisting of n 0's followed by n 1's.
  {01, 0011, 000111, …}

Computational Theory: Introduction

- The empty language ϕ.
- اللغة الفارغة يعبر عنها بـ ϕ .
- The language $\{\varepsilon\}$ consisting of the empty string.
- اللغة $\{\varepsilon\}$ تحتوي على سلسلة الفراغ.

$$\phi \neq \{\varepsilon\}$$

Question: How many languages are possible over Σ^* ?

Answer: infinite

Computational Theory: Introduction

Grammar Concepts مفاهيم القواعد

- A grammar G is a quadruple G = (V, T, S, P) where:
- القواعد G , هي الرباعي G = (V, T, S, P) بحيث:
  - V is a finite set of objects called variables.
  - V هي مجموعة منتهية من الكائنات تسمى المتغيرات.
  - T is a finite set of objects called terminal symbols.
  - T هي مجموعة منتهية من الكائنات تسمى الرموز الطرفية.
  - $S \in V$, S is a special symbol called the start symbol.
  - $S , V \ni S$ هو رمز خاص يسمى رمز البداية.
  - P is a finite set of productions.
  - P هي مجموعة منتهية من الانتاجات.
  - V and T are nonempty and disjoint.
  - V و T هما غير خاليتين و منفصلتين.

Computational Theory: Introduction

- Productions have form $x \rightarrow y$ where:
- الانتاجات تأخذ الصيغة $x \rightarrow y$ حيث:
  - $x \in (V \cup T)^+$, i.e., x is some non-null string of terminals and variables.
  - $x \in (V \cup T)^+$, ما يعني, x هو بعض السلاسل غير الفارغة من الرموز الطرفية و المتغيرات.
  - $y \in (V \cup T)^*$, i.e., y is some, possibly null, string of terminals and variables.
  - $y \in (V \cup T)^*$, ما يعني, y هو بعض السلاسل المحتمل فارغة من الرموز الطرفية و المتغيرات.

Computational Theory: Introduction

- $w_1 \overset{*}{\Rightarrow} w_n$ means that w_1 derives w_n in zero or more production steps.
- $w_1 \overset{*}{\Rightarrow} w_n$, تعني ان w_1 يشتق w_n في صفر او اكثر من خطوات الإنتاج.

A derivation of some sentence $w \in L(G)$ is a sequence $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \ldots \Rightarrow w_n \Rightarrow w$

Computational Theory: Introduction

- Mathematical abstractions (models) can be used to represent real systems
- التجريدات الرياضية (النماذج) يمكن ان تستخدم لتمثيل الأنظمة الحقيقية
- Formal reasoning can improve our ability to describe, design and build systems
- التفكير المنطقي الشكلي يمكن ان تطور مقدرتنا على وصف, تصميم, و بناء النظم
  - Precisely define requirements — تعريف المتطلبات بدقة
  - Uncover design flaws — اكتشاف عيوب التصميم
  - Produce rational implementations — انتاج تنفيذات منطقية (عقلانية)
- Different models and logics have different strengths and weaknesses
- النماذج و المنطق المختلفة لها نقاط قوة و نقاط ضعف مختلفة

Computational Theory: Introduction

Chomsky Classification of Grammars

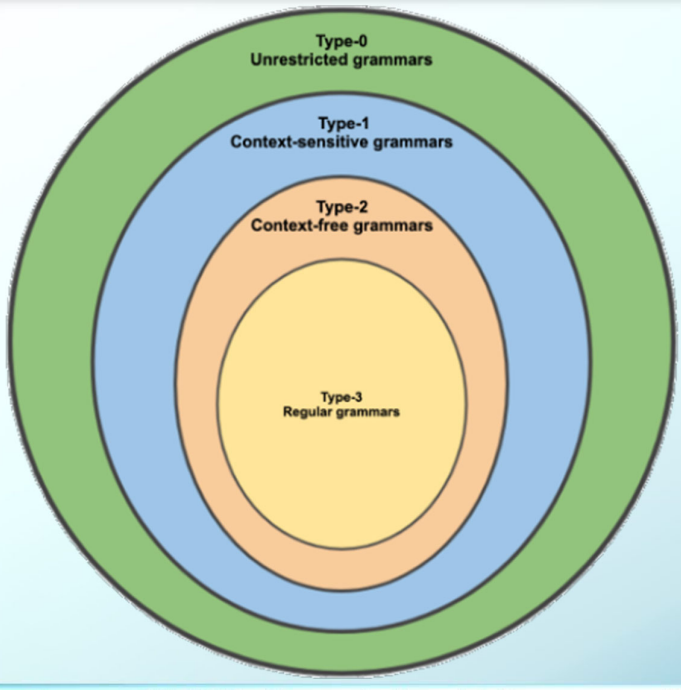
According to Noam Chomsky, there are **four** types of grammars:

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown automaton
Type 3	Regular grammar	Regular language	Finite state automaton

Chomsky Classification of Grammars

اعتمادا على العالم نعوم تشومسكي, توجد **أربعة** أنواع من القواعد:

نوع القواعد	القواعد المقبولة	اللغة المقبولة	الاتمتة
Type 0	قواعد غير مقيدة	لغة قابلة للعد بشكل متكرر	آلة تورينج
Type 1	قواعد حساسة للسياق	لغة حساسة للسياق	أوتوماتك محدود خطيًا
Type 2	قواعد حرة السياق	لغة حرة السياق	آلة الدفع للأسفل
Type 3	قواعد المنتظمة	لغة منتظمة	آلة الحالة المنتهية



Type - 3 Grammar

- **Type-3 grammars** generate regular languages.
- Type-3 grammars must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal.
- The productions must be in the form **X → a or X → aY**
  where **X, Y ∈ N** (Non terminal), and **a ∈ T** (Terminal)
- The rule **S → ε** is allowed if **S** does not appear on the right side of any rule.
- Example

$$X \rightarrow \varepsilon,\ X \rightarrow a \mid aY,\ Y \rightarrow b$$

قواعد النوع – 3 -

- **قواعد النوع – 3 -** , تنتج لغات منتظمة.
- يجب أن تحتوي قواعد النوع الثالث على رمز واحد غير طرفي على الجانب الأيسر وجانب أيمن يتكون من رمز واحد أو رمز واحد يتبعه رمز واحد غير طرفي.
- الانتاجات تأخذ الصيغة **X → a or X → aY**
  حيث **X, Y ∈ N** (غير طرفية) و **a ∈ T** (طرفي)
- القاعدة **S → ε** مسموح بها اذا كان **S** لا يظهر في الجانب الأيمن أي قاعدة.

✧ مثال

$$X \rightarrow \varepsilon,\ X \rightarrow a \mid aY,\ Y \rightarrow b$$

Type - 2 Grammar

- **Type-2 grammars** generate context-free languages.
- The productions must be in the form $A \rightarrow \gamma$ where $A \in N$ (Non terminal), and $\gamma \in (T \cup N)^*$ (String of terminals and non-terminals).
- These languages generated by these grammars are be recognized by a non-deterministic pushdown automaton.
- Example

$$S \rightarrow X|a,\ X \rightarrow a,\ X \rightarrow aX,\ X \rightarrow abc,\ X \rightarrow \varepsilon$$

قواعد النوع – 2 -

- **قواعد النوع – 2 -** , تنتج لغات حرة السياق.
- الانتاجات يجب أن تكون بالصيغة $A \rightarrow \gamma$ حيث $A \in N$ (غير طرفية) و $\gamma \in (T \cup N)^*$ (سلسلة من الطرفيات و الغير طرفيات)
- يمكن التعرف على اللغات التي تم إنشاؤها بواسطة هذه القواعد بواسطة آلة الدفع غير المحدودة.
  - مثال

$$S \rightarrow X|a,\ X \rightarrow a,\ X \rightarrow aX,\ X \rightarrow abc,\ X \rightarrow \varepsilon$$

Type - 1 Grammar

- **Type-1 grammars** generate context-sensitive languages.
- The productions must be in the form **α A β → α γ β** where **A ∈ N** (Non-terminal), and **α, β, γ ∈ (T ∪ N)*** (Strings of terminals and non-terminals).
- The strings **α** and **β** may be empty, but **γ** must be non-empty
- The rule **S → ε** is allowed if S does not appear on the right side of any rule. The languages generated by these grammars are recognized by a linear bounded automaton.
- Example

```
AB → AbBc,  A → bcA,  B → b
```


قواعد النوع – 1 -

- **قواعد النوع – 1 -** , تنتج لغات حساسة للسياق.
- الانتاجات تأخذ الصيغة **α A β → α γ β** حيث **A ∈ N** (غير طرفية) و **α, β, γ ∈ (T ∪ N)*** (سلسلة من الطرفيات و الغير طرفيات)
- السلاسل **α** و **β** يمكن ان تكون فارغة, لكن **γ** يجب الا تكون فارغة.
- القاعدة **S → ε** مسموح بها اذا كان **S** لا يظهر في الجانب الأيمن أي قاعدة. يتم التعرف على اللغات التي يتم إنشاؤها بواسطة هذه القواعد بواسطة آلة محدودة خطيًا
  - مثال

```
AB → AbBc,  A → bcA,  B → b
```

Type - 0 Grammar

- **Type-0 grammars** generate Turing-recognizable (*recursively enumerable*) languages.
- The productions have no restrictions. They are any phase structure grammar including all formal grammars.
- They generate the languages that are recognized by a Turing machine.
- The productions can be in the form of **α** → **β**
  where **α** is a string of terminals and non-terminals with at least one non-terminal and **α** cannot be null. **β** is a string of terminals and non-terminals.
- Example

  S → ACaB, Bc → acB, CB → DB, aD → Db

قواعد النوع – 0 -

- **قواعد النوع – 0 -** , تنتج لغات تعرفها آلة تورينج (معدودة تعاودياً).
- لا توجد قيود على الإنتاجات. فهي تتضمن أي بنية لقواعد النحو بما في ذلك جميع القواعد النحوية الشكلية.
- تنتج اللغات التي يتم التعرف عليها بواسطة آلة تورينج.
- يمكن أن تكون الإنتاجات في شكل **α** → **β**
  حيث **α** عبارة عن سلسلة من الرموز الطرفية وغير الطرفية مع وجود رمز غير طرفي واحد على الأقل ولا يمكن أن يكون **α** فارغًا. **β** عبارة عن سلسلة من الرموز الطرفية وغير الطرفية
  - مثال

  S → ACaB, Bc → acB, CB → DB, aD → Db

Computational Theory: Introduction