# **Formal language**

## Definition

**# Source**: [**Wiki source**](https://drive.google.com/drive/u/1/folders/1kIQnffQbC2UeTqdXW54MEtEYv-wQ6Wom): [Formal language](https://drive.google.com/file/d/11QOJGdPrfPeuXG4LXDCBjFsqjhib9G_h/view?usp=drive_link) -> at - (**Top**)

In [logic](https://en.wikipedia.org/wiki/Logic), [mathematics](https://en.wikipedia.org/wiki/Mathematics), [computer science](https://en.wikipedia.org/wiki/Computer_science), and [linguistics](https://en.wikipedia.org/wiki/Linguistics), a formal language is a set of [strings](https://en.wikipedia.org/wiki/String_(computer_science)) whose symbols are taken from a set called "[alphabet](https://en.wikipedia.org/wiki/Formal_language#Definition)".

The alphabet of a formal language consists of symbols that concatenate into strings (also called "words").[[1]](https://en.wikipedia.org/wiki/Formal_language#cite_note-1) Words that belong to a particular formal language are sometimes called [*well-formed words*](https://en.wikipedia.org/wiki/Formal_language#Definition). A formal language is often defined by means of a [formal grammar](https://en.wikipedia.org/wiki/Formal_grammar) such as a [regular grammar](https://en.wikipedia.org/wiki/Regular_grammar) or [context-free grammar](https://en.wikipedia.org/wiki/Context-free_grammar).

## Examples

**# Source**: [**Wiki source**](https://drive.google.com/drive/u/1/folders/1kIQnffQbC2UeTqdXW54MEtEYv-wQ6Wom): [Formal language](https://drive.google.com/file/d/11QOJGdPrfPeuXG4LXDCBjFsqjhib9G_h/view?usp=drive_link) -> at - (**Examples**)

**The following rules describe a formal language *L* over the alphabet Σ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, =}:**

* **Every nonempty string that does not contain "+" or "=" and does not start with "0" is in *L*.**
* **The string "0" is in *L*.**
* **A string containing "=" is in *L* if and only if there is exactly one "=", and it separates two valid strings of *L*.**
* **A string containing "+" but not "=" is in *L* if and only if every "+" in the string separates two valid strings of *L*.**
* **No string is in *L* other than those implied by the previous rules.**

**Under these rules, the string "23+4=555" is in *L*, but the string "=234=+" is not. This formal language expresses** [**natural numbers**](https://en.wikipedia.org/wiki/Natural_number)**, well-formed additions, and well-formed addition equalities, but it expresses only what they look like (their** [**syntax**](https://en.wikipedia.org/wiki/Syntax)**), not what they mean (**[**semantics**](https://en.wikipedia.org/wiki/Semantics)**). For instance, nowhere in these rules is there any indication that "0" means the number zero, "+" means addition, "23+4=555" is false, etc.**

## Key Concepts and Definitions in Formal Systems

### 1-Letter (alphabet)

#### Definition

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

\*A **letter** in a writing system is a symbol or character used to represent a basic unit of sound in a language, which is typically part of an alphabet. In alphabetic **writing systems**, letters correspond to **phonemes**—the smallest units of sound in spoken language. For example, in the English language, letters like "A," "B," and "C" each represent specific sounds.

\***Letters** are combined in different ways to form words and sentences. Writing systems that use letters are often contrasted with other types of writing systems, such as logographic (where symbols represent entire words or concepts, like in Chinese) or syllabic (where symbols represent syllables, like in the Japanese kana).

\*In summary, a letter is a fundamental symbol in an alphabetic writing system, representing individual sounds that, when combined, form words and communicate meaning.

#### Examples in writing systems

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

1. English Alphabet (Latin-based alphabet):

* **A**: Represents the sound /æ/ or /ɑ/ depending on the word.
* **B**: Represents the sound /b/.
* **C**: Represents the sound /k/ or /s/ depending on the context.

2. Greek Alphabet:

* **Α (Alpha)**: Represents the sound /a/ as in "father."
* **Β (Beta)**: Represents the sound /b/.
* **Γ (Gamma)**: Represents the sound /g/ as in "go."

3. Cyrillic Alphabet (used in Russian, among other languages):

* **А (A)**: Represents the sound /a/.
* **Б (Be)**: Represents the sound /b/.
* **В (Ve)**: Represents the sound /v/.

4. Hebrew Alphabet:

* **א (Aleph)**: Represents a glottal stop, often silent in modern Hebrew.
* **ב (Bet)**: Represents the sound /b/.
* **ג (Gimel)**: Represents the sound /g/.

5. Arabic Alphabet:

* **ا (Alif)**: Represents a glottal stop or a long /a/ sound.
* **ب (Ba)**: Represents the sound /b/.
* **ج (Jeem)**: Represents the sound /ʤ/ (like "j" in "jump").

6. Devanagari Script (used for Hindi, Sanskrit, etc.):

* **अ (A)**: Represents the sound /ʌ/ (like "a" in "sofa").
* **ब (Ba)**: Represents the sound /b/.
* **च (Cha)**: Represents the sound /ʧ/ (like "ch" in "chat").

#### Key Concepts and Definitions in Grapheme

##### Grapheme

###### Definition

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

A **grapheme** is the smallest unit of a writing system that represents a single sound or a combination of sounds. It can be a letter, character, or symbol that corresponds to a specific phoneme (sound) in a language. Graphemes are the building blocks of written language and are used to represent speech sounds visually.

###### Examples

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

1. English:

* **Single-letter graphemes**:  
  + *"c"*, *"a"*, *"t"* in the word *"cat"*
* **Two-letter graphemes** (digraphs):  
  + *"ch"* in *"chat"*
  + *"sh"* in *"shoes"*
* **Three-letter graphemes** (trigraphs):  
  + *"sch"* in *"school"*
  + *"tch"* in *"match"*

2. Spanish:

* **Single-letter graphemes**:  
  + *"c"*, *"a"*, *"r"* in the word *"carro"* (car)
* **Diacritics** (marks that alter the sound of a letter):  
  + *"é"* (accent mark) in *"café"*
  + *"ñ"* in *"niño"* (child)

3. French:

* **Single-letter graphemes**:  
  + *"b"*, *"é"*, *"r"* in *"béret"*
* **Diacritics**:  
  + *"à"*, *"é"*, *"ç"*

4. Chinese:

* **Chinese characters** (each represents a whole syllable, which can correspond to one or more graphemes):  
  + *"你"* (nǐ) meaning "you"
  + *"好"* (hǎo) meaning "good"

In Chinese, graphemes are typically characters, each representing a syllable, and may represent one or more sounds depending on the character.

5. Arabic:

* **Single-letter graphemes**:  
  + *"ب"* (b) as in *"باب"* (baab) meaning "door"
  + *"ر"* (r) as in *"رسول"* (rasool) meaning "messenger"
* **Combined graphemes**:  
  + The Arabic script is cursive, meaning the shape of the grapheme may change depending on its position in the word.

Each of these graphemes corresponds to a unit of meaning or sound in the written form of the language.

### 2-Word and morphemes

#### Word Definition

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

**Word (in linguistics)**: A unit of language that carries meaning and consists of one or more sounds or symbols. Words can be spoken or written, and they can stand alone or combine with other words to form phrases or sentences.

#### Morphemes Definition

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

A **morpheme** is the smallest unit of meaning in a language. Unlike a word, which is a complete unit of meaning, a morpheme can be a part of a word (a prefix, suffix, or root) that carries meaning. Morphemes can be classified into two main types:

#### Word VS Morphemes

**# Source**: **Chatgpt (GPT-4-turbo)**

\*A word is a complete unit of meaning that can stand alone, such as "book" or "run." A word can also consist of more than one morpheme, like "cats," which contains two morphemes: "cat" and "-s."

\*Morphemes are the smallest units of meaning. For example, "un-" in "undo" and "-ed" in "walked" are bound morphemes, meaning they cannot stand alone. "Happy" is a free morpheme because it can stand alone, while "unhappy" consists of two morphemes: "un-" and "happy." Similarly, "dogs" contains three morphemes: "dog," "-s" (plural marker), and "s" itself as a bound morpheme.

\*The concept of "word" is distinguished from that of a [morpheme](https://en.wikipedia.org/wiki/Morpheme), which is the smallest unit of language that has a meaning, even if it cannot stand on its own.[[1]](https://en.wikipedia.org/wiki/Word#cite_note-CDL-1)

\*Words are made out of at least one morpheme. Morphemes can also be joined to create other words in a process of [morphological derivation](https://en.wikipedia.org/wiki/Morphological_derivation).[[2]](https://en.wikipedia.org/wiki/Word#cite_note-RDLL-2): 768

\* In English and many other languages, the morphemes that make up a word generally include at least one [root](https://en.wikipedia.org/wiki/Root_(linguistics)) (such as "rock", "god", "type", "writ", "can", "not") and possibly some [affixes](https://en.wikipedia.org/wiki/Affix) ("-s", "un-", "-ly", "-ness"). Words with more than one root ("[type][writ]er", "[cow][boy]s", "[tele][graph]ically") are called [compound words](https://en.wikipedia.org/wiki/Compound_(linguistics)). In turn, words are combined to form other elements of language, such as [phrases](https://en.wikipedia.org/wiki/Phrase) ("a red rock", "put up with"), [clauses](https://en.wikipedia.org/wiki/Clause) ("I threw a rock"), and [sentences](https://en.wikipedia.org/wiki/Sentence_(linguistics)) ("I threw a rock, but missed").

### 3-Writing system

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

A **writing system** is a method of visually representing verbal communication through symbols or characters. It enables people to convey ideas, information, and emotions in a written form. Writing systems can be used to document language, preserve history, or communicate across distances.

There are different types of writing systems, which include:

1. **Alphabetic systems**: These use a set of letters or symbols, each representing individual sounds or phonemes in the language. Examples include the Latin alphabet (used in English) and the Cyrillic alphabet (used in Russian).
2. **Syllabic systems**: In these systems, symbols represent syllables rather than individual sounds. Japanese Kana is a well-known example.
3. **Logographic systems**: These systems use symbols or characters to represent entire words or concepts, rather than individual sounds or syllables. Chinese characters are an example of this system.
4. **Pictographic systems**: These use pictures or drawings to represent words or ideas. Ancient Egyptian hieroglyphs and early forms of writing like cave paintings are examples.
5. **Abjad systems**: These primarily represent consonants, with vowels either omitted or indicated with diacritics. The Arabic and Hebrew scripts are examples of abjads.
6. **Abugida systems**: These are similar to syllabaries but with an emphasis on consonants and modifications that represent vowels. The Devanagari script used in Hindi and Sanskrit is an example.

Each writing system is unique to the language or culture that created it, and over time, these systems have evolved to meet the needs of communication in society.

### 4-Alphabet

#### Definition

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

In language, an **alphabet** is a standardized set of letters or symbols used to represent the basic speech sounds (phonemes) of a language. These symbols are used to form words and sentences. The alphabet typically includes a series of letters arranged in a specific order.

Alphabets are the building blocks of written language and are crucial for communication in languages that use them.

#### 

#### Examples

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

\*For example, the **English alphabet** consists of 26 letters: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z. Each of these letters corresponds to one or more sounds in spoken English.

\*Different languages use different alphabets, such as the **Cyrillic alphabet** in Russian or the **Hebrew alphabet** in Hebrew.

### 5-Well-formedness

#### Definition

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

Well-formedness in formal languages refers to the structural correctness of strings (sequences of symbols) according to the syntactic rules of the language. In formal language theory, a language is defined by a set of rules or grammar that dictate how symbols can be combined to form valid expressions or sentences.

#### Examples

##### 1. Mathematical Expressions (Context-Free Grammar Example)

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

Let's consider a simple formal language defined by the following grammar:

* S → aSb | ε

Here, S is the start symbol, and a and b are terminal symbols. The grammar generates strings where there are equal numbers of as followed by bs.

**Well-formed (Valid) Examples:**

* **ε** (empty string): This is valid because it matches the production S → ε.
* **ab**: This is well-formed because it matches S → aSb followed by S → ε.
* **aabb**: This is well-formed because it matches S → aSb twice, and then S → ε is applied to complete the string.
* **aaabbb**: This is well-formed because it can be derived from S → aSb applied three times and finished with S → ε.

**Ill-formed (Invalid) Examples:**

* **abbb**: This is ill-formed because there are more bs than as. The grammar requires a one-to-one correspondence between as and bs.
* **aab**: This is ill-formed because there are fewer bs than as, violating the grammar's rule of equal numbers of as and bs.

##### 2. Programming Language Expressions (Python)

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

In programming languages, well-formedness refers to adhering to the syntactic rules of the language (like Python's grammar).

**Well-formed (Valid) Examples:**

* **x = 5**: This is well-formed because it follows the assignment rule in Python: a variable (x), followed by the assignment operator (=), and a valid expression (5).
* **if x > 5:**: This is well-formed because it follows the structure of a conditional statement in Python, with a valid expression (x > 5) and the colon (:) marking the start of the indented block.
* **for i in range(10):**: This is well-formed because it follows the correct syntax for a for loop in Python, with valid expressions for the loop variable and range.

**Ill-formed (Invalid) Examples:**

* **x = ;**: This is ill-formed because the right-hand side of the assignment is missing a valid expression.
* **if x > 5**: This is ill-formed because it is missing a colon (:) at the end of the condition, which is required for the statement to be valid in Python.
* **for in range(10):**: This is ill-formed because it is missing the loop variable after the for keyword.

##### 3. Parenthesis Matching (Context-Free Grammar Example)

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

Consider a simple formal language for balanced parentheses, where the grammar is:

* S → (S) | SS | ε

**Well-formed (Valid) Examples:**

* **()**: This is well-formed because it matches the rule S → (S) with S → ε for the inner part.
* **(())**: This is well-formed because it matches S → (S) applied twice, with the inner part also following the same rule.
* **()()**: This is well-formed because it can be generated by S → SS, where each part is valid on its own.

**Ill-formed (Invalid) Examples:**

* **(()**: This is ill-formed because there is an unmatched opening parenthesis.
* **)**: This is ill-formed because there is a closing parenthesis without a matching opening parenthesis.
* **(()))**: This is ill-formed because there are too many closing parentheses.

##### 4. Regular Expressions

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

In the context of regular languages, well-formedness can refer to whether a regular expression is syntactically correct.

Well-formed (Valid) Examples:

* **a\***: This is well-formed because it follows the correct syntax for matching zero or more occurrences of the letter a.
* **(a|b)\***: This is well-formed because it uses parentheses and the | (or) operator correctly to match zero or more occurrences of either a or b.
* **ab+**: This is well-formed because it matches an a followed by one or more bs.

Ill-formed (Invalid) Examples:

* **a+\***: This is ill-formed because the + operator is misused before the \* operator, which is not syntactically correct.
* **(**: This is ill-formed because there is an unmatched opening parenthesis.
* **a|b**: This is ill-formed because the | operator must be surrounded by parentheses to indicate an explicit grouping, like (a|b).

### 6-Formal grammar

#### Definition

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

**Formal grammar** is a set of rules that define the structure of valid strings in a formal language. It provides a framework for generating all possible valid sentences (or strings) that belong to the language. A formal grammar defines how symbols in the language can be combined or arranged to produce valid sequences, and it is typically composed of a finite set of production rules.

#### Components of Formal Grammar:

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

A formal grammar consists of the following elements:

1. **Alphabet (Σ)**: A finite set of symbols or characters that make up the strings in the language. These symbols are called **terminals**. For example, in a programming language, these could be keywords, operators, or variable names.
2. **Non-terminals (V)**: Symbols used to define or generate the structure of the language. Non-terminals are placeholders that can be replaced by other non-terminals or terminals using production rules. For example, in a mathematical expression, "expression" and "term" might be non-terminal symbols.
3. **Start symbol (S)**: A special non-terminal from which production starts. This is the symbol that can eventually generate any string in the language. It's the root of the derivation process.
4. **Production rules (P)**: A set of rules that describe how non-terminal symbols can be replaced by combinations of terminals and other non-terminals. A rule is typically written in the form:  
    A→α  
    where A is a non-terminal, and α\alphaα is a string that can contain both terminals and non-terminals.

#### Example: A Simple Grammar for Arithmetic Expressions

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

Consider a simple formal grammar for a language of arithmetic expressions consisting of addition, subtraction, and multiplication.

* **Alphabet (Σ)**: {+, -, \*, (, ), a, b, c, ...} (the digits and operators)
* **Non-terminals (V)**: {Expression, Term, Factor}
* **Start symbol (S)**: Expression
* **Production rules (P)**:  
  1. Expression → Expression + Term | Expression - Term | Term
  2. Term → Term \* Factor | Factor
  3. Factor → ( Expression ) | number

In this grammar, an **Expression** is made up of terms joined by + or - operators. A **Term** is made up of factors joined by the \* operator. A **Factor** can be either a number or a nested **Expression** enclosed in parentheses.

This grammar generates valid arithmetic expressions such as:

* a + b
* a \* (b + c)
* (a + b) \* c

#### Example of a Regular Grammar (Type 3)

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

A **regular grammar** is the simplest type of grammar, often used to define regular languages, which can be recognized by finite automata. Regular grammars are commonly used for lexical analysis in programming languages.

**Grammar:**

* **Alphabet**: {a, b}
* **Non-terminals**: {S}
* **Start symbol**: S
* **Production rules**:  
  1. S → aS | bS | ε

**Explanation:**

* The start symbol S can be replaced by either aS or bS, meaning the string can contain any number of as and bs in any order. The empty string ε is also allowed, meaning the string can also be empty.

**Valid Strings:**

* a
* b
* ab
* ba
* aabbb
* ε (the empty string)

**Invalid Strings:**

* abbbba (because it doesn’t match the production rules; S → aS or S → bS must occur sequentially and can't end in a without a preceding b).

#### Types of Formal Grammars:

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

Formal grammars are classified into different types based on the complexity of the production rules. These classifications are part of the **Chomsky hierarchy**, which includes:

1. **Type 0 (Unrestricted Grammar)**:  
   * These grammars have no restrictions on the form of production rules. They can generate any language that is Turing-recognizable, which means they can describe any computation that a Turing machine can perform.
   * Example: A → BCD, where A, B, C, and D are non-terminals, and the production can have any form.
2. **Type 1 (Context-sensitive Grammar)**:  
   * In context-sensitive grammars, the production rules must satisfy certain conditions, such as the left-hand side of a rule being as long as or longer than the right-hand side. These grammars can describe more complex languages than Type 2 grammars but are less powerful than Type 0 grammars.
   * Example: AB → AC (Here, the number of symbols on the right-hand side is greater than or equal to the left-hand side).
3. **Type 2 (Context-free Grammar)**:  
   * In context-free grammars, the left-hand side of every production rule consists of a single non-terminal symbol. These grammars can describe many important programming languages and computational models, such as arithmetic expressions or function calls in programming.
   * Example: A → aB | bC.
4. **Type 3 (Regular Grammar)**:  
   * Regular grammars are the simplest type of grammars. The left-hand side of every production rule consists of a single non-terminal symbol, and the right-hand side consists of at most one terminal symbol followed by at most one non-terminal symbol. These grammars describe regular languages, which can be recognized by finite automata.
   * Example: A → aB | bC.

#### Importance of Formal Grammar:

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

* **Language Definition**: Formal grammars allow us to precisely define programming languages, mathematical languages, and natural languages.
* **Syntax Checking**: They are used to create parsers that check whether a given string (e.g., a program or an expression) is well-formed according to the rules of the language.
* **Compilers**: Compilers use formal grammars to translate source code into machine code by analyzing the syntactic structure of the code.
* **Natural Language Processing**: In linguistics and artificial intelligence, formal grammars are used to model the syntax of natural languages, helping in tasks such as speech recognition and machine translation.

#### Example: Simple Context-Free Grammar

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

Consider the grammar for a language that generates balanced parentheses (strings with an equal number of opening and closing parentheses):

* **Alphabet**: { (, ) }
* **Non-terminals**: { S }
* **Start symbol**: S
* **Production rules**:  
  1. S → ( S )
  2. S → SS
  3. S → ε

This grammar generates strings like:

* ()
* (())
* ()()
* (()())

#### Conclusion:

**# Source**: **Chatgpt (GPT-4-turbo) at [4/3/2025]**

A **formal grammar** is a mathematical tool for defining the syntactic structure of a language. It helps to describe how symbols in a language can be arranged according to specific rules. Formal grammars are fundamental in fields such as computer science (for defining programming languages), computational linguistics (for natural language processing), and automata theory (for recognizing language patterns).