Regular grammar

In theoretical computer science and formal language theory, a **regular grammar** is a grammar that is *right-regular* or *left-regular*. While their exact definition varies from textbook to textbook, they all require that

- all production rules have at most one non-terminal symbol;
- that symbol is either always at the end or always at the start of the rule's right-hand side.

Every regular grammar describes a regular language.

Strictly regular grammars

A **right-regular grammar** (also called right-linear grammar) is a formal grammar (N, Σ, P, S) in which all production rules in P are of one of the following forms:

1. $A \to a$
2. $A \to aB$
3. $A \to \varepsilon$

where A, B, $S \in N$ are non-terminal symbols, $a \in \Sigma$ is a terminal symbol, and ε denotes the empty string, i.e. the string of length 0. S is called the start symbol.

In a **left-regular grammar**, (also called left-linear grammar), all rules obey the forms

1. $A \to a$
2. $A \to Ba$
3. $A \to \varepsilon$

The language described by a given grammar is the set of all strings that contain only terminal symbols and can be derived from the start symbol by repeated application of production rules. Two grammars are called weakly equivalent if they describe the same language.

Rules of both kinds must not be mixed; for example, the grammar with rule set { $S \to aT$, $T \to Sb$, $S \to \varepsilon$ } is not regular, and describes the language $\{a^i b^i : i \in \mathbb{N}\}$, which is not regular, either.

Some textbooks and articles disallow empty production rules, and assume that the empty string is not present in languages.

Extended regular grammars

An *extended right-regular grammar* is one in which all rules obey one of

1. $A \to w$, where A is a non-terminal in N and w is in a (possibly empty) string of terminals Σ^*
2. $A \to wB$, where A and B are in N and w is in Σ^*.

Some authors call this type of grammar a *right-regular grammar* (or *right-linear grammar*)[1] and the type above a *strictly right-regular grammar* (or *strictly right-linear grammar*).[2]

An *extended left-regular grammar* is one in which all rules obey one of

1. $A \to w$, where A is a non-terminal in N and w is in Σ^*
2. $A \to Bw$, where A and B are in N and w is in Σ^*.

Examples

An example of a right-regular grammar G with $N = \{S, A\}$, $\Sigma = \{a, b, c\}$, P consists of the following rules

$S \to aS$
$S \to bA$
$A \to \varepsilon$
$A \to cA$

and S is the start symbol. This grammar describes the same language as the regular expression a*bc*, viz. the set of all strings consisting of arbitrarily many "*a*"s, followed by a single "*b*", followed by arbitrarily many "*c*"s.

A somewhat longer but more explicit extended right-regular grammar G for the same regular expression is given by $N = \{S, A, B, C\}$, $\Sigma = \{a, b, c\}$, where P consists of the following rules:

$S \to A$
$A \to aA$
$A \to B$
$B \to bC$
$C \to \varepsilon$
$C \to cC$

...where each uppercase letter corresponds to phrases starting at the next position in the regular expression.

As an example from the area of programming languages, the set of all strings denoting a floating point number can be described by an extended right-regular grammar G with $N = \{S,A,B,C,D,E,F\}$, $\Sigma = \{0,1,2,3,4,5,6,7,8,9,+,-,.,e\}$, where S is the start symbol, and P consists of the following rules:

$S \to +A$	$A \to 0A$	$B \to 0C$	$C \to 0C$	$D \to +E$	$E \to 0F$	$F \to 0F$
$S \to -A$	$A \to 1A$	$B \to 1C$	$C \to 1C$	$D \to -E$	$E \to 1F$	$F \to 1F$
$S \to A$	$A \to 2A$	$B \to 2C$	$C \to 2C$	$D \to E$	$E \to 2F$	$F \to 2F$
	$A \to 3A$	$B \to 3C$	$C \to 3C$		$E \to 3F$	$F \to 3F$

A → 4A	B → 4C	C → 4C	E → 4F	F → 4F
A → 5A	B → 5C	C → 5C	E → 5F	F → 5F
A → 6A	B → 6C	C → 6C	E → 6F	F → 6F
A → 7A	B → 7C	C → 7C	E → 7F	F → 7F
A → 8A	B → 8C	C → 8C	E → 8F	F → 8F
A → 9A	B → 9C	C → 9C	E → 9F	F → 9F
A → .B		C → eD		F → ε
A → B		C → ε		

Expressive power

There is a direct one-to-one correspondence between the rules of a (strictly) right-regular grammar and those of a nondeterministic finite automaton, such that the grammar generates exactly the language the automaton accepts.[3] Hence, the right-regular grammars generate exactly all regular languages. The left-regular grammars describe the reverses of all such languages, that is, exactly the regular languages as well.

Every strict right-regular grammar is extended right-regular, while every extended right-regular grammar can be made strict by inserting new non-terminals, such that the result generates the same language; hence, extended right-regular grammars generate the regular languages as well. Analogously, so do the extended left-regular grammars.

If empty productions are disallowed, only all regular languages that do not include the empty string can be generated.[4]

While regular grammars can only describe regular languages, the converse is not true: regular languages can also be described by non-regular grammars.

Mixing left-regular and right-regular rules

If mixing of left-regular and right-regular rules is allowed, we still have a linear grammar, but not necessarily a regular one. What is more, such a grammar need not generate a regular language: all linear grammars can be easily brought into this form, and hence, such grammars can generate exactly all linear languages, including non-regular ones.

For instance, the grammar G with $N = \{\mathrm{S}, \mathrm{A}\}$, $\Sigma = \{\mathrm{a}, \mathrm{b}\}$, P with start symbol S and rules

S → aA
A → Sb
S → ε

generates $\{a^i b^i : i \geq 0\}$, the paradigmatic non-regular linear language.

See also

- Regular expression, a compact notation for regular grammars
- Regular tree grammar, a generalization from strings to trees
- Prefix grammar
- Chomsky hierarchy
- Perrin, Dominique (1990), "Finite Automata", in Leeuwen, Jan van (ed.), *Formal Models and Semantics*, Handbook of Theoretical Computer Science, vol. B, Elsevier, pp. 1–58
- Pin, Jean-Éric (Oct 2012). *Mathematical Foundations of Automata Theory* (http://www.liafa.jussieu.fr/~jep/PDF/MPRI/MPRI.pdf) (PDF)., chapter III

References

1. John E. Hopcroft and Jeffrey D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation* (https://archive.org/details/introductiontoau00hopc). Reading/MA: Addison-Wesley. ISBN 0-201-02988-X. Here: p.217 (left, right-regular grammars as subclasses of context-free grammars), p.79 (context-free grammars)
2. Hopcroft and Ullman 1979 (p.229, exercise 9.2) call it a normal form for right-linear grammars.
3. Hopcroft and Ullman 1979, p.218-219, Theorem 9.1 and 9.2
4. Hopcroft and Ullman 1979, p.229, Exercise 9.2