Title: Pattern language (formal languages)

URL: https://en.wikipedia.org/wiki/Pattern_language_(formal_languages)

PageID: 40946774

Categories: Category:Formal languages, Category:Machine learning, Category:Theoretical computer science



---

In theoretical computer science , a pattern language is a formal language that can be defined as the set of all particular instances of a string of constants and variables. Pattern Languages were introduced by Dana Angluin in the context of machine learning . [ 1 ]

Definition

Given a finite set $\Sigma$ of constant symbols and a countable set X of variable symbols disjoint from $\Sigma$, a pattern is a finite non-empty string of symbols from $\Sigma \cup X$ .

The length of a pattern p , denoted by | p |, is just the number of its symbols.

The set of all patterns containing exactly n distinct variables (each of which may occur several times) is denoted by P n , the set of all patterns at all by P * .

A substitution is a mapping $f : P^* \to P^*$ such that [ note 1 ]

f is a homomorphism with respect to string concatenation ($\cdot$), formally: $\forall p, q \in P^* . f(p \cdot q) = f(p) \cdot f(q)$;

f is non-erasing, formally: $\forall p \in P^* . f(p) \neq \varepsilon$, where $\varepsilon$ denotes the empty string ; and

f respects constants, formally: $\forall s \in \Sigma . f(s) = s$ .

If $p = f(q)$ for some patterns $p, q \in P^*$ and some substitution f , then p is said to be less general than q , written $p \leq q$ ;

in that case, necessarily $|p| \geq |q|$ holds.

For a pattern p , its language is defined as the set of all less general patterns that are built from constants only, formally: $L(p) = \{ s \in \Sigma^+ : s \leq p \}$, where $\Sigma^+$ denotes the set of all finite non-empty strings of symbols from $\Sigma$.

For example, using the constants $\Sigma = \{ 0, 1 \}$ and the variables $X = \{ x, y, z, ... \}$, the pattern 0 x 10 xx 1 $\in P_1$ and xxy $\in P_2$ has length 7 and 3, respectively.

An instance of the former pattern is 00 z 100 z 0 z 1 and 01 z 101 z 1 z 1, it is obtained by the substitution that maps x to 0 z and to 1 z , respectively, and each other symbol to itself. Both 00 z 100 z 0 z 1 and 01 z 101 z 1 z 1 are also instances of xxy . In fact, L (0 x 10 xx 1) is a subset of L ( xxy ). The language of the pattern x 0 and x 1 is the set of all bit strings which denote an even and odd binary number , respectively. The language of xx is the set of all strings obtainable by concatenating a bit string with itself, e.g. 00, 11, 0101, 1010, 11101110 $\in L(xx)$.

Properties

The problem of deciding whether $s \in L(p)$ for an arbitrary string $s \in \Sigma^+$ and pattern p is NP-complete (see picture),

and so is hence the problem of deciding $p \leq q$ for arbitrary patterns p , q . [ 2 ]

The class of pattern languages is not closed under ...

union: e.g. for $\Sigma = \{0,1\}$ as above , $L(01) \cup L(10)$ is not a pattern language;

complement: $\Sigma^+ \setminus L(0)$ is not a pattern language;

intersection: $L(x\,0\,y) \cap L(x\,1\,y)$ is not a pattern language;

Kleene plus : L (0) + is not a pattern language;

homomorphism: f ( L ( x )) = L (0) + is not a pattern language, assuming f (0) = 0 = f (1);

inverse homomorphism : $f^{-1}(111) = \{ 01, 10, 000 \}$ is not a pattern language, assuming f (0) = 1 and f (1) = 11.

The class of pattern languages is closed under ...

concatenation: L ( p )· L ( q ) = L ( p · q );

reversal: L ( p ) rev = L ( p rev ). [ 3 ]

If $p , q \in P 1$ are patterns containing exactly one variable, then $p \leq q$ if and only if $L ( p ) \subseteq L ( q )$;

the same equivalence holds for patterns of equal length. [ 4 ] For patterns of different length, the above example p = 0 x 10 xx 1 and q = xxy shows that $L ( p ) \subseteq L ( q )$ may hold without implying $p \leq q$ .

However, any two patterns p and q , of arbitrary lengths, generate the same language if and only if they are equal up to consistent variable renaming. [ 5 ] Each pattern p is a common generalization of all strings in its generated language L ( p ), modulo associativity of (·).

Location in the Chomsky hierarchy

In a refined Chomsky hierarchy , the class of pattern languages is a proper superclass and subclass of the singleton [ note 2 ] and the indexed languages , respectively, but incomparable to the language classes in between; due to the latter, the pattern language class is not explicitly shown in the table below .

The class of pattern languages is incomparable with the class of finite languages , with the class of regular languages , and with the class of context-free languages :

the pattern language L ( xx ) is not context-free (hence neither regular nor finite ) due to the pumping lemma ;

the finite (hence also regular and context-free) language { 01, 10 } is not a pattern language.

Each singleton language is trivially a pattern language, generated by a pattern without variables.

Each pattern language can be produced by an indexed grammar :

For example, using $\Sigma = \{ a , b , c \}$ and $X = \{ x , y \}$,

the pattern a x b y c x a y b is generated by a grammar with nonterminal symbols $N = \{ S x , S y , S \} \cup X$ , terminal symbols $T = \Sigma$, index symbols $F = \{ a x , b x , c x , a y , b y , c y \}$, start symbol S x , and the following production rules:

An example derivation is:

S x [] ⇒ S x [ b x ] ⇒ S x [ a x b x ] ⇒ S y [ a x b x ] ⇒ S y [ c y a x b x ] ⇒ S [ c y a x b x ] ⇒ a x [ c y a x b x ] b y [ c y a x b x ] c x [ c y a x b x ] a y [ c y a x b x ] b ⇒ a x [ a x b x ] b y [ c y a x b x ] c x [ c y a x b x ] a y [ c y a x b x ] b ⇒ a a x [ b x ] b y [ c y a x b x ] c x [ c y a x b x ] a y [ c y a x b x ] b ⇒ a ab x [] b y [ c y a x b x ] c x [ c y a x b x ] a y [ c y a x b x ] b ⇒ a ab b y [ c y a x b x ] c x [ c y a x b x ] a y [ c y a x b x ] b ⇒ ... ⇒ a ab b c y [] c x [ c y a x b x ] a y [ c y a x b x ] b ⇒ a ab b c c x [ c y a x b x ] a y [ c y a x b x ] b ⇒ ... ⇒ a ab b c c ab x [] a y [ c y a x b x ] b ⇒ a ab b c c ab a y [ c y a x b x ] b ⇒ ... ⇒ a ab b c c ab a c y [] b ⇒ a ab b c c ab a c b

In a similar way, an index grammar can be constructed from any pattern.

Learning patterns

Given a sample set S of strings, a pattern p is called descriptive of S if $S \subseteq L ( p )$, but not $S \subseteq L ( q ) \subset L ( p )$ for any other pattern q .

Given any sample set S , a descriptive pattern for S can be computed by

enumerating all patterns (up to variable renaming) not longer than the shortest string in S ,

selecting from them the patterns that generate a superset of S ,

selecting from them the patterns of maximal length, and

selecting from them a pattern that is minimal with respect to ≤. [ 6 ]

Based on this algorithm, the class of pattern languages can be identified in the limit from positive examples. [ 7 ]

Notes

References

v

t

e

Type-0

—

Type-1

—

—

—

—

—

Type-2

—

—

Type-3

—

—

Unrestricted

(no common name)

Context-sensitive

Positive range concatenation

Indexed

—

Linear context-free rewriting systems

Tree-adjoining

Context-free

Deterministic context-free

Visibly pushdown

Regular

—

Non-recursive

Recursively enumerable

Decidable

Context-sensitive

Positive range concatenation *

Indexed *

—

Linear context-free rewriting language

Tree-adjoining

Context-free

Deterministic context-free

Visibly pushdown

Regular

Star-free

Finite

Turing machine

Decider

Linear-bounded

PTIME Turing Machine

Nested stack

Thread automaton

restricted Tree stack automaton

Embedded pushdown

Nondeterministic pushdown

Deterministic pushdown

Visibly pushdown

Finite

Counter-free (with aperiodic finite monoid)

Acyclic finite