Syntactic Pattern Recognition

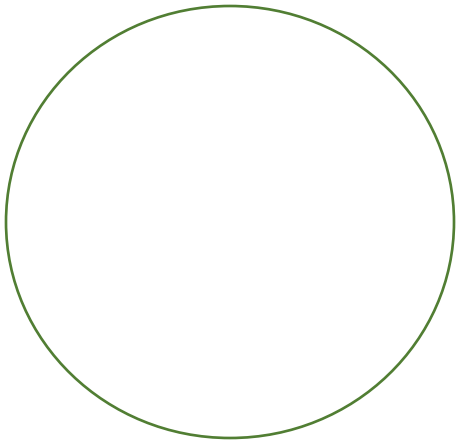
Syntactic Pattern Recognition

- Statistical pattern recognition attempts to classify patterns based on a set of extracted features and an underlying statistical model for the generation of these patterns. Ideally, this is achieved with a rather straightforward method.
  - determine the feature vector,
  - train the system,
  - classify the patterns
- Patterns that include structural or relational information are difficult to represent as feature vectors.
- Syntactic pattern recognition uses this structural information for classification and description.

Some simple geometrical patterns

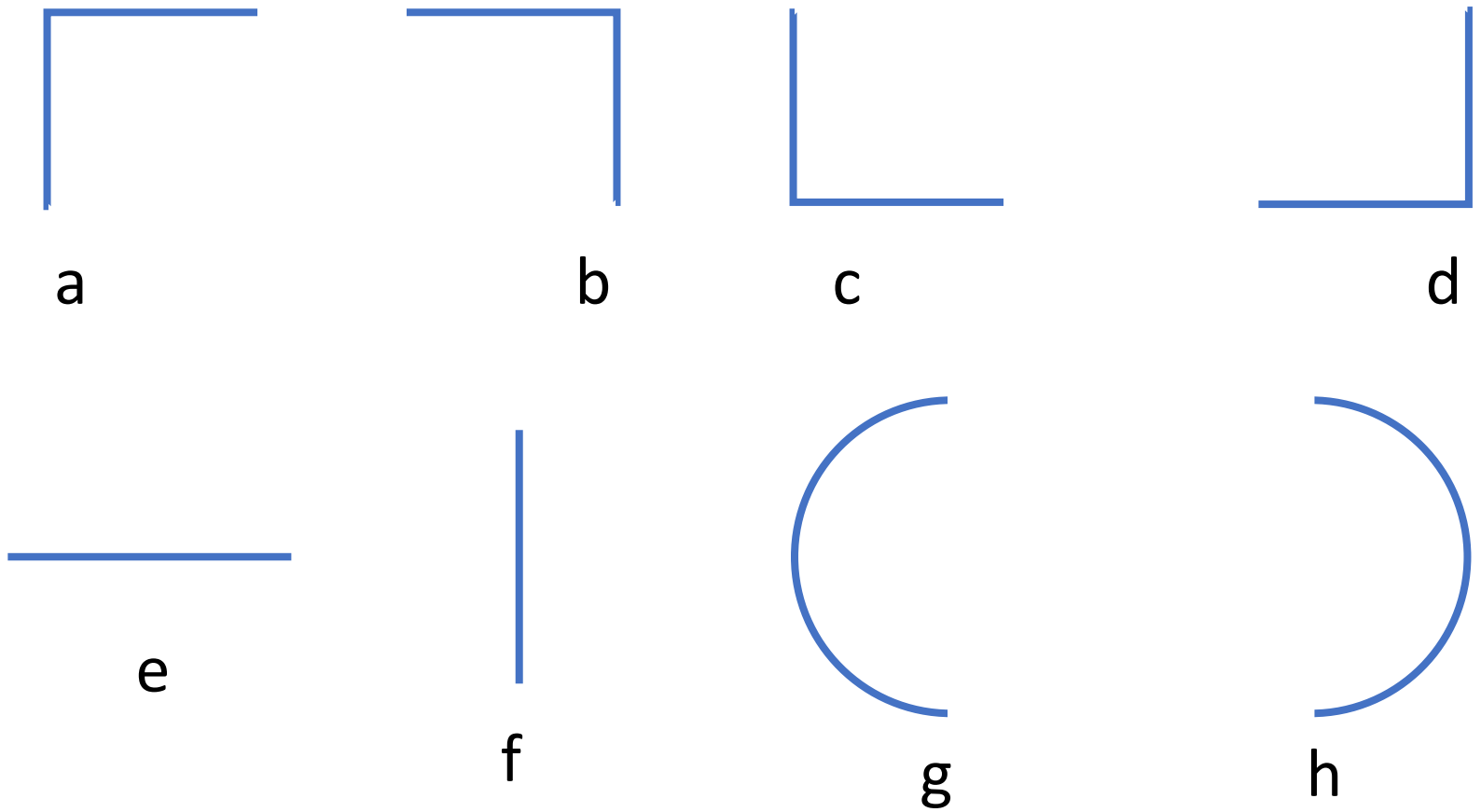
Square

Rectangle

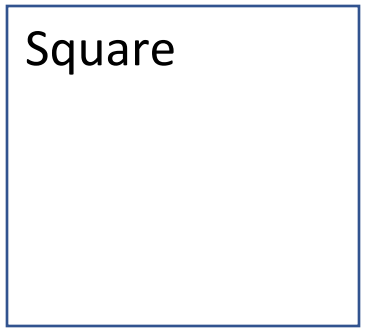


Circle

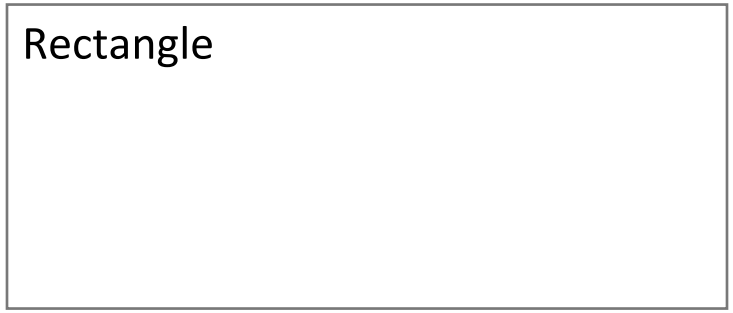
Primitive patterns or sub-patterns required for classification



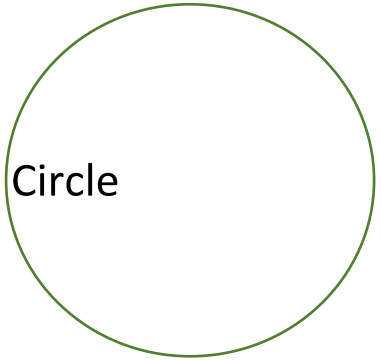
Description of patterns in terms of sub-patterns



a e b f d e c

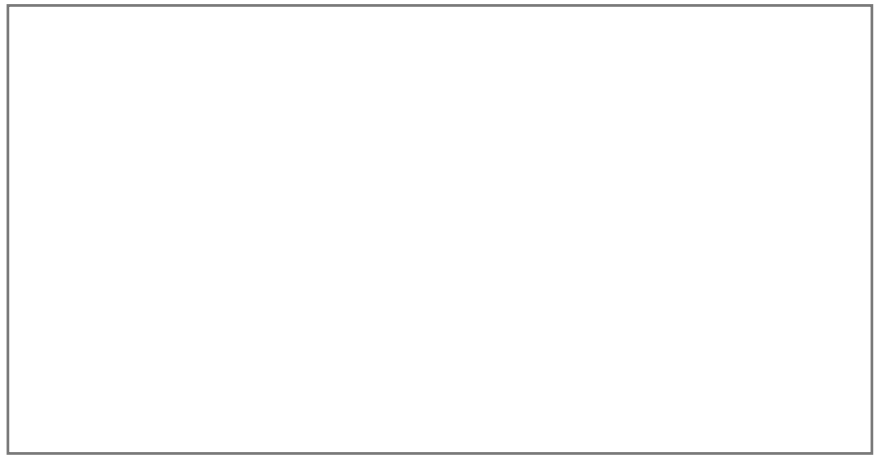


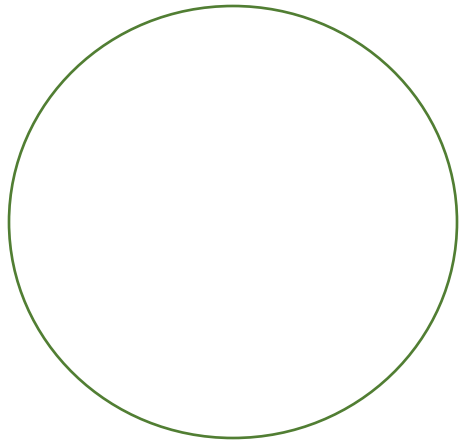
a e e b f d e e c



g h

Problems in classical approach of PR in handling patterns having typical geometrical shapes





Syntactic Pattern Recognition

- Grammars can be used to create a definition of the structure of each pattern class.
- Then, recognition and classification are done using either:
  - Parsing using formal grammars (called Syntactic Pattern Recognition)

    or
  - Relational graph matching(called Hierarchical approach)

Classification

- Classification can be done based on a measure of structural similarity in patterns.
- Each pattern class can be represented by a structural representation or description.

Description

- A description of the pattern structure is useful for recognizing entities when a simple classification isn’t possible.
- Can also describe aspects that cause a pattern not to be assigned to a particular class.
- In complex cases, recognition can only be achieved through a description for each pattern rather than through classification.

Definitions

- The simplest sub-patterns are called pattern primitives, and should be much easier to recognize than the overall patterns.
- The language used to describe the structure of the patterns in terms of sets of pattern primitives is called the pattern description language.
- The pattern description language will have a grammar that specifies how primitives can be composed into patterns

Syntax Analysis

- When a primitive within the pattern is identified, syntax analysis (parsing) is performed on the sentence describing the pattern to determine if it is correct with respect to the grammar.
- Syntax analysis also gives a structural description of the sentence associated with the pattern.
- One advantage of this approach is that a grammar (rewriting) rule can be applied many times.

Pattern Description Using Grammar

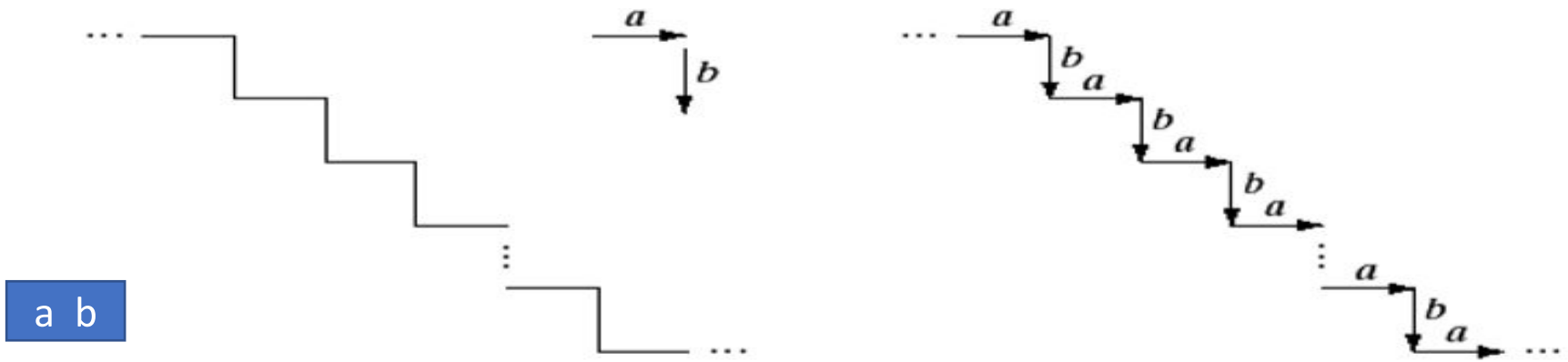


Fig 1: (a) Staircase structure (b) structure encoded in terms of primitive a and b to yield the description …ababab…

Pattern Description Using Grammar

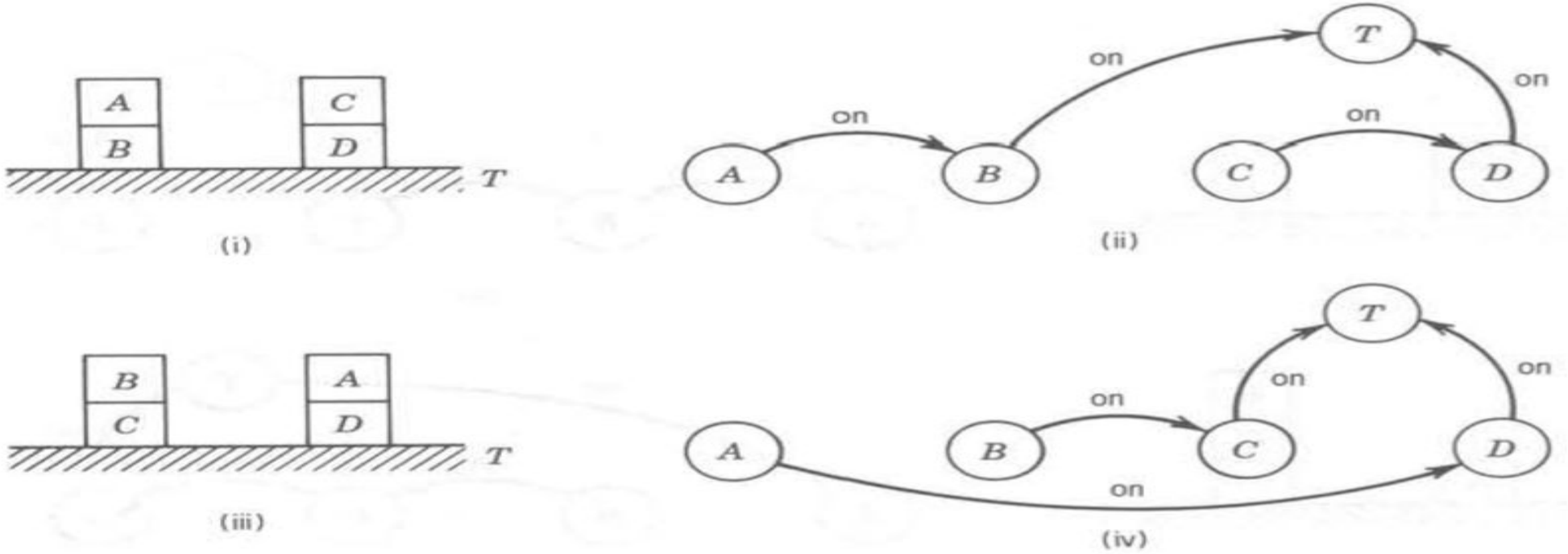


Fig 2: A grammar describing four blocks arranged in 2-block stacks. (i) An example. (ii) Graphical description corresponding to (i). (iii) Another example. (iv) Graphical description corresponding to (iii).

Pattern Description Using Grammar

 A grammar describing four blocks arranged in 2-block stacks:

  V_T = { table , block , + , ↑ } (terminal symbols)

V_N = { DESK,LEFT STACK,RIGHT STACK}
(non-terminal symbols)

S = DESK $\in$ V_N (root symbol)

P = {DESK $\rightarrow$ LEFT STACK + RIGHT STACK
DESK $\rightarrow$ RIGHT STACK + LEFT STACK
LEFT STACK $\rightarrow$ block ↑ block ↑ table
RIGHT STACK $\rightarrow$ block ↑ block ↑ table}
(production rules)

Pattern Description Using Grammar

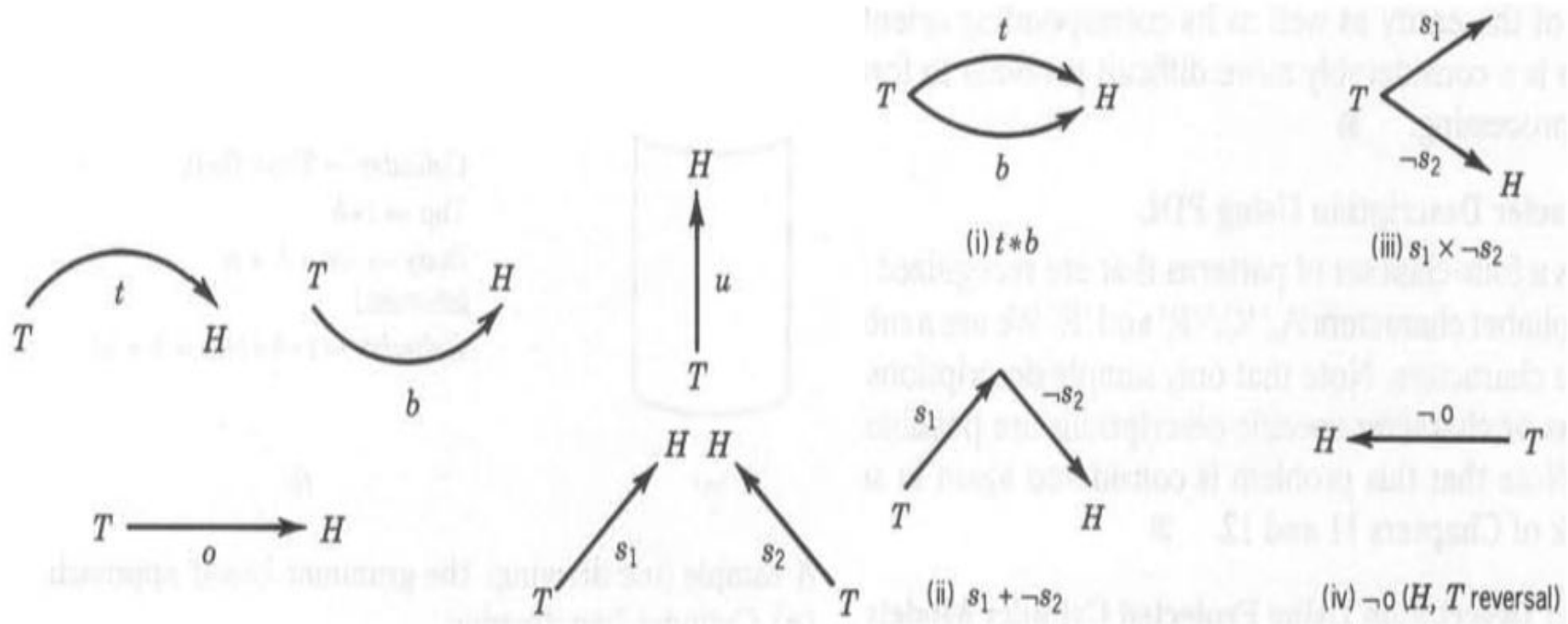


Fig 3: A 2-D line drawing picture description grammar

A set of terminal symbols { t, b, u, o, s ,*,-,+} where

+ represents head to tail concatenation,

* represents head-head and tail-tail attachment,

– represents the head-tail reversal. (x should be * in (iii))

Pattern Description Using Grammar

- .

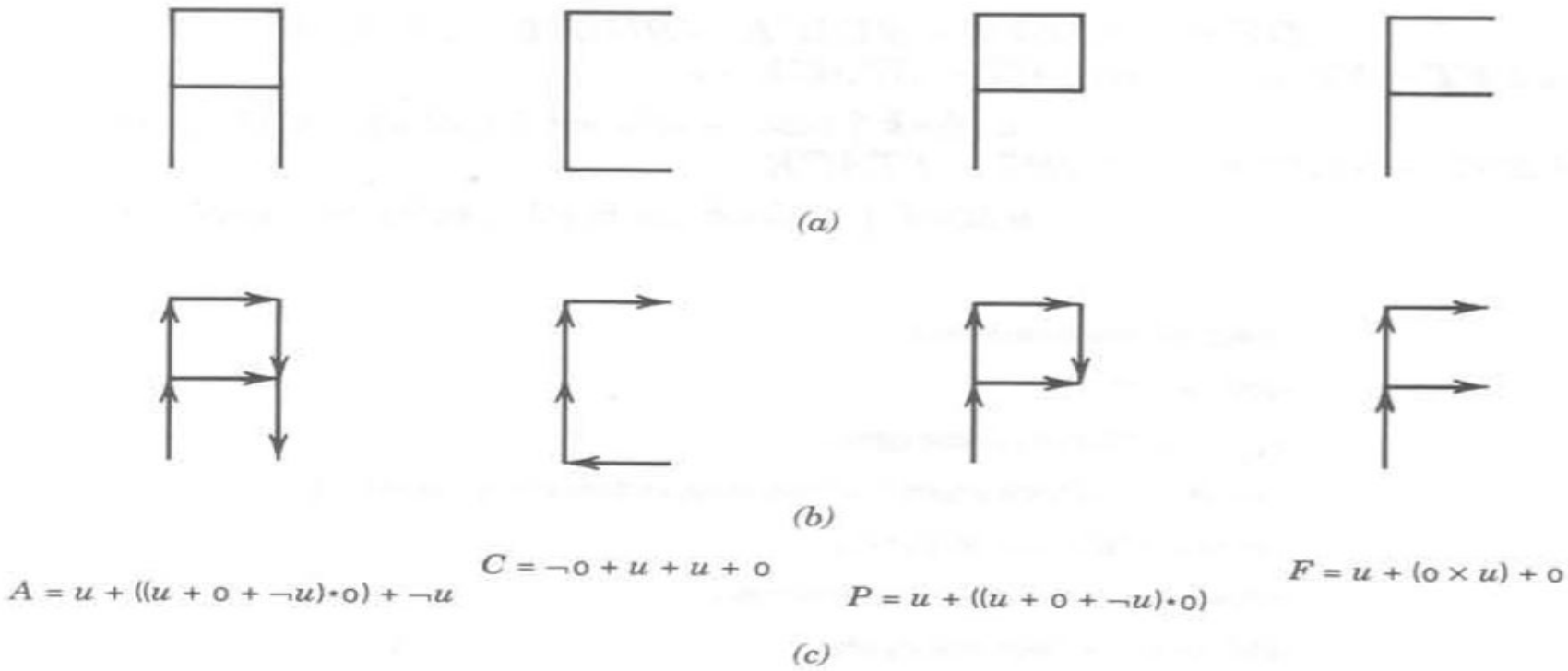
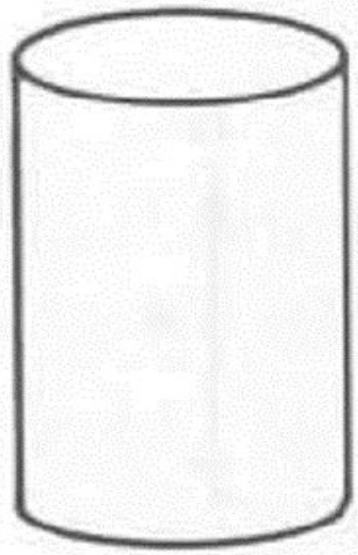


Fig 4: Representation of four characters using the line drawing picture description grammar (a) Pattern data. (b) Primitive representation and interconnection. (c) Corresponding descriptions

Pattern Description Using Grammar



$$Cylinder \rightarrow Top * Body$$
$$Top \rightarrow t * b$$
$$Body \rightarrow \neg u + b + u$$

(alternate:)

$$Cylinder \rightarrow t * b * (\neg u + b + u)$$

Fig 5: Representation of a cylinder using the line drawing picture description grammar

When to Use It

- Picture recognition and scene analysis are problems in which there are a large number of features and the patterns are complex.
  - For example, recognizing areas such as highways, rivers, and bridges in satellite pictures.
- In this case, a complex pattern can be described in terms of a hierarchical composition of simpler sub patterns.

Syntactic System

.

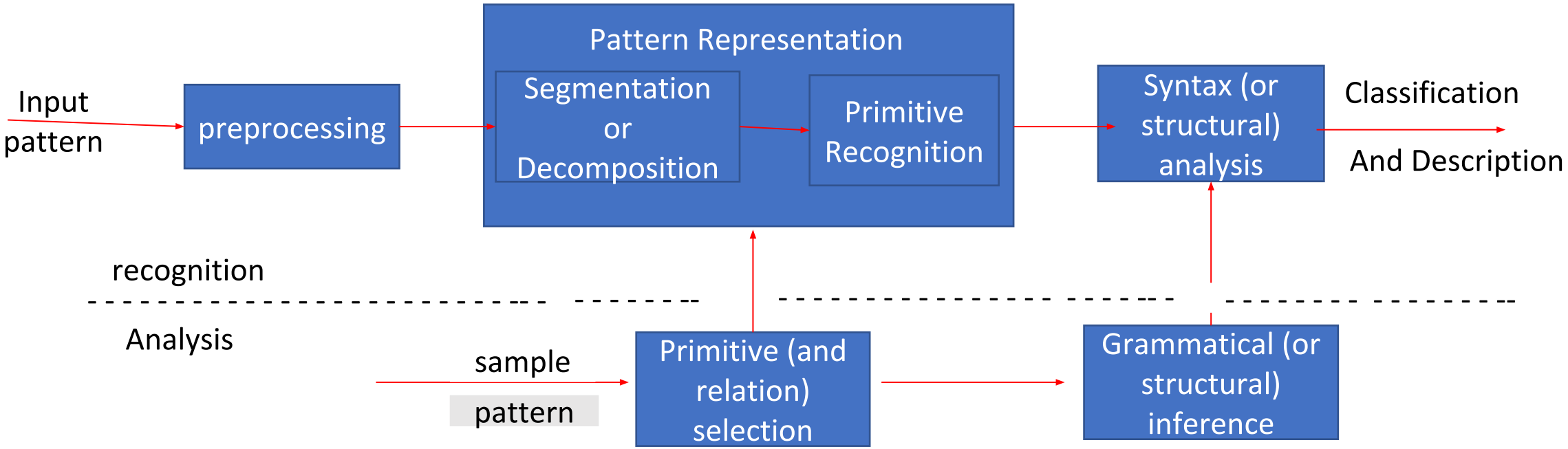


Fig 6 :Block Diagram of a syntactic pattern recognition system

Syntactic System

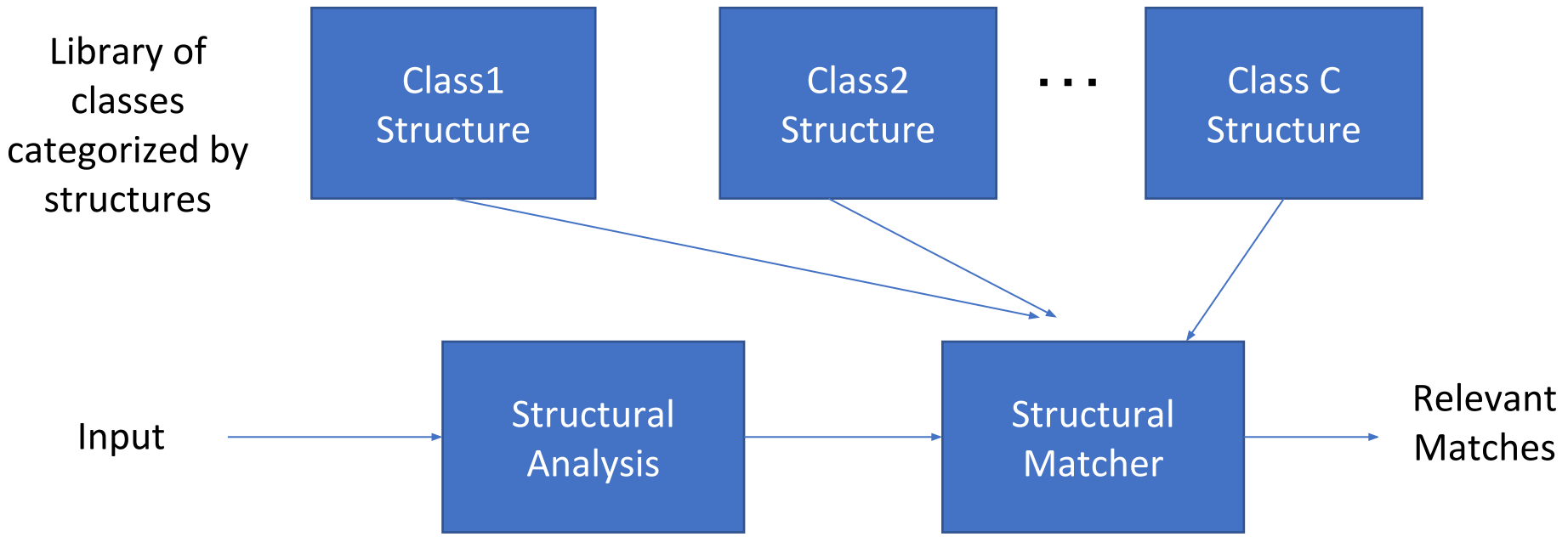


Fig 7: Using Syntactic Patter Recognition for Classification

Syntactic System

- Consists of two main parts:
  - Analysis -primitive selection and grammatical or structural inference
  - Recognition -preprocessing, segmentation or decomposition, primitive and relation recognition, and syntax analysis.
- Preprocessing includes the tasks of pattern encoding and approximation, filtering, restoration, and enhancement.
- After preprocessing, the pattern is segmented into sub-patterns and primitives using predefined operations.
- Sub-patterns are identified with a given set of primitives, so each pattern is represented by a set of primitives with the specified syntactic operations.

Syntax Parsing

- For example, using the concatenation operation, each pattern is recognized by a string of concatenated primitives.
- At this point, the parser will determine if the pattern is syntactically correct.
  - It belongs to the class of patterns described by the grammar if it is correct.
- During parsing/syntax analysis, a description is produced in terms of a parse tree, assuming the pattern is syntactically correct.
- If it isn’t correct, it will either be rejected or analyzed based on a different grammar, which could represent other possible pattern classes.

Matching

- The simplest form of recognition is template matching, in which a string of primitives representing an input pattern is compared to strings of primitives representing reference patterns.
- The input pattern is classified in the same class as the prototype that is the best match, which is determined by a similarity criterion.

Matching vs. Complete Parsing

- In this case, the structural description is ignored.
- The opposite approach is a complete parsing that uses the entire structural description.
- There are many intermediate approaches; for example, a series of tests designed to test the occurrence of certain primitives, sub-patterns, or combinations of these. The result of these tests will determine a classification

- Parsing is required if the problem necessitates using a complete pattern description for recognition.
- Efficiency of the recognition process is improved by simpler approaches that do not require a complete parsing.
- Basically, parsing can be expensive, so don’t use it unnecessarily.

Inferring Grammars

- Grammatical inference machine similar to “learning” in the discriminant approach; it infers a grammar from a set of training patterns.
- The inferred grammar can then be used for pattern description and syntax analysis.

Hierarchical Approach

- The hierarchical approach comes from the similarity that can be seen between the structure of patterns and the syntax or grammar of languages.
- Basic idea: use graphs to represent structural relationships
  - Primitives: nodes/vertices of a graph
  - Relationships: arcs/edges of a graph
- Essential component: graph matching
- Following this analogy, patterns can be built up from sub-patterns in a number of ways, similarly to how one builds words by concatenating characters, and builds a phrase or sentence by concatenating words.

Definitions

- Homomorphism:
  - A function that relabels one graph into another.
  - Merging nodes is allowed so long as it preserves the edge relationships.
- Isomorphism:
  - A homomorphism that is 1-to-1 and onto.
  - Merging nodes not allowed—exact match with nothing changed but labels.

Quick Rejection Criteria

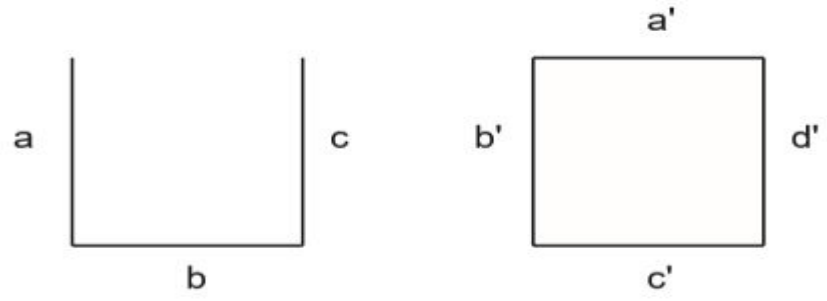
- For a match to exist, several things must hold:
  - same number of vertices
  - same number of nodes
  - in-degree of vertices must match
  - out-degree of vertices must match
  - cycles and lengths must match
- Failure of any of these to match means you can reject the match, but determine the isomorphism requires that you actually find the relabeling

Finding Isomorphisms

- Use an adjacency matrix representation of the graph
- Reorder the rows and corresponding columns
- Compare all possible reorderings
- Comparison: $O(n^2)$
- Possible Permutations: $O(n!)$
- Only really feasible for small n.

Subisomorphisms

- Harder problem: is one pattern part of another one?
- Equivalent to finding subgraph that is isomorphic to another: subisomorphism.
- Finding subisomorphisms not only requires determining isomorphism between graph and subgraph, but also considering all possible subgraphs.

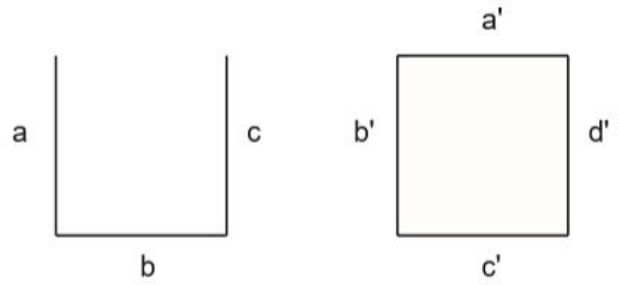


Graph Similarity

- What if we want to find not only identical matches but similar patterns?
- What does it mean for graphs to be similar?
- One way is to count modifications required to make one match the other
  - inserting an edge
  - deleting an edge
  - inserting a node
  - deleting a node
  - splitting a node
  - merging two nodes

Match Graphs

- I Create a new graph where
  - each node is a pairing between a node from one attributed graph and compatible node from the other, and
  - each edge represents consistency (mutual compatibility) between two pairing
- A subisomorphism is indicated by a completely-connected subgraph (clique) of the match graph.
- The best match is the largest clique.



Attributed Graphs

- Idea: can relabel graph nodes, but must match apples to apples and oranges to oranges.
- An attributed graph has attributes attached to each vertex.
- Matching attributed graphs must not only produce a (sub)isomorphism but must also match attributes