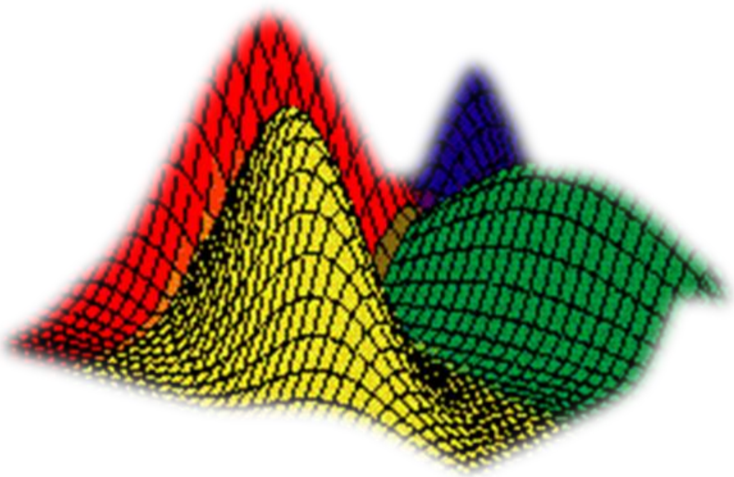


CSE 473: Pattern Recognition

Recall the Pattern Recognition Approaches So Far

- Determine feature vector $\underline{x}$
- Train a system
- Classify the unknown pattern

Recall the Pattern Recognition Approaches So Far

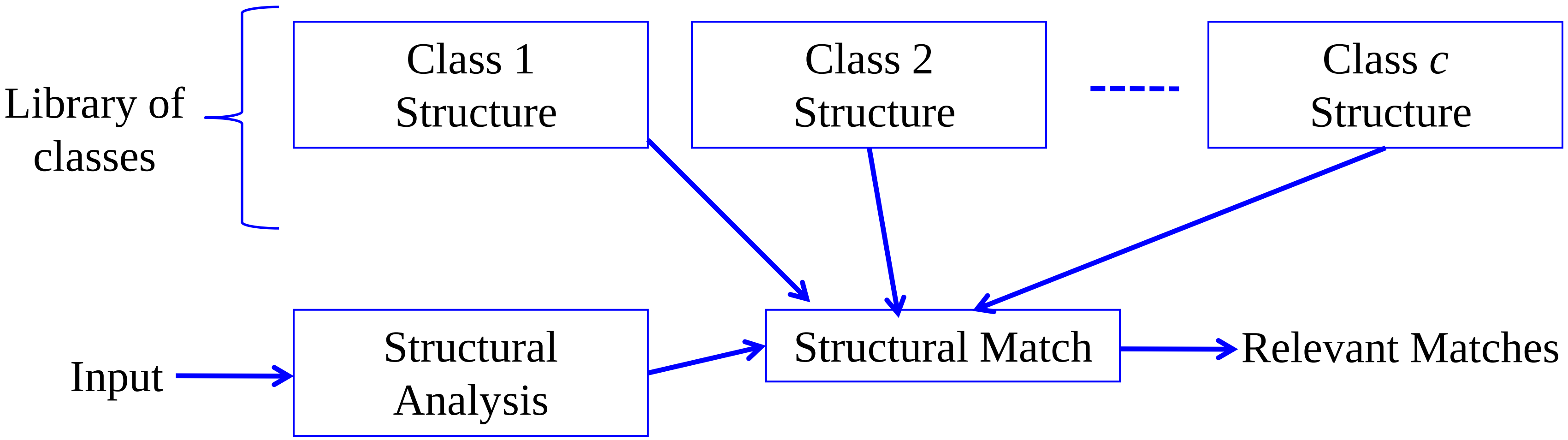
- Determine feature vector $\underline{x}$
- Train a system
- Classify the unknown pattern

- However, many patterns
  - are structural
  - have relational information
  - are difficult to extract *traditional* feature vectors

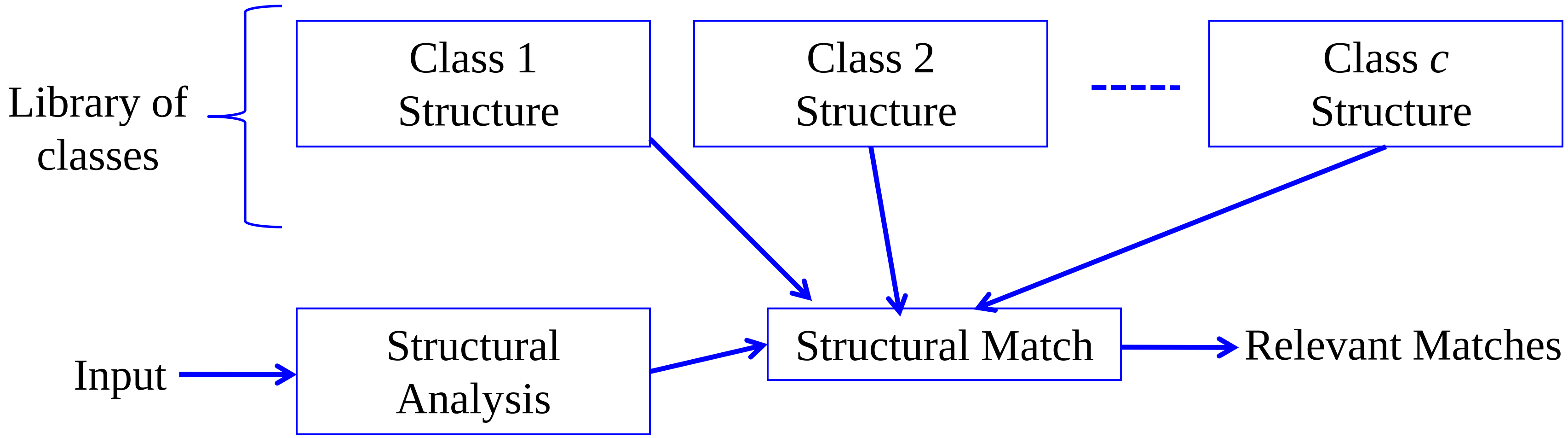
Alternate approach:

Syntactic /Structural Pattern Recognition

Syntactic Pattern Recognition (SyntPR)

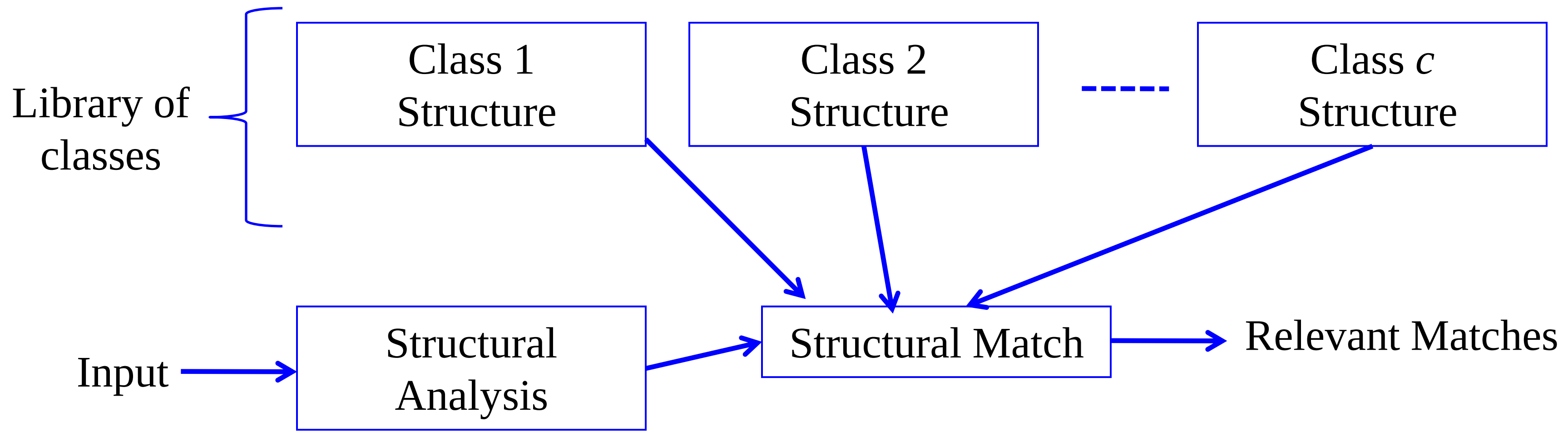


Syntactic Pattern Recognition (SyntPR)



- SyntPR is used for
  - classification
  - description

Syntactic Pattern Recognition (SyntPR)



- SyntPR assumes
  - Pattern is quantifiable
- Quantification is through
  - Formal grammar
  - Relational description or graph

Syntactic Pattern Recognition (SyntPR)

- Formal grammar uses
  - parsing
  - hierarchical decomposition

- Graph based approaches uses matching

SyntPR using Grammars

- Formal grammar can provide
  - efficient use of production rules
  - hierarchical decomposition
  - analysis, description and classification of complex patterns

SyntPR using Grammars

- Example: the baby walks smoothly

SyntPR using Grammars

- Example: the baby walks smoothly

- Use production rules to analyze the sentence:
  - <sentence>

SyntPR using Grammars

- Example: the baby walks smoothly
- Use production rules to analyze the sentence:
  - <sentence>
  - <noun phrase> <verbal phrase>

SyntPR using Grammars

- Example: the baby walks smoothly
- Use production rules to analyze the sentence:
  - <sentence>
  - <noun phrase> <verbal phrase>
  - <article> <noun> <verbal phrase>

SyntPR using Grammars

- Example: the baby walks smoothly

- Use production rules to analyze the sentence:
  - <sentence>
  - <noun phrase> <verbal phrase>
  - <article> <noun> <verbal phrase>
  - the baby <verbal phrase>
    -
    -

Preliminaries of Grammars

- An alphabet, V: nonempty, finite set of primitive symbols
  - Example: $V = \{a, b, c, \ldots, z\}$

Preliminaries of Grammars

- An alphabet, V: nonempty, finite set of primitive symbols
  - Example: $V = \{a, b, c, \ldots, z\}$

- String of different sizes:
  - V
  - $V^2 = V \circ V$, Example: ac, bx, jk, . . .
  - $V^+ = V \cup V^2 \cup V^3 \cup \ldots$
  - $V^* = \{\epsilon\} \cup V \cup V^2 \cup V^3 \cup \ldots$

Preliminaries of Grammars

- Language, *L*: set of strings
  - $L \subseteq V^*$
  - Union: $L_1 \bigcup L_2 = \{s \mid s \in L_1 \text{ or } s \in L_2\}$

Preliminaries of Grammars

- Language, L: set of strings
  - $L \subseteq V^*$
  - Union: $L_1 \bigcup L_2 = \{s \mid s \in L_1 \text{ or } s \in L_2\}$
  - Concatenation:

$$L_1 \circ L_2 = \{s \mid s = s_1 s_2 \text{ where } s_1 \in L_1 \text{ and } s_2 \in L_2\}$$

Elements of a Grammar (G)

- Set of Terminals, V_T
- Set of Non-terminals, V_N
- Set of Productions, P
- Starting Symbol, S

$$G = (V_T, V_N, P, S)$$

Elements of a Grammar (G)

- Set of Terminals, V_T
- Set of Non-terminals, V_N
- Set of Productions, P
- Starting Symbol, S

$$G = (V_T, V_N, P, S)$$

- The language generated by G is $L(G)$

Application Modes of a Grammar

- Generative

- Analytic

Application Modes of a Grammar

- Generative
  - Given a grammar, generate a sentence (string of terminal symbols)

- Analytic

Application Modes of a Grammar

- Generative
  - Given a grammar, generate a sentence (string of terminal symbols)

- Analytic
  - Given a grammar, determine:
    - is sentence, s, generated from G? that is, $s \overset{?}{\in} L(G)$
    - If so, get the structure of the sentence

Application of Grammar in PR

- Analytic
  - Given a grammar, determine:
    - is sentence, *s*, generated from G? that is, $s \overset{?}{\in} L(G)$
    - If so, get the structure of the sentence

- A class can be characterized using a Grammar, *G*
- The unknown pattern can be represented as a sentence, *s*

- Determine: $s \overset{?}{\in} L(G)$

Notations of Grammars

- Capital letters, A, B, C:
  - Non-terminals, V_N

Notations of Grammars

- Capital letters, A, B, C:
  - Non-terminals, V_N

- Small letters, a, b, c:
  - Terminals, V_T

Notations of Grammars

- Capital letters, A, B, C:
  - Non-terminals, V_N
- Small letters, a, b, c:
  - Terminals, V_T
- Greek letters, α, β:
  - strings consists of terminals and/or non-terminals, $(V_T \cup V_N)^*$

Types of Grammars

$$\alpha_1 \rightarrow \beta_2$$

Types of Grammars

$$\alpha_1 \rightarrow \beta_2$$

- Type 0: Free or Unrestricted
  - No restriction on any side

Types of Grammars

$$\alpha_1 \rightarrow \beta_2$$

- Type 0: Free or Unrestricted
  - No restriction on any side

- Type 1: Context sensitive
  - $\beta_2 \neq \varepsilon$
  - $|\alpha_1| \leq |\beta_2|$

Types of Grammars

$$\alpha_1 \rightarrow \beta_2$$

- Type 0: Free or Unrestricted
  - No restriction on any side

- Type 1: Context sensitive
  - $\beta_2 \neq \varepsilon$
  - $|\alpha_1| \leq |\beta_2|$
  - $\alpha \alpha_i \beta \rightarrow \alpha \beta_i \beta$

Types of Grammars

$$\alpha_1 \to \beta_2$$

- Type 1: Context sensitive
  - $\beta_2 \neq \varepsilon$
  - $|\alpha_1| \leq |\beta_2|$
  - $\alpha\alpha_i\beta \to \alpha\beta_i\beta$

- Type 2: Context Free
  - $\alpha_1 = S_1 \in V_N$
  - $S_1 \to \beta_2$

Types of Grammars

$$\alpha_1 \to \beta_2$$

- Type 2: Context Free
  - $\alpha_1 = S_1 \in V_N$
  - $S_1 \to \beta_2$

- Type 3: Regular or Finite State grammar (FSG)
  - $S_1 \to a$
  - $S_1 \to aS_2$

Comparison of Grammars

$$T_0 \qquad T_1 \qquad T_2 \qquad T_3$$ Grammar Types

$$L(T_0) \supset L(T_1) \supset L(T_2) \supset L(T_3)$$ Languages

Increasing Production Constraints →

← Increasing Description Capability

← Increasing Recognition Capability

Graphical Representation of FSG

$$V_T = \{a, b\}$$

$$V_N = \{S, A_1, A_2\}$$

$$P = \{S \to aA_2$$

$$S \to bA_1$$

$$A_1 \to a$$

$$A_1 \to aA_1$$

$$A_2 \to b\}$$

Graphical Representation of FSG

$$V_T = \{a, b\}$$

$$V_N = \{S, A_1, A_2\}$$

$$P = \{S \rightarrow aA_2$$

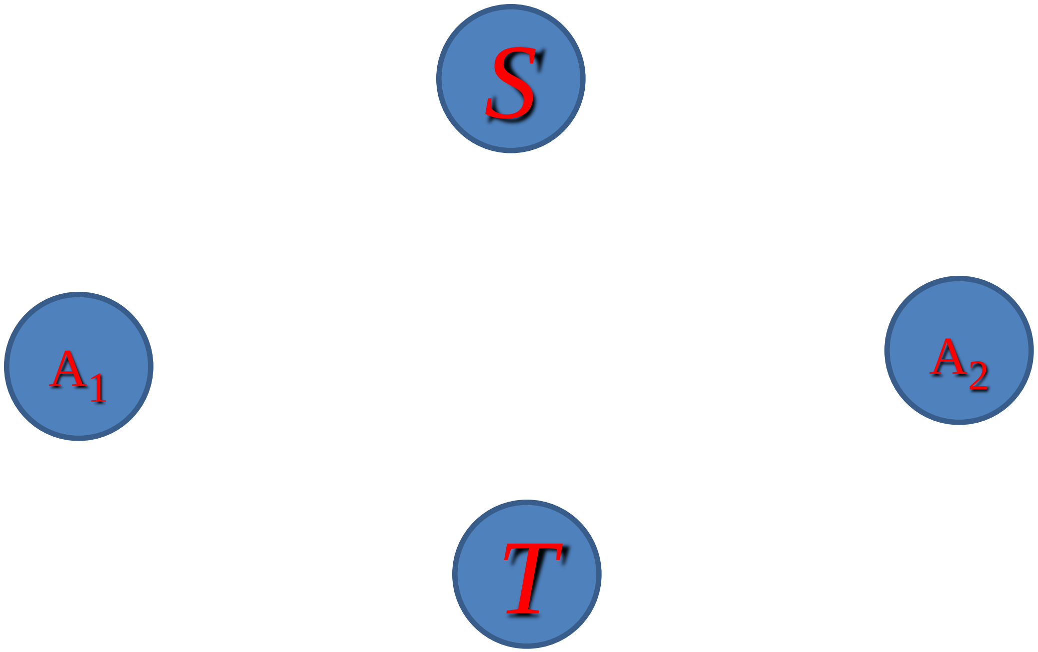
$$S \rightarrow bA_1$$

$$A_1 \rightarrow a$$

$$A_1 \rightarrow aA_1$$

$$A_2 \rightarrow b\}$$

- Get nodes for each non-terminal symbols including *S* and *T*



Graphical Representation of FSG

$$V_T = \{a, b\}$$

$$V_N = \{S, A_1, A_2\}$$

$$P = \{S \to aA_2$$

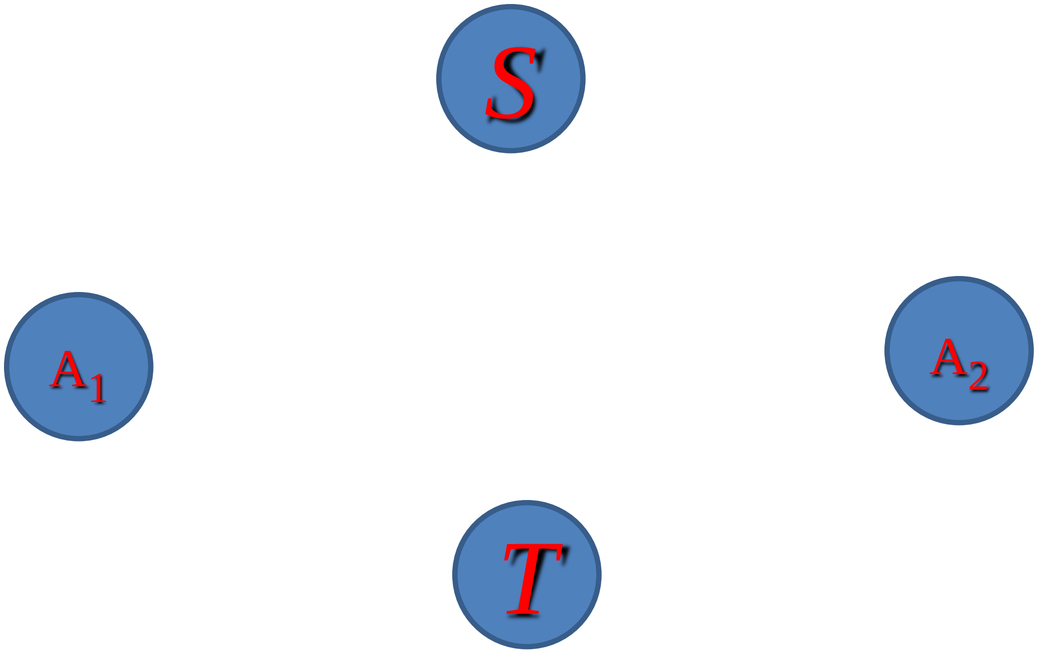
$$S \to bA_1$$

$$A_1 \to a$$

$$A_1 \to aA_1$$

$$A_2 \to b\}$$

- For each A_i->aA_j, make edge from A_i to A_j labeled with a



Graphical Representation of FSG

$$V_T = \{a, b\}$$

$$V_N = \{S, A_1, A_2\}$$

$$P = \{\boxed{S \to aA_2}$$

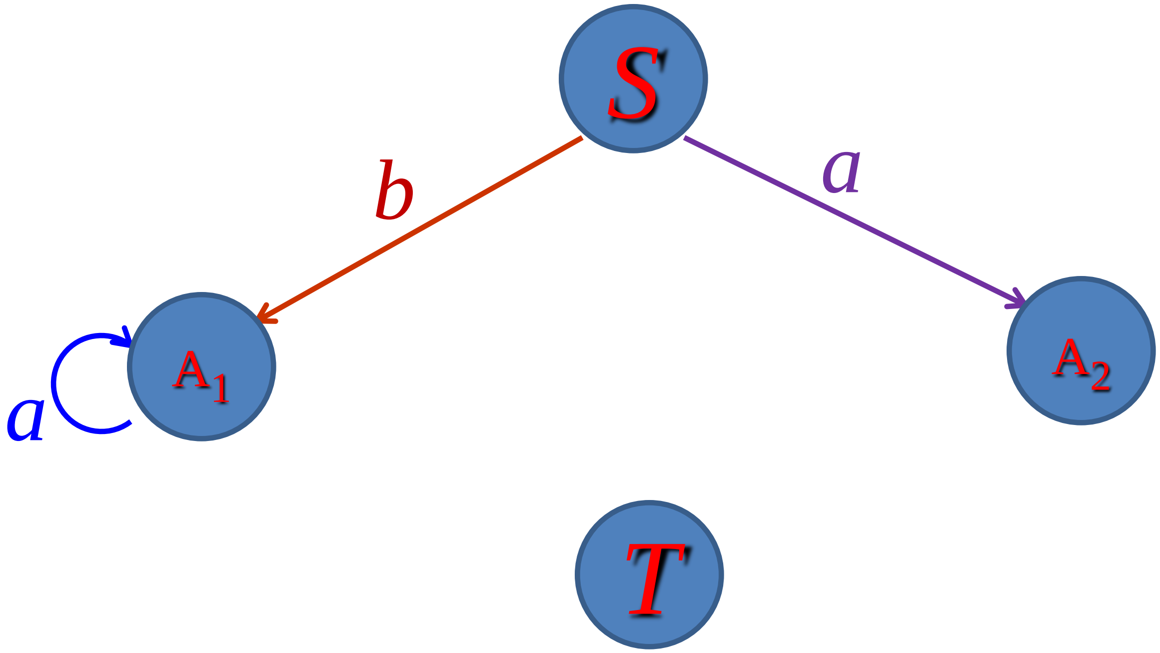
$$\boxed{S \to bA_1}$$

$$A_1 \to a$$

$$\boxed{A_1 \to aA_1}$$

$$A_2 \to b\}$$

- For each A_i->aA_j, make edge from A_i to A_j labeled with a



Graphical Representation of FSG

$V_T = \{a, b\}$

$V_N = \{S, A_1, A_2\}$

$P = \{S \rightarrow aA_2$

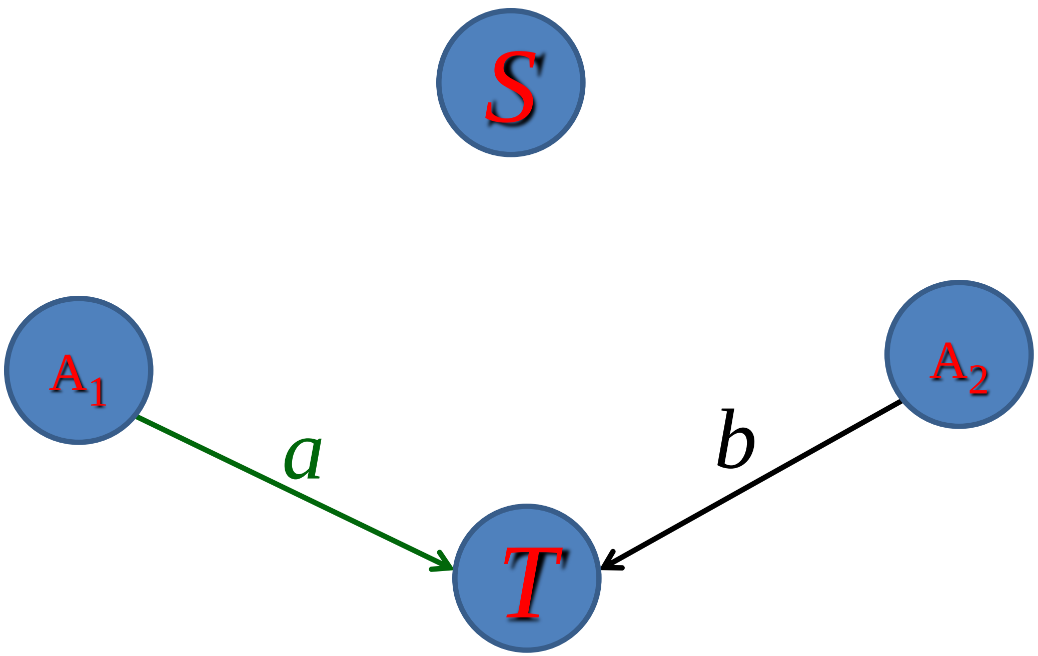
$S \rightarrow bA_1$

$\boxed{A_1 \rightarrow a}$

$A_1 \rightarrow aA_1$

$\boxed{A_2 \rightarrow b}\}$

- For each A_i->a, make edge from A_i to T labeled with a



Graphical Representation of FSG

$$V_T = \{a, b\}$$

$$V_N = \{S, A_1, A_2\}$$

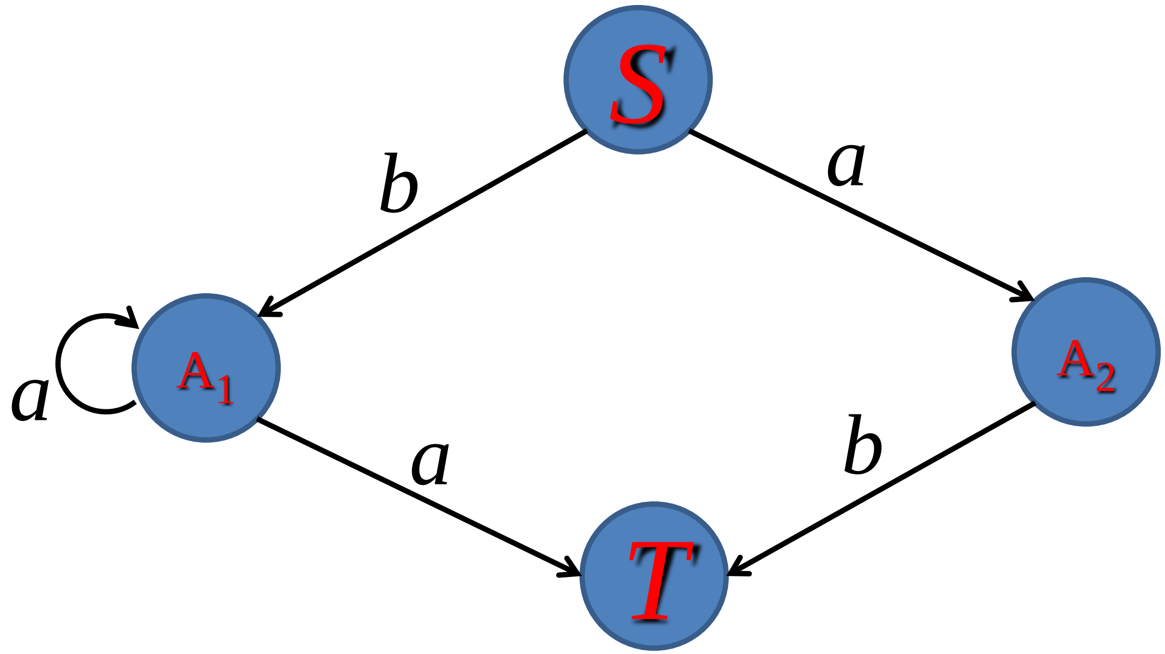
$$P = \{S \to aA_2$$

$$S \to bA_1$$

$$A_1 \to a$$

$$A_1 \to aA_1$$

$$A_2 \to b\}$$



Other Definitions Related To Grammar

- Production vs. Derivation

$$\alpha_1 \to \beta_2 : \quad \text{Production}$$

$$x \Rightarrow x_n : \quad \text{Derivation}$$

- Recursive grammar, if *G* allows

$$S_1 \Rightarrow \alpha S_1 \beta$$

Other Definitions Related To Grammar

- Recursive grammar, if *G* allows

$$S_1 \Rightarrow \alpha S_1 \beta$$

- Cycle Free, if there is no derivation like this

$$x \Rightarrow x_1 \Rightarrow x_2 \cdots \Rightarrow x_n \Rightarrow x$$

Other Definitions Related To Grammar

- Ambiguous Grammar, if *G* allows

- $x \Rightarrow x_1 \Rightarrow x_2 \cdots \Rightarrow x_i' \cdots \Rightarrow x_n \Rightarrow S$
- $x \Rightarrow x_1 \Rightarrow x_2 \cdots \Rightarrow x_i \cdots \Rightarrow x_n \Rightarrow S$

Other Definitions Related To Grammar

- Ambiguous Grammar, if G allows
- $$x \Rightarrow x_1 \Rightarrow x_2 \cdots \Rightarrow \boxed{x_i'} \cdots \Rightarrow x_n \Rightarrow s$$
- $$x \Rightarrow x_1 \Rightarrow x_2 \cdots \Rightarrow \boxed{x_i} \cdots \Rightarrow x_n \Rightarrow s$$
- More that one derivation for the same s
- More that one parse tree for the same s

Other Definitions Related To Grammar

- Equivalence of Grammars
  - G_1 and G_2 are equivalent iff $L(G_1) = L(G_2)$

Other Definitions Related To Grammar

- Equivalence of Grammars
  - G_1 and G_2 are equivalent iff $L(G_1) = L(G_2)$

- covering
  - $G_1 = (V_T^1, V_N^1, P^1, S^1)$ covers $G_2 = (V_T^2, V_N^2, P^2, S^2)$ if there exists mapping f such that
    - $V_N^1 = f(V_N^2)$
    - $S^1 = f(S^2)$
    - P^1 is obtained from P^2 replacing corresponding symbols

Example of Some Grammars

$$\boxed{\begin{array}{l} S \to aAa \\ A \to a \\ A \to b \end{array}}$$

?

$$\boxed{\begin{array}{l} S \to SC \\ CB \to Cb \\ aB \to ab \\ bB \to bb \end{array}}$$

?

$$\boxed{\begin{array}{l} S \to aA_1 \\ S \to bA_1 \\ A_1 \to a \\ A_1 \to b \end{array}}$$

?

Example of Some Grammars

$$
\begin{aligned}
S &\rightarrow aAa \\
A &\rightarrow a \\
A &\rightarrow b
\end{aligned}
$$

CFG

$$
\begin{aligned}
S &\rightarrow SC \\
CB &\rightarrow Cb \\
aB &\rightarrow ab \\
bB &\rightarrow bb
\end{aligned}
$$

CSG

$$
\begin{aligned}
S &\rightarrow aA_1 \\
S &\rightarrow bA_1 \\
A_1 &\rightarrow a \\
A_1 &\rightarrow b
\end{aligned}
$$

FSG

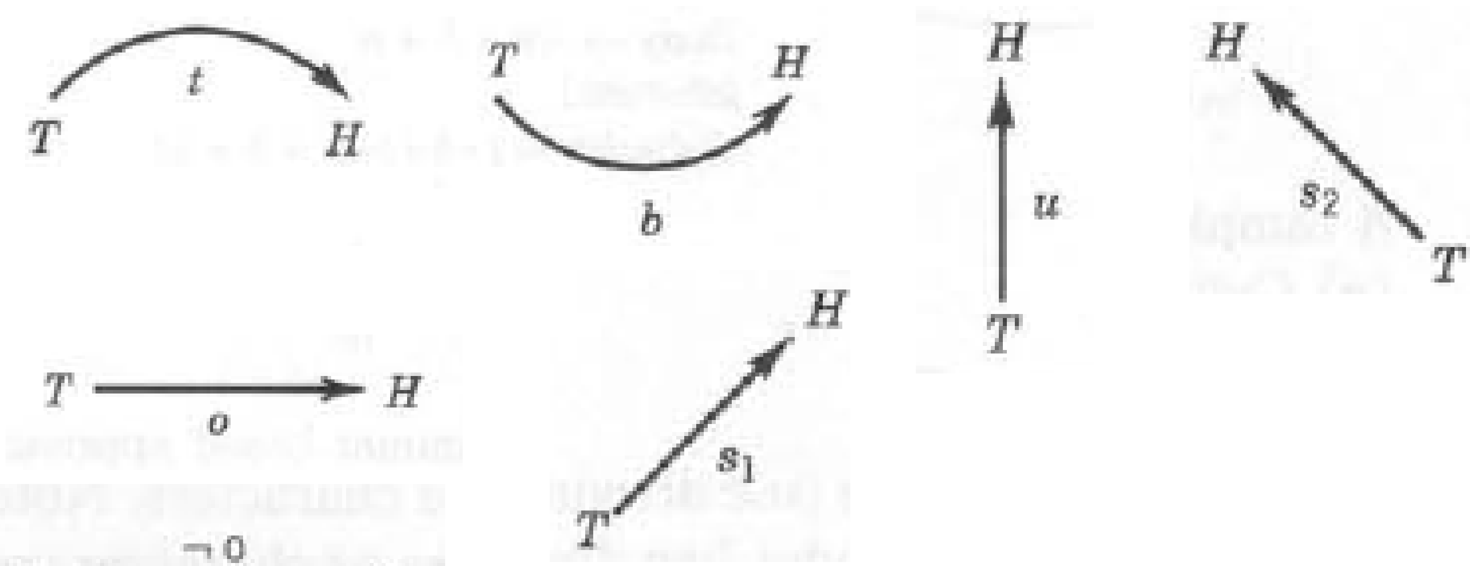
Chomsky Normal Form

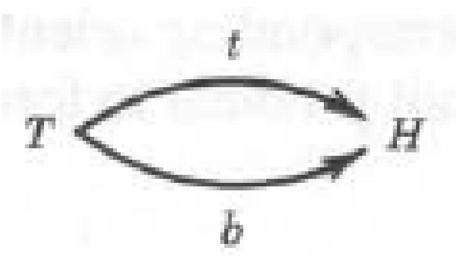
$$S \rightarrow BC \qquad \text{where } A,\ B,\ C \in V_N$$

$$A \rightarrow a \qquad \text{where } A \in V_N \text{ and } a \in V_T$$

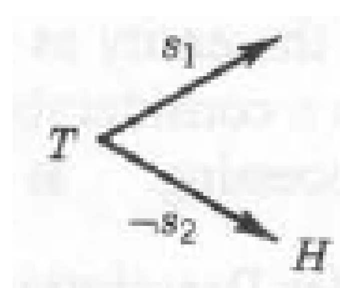
Example: A Line Drawing Grammar

Terminal Symbols

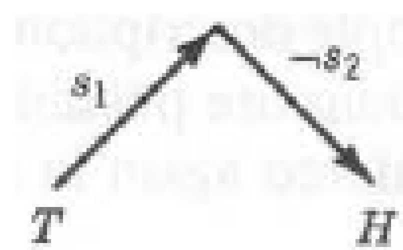




$t * b$



$s_1 \times \neg s_2$



$s_1 + \neg s_2$

Example: A Line Drawing Grammar

$$G_{cyl} = (V_T^{cyl}, V_N^{cyl}, P^{cyl}, S^{cyl})$$

$$V_T^{cyl} = \{t, b, u, o, s_1, s_2, *, \neg, +\}$$

$$V_N^{cyl} = \{top, body, Cylinder\}$$

$$S^{cyl} = Cylinder$$

$$P^{cyl} = \{Cylinder \rightarrow top * body$$

$$top \rightarrow t * b$$

$$body \rightarrow \neg u + b + u\}$$

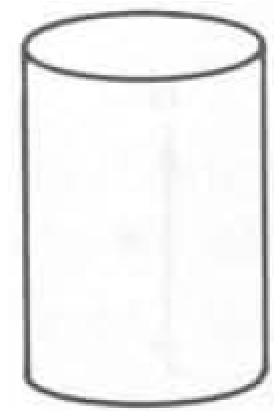
Example: Cylinder

$$G_{cyl} = (V_T^{cyl}, V_N^{cyl}, P^{cyl}, S^{cyl})$$

$$V_T^{cyl} = \{t, b, u, o, s_1, s_2, *, \neg, +\}$$

$$V_N^{cyl} = \{top, body, Cylinder\}$$

$$S^{cyl} = Cylinder$$

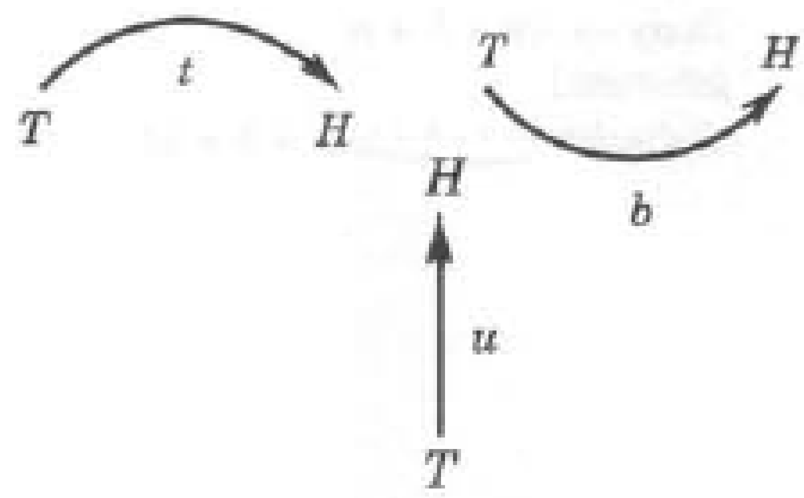


$$Cylinder \rightarrow t * b * (\neg u + b + u)$$

$$P^{cyl} = \{Cylinder \rightarrow top * body$$

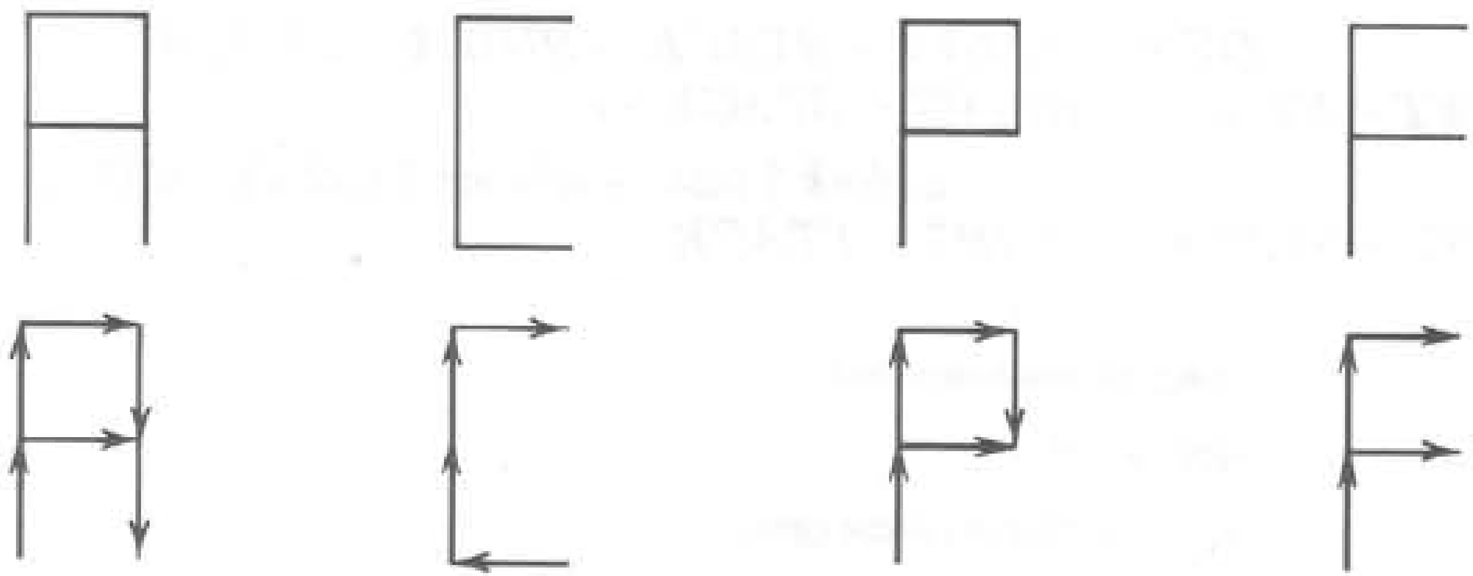
$$top \rightarrow t * b$$

$$body \rightarrow \neg u + b + u\}$$

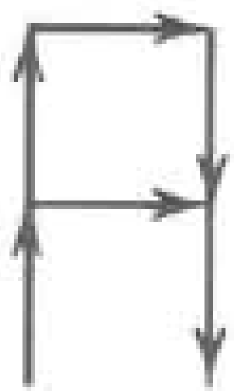


Example 2: Characters

Example 2: Characters



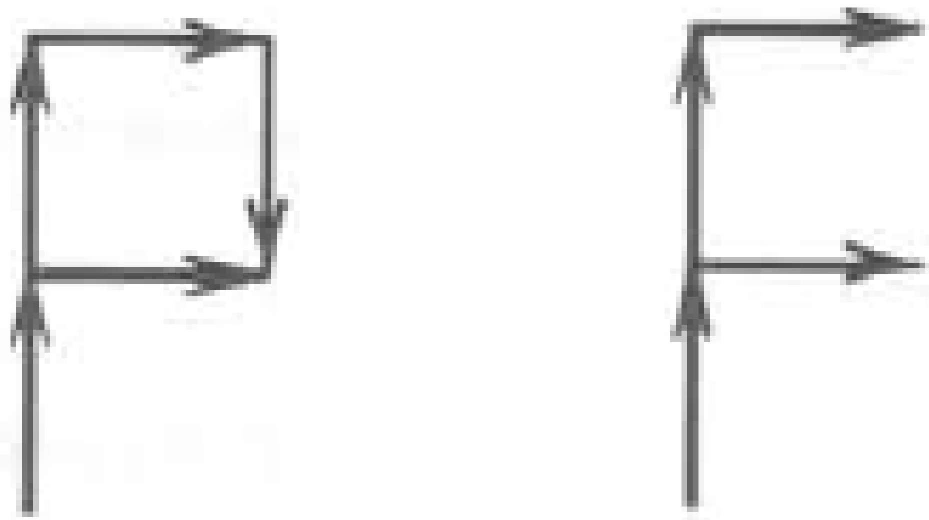
Example 2: Characters



$$A = u + ((u + o + \neg u)*o) + \neg u$$

$$C = \neg o + u + u + o$$

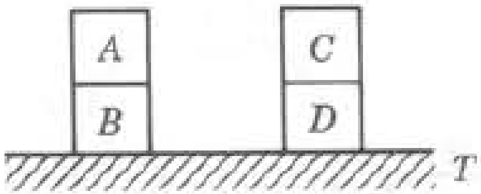
Example 2: Characters

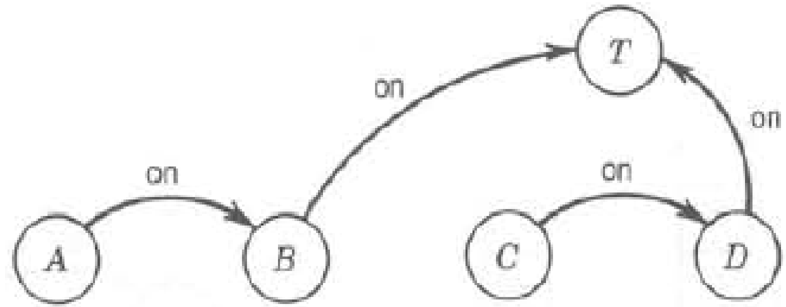


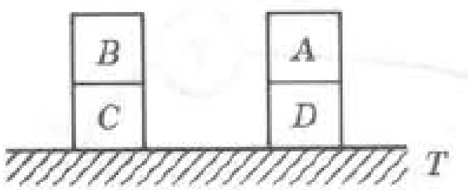
$$P = u + ((u + 0 + -u)*0)$$

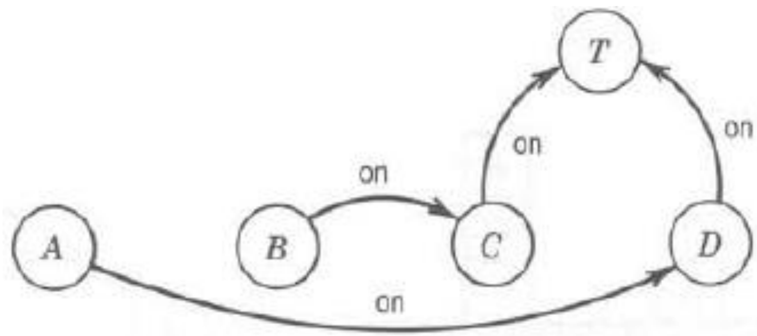
$$F = u + (0 \times u) + 0$$

Example 3: Block World Representation

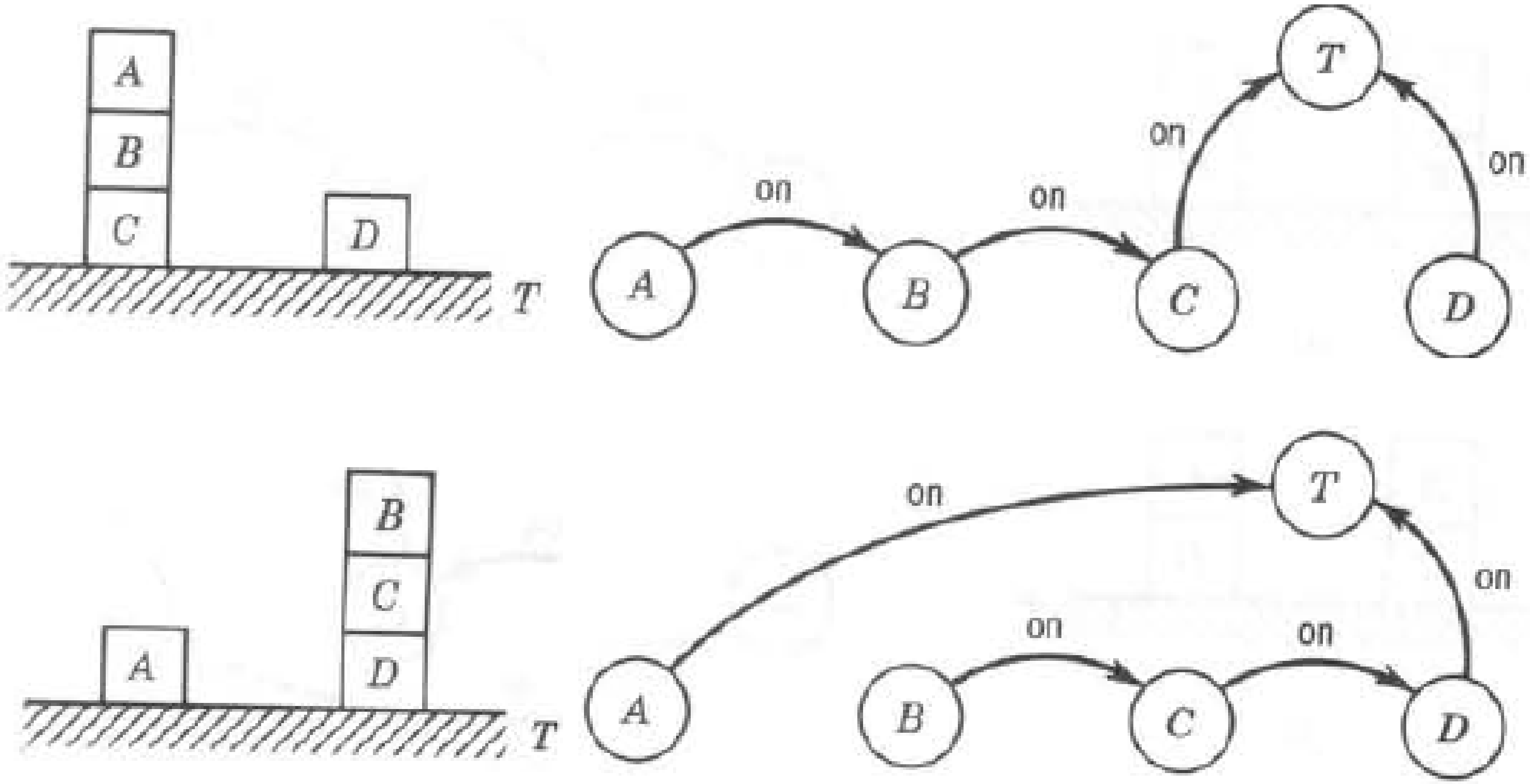








Example 3: Block World Representation



Example 3: Block World Representation

$V_T = \{table, block, +, \uparrow\}$ (terminal symbols)

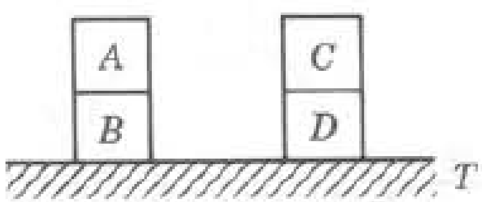
$V_N = \{DESC, LEFT\_STACK, RIGHT\_STACK\}$

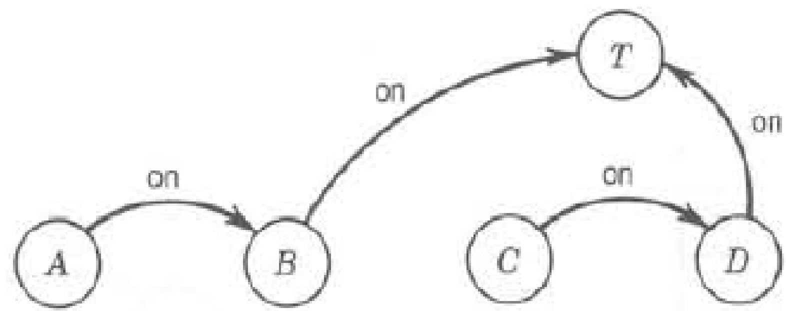
$$S = DESC \in V_N$$

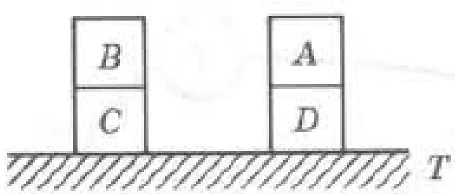
$$\begin{aligned} P = \{&DESC \rightarrow LEFT\_STACK + RIGHT\_STACK \\ &DESC \rightarrow RIGHT\_STACK + LEFT\_STACK \\ &LEFT\_STACK \rightarrow block \uparrow block \uparrow table \\ &RIGHT\_STACK \rightarrow block \uparrow block \uparrow table\} \end{aligned}$$

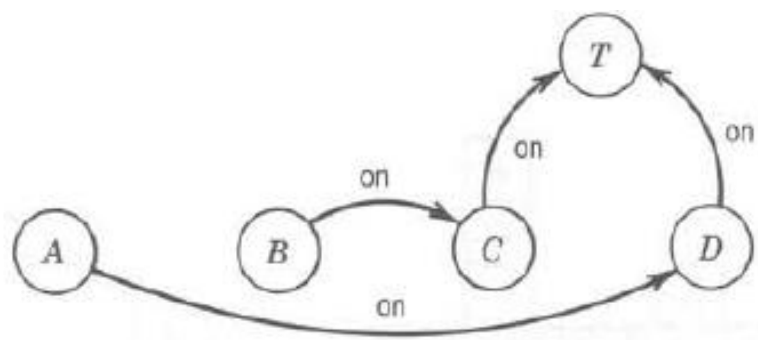
Example 3: Block World Representation

$$
\begin{aligned}
P = \{ & DESC \rightarrow LEFT\_STACK + RIGHT\_STACK \\
& DESC \rightarrow RIGHT\_STACK + LEFT\_STACK \\
& LEFT\_STACK \rightarrow block \uparrow block \uparrow table \\
& RIGHT\_STACK \rightarrow block \uparrow block \uparrow table \}
\end{aligned}
$$

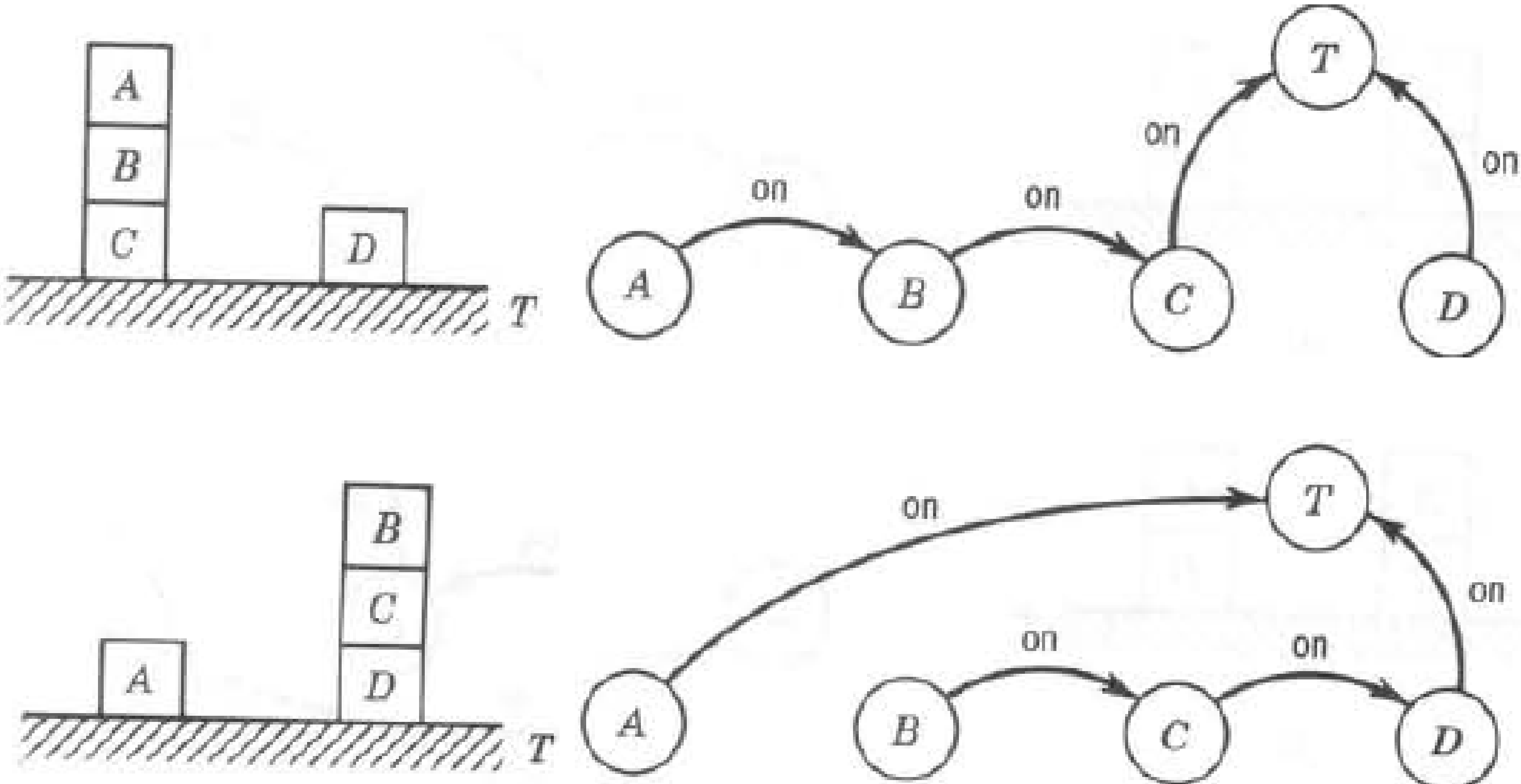




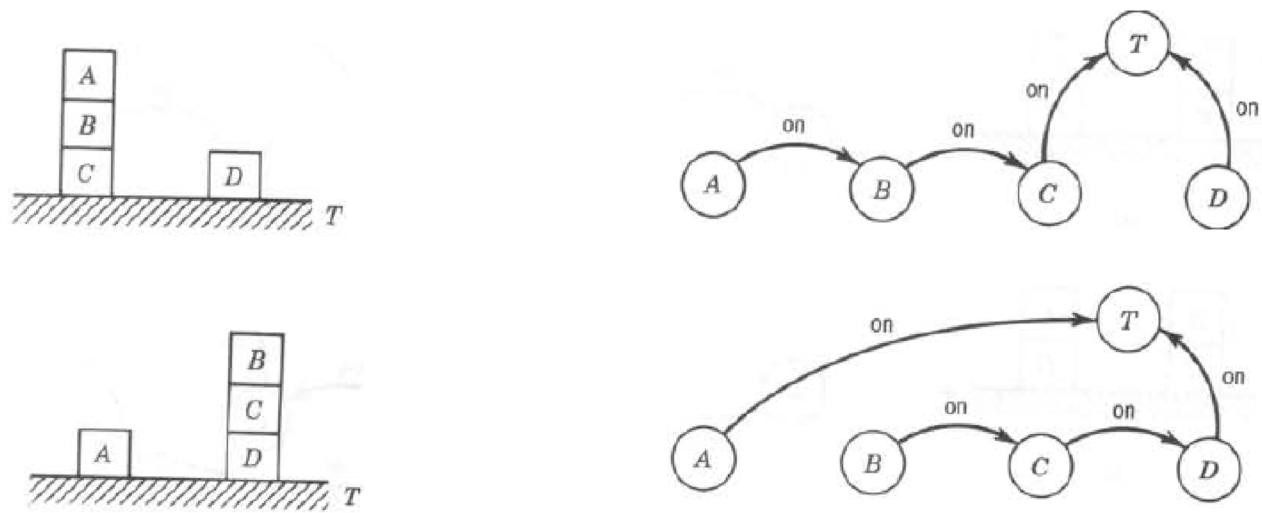




Example 3: Block World Representation



Example 3: Block World Representation



$$
\begin{aligned}
P = \{ & DESC \rightarrow LEFT\_STACK + RIGHT\_STACK \\
& LEFT\_STACK + RIGHT\_STACK \rightarrow \\
& \qquad block \uparrow table + block \uparrow block \uparrow block \uparrow table \\
& LEFT\_STACK + RIGHT\_STACK \rightarrow \\
& \qquad block \uparrow block \uparrow block \uparrow table + block \uparrow table \}
\end{aligned}
$$