MLI

# MACHINE LEARNING and INFERENCE

## An Overview of Programs and Examples of their Performance

A collection prepared by
Ryszard Michalski, Lucja Iwanska, Kaihu Chen, Heedong Ko and Peter Haddawy

| | |
|---|---|
| AQ | Learning Attributional Descriptions |
| INDUCE | Learning Structural Descriptions |
| CLUSTER | Conceptual Clustering |
| SPARC/G | Qualitative Prediction |
| SPARC/E | Predicting Card Sequences |
| ABACUS | Qualitative Discovery |
| VPL | Variable Precision Logic |

September, 1986

Artificial Intelligence Laboratory
Department of Computer Science
University of Illinois at Urbana–Champaign

# AQ

## Learning Attributional Descriptions

## BRIEF DESCRIPTION:

This is a multipurpose program that learns, in one step or incrementally, attributional descriptions of concepts from their examples. In this process the program uses background knowledge, which includes the definition of attributes and their types, rules and concepts it already knows, and a preference criterion that evaluates candidate hypotheses. There have been many versions of the program. Here we present examples from two versions of the program, AQ11 (*no-memory incremental learning*) and AQ15 (*full-memory incremental learning*).

## REFERENCES:


R. S. Michalski and J. B. Larson, "Incremental Generation of VL1 Hypotheses: the Underlying Methodology and the Description of Program AQ11," ISG 83-5, UIUCDCS-F-83-905, Department of Computer Science, University of Illinois, Urbana, 1983.

I. Mozetic, "Compression of the ECG Knowledge-base Using the AQ Induction Learning Algorithm", ISG 85-13, UIUCDCS-F-85-943, Department of Computer Science, University of Illinois, Urbana, 1985.

J. Hong, I. Mozetic and R. S. Michalski, 1985, "AQ15: Learning Attribute-Based Description from Examples, Algorithm and User's Guide", UIUCDCS-F-85-949, ISG Report 85-17, 1986.


## EXAMPLES:

- Learning Diagnostic Rules from Examples (Soybean Disease)
- Automated Knowledge Base Development & Refinement (Cardiac Arrhythmia)

# AQ11–EXAMPLE: LEARNING DIAGNOSTIC RULES FROM EXAMPLES (Soybean Disease)

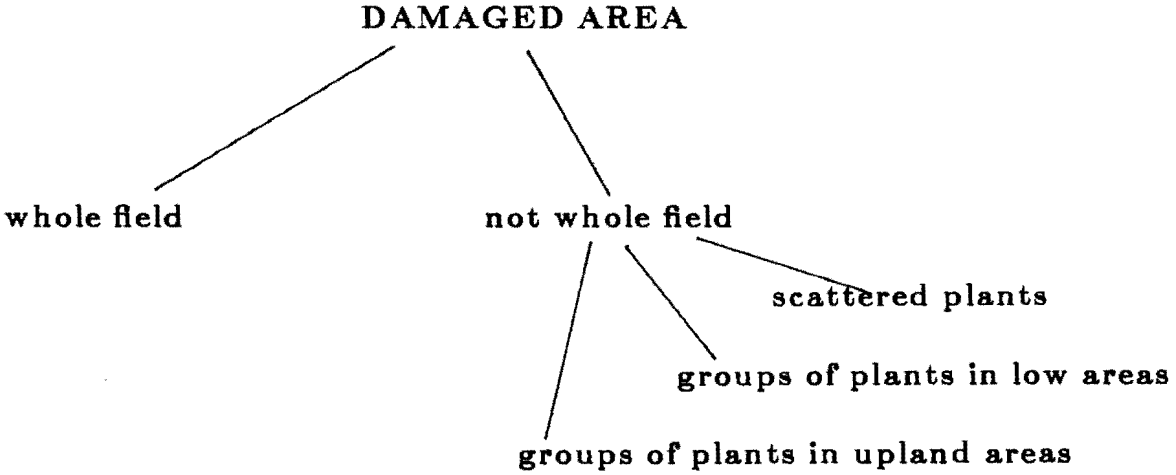
Learn diagnostic rules for the most common 15 soybean diseases from selected examples of diseases. The diseases are:

- Diaporthe Stem Canker
- Charcoal Rot
- Rhizoctonia Root Rot
- Phytophora Root Rot
- Brown Stem Rot
- Powdery Mildew
- Downy Mildew
- Brown Spot
- Bacterial Blight
- Bacterial Pustule
- Purple Seed Stain
- Anthracnose
- Phyllosticta Leaf Spot
- Alternaria Leaf Spot
- Frog Eye Leaf Spot

INPUT:

■ A collection of 230 cases (learning instances) of the diseases. Each instance was described in terms of 35 attributes characterizing the plant and its environment (Figure 1.). The individual descriptions were produced by experts, who were given a questionnaire to complete. An example of a completed questionnaire for a case of Brown Spot is shown in Figure 2.

■ Background knowledge: It contains attributes, their types and legal value sets, preference criteria for selecting among the rule candidates, and various rules characterizing relationships among the attributes. Here is an example of specification of the legal value set for the attribute *damaged area* (the value set is a hierarchy):



# Attributes Characterizing the Plant and its Environment

| | Number of values |
|---|---|
| 1. *Environmental descriptors* | |
| 1.1 Time of occurrence | (7) |
| 1.2 Plant stand | (2) |
| 1.3 Precipitation | (3) |
| 1.4 Temperature | (3) |
| 1.5 Occurrence of hail | (2) |
| 1.6 Number years crop repeated | (10) |
| 1.7 Damaged area | (4) |
| 2. *Plant global descriptors* | |
| 2.1 Severity | (3) |
| 2.2 Seed treatment | (3) |
| 2.3 Seed germination | (3) |
| 2.4 Plant height | (2) |
| 3. *Plant local descriptors* | |
| 3.1 Condition of leaves | (2) |
| 3.1.1 Leafspots—halos | (3) |
| 3.1.2 Leafspots—margin | (3) |
| 3.1.3 Leafspot size | (3) |
| 3.1.4 Leaf shredding or shot holing | (2) |
| 3.1.5 Leaf malformation | (2) |
| 3.1.6 Leaf mildew growth | (3) |
| 3.2 Condition of stem | (2) |
| 3.2.1 Presence of lodging | (2) |
| 3.2.2 Stem cankers | (4) |
| 3.2.3 Canker lesion color | (4) |
| 3.2.4 Fruiting pod on stem | (2) |
| 3.2.5 External decay | (3) |
| 3.2.6 Mycelium on stem | (2) |
| 3.2.7 Internal discoloration | (3) |
| 3.2.8 Sclerotia—internal or external | (2) |
| 3.3 Condition of fruits—pods | (4) |
| 3.3.1 Fruit spots | (5) |
| 3.4 Condition of seed | (2) |
| 3.4.1 Mold growth | (2) |
| 3.4.2 Seed discoloration | (2) |
| 3.4.3 Seed size | (2) |
| 3.4.4 Seed shrivelling | (2) |
| 3.5 Condition of roots | (3) |

FIGURE 1.

# A Completed Questionnaire Describing a Diseased Plant (Brown Spot)

*Environmental descriptors*
- Time of occurrence = July
- Plant stand = normal
- Precipitation = above normal
- Temperature = normal
- Occurrence of hail = no
- Number years crop repeated = 4
- Damaged area = whole fields

*Plant global descriptors*
- Severity = potentially severe
- Seed treatment = none
- Seed germination = less than 80%
- Plant height = normal

*Plant local descriptors*
- Condition of leaves = abnormal
  - Leafspots—halos = withour yellow halos
  - Leafspots—margin = without watersoaked margin
  - Leafspot size = greater than 1/8 inch
  - Leaf shredding or shot holding = present
  - Leaf malformation = absent
  - Leaf mildew growth = absent
- Condition of stem = abnormal
  - Presence of lodging = no
  - Stem cankers = above the second node
  - Canker lesion color = brown
  - Fruiting bodies on stem = present
  - External decay = absent
  - Mycelium on stem = absent
  - Internal discoloration of stem = none
  - Sclerotia—internal or external = absent
- Condition of fruits—pods = normal
  - Fruit spots = absent
- Condition of seed = normal
  - Mold growth = absent
  - Seed discoloration = absent
  - Seed size = normal
  - Seed shriveling = absent
- Condition of roots = normal

*Diagnosis*
*Diaporthe stem canker*( ) *Charcoal rot*( ) *Rhizoctonia root rot*( )
*Phytophthora root rot*( ) *Brown stem root rot*( ) *Powdery mildew*( )
*Downy mildew*( ) *Brown spot*(X) *Bacterial blight*( )
*Bacterial pustule*( ) *Purple seed stain*( ) *Anthracnose*( )
*Phyllosticta leaf spot*( ) *Alternaria leaf spot*( ) *Frog eye leaf spot*( )

FIGURE 2.

Here is a sample of rules characterizing constraints which hold among various attribute values (in the rules * denotes *does not apply* and $=>$ denotes the logical implication):

1. [leaves = normal] ⇒ [leafspots halos = *][leafspots margin = *]
   [leafspot size = *][leaf shredding = *]
   [leaf malformation = *][leaf mildew growth = *]
2. [leafspots halos = absent] ⇒ [leafspots margin = *][leafspot size = *]
3. [stem = normal] ⇒ [presence of lodging = *][stem cankers = *]
   [canker lesion color = *][fruiting bodies on stem = *]
   [external decay of stem = *][mycelium on stem = *]
   [internal discoloration = *]
   [sclerotia internal or external = *]
4. [fruit pods = normal] ⇒ [fruit spots = *]
5. [seed = normal] ⇒ [seed mold growth = *][seed discoloration = *]
   [seed size = *][seed shriveling = *]

These rules specify conditions under which attributes are not applicable. Rule 1, for example, states that if leaves are normal, then it is not meaningful to ask for properties of diseased leaves.

OUTPUT:

A diagnostic rule for each disease. Here are two examples of rules created by the program. For comparison, each rule is accompanied by a corresponding expert supplied rule for the same disease.

## ■ Rules for Diaporthe Stem Canker:

An Inductively-Derived Rule:

[time=Jul...Oct][precipitation>=normal][leaf-malformation=absent]
[stem=abnormal][external-decay=firm & dry]
[fruit-pods=n][stem-cankers=above-second-node] **(10, 10)**
=> [class(disease)=Diaporthe-Stem-Canker]

where a pair of numbers (x, y) characterizes an individual conjunctive condition of a rule: x indicates the number of examples uniquely covered by the given conjunction, y indicates the total number of examples covered by the conjunction.

The Corresponding Expert-Derived Rule:

Qs( [time=Aug...Sept][precipitation:↑EP][fruit-pods=normal]
[stem-cankers=above-second-node][fruiting-bodies=present])
+
Qc( [temperature≥normal][canker-lesion-color=brown]
[number-years-crop-repeated:↑ER1])
=> [class(disease)=Diaporthe-Stem-Canker])

where

Qs is a weight indicating significant conditions,
Qc is a weight indicating corroborative conditions,
EP is a weight assigning function which is defined as follows:
EP = 1.0 if precipitation is above normal
= 0.7 if precipitation is normal
= 0.2 otherwise.
↑ indicates that weight assigning function
is monotonically increasing,
ER1 is a weight assigning function which is defined as follows:
ER1 = 1.0 if the number of years crop repeated $\geq 3$
= 0.8 if the number of years crop repeated = 2
= 0.7 if the number of years crop repeated = 1
= 0.2 if crop not repeated.

## ■ Rules for Phytophthora Root Rot:

### Inductively–Derived Rule:

[plant–stand > normal][precipitation ≥ normal][temperature ≤ normal]
[plant–height = abnormal][leaves = abnormal]
[leaf–malformation = absent][stem = abnormal] **(6, 24)**

v

[time = Apr..Aug][plant–stand = abnormal][damaged–area = low]
[plant–height = abnormal][leaves = abnormal][stem = abnormal]
[external–decay = absent,soft and watery] **(16, 34)**

=> [class(disease) = Phytophora–root–rot]

### The Corresponding Expert–Derived Rule:

Qs( [time = ∩ET][plant–stand < normal]
([time = Apr...Jun] => [precipitation = normal])
([time = Jul...Aug] => [precipitation = above–normal])
([time = Apr] => [temperature = above–normal])
([time = May...Aug] => [temperature = above n])[damaged–areas = low]
[plant–growth = abnormal][leaves = abnormal][stem = abnormal]
[stem–cankers = at or slightly above soil line]
([time = May...Aug] => [canker lesion color = dark brown or black])
[roots = rotted])

+

Qc( [number–years crop repeated ≥ 2])
=> [diagnosis(disease) = Phytophthora–root–rot]

where

ET is a weight assigning function which is defined as follows:

ET = 1.0 if time of occurrence = May..July
= 0.7 if time of occurrence = April v August
= 0.2 otherwise.

∩*fn* indicates that the function *fn* has maximum around some mean and decreases with the distance from this mean.

The inductively–derived rule has two conjunctive statements linked by disjunction. In contrast, the expert–derived rule can be viewed as one long conjunction in which some conditions are considered significant and some as confirmatory.

# COMPARISON OF EXPERT–DERIVED AND INDUCTIVELY–DERIVED RULES

Both inductively–derived and expert–derived rules were tested on 340 testing examples. The following table summarizes the performance of the rules.

| Type | % correct diagnosis | % preferred diagnosis | % not diagnosed | Indecision ratio | Threshold |
|---|---|---|---|---|---|
| Inductively–derived | 100.0 | 97.6 | – | 2.64 | 0.80 |
| Expert–derived | 96.2 | 71.8 | 2.1 | 2.90 | 0.65 |

where

*% correct diagnosis* denotes the percentage of the cases in which one of the program's alternative diagnoses was judged by an expert as correct diagnosis;

*% preferred diagnosis* denotes the percentage of cases in which the program's diagnosis with the highest score (among alternative diagnoses) was judged by an expert as correct.

*Indecision ratio*: the ratio of the number of alternative diagnosis for the events of the given disease over the number of testing events in the set.

*Threshold*: if the degree of match between the observed symptoms and a rule is equal or exceeds the threshold, than the diagnostic hypothesis indicated by the rule is added to an alternative diagnosis list, otherwise the hypothesis is rejected.

# AQ–EXAMPLE: AUTOMATED KNOWLEDGE BASE DEVELOPMENT & REFINEMENT (Cardiac Arrhythmia)

This example illustrates an application of inductive learning to a problem in which learning instances cannot be obtained directly. In this example, the problem is to refine an arrhythmia knowledge base. Learning instances cannot be obtained directly because it is not ethical to introduce a malfunction to a human heart in order to determine the corresponding ECG (electrocardiogram).

Our approach to such a problem is illustrated in Figure 1. Using experts and medical literature as sources of information, a simulation model of the process is built (Figure 2). By introducing to the model various malfunctions and observing how they affect the measurable properties of the system, an initial knowledge base of learning instances is created. The model contains 943 rules, from which 5240 instances of heart disorders were generated.

Each instance given to the program describes a heart disorder in terms of 10 attributes, such as P–wave, rhythm, PR interval, P–QRS relation, etc. By applying AQ15 (version NEWGEM) to these instances, a refined arrhythmia knowledge base was created.

# AUTOMATED KNOWLEDGE BASE DEVELOPMENT AND REFINEMENT

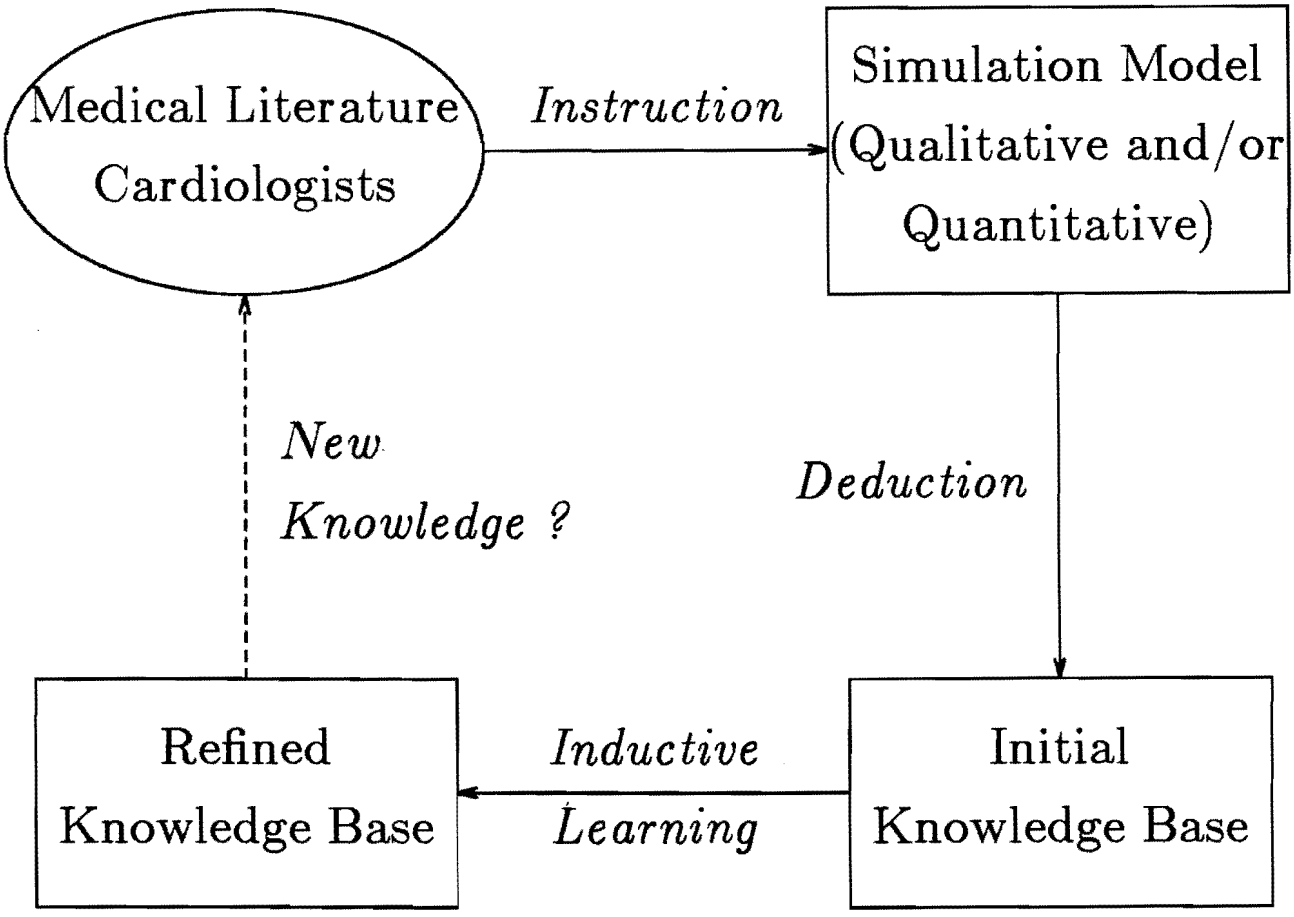
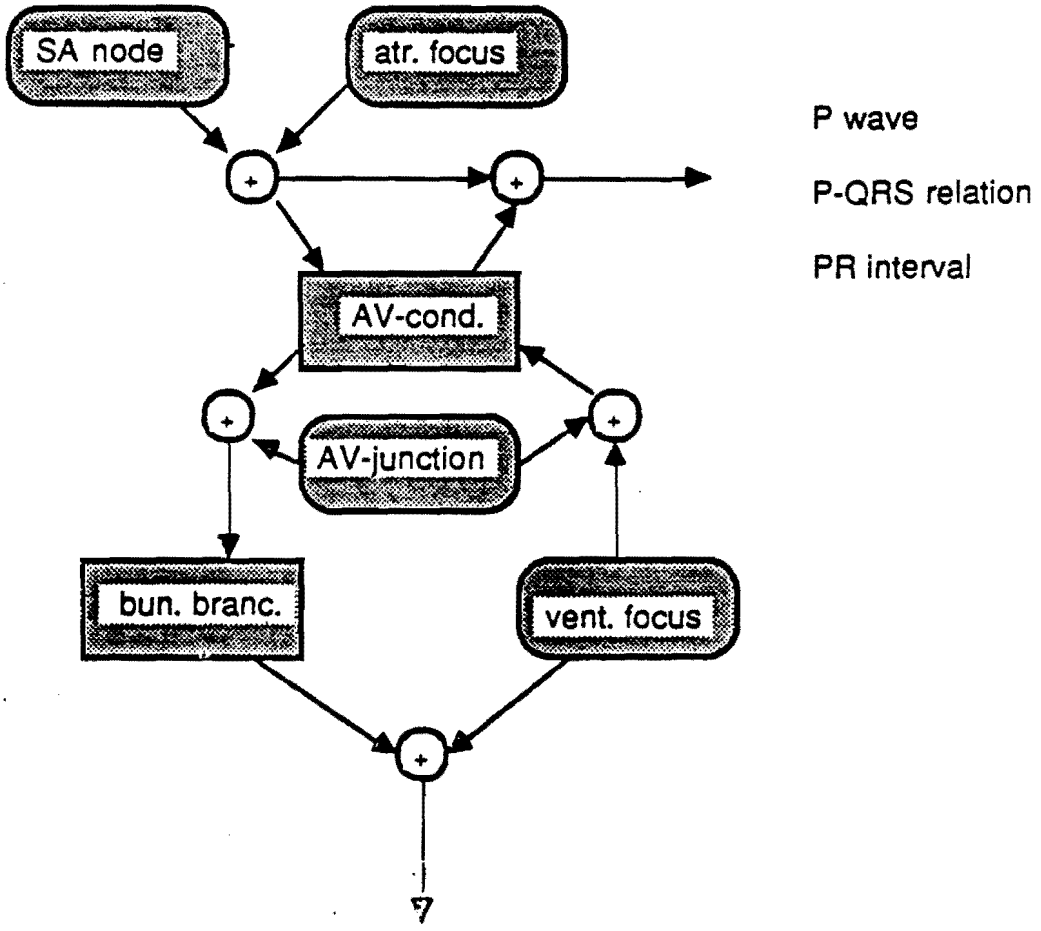



FIGURE 1

# Model of the Heart




ECG:

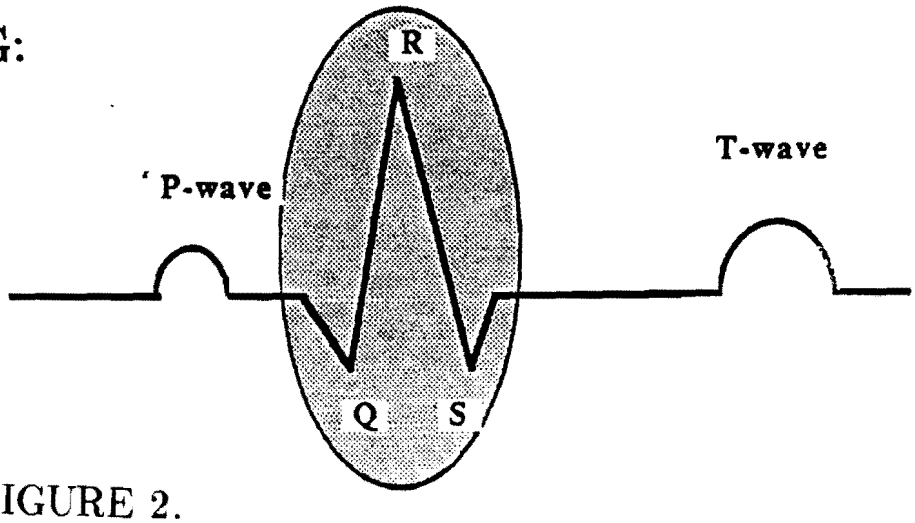



FIGURE 2.

INPUT:

■ The knowledge base obtained from simulation. It contains 5240 cases of 25 types of heart disorders: Sinus Tachycardia, Wenckebach, Atria Ectopic Beats, AV block 3, and others. Each case relates multiple heart disorder to a symbolic ECG description (ECG is described with ten many–valued attributes).

Below is an example of a description for Sinus Tachycardia:

[Rhythm = irregular][P_wave = normal][Rate_of_P = 100_250]
[P_QRS = after_P_some_QRS_miss][PR_interval = prolonged]
[QRS_complex = normal][Rate_of_QRS = 60_100 ∨ 100_250]
[Ectopic_P = abnormal][Ectopic_PR = prolonged][Ectopic_QRS = normal]
=> [class(Disorder) = Sinus Tachycardia]

■ Background knowledge: It contains a specification of attributes that characterize an ECG, their types and legal value sets, preference criteria for selecting among the rule candidates, and various rules characterizing relationships among the attributes.

OUTPUT:

A refined knowledge base that contains two types of rules:

(1) *Characteristic descriptions of simple heart disorders*, similar to definitions of diseases in the medical literature.
(2) *Diagnostic rules*, for efficient ECG diagnosis.

Here is an example of a characteristic description generated for Sinus Tachycardia:

$$[\text{P-wave}{=}\text{normal}][\text{Rate-of-P}{=}100..250] \\ <= [\text{class(disorder)} = \text{Sinus-Tachycardia}]$$

Corresponding descriptions from medical literature:

Goldman (1976) : A *regular* sinus *rhythm* with a *rate in excess of 100.*

Phibbs (1973) : In sinus tachycardia the *rate is over 100.* It will rarely exceed 160.

Here is an example of a diagnostic rule formulated by the program:

$$[\text{P\_QRS} = \text{after\_P\_some\_QRS\_miss}] \Rightarrow \\ [\text{AV\_cond} = \text{Wenckebach} \vee \text{Mobitz2}] \vee \\ [\text{Atr\_focus} = \text{Atrial\_flutter} \vee \text{Atrial\_fibrillation}][\text{AV\_cond} = \text{none}]$$

If after some P-waves QRS complexes are missing, then there is either a conduction disturbance of the type Wenckebach or Mobitz2, or there is Atrial Flutter or Atrial Fibrillation, without any conduction disturbance. To illustrate specific diagnoses, the right-hand sides of the refined rules are logically intersected, yielding one or more arrhythmias (a patient may have several arrhythmias).

The table below illustrates the Knowledge Base refinement achieved by the inductive learning process:

| | Initial Knowledge Base | Refined Knowledge Base |
|---|---|---|
| Rules/Events | 943/5240 | 88 |
| Storage size | 900 kB | 28 kB |

The refined knowledge base has about 10 times fewer rules and occupies about 30 times less memory than the initial knowledge base.

# INDUCE

## Learning Structural Descriptions from Examples

## BRIEF DESCRIPTION:

This is a multipurpose inductive learning system that learns structural descriptions of concepts from their examples. Examples are presented to the system in the form of a logical statement involving attributes of the objects and their parts, and the relations among the parts. The description language is based on an extended predicate calculus, called variable–valued logic system VL2. One version, INDUCE/2, learns descriptions in one step, and the other version, INDUCE/4, learns descriptions either in one step or incrementally. Both programs have a capability for a rather extensive use of domain knowledge, represented in terms of the so–called *l–rules* and *a–rules*, and for automatically constructing new attributes based on that knowledge (constructive induction).

## REFERENCES:

W. A. Hoff, R. S. Michalski and R. E. Stepp, "INDUCE/2: A Program for Learning Structural Descriptions from Examples," UIUCDCS–F–83–904, 1983.

G. Mehler, J. Bentrup and J. Riedesel, INDUCE/4: "A Program for Incrementally Learning Structural Descriptions from Examples", Department of Computer Science, University of Illinois, Urbana, UIUCDCS–F–86–958, 1986.

## EXAMPLES:

- Geometrical Figures (2 classes)
- Geometrical Figures (3 classes)
- Cells
- Trains
- Chemical Compounds

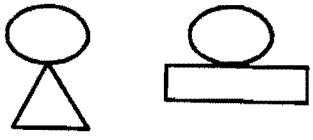
# Representation of Objects and Object Classes

Structured objects: represented by extended first order logic expressions using Variable–valued logic system VL2:

(structural description)

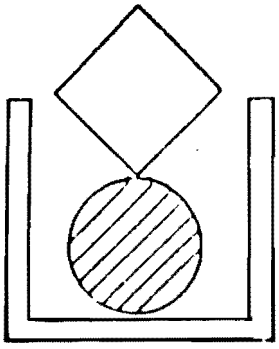
| OBJECT | OBJECT DESCRIPTION |
|---|---|
|  | [contains(object,part1,part2)][ontop(part1,part2)] & [shape(part1)=square][shape(part2)=circle] & [color(part1)=red][color(part2)=red] & [size(part1)=small][size(part2)=small] |

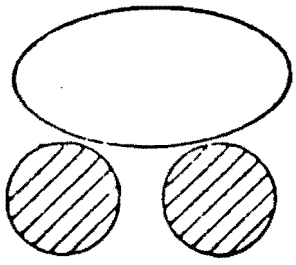
| CLASS | CLASS DESCRIPTION |
|---|---|
|  | ∀ object [contains(object,part1,part2)] & [ontop(part1,part2)][shape(part1)=oval] & [color(part2)=red] |

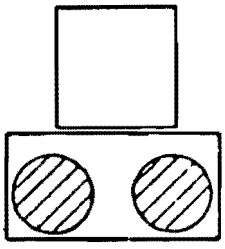


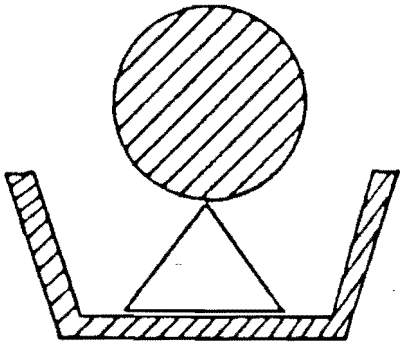
# INDUCE–EXAMPLE: GEOMETRICAL FIGURES (2 classes)

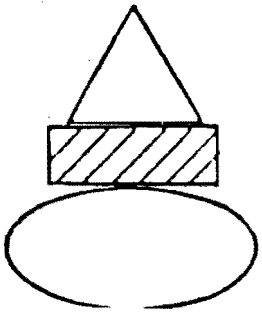
Find rules distinguishing between two groups of figures, G1 and G2.

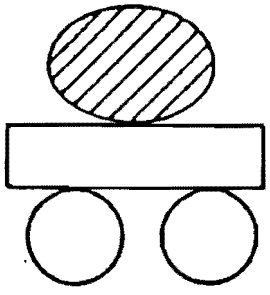












GROUP G1 GROUP G2

## INPUT:

- A description of each figure along with its group. Below is an example of such a description for the top figure in group G1 (in the expression the figure is denoted OBJ).

  [contains(OBJ,part1,part2,part3)][ontop(part1,part2)]
  [ontop(part2,part3)][size(part1)=medium][size(part2)=medium]
  [size(part3)=large][texture(part1)=clear][texture(part2)=shaded]
  [texture(part3)=clear][shape(part1)=diamond][shape(part2)=circle]
  [shape(part3)=u-shape] => [class(OBJ) = G1]

- Background knowledge containing the definition of initial descriptors and their types; constructive induction rules for generating new attributes are given by a number of l-rules and a-rules such as number of objects whose size is small, medium and large. An example of l-rule concerning ontop relation is:

  $\forall$ i, j, k [ontop(part(i),part(j)][ontop(part(j),part(k))]
  => [ontop(part(i),part(k)]

  If part(i) is on top of part(j), and part(j) is on top of part(k), then part(i) is on top of part(k).

## OUTPUT:

A sample of rules discovered by the program:

- [contains(OBJ,part)][top(part)][shape(part) = polygon]
  => [class(OBJ) = G1]

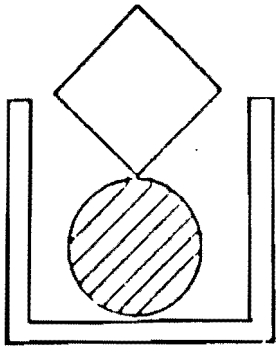
If the top part of the object is a polygon, then it belongs to class G1.

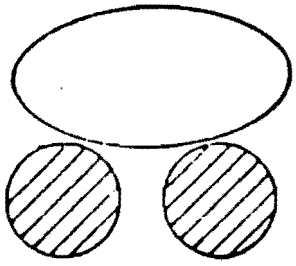
- [contains(OBJ,part)][top(part)][shape(part) = oval]
  => [class(OBJ) = G2]

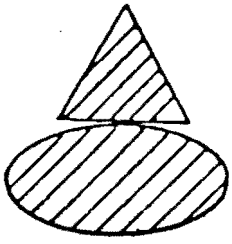
If the top part of the object is an oval in shape, then it belongs to class G2.

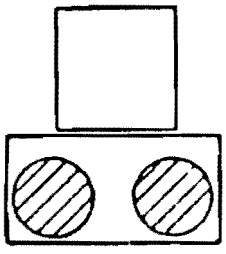
# INDUCE–EXAMPLE: GEOMETRICAL FIGURES (3 classes)

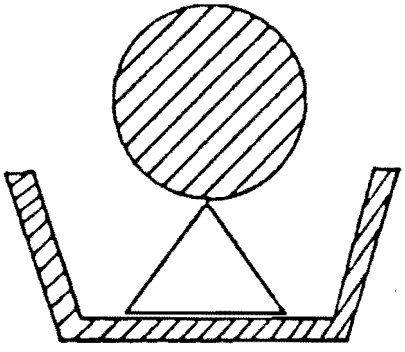
This is an extension of previous example. Here the program has to learn an additional concept (G3) in the context of the concepts learned before (G1, G2).

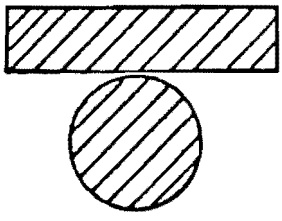


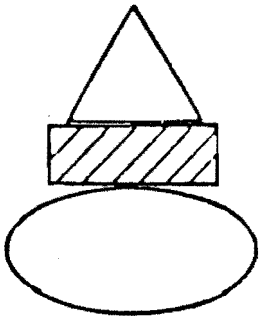


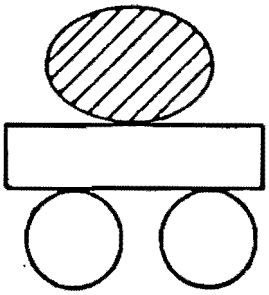


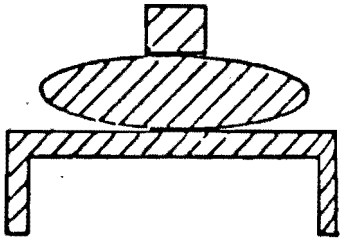












GROUP G1 GROUP G2 GROUP G3

INPUT:

- A description of each figure along with its class. For illustration, a description of the top figure in group G1 is shown below (in the expression the figure is denoted OBJ).

  [contains(OBJ,part1,part2,part3)][ontop(part1,part2)]
  [ontop(part2,part3)][size(part1)=medium][size(part2)=medium]
  [size(part3)=large][texture(part1)=clear][texture(part2)=shaded]
  [texture(part3)=clear][shape(part1)=diamond][shape(part2)=circle
  [shape(part3)=u-shape] => [class(OBJ) = G1]

- Background knowledge containing the definition of initial descriptors and their types; constructive induction rules for generating new attributes are given by a number of l-rules and a-rules such as number of objects whose size is small, medium and large. An example of l-rule concerning ontop relation is:

  $\forall$ i, j, k [ontop(part(i),part(j)][ontop(part(j),part(k))]
  => [ontop(part(i),part(k)]

  If part(i) is on top of part(j), and part(j) is on top of part(k), then part(i) is on top of part(k).

# OUTPUT:

A sample of rules discovered by the program:

- [contains(OBJ,part)][top(part)][shape(part) = polygon]
  [texture(part) = clear] => [(class(OBJ) = G1]

If the top part of the object is a polygon with clear texture, then it belongs to class G1. (Notice that in contrast with the two–class case, the rule has an additional condition that the texture is clear.)

- [contains(OBJ,part)][top(part)][shape(part)=oval]
  => [class(OBJ)=G2]

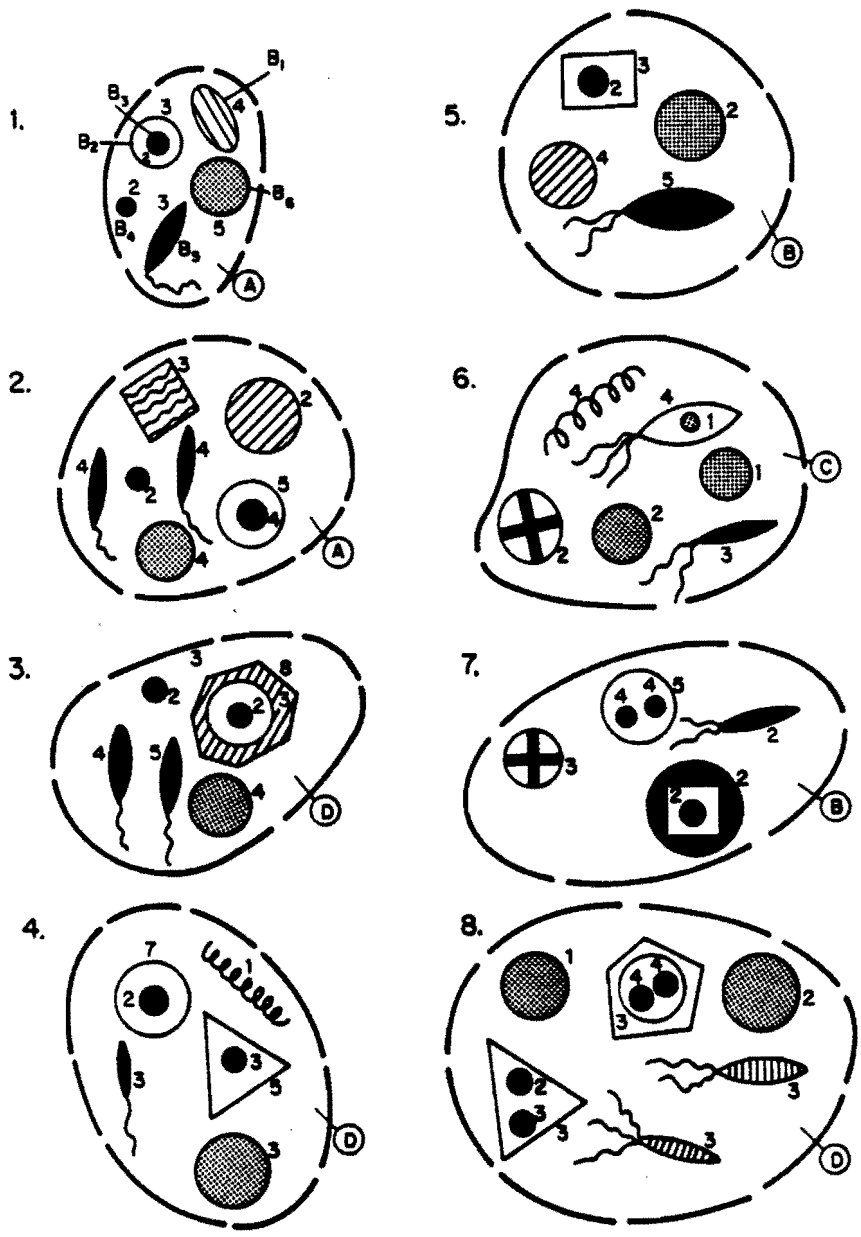
If the top part of the object is an oval shape, then it belongs to class G2.

- $\forall$ part [contains(OBJ,part)][texture(part) = shaded]
  => [class(OBJ) = G3]

If all parts of the object are shaded, then the object belongs to class G3.

# INDUCE–EXAMPLE: CELLS

Find a simple rule distinguishing between cancerous and noncancerous cells.




CANCEROUS CELLS NONCANCEROUS CELLS

## INPUT:

■ A description of each cell along with its class. Below is an example of a description for Cell 1.

[contains(Cell1, B1, B2,..., B6][cir(Cell1)=8][pplasm(Cell1)=A]
[shape(B1)=ellipse][texture(B1)=stripes][weight(B1)=4]
[orient(B1)=NW]...[shape(B5)=boat][texture(B5)=solid][weight(B5)=4]
[orient(B5)=NE][no-tails(B5)=1][shape(B6)=circle][texture(B6)=shaded]
[weight(B6)=5] => [class(Cell1)=CANCEROUS]

where B1,...,B6 are bodies in Cell 1.

■ Background knowledge containing the definition of descriptors and their types, rules for defining the structure of structured descriptors, rules to construct new attributes, a preference criterion that evaluates candidate hypotheses. Here is an example of a rule defining an element of the generalization structure of a structured attribute.

[shape=circle v ellipse] => [shape=oval]

## OUTPUT:

A sample of rules describing cancerous cells discovered by the program:

• ∃ B, [contains(Cell,B)][texture(B)=shaded][weight(B) ≥3]
=> [class(Cell) = CANCEROUS]
If the texture of a body in a cell is shaded, and the weight of the body is greater than 3, then it is a cancerous cell.

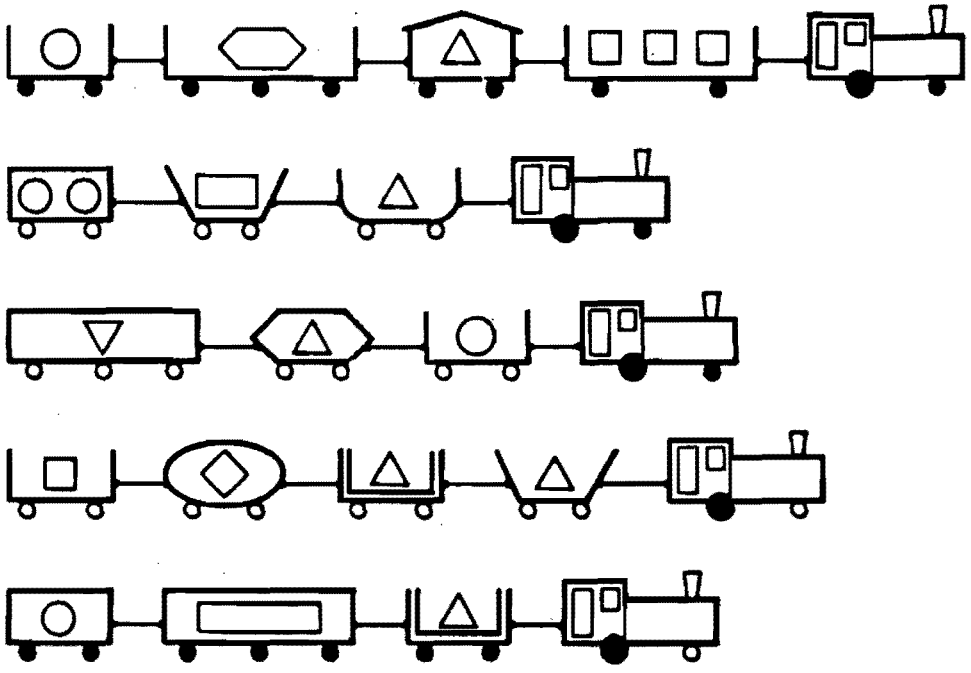
• [circ = even] => [class(Cell) = CANCEROUS]
If the number of segments in the circumference of the cell is even, then it is a cancerous cell.

• ∃ B [shape(B)=boat][no-tails(B)=1] => [class(Cell) = CANCEROUS]
If the cell has at least one 'boat' shape body with a single tail, then it is a cancerous cell.

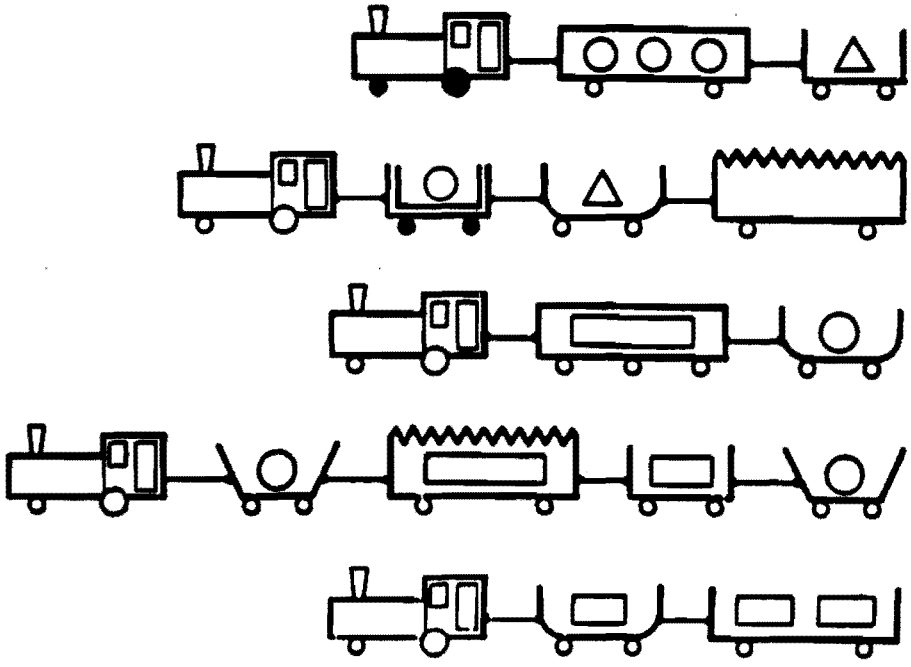
# INDUCE–EXAMPLE: TRAINS

Discover simple rules discriminating between Eastbound and Westbound trains.

EASTBOUND TRAINS:



WESTBOUND TRAINS:



## INPUT:

■ A description of each train along with its class. For illustration, here is a description of the first Eastbound train:

[contains(Train1,Car1,Car2,Car3,Car4,Car5)][infront(Car1,Car2)]
[car-type(Car1)=engine][car-length(Car1)=long]
[number-of-wheels(Car1)=2][wheel-color(Car1)=black]
• • •
[infront(Car4,Car5)][car-type(Car5)=open-box][car-length(Car5)=short]
[number-of-wheels(Car5)=2][wheel-color(Car5)=black]
[number-of-items-carried(Car5)=1][cargo-shape(Car5)=ball]
=> [class(Train1) = Eastbound]

■ Background knowledge containing the definition of descriptors and their types, rules for defining the structure of structured descriptors, rules to construct new attributes, a preference criterion that evaluates candidate hypotheses. Here is an example of a rule defining an element of the generalization structure of a structured attribute.

[car-shape=open-box v u-shaped] => [car-shape=open-top]

If shape of the car is open box or u-shaped, then the car has an open top.

## OUTPUT:

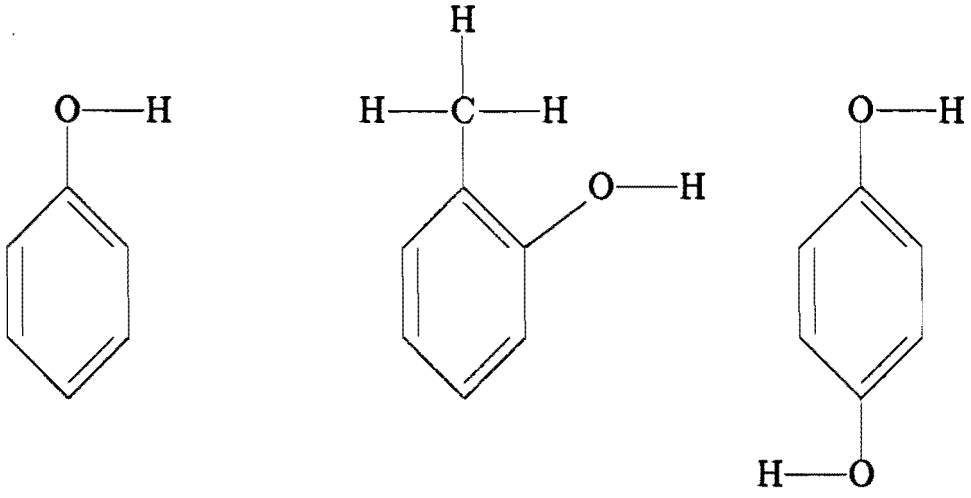
A sample of rules discovered by the program:

• [contains(Train,Car)][length(Car) = short][car-shape(Car) = closed-top]
=> [class(Train)= Eastbound]
If the train contains a short car with a closed top, then it is an Eastbound train.

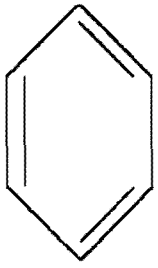
• [number-of-cars = 3] V [contains(Train,Car)][car-shape(Car) = jagged-to
=> [class(Train)= Westbound]
If the train has three cars or has a car with a jagged top, then it is a Westbound train.

# INDUCE–EXAMPLE: ORGANIC COMPOUNDS

Given are different organic compounds of the phenol group. The problem is to find a characteristic description of this chemical group. A characteristic description of a class states properties common to all objects in the class.




PHENOL O–METHYLPHENOL PENADIPHENOL

where  — Aromatic Group

INPUT:

■ A description of each compound along with its class. For illustration, a description of the compound phenol is shown below.

[contains(Phenol,Atom1,Atom2,...,Atom8)][type(Atom1)=carbon]
[type(Atom2)=carbon][type(Atom3)=carbon][type(Atom4)=carbon]
[type(Atom5)=carbon][type(Atom6)=carbon][type(Atom7)=oxygen]
[type(Atom8)=hydrogen][single-bond(Atom1,Atom2)]
[double-bond(Atom2,Atom3)][single-bond(Atom3,Atom4)]
[double-bond(Atom4,Atom5)][single-bond(Atom5,Atom6)]
[double-bond(Atom6,Atom1)][single-bond(Atom6,Atom7)]
[single-bond(Atom7,Atom8)] => [class(Phenol) = PHENOL-GROUP]

■ Background knowledge rules containing the definition of descriptors and their types, a preference criterion that evaluates candidate hypotheses, and rules characterizing the known chemical properties and different chemical groups. An example of a latter rule for aromatic group is:

[contains(Group,Atom1,Atom2,Atom3,Atom4,Atom5,Atom6]
[type(Atom1)=carbon][type(Atom2)=carbon][type(Atom3)=carbon]
[type(Atom4)=carbon][type(Atom5)=carbon][type(Atom6)=carbon]
[double-bond(Atom1,Atom2)][single-bond(Atom2,Atom3)]
[double-bond(Atom3,Atom4)][single-bond(Atom4,Atom5)]
[double-bond(Atom5,Atom6)][single-bond(Atom6,Atom1)]
=> [Type(Group)=aromatic]

OUTPUT:

A description produced by the program:

[class(Compound) = PHENOL-GROUP] <=
[contains(Compound,Group1,Group2)][Attached(Group1,Group2)]
[Type(Group1) = aromatic][Type(Group2) = methyl v OH]
The phenol group has an aromatic group attached to a methyl or OH group.

# CLUSTER

## Conceptual Clustering: Inventing Meaningful Classification of Objects

## BRIEF DESCRIPTION:

Given a set of objects, the program invents a hierarchy of classes into which objects can be classified. Classes are defined by a conjunctive description. For the version CLUSTER/2, objects are described by stating values of attributes (an attributional description). For the version CLUSTER/S, objects are described by stating values of attributes of objects and their parts, and relations binding the parts (a structural description).

## REFERENCES:

R. Michalski and R. E. Stepp III, "Learning From Observations: Conceptual Clustering", in "Machine Learning– An Artificial Intelligence Approach", vol.I, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M., Tioga Publishers, 1983

R. E. Stepp III, "Conjunctive Conceptual Clustering: A Methodology and Experimentation", Ph.D. Thesis, UIUCDCS–R–84–1189, 1984.

R. Michalski and R. E. Stepp III, "Conceptual Clustering: Inventing Goal–Oriented Classifications of Structured Objects", in "Machine Learning– An Artificial Intelligence Approach", vol.II, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M., Morgan Kaufmann Publishers, 1986

## EXAMPLES:

- Spanish Songs
- Grouping Trains
- Microcomputers

# CLUSTER/2–EXAMPLE: SPANISH SONGS

The problem is to build a classification of 100 Spanish folk songs.

INPUT:

- A description of each song in terms of the following 22 attributes (the number in parentheses indicates the number of legal values of the attribute):

| | | | |
|---|---|---|---|
| 1. | Tonal range | (8) | The diatonic distance between the lowest and the highest notes of the song |
| 2. | Number of tones | (8) | A count of all different notes used throughout the song |
| 3. | Phrygian | (2) | Song uses phrygian scale |
| 4. | Dance | (2) | People sometimes dance to this song |
| 5. | Serenade | (2) | Song is a serenade |
| 6. | Love | (2) | Song's subject pertains to love |
| 7. | Religion | (2) | Song's subject pertains to religion |
| 8. | Panegyric | (2) | Song praises a particular town or region |
| 9. | Solo | (2) | Performed by a solo singer |
| 10. | Instruments | (2) | Any sort of instrumental accompaniment |
| 11. | Female | (2) | Women participate in the singing of this song |
| 12. | Mixed | (2) | Male–female (mixed) singing takes place in the song |
| 13. | Number of phrases | (4) | Number of different musical phrases used throughout the song |
| 14. | Meter | (3) | No regular rhythm |
| 15. | Harmonic–structure | (2) | Harmonic structure is monophonic or polyphonic |
| 16. | Home | (2) | Song is homophonic |
| 17. | Harmony | (3) | No harmonic accompaniment, non–phrygian harmonic accompaniment, phrygian harmonic accompaniment |
| 18. | Rubato | (6) | Degree of rhythmic freedom |
| 19. | Melisma | (6) | Degree to which each syllable has several different notes to be sung |
| 20. | Embel–ishment | (6) | Degree to which single notes are enhanced by additional ornamental notes |
| 21. | Tension | (6) | Neck, throat muscle tightness as evidenced by the tessiture |
| 22. | Blend | (7) | Degree of togetherness of the different performers |

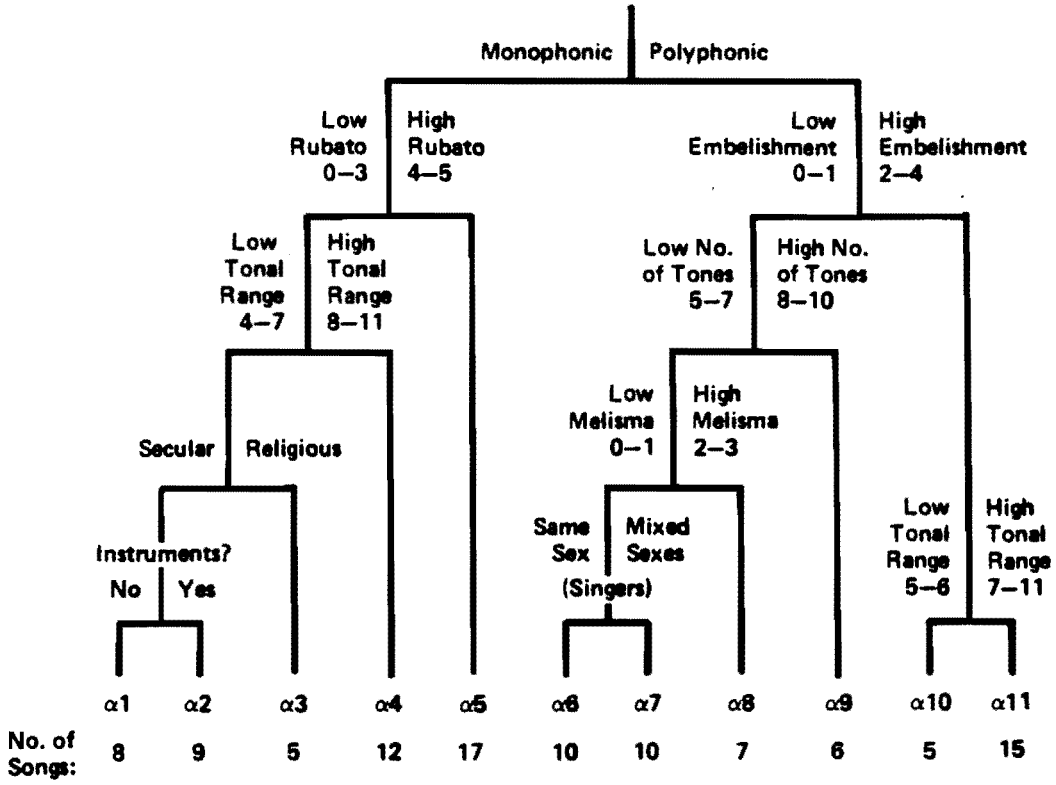
- Background knowledge containing a specification of attributes and their types. The linear attributes are: tonal range, number of tones, phrases, degree of rubato, melisma, embelishment, and tension. Others are nominal types. The quality of clustering is judged by a continuous measure that includes simplicity of the class descriptions, inter–cluster difference, discrimination index and dimensionality reduction.

## OUTPUT:

- A classification constructed by the program (Figure 1).

- Descriptions of individual classes (see example on Figure 1). All the attributes as well as relevant data were provided by musicologist Pablo Poveda, who studied the problem of classifying Spanish folk songs using numerical taxonomy. However, the traditional methods were not very satisfying, because the generated clusters lacked descriptions, and therefore were difficult to interpret.

  One interesting aspect of the generated hierarchy is that the value sets of some variables have been split into ranges. For example, while producing the second level clustering of the monophonic folk songs, the range of the degree of "rubato" was split into two ranges 0..3 and 4..5 which can be characterized as "low" and high", respectively.




A description for the class $\alpha 1$ (corresponding to the path from the root to the leaf marked $\alpha 1$).

```
[class(song) = α1] <=
    [harmonic-structure=monophonic][rubato=low]
    [tonal-range=low][religion=secular][instruments-used=no]
    [no-of-distinct-tones=5..8][dance=no][panegiric=no]
    [no-of-phrases=1..2][melisma=0..2][tension=1..3]
```

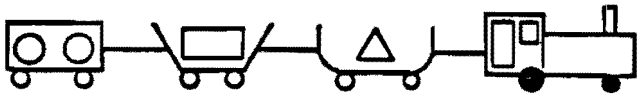
A Description Constructed by the Program

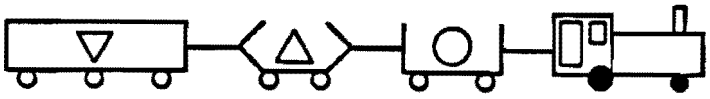
Figure 1

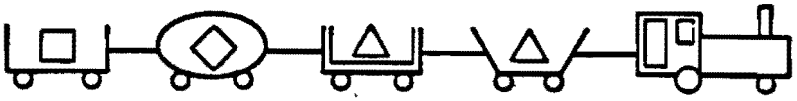
# CLUSTER/S–EXAMPLE: TRAINS

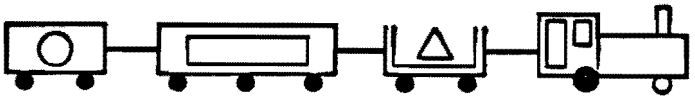
This example uses the same trains as TRAINS example for INDUCE program. The difference is that trains have no prior classification. The problem is to create a meaningful classification of the trains.

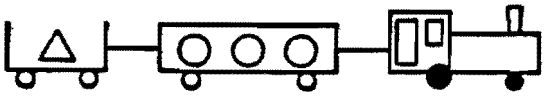
A. 

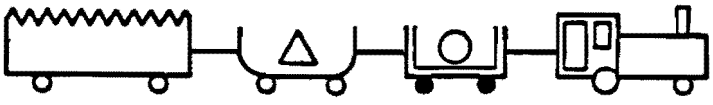
B. 

C. 

D. 

E. 

F. 

G. 

H. 

I. 

J. 

*How to classify these trains?*

# INPUT:

- Below is a description of the first train (A):

[contains(Train1,Car1,Car2,Car3,Car4,Car5)][infront(Car1,Car2)]
[infront(Car2,Car3)][infront(Car3,Car4)][infront(Car4,Car5)]
[car-shape(Car1)=engine][car-length(Car1)=long]
[number-of-wheels(Car1)=2][wheel-type(Car1)=black]
[number-of-items-carried(Car1)=0]

• • •

[car-shape(Car5)=open-rectangle][car-length(Car5)=short]
[number-of-wheels(Car5)=2][wheel-type(Car5)=black]
[number-of-items-carried(Car5)=1][cargo-shape(Car5)=circle-load]

- Background knowledge containing a specification of the descriptors, their types, and the structure of their domains; constructive induction rules for generating new attributes from the initially given information. Among the latter are "counting rules" that define new attributes such as the number of cars whose cargo shape is a circle.

## OUTPUT:

The program generates several hierarchies reflecting different goals. The goals of classification are linked to the relevant features of objects via a Goal Dependency Network (GDN), which is a part of program's background knowledge. By using a GDN, the relevant object features are determined, and then the program constructs a classification hierarchy using the method of conceptual clustering. In determining a classification, the system is guided by a "clustering quality criterion". The default criterion is a continuation of the "minimum sparseness" (minimum generality) and maximum simplicity of descriptions. The criterion is flexible and can be changed by a user.

- Result 1 (see Figure 1)

This hierarchy was generated assuming that the general goal is to determine a classification based on simple geometrical properties of trains. A Goal Dependency Network is shown in Figure 2. The clustering quality criteria were 95% tolerance for simplicity and minimum sparseness.

- Result 2 (see Figure 3)

This hierarchy was generated assuming that the general goal is to determine a classification reflecting the survival or safety aspects of trains. The Goal Dependency Network is shown in Figure 4. An example of l-rule for has_toxic_chemicals is:

```
[contains(Train,Car)][car-shape(Car)=open-top]
[cargo-shape(Car)=circle-load][number-of-items-carried(Car)=1]
      <=>  [has_toxic_chemicals(Train)]
```

A toxic chemical container will be identified as a single sphere (circle) riding in an open-top car. The clustering quality criterion was that the derived attribute "has_toxic_chemicals(Train)" was given a large preference score to reflect its importance in the resulting description.

A. 

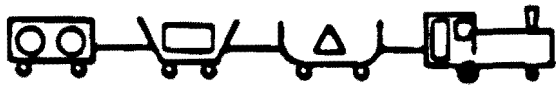
E. 

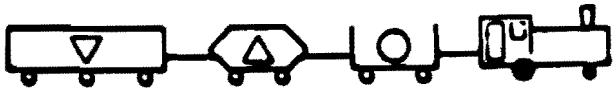
F. 

H. 

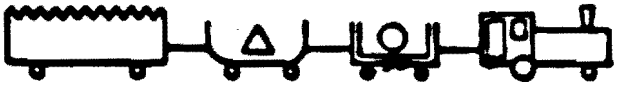
J. 

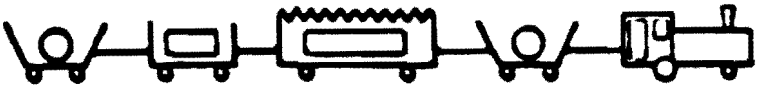
**Class 1: "There are 2 different car shapes in the train"**

B. 

C. 

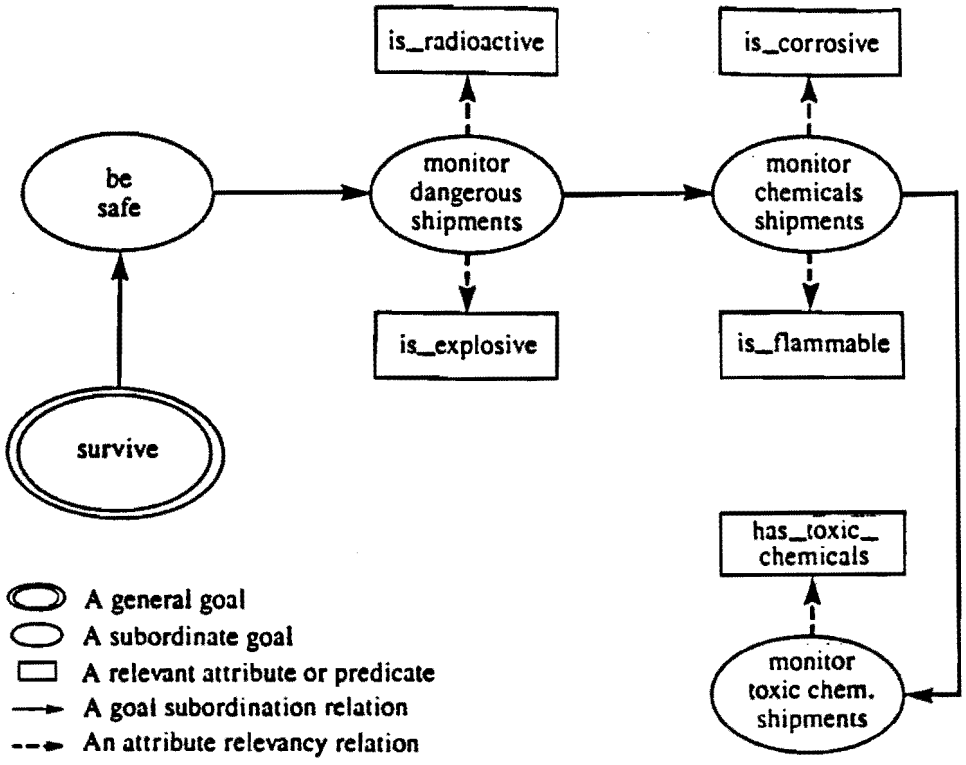
D. 

G. 

I. 

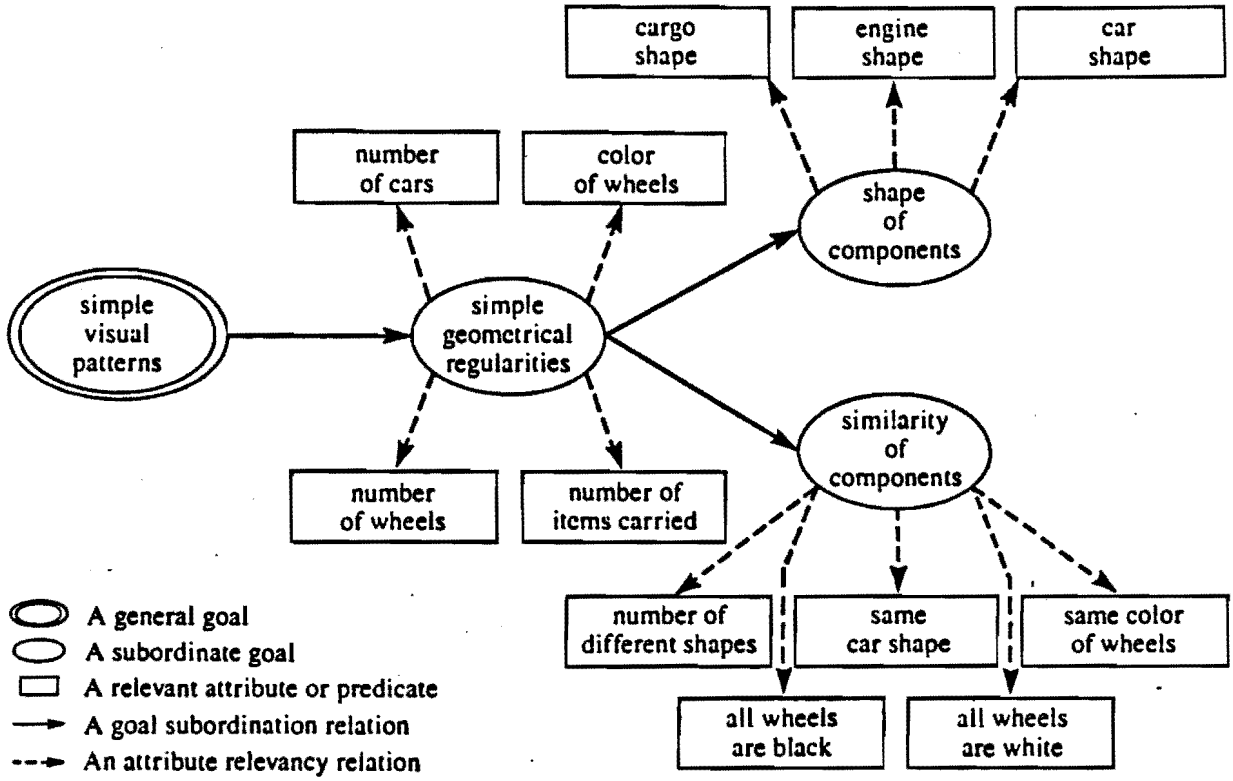
**Class 2: "There are 3 or more different car shapes in the train"**

A Description Constructed by the Program
Figure 1




A GDN for the goal of finding simple visual patterns.

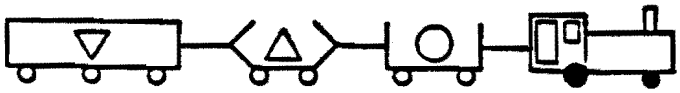
**Figure 2**

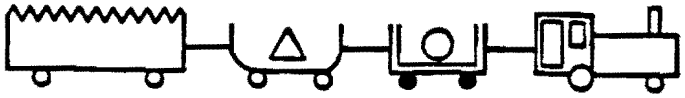



A hypothetical GDN for dangerous train shipments.

Figure 3

A. 

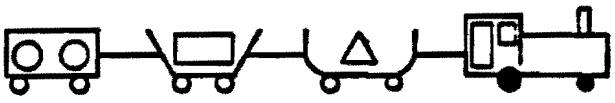
C. 

G. 

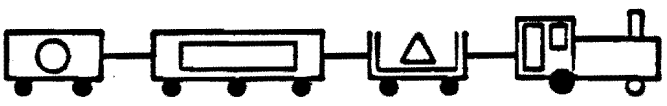
H. 

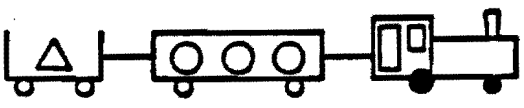
I. 

"These trains are carrying toxic chemicals."

B. 

D. 

E. 

F. 

J. 

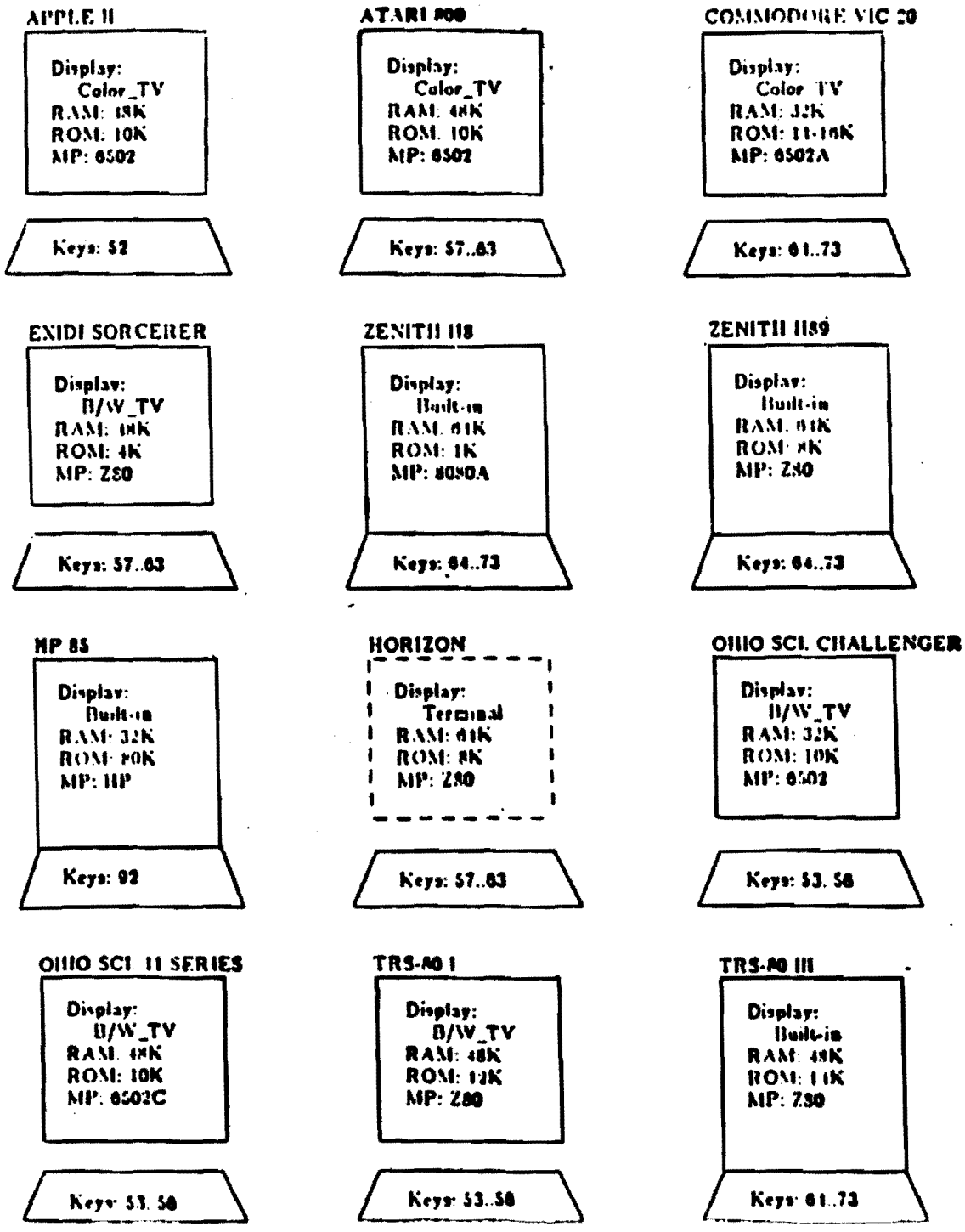
"These trains are not carrying toxic chemicals."

A Description Constructed by the Program

Figure 4

# CLUSTER–EXAMPLE: Microcomputers

Given examples of popular microcomputers (see picture below), the program is supposed to develope a meaningful classification of them.



INPUT:

- A description of each microcomputer in the collection is given in terms of attributes viewed as potentially relevant to its meaningful classification. Here, each microcomputer was described in terms of the type of microprocessor, RAM size, ROM size, display type and number of keys in its keyboard. Below is a sample of descriptions:

  APPLE II: [Microprocessor=6502][RAM=48K][ROM=10K][Display=Color–TV][Keys=52]
  ATARI 800: [Microprocessor=6502][RAM=48K][ROM=10K][Display=Color–TV][Keys=57..63]

  • • •

  TRS–80 III: [Microprocessor=Z80][RAM=48K][ROM=14K][Display=Built–in][Keys=64..73]

- Background knowledge containing information about each attribute. The microprocessor may be 6502, 6502A, 6502C, Z80, 8080A or HP. The size of RAM ranges between 32K and 64K. The size of ROM ranges between 4K and 80K. The display may be a color TV, black and white TV, terminal or built–in type. The number of keys are between 52 and 92. The classification quality criterion was to have "minimum sparseness" of descriptions, and have class descriptions maximally specific.

# OUTPUT:

- Classification of microcomputers.

  CLASSES(microcomputers) = {HP, AVCO, SHT, SHTI}
  HP= {HP 85}
  AVCO= {Apple II, Atari 800, VIC 20, Challenger, Ohio Sci. 11}
  SHT= {Sorcerer, Horizon, TRS 80 III}
  SHTI= {Sorcerer, Horizon, TRS 80 I}

- The rules distinguishing the invented classes.

  - [Class = HP] =>
    [Microprocessor=HP][ROM=80K]
    [Display=Built-in][Keys=92]
    Computers of the class HP have HP microprocessor, 80K ROM, built-in display and 92 keys in the keyboard.

  - [Class = AVCO] =>
    [Microprocessor = 6502x][ROM = 10K .. 16K]
    [Display = TV][Keys = 52 .. 73]
    Computers of the class AVCO have have 6502x microprocessor, ROM between 10K and 16K, TV display and between 52 and 73 keys.

  - [Class = SHT] =>
    [Microprocessor=8080x][Display=Built-in]
    Computers of the class SHT have 8080x microprocessor and built-in display.

  - [Class = SHTI] =>
    [Microprocessor=8080x][Display $\neq$ Built-in]
    Computers of the class SHTI have 8080x microprocessor and do not have built-in display.

# SPARC/G

# Qualitative Prediction

## BRIEF DESCRIPTION:

Given a description of a discrete process (a sequence of events or objects) and background knowledge characterizing domain properties and constraints, the program is supposed to discover rules governing the process and to qualitatively predict its plausible continuation. The program formulates rules using three description models: disjunctive normal form, decomposition and periodic model. The models specify possible forms of the rules and their evaluation criteria. The following examples demonstrate the program in its early stage; future extension will utilize to a greater extent domain specific knowledge and deductive inference so that the program will be able to learn rules that predict more complex processes.

## REFERENCES:

R. S. Michalski, H. Ko and K. Chen, "Qualitative Prediction: A Method and a Program SPARC/G," ISG 85-12, UIUCDCS-F-85-942, Department of Computer Science, University of Illinois, Urbana, February 1985

## EXAMPLES:

- Predicting a Sequence of Geometric Figures
- Man-Robot Skill Transfer
- Discovering Code for a Safe Passage through Mined Channels

# QUALITATIVE PREDICTION
# and
# PROCEDURE LEARNING

## PROJECT SPARC/G

Goal: Qualitative prediction of processes

- A form of PART–TO–WHOLE generalization
- Some Application
  - Agricultural Prediction
    (Insect Infestation Forecast)
  - Medical Diagnosis and Prognosis
  - Robotics
  - Eleusis Game for Modeling Scientific Discovery
  - Stock Market Forecast

# The SPARC/G Methodology

Given:

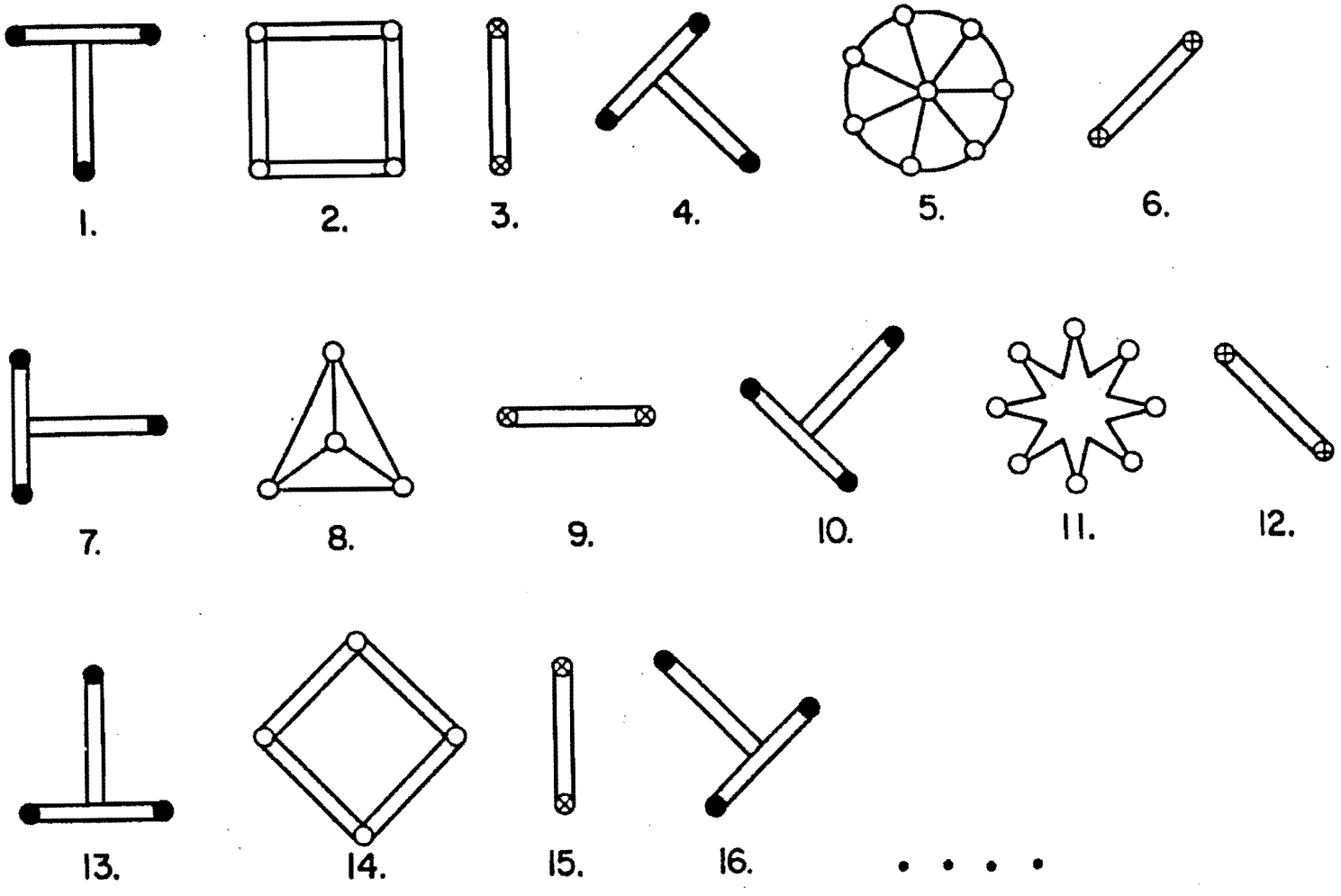
- Sequence of observations (events, situations)
- Background knowledge
- Evaluation criteria
- Goal of prediction
- Model of prediction

Find:

Descriptions which explain the given sequence of observations and predict future events or situations.

## SPARC/G–EXAMPLE: Geometric Figures

Given a sequence of geometric objects (see picture below), the program is supposed to discover the underlying rules that describe the sequence, and based on them predict the properties of the next objects.




*What is the next figure?*

INPUT:

- A description of a sequence is given in terms of all attributes that are viewed as potentially relevant for predicting the sequence. Here, a geometric object in the sequence is described by the number of nodes, the texture of the nodes, the shape, and the orientation of the object:

  1: [number–of–nodes=3][node–texture=solid][shape=T–shape][orientation=0]
  2: [number–of–nodes=4][node–texture=blank][shape=any][orientation=0]

  • • •

  16:[number–of–nodes=3][node–texture=solid][shape=T–shape][orientation=+45]

- Background knowledge containing information about the legal value sets of the attributes, models and preference criteria for evaluating competing hypotheses. The node texture may be blank, cross or solid. The shape of the figure may be T–shape, bar or other shapes. The number of nodes ranges between 2 and 8. The orientation ranges between 0 and 360 degrees and it is periodic. Knowledge for constructing difference attribute of orientation between adjacent figures of the sequence takes into consideration that orientation of 360 degrees is the same as 0 degrees from the periodicity of the domain. Other new attributes are also computed.

OUTPUT:

One of the descriptions created by the program:

Period( [number-of-nodes=3][node-texture=solid]
[shape=T-shape][orientation(i+1)=orientation(i)+45],

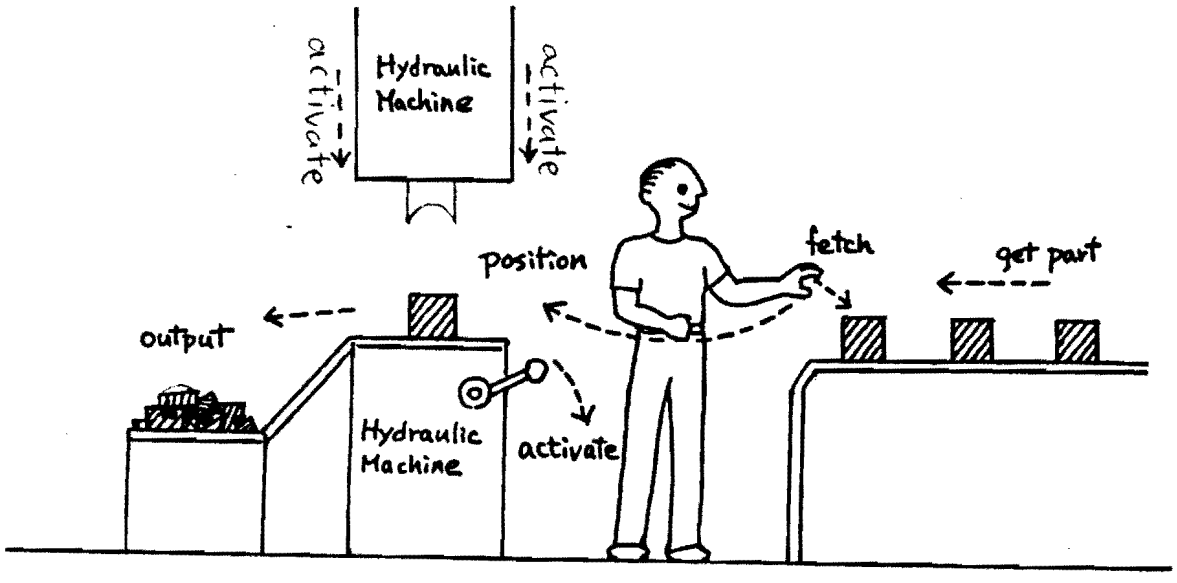
[number-of-nodes>=4][node-texture=blank][shape=any]
Period( [number-of-nodes=4], [number-of-nodes=8]),

[number-of-nodes=2][node-texture=cross]
[shape=bar][orientation(i+1)=orientation(i)-45] )

The description is based on the periodic model. It consists of three repeating phases (separated by ","). The objects in the fist phase have 3 nodes, solid node-texture, are T-shaped, and are rotating counter clockwise by 45 degrees. The objects in the second phase have 4 or more nodes, blank node-texture, and can be of any shape. Within this phase there is an embedded two-phase periodic description stating that the number of nodes alternates between 4 and 8. The objects in the third phase have a pair of nodes, cross node-texture, are of bar shape, and are rotating clockwise by 45 degrees. Thus, one can predict that the next object in the sequence will likely to have 8 number of nodes, blank node-texture and may be of any shape.

## SPARC/G–EXAMPLE: Man–Robot Skill Transfer

Assuming that a robot intend to acquire the knowledge of working on an assembly line by observing the actions of an assembly line worker. Five actions are defined for the assembly worker: get part, fetch part, position the part in the hydraulic machine, active the hydraulic machine, and output the finished part. Five states are also defined for each part: in–place, not–in–place, finished, held–by–worker, and undetermined. The trace is represented as a sequence of actions of the worker, and the states of the part being handled.



INPUT:

- Below is a sequence of descriptions of the states, and the actions taken by the assembly line worker:

  [part-status=not-properly-positioned][worker-action=get-part]
  [part-status=held][worker-action=fetch]
  [part-status=in-position][worker-action=put-into-hydraulic-machine]
  [part-status=finished][worker-action=activate-hydraulic-machine]
  [part-status=undetermined][worker-action=output]
  [part-status=undetermined][worker-action=get-part]

  • • •

  [part-status=not-properly-positioned][worker-action=get-part]
  [part-status=not-properly-positioned][worker-action=fetch]
  [part-status=held][worker-action=fetch]
  [part-status=not-properly-positioned][worker-action=put-into-hydraulic-machine]

- Background knowledge: (1) A set of five actions performed by the assembly worker: get part, fetch part, position the part in the hydraulic machine, activate the hydraulic machine, and output the finished part. (2) A set of five states are also defined for each part: in-place, not-in-place, finished, held-by-worker, and undetermined. The actions and states are considered to be attributes of nominal type. Information about the legal value sets of the attributes, models and preference criteria for evaluating competing hypotheses, and the knowledge for constructing new attributes are also given.

OUTPUT:

The rules discovered by the program:

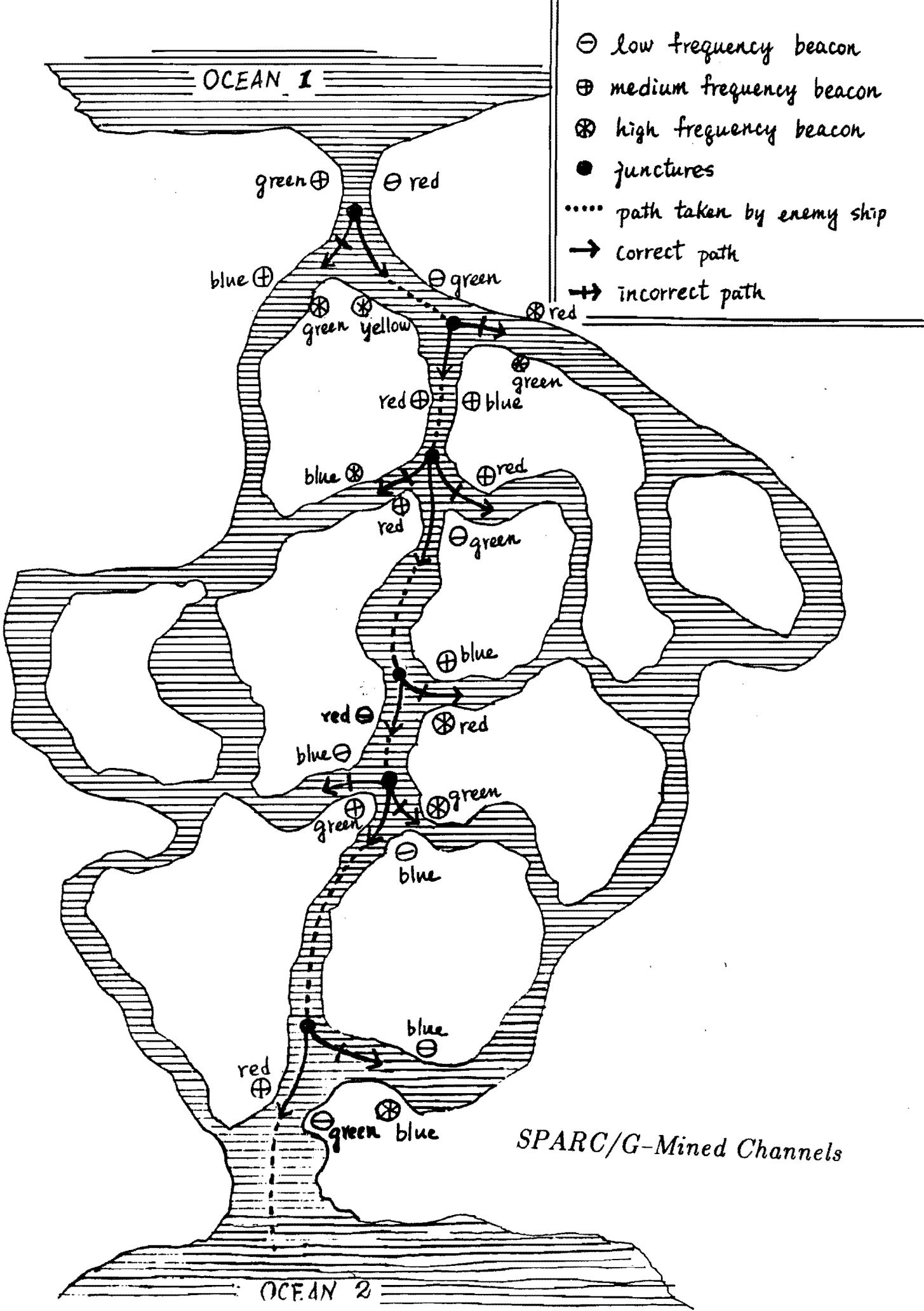
[part–status1 = held] → [action0 = put–into–hydraulic–machine]
[part–status1 = finished] → [action0 = output]
[part–status1 = not–properly–positioned] → [action0 = fetch]
[part–status1 = in–position] → [action0 = activate–hydraulic–machine]
[part–status1 = undetermined] → [action0 = get–part]

If a part is being held, then put it into the hydraulic machine.
If a part is finished, then output it.
If a part is in position, then activate the hydraulic machine.
If a part is nonexistent, then retrieve another part.

## SPARC/G–EXAMPLE: Discovering Code for a Safe Passage through Mined Channels

Suppose there are channels connecting Ocean1 and Ocean2 populated by a number of small islands. The passages around the islands are full of mines that are activated or deactivated remotely by the enemy at different times. The enemy ships can find a safe passage at any given time by knowing a secret code based on the colors and frequencies of the beacons before and after the junctions of the passage. The problem is to discover the code by observing an enemy ship passing through the channels. The channels are shown in the figure "SPARC/G–Mined Channels" (next page).

This example illustrates respectable performance of the program for a problem of relatively high complexity (approximately $10^{120}$ elements in the description space. See discussions in the following paragraphs).


low frequency beacon
medium frequency beacon
high frequency beacon
junctures
path taken by enemy ship
correct path
incorrect path
OCEAN 1
green
red
blue
green
green
yellow
red
green
red
blue
blue
red
red
green
blue
red
red
blue
green
green
blue
blue
red
green
blue
SPARC/G-Mined Channels
OCEAN 2

INPUT:

- A description of the sequence of states of beacons on the passage of the enemy ship is given below.

  [color(LB)=red][frequency(LB)=low][color(RB)=green][frequency(RB)=medium] => [safe]
  [color(LB)=green][frequency(LB)=high][color(RB)=blue][frequency(RB)=medium] => [unsafe]
  [color(LB)=green][frequency(LB)=low][color(RB)=red][frequency(RB)=high] => [safe]

  • • •

  [color(LB)=green][frequency(LB)=low][color(RB)=red][frequency(RB)=medium] => [safe]

  where

  LB – left beacon
  RB – right beacon

- Background knowledge: A beacon is identified by its color and the frequency of its flashing light. The color may be red, green or blue and the frequency may be low, medium or high. The values of the frequency is linearly ordered: low < medium < high. The program is also given the knowledge for constructing new attributes from the given ones.

## OUTPUT:

The program generated the following rules based on the decomposition model:

$$[\text{color(CLB)} = \text{red}] \Rightarrow [\text{frequency(NRB)} > \text{frequency(CRB)}]$$
$$[\text{color(CLB)} = \text{green}] \Rightarrow [\text{frequency(NRB)} < \text{frequency(CRB)}]$$
$$[\text{color(CLB)} = \text{blue}] \Rightarrow [\text{frequency(NRB)} = \text{frequency(CRB)}]$$

where

CRB --- current right beacon
NRB --- next right beacon
CLB --- current left beacon
NLB --- next left beacon

The passage is safe if: the color of CLB is red and the frequency of CRB is lower than that of NRB or the color of CLB is green and the frequency of CRB is higher than that of NRB or the color of CLB is blue and the frequency of CRB is the same as that of NRB.

This is a rather impressive result in view of the complexity of the problem. The complexity can be considered as follows: given that two attributes, *frequency* and *color*, characterize every beacon, assume that the three rule models allow descriptions of no more than three disjuncts, a lookback of no more than three (which implies that only the nearest four pairs of beacons are important in solving the secret code), and a phase of no more than three for periodic model. The cardinality of such a description space is approximately $10^{120}$.

# SPARC/E

## Discovering Secret Rules in Eleusis

BRIEF DESCRIPTION:

The program discovers secret rules in the card game ELEUSIS that models a process of scientific discovery. It also plays the card game using the discovered rules. The rule discovery part of the program is a specialization of the program SPARC/G. The program has proven itself to be a strong player in actual ELEUSIS games against human players.

REFERENCES:

R. S. Michalski, H. Ko and K. Chen, "SPARC/E(V.2) An Eleusis Rule Generator and Game Player", UIUCDCS–F–85–941, ISG 85–11, 1985.

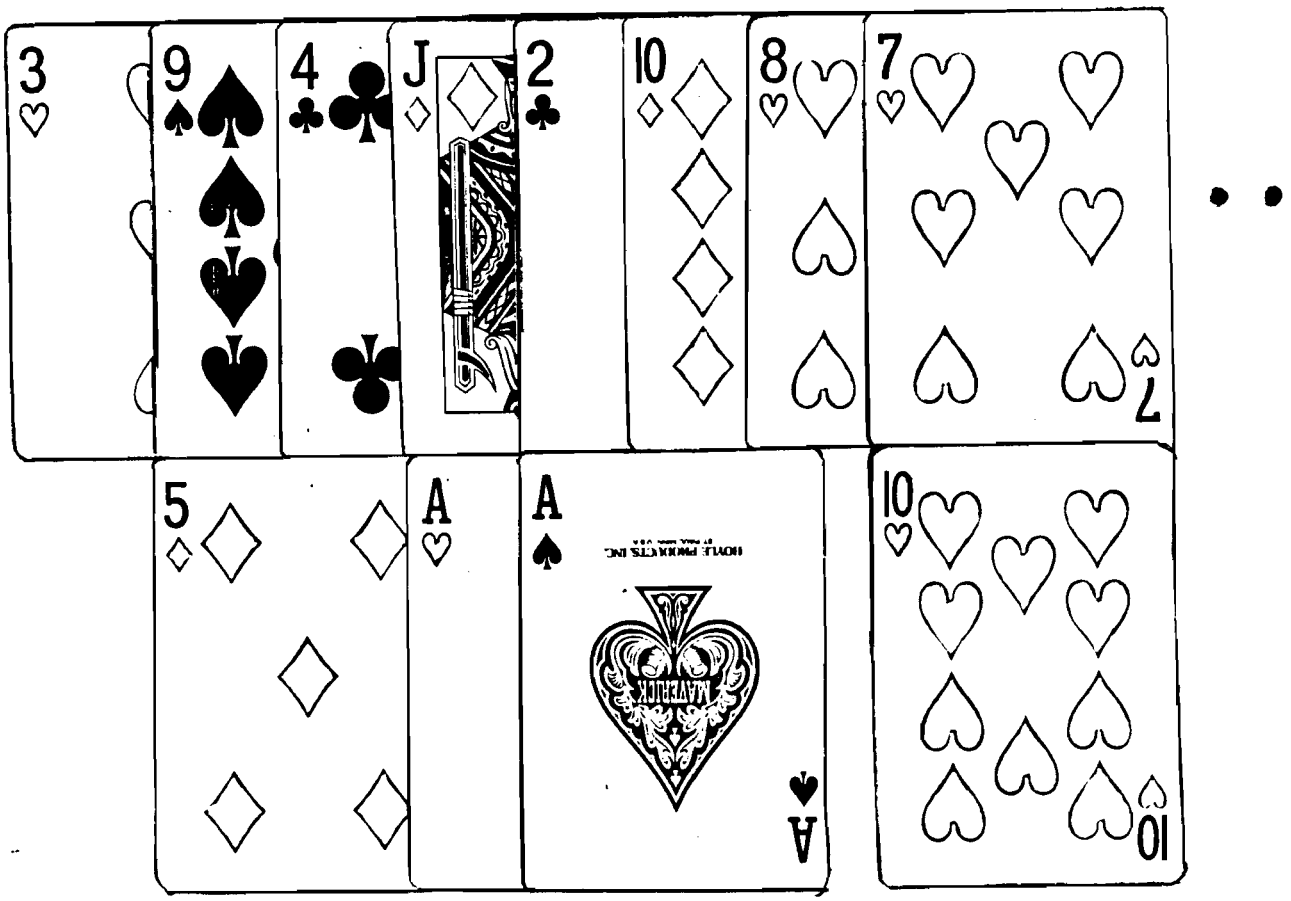
EXAMPLES:

- Game 1 (from Scientific American)
- Game 2 (from Donald Michie)
- Game 3

SPARC/E–EXAMPLE: Game 1 (from Scientific American)

The problem was taken from the article: "MATHEMATICAL GAMES: On Playing New Eleusis, the Game that Simulates the Search for Truth" by Martin Gardner, *Scientific American, October 1977.* The cards in the mainstream represent a sequence satisfying a "secret rule" to be discovered. Each downward card sequence below a card in the mainstream represents an incorrect continuation of the sequence after the card.

Mainstream →



*What is the secret rule?*

## INPUT:

- A description of the sequence of cards:

  [suit=heart][rank=3] => correct
  [suit=spade][rank=9] => correct
  [suit=diamond][rank=5] => incorrect

  • • •

  [suit=diamond][rank=7] => correct

- Background knowledge containing a specification of attributes, their types and domains, and various constructive induction rules. Two initial attributes are: suit and rank. Suit of a card can be club, diamond, heart or spade. Its rank ranges from Ace to King. Constructive induction rules define new attributes as function of initial ones. For example, face card is a card whose rank is Jack, Queen and King. Color of the card is black if its suit is club or spade and red if diamond or heart. The parity of a card is even if its rank is even and odd if its rank is odd. The program is also given models and their preference criteria for candidate rules. Description models constrain the scope of legal syntactic structures of the rules.

## OUTPUT:

The rule discovered by the program:

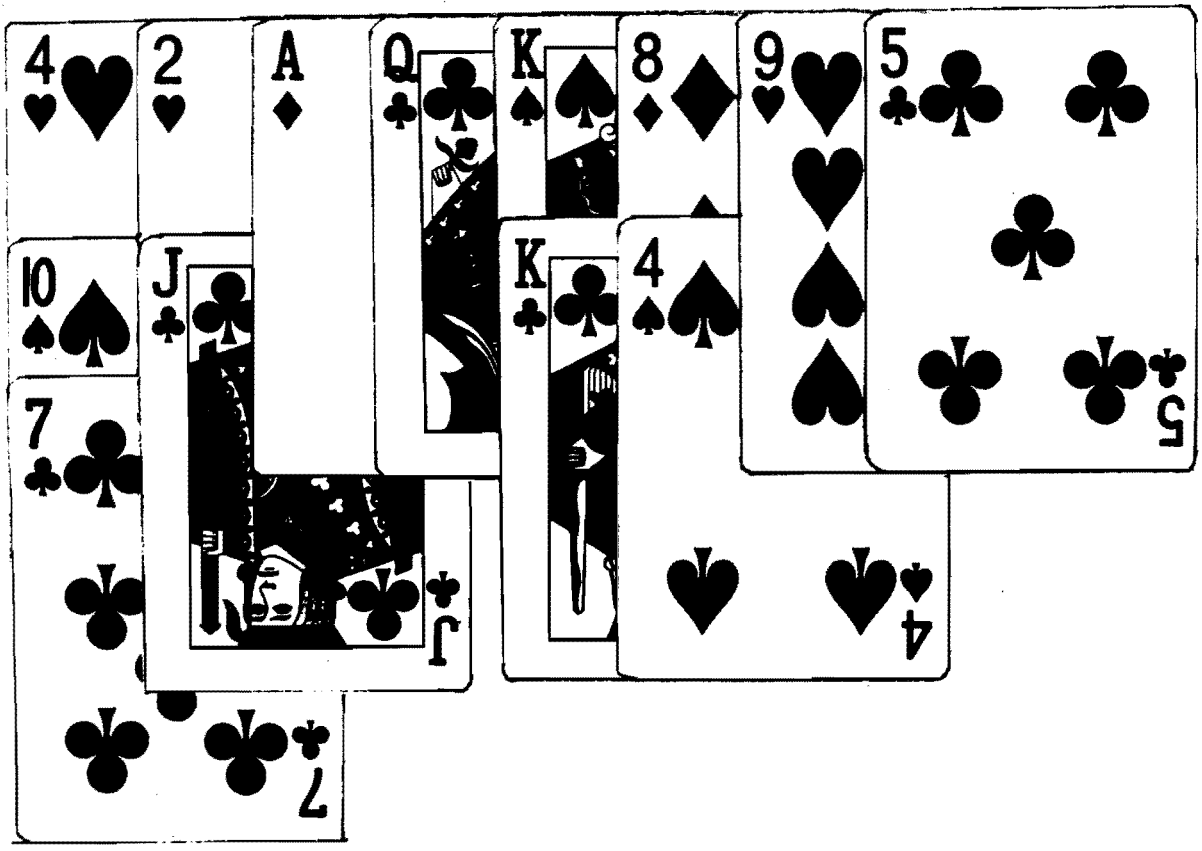
[parity(card1)=odd] => [color(card0)=black]
[parity(card1)=even] => [color(card0)=red]

If the rank of the previous card is odd, then play a black card;
if the rank of the previous card is even, then play a red card.
This is precisely the rule invented by the dealer of the game.

# SPARC/E–EXAMPLE: Game 2 (from Michie)

Given the following layout of cards, find a secret rule governing the order of the cards (the rule was originated by Donald Michie):

Mainstream →



*What is the secret rule?*

INPUT:

- A description of each card in the sequence, together with an indication whether it is a correct or incorrect sequence continuation.

  [suit=heart][rank=4] => correct
  [suit=spade][rank=10] => incorrect

  • • •

  [suit=club][rank=5] => correct

- Background knowledge containing a specification of attributes, their types and domains, and various constructive induction rules. Two initial attributes are: suit and rank. Suit of a card can be club, diamond, heart or spade. Its rank ranges from Ace to King. Constructive induction rules define new attributes as function of initial ones. For example, face card is a card whose rank is Jack, Queen and King. Color of the card is black if its suit is club or spade and red if diamond or heart. The parity of a card is even if its rank is even and odd if its rank is odd. The program is also given models and their preference criteria for candidate rules. Description models constrain the scope of legal syntactic structures of the rules.

OUTPUT:

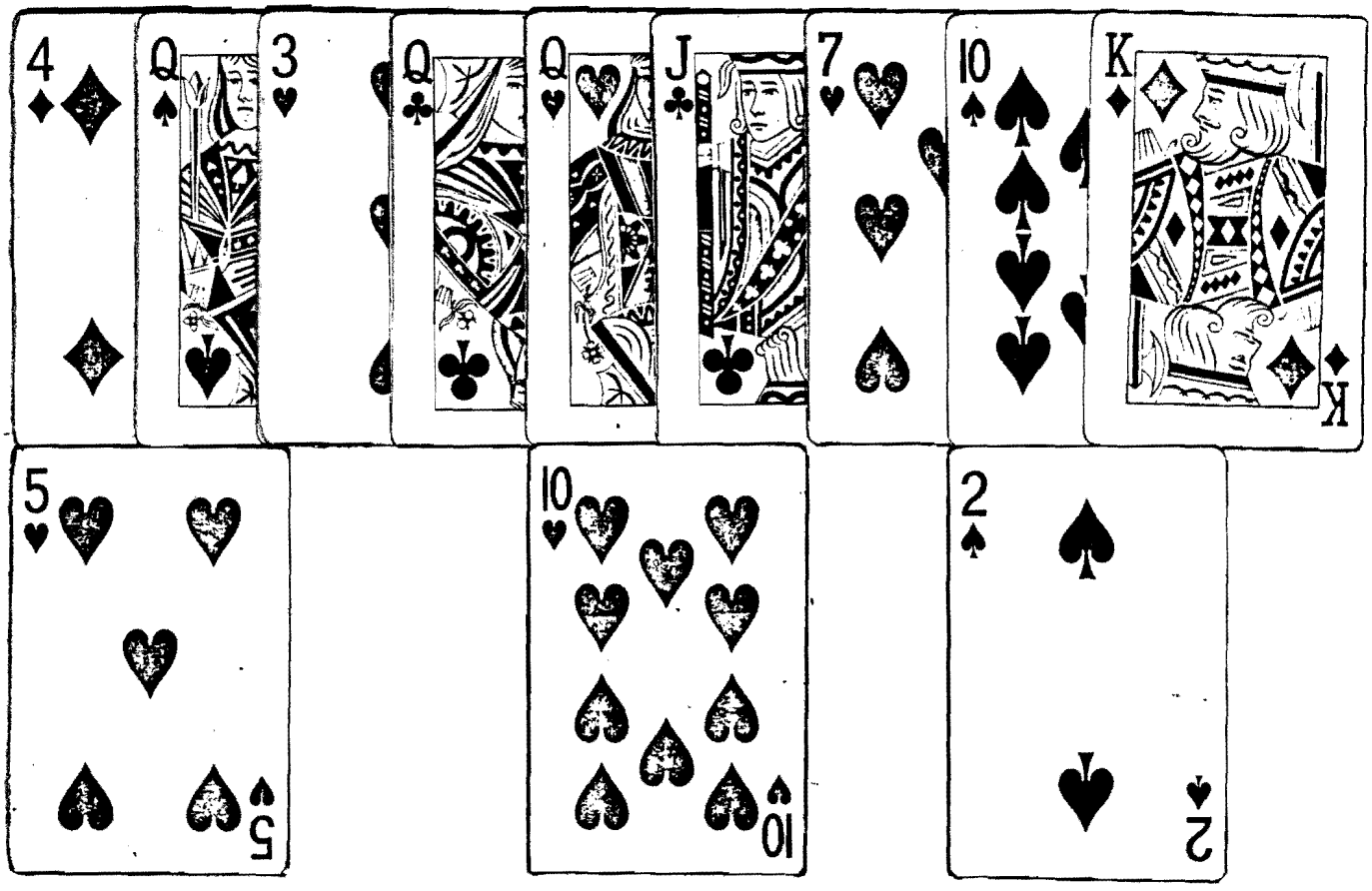
The rule discovered by the program:

[parity(card1)=odd] => [color(card0)<>color(card1)]
[parity(card1)=even] => [color(card0)=color(card1)]

If the value of the previous card is odd, then play a card of different color. If the value of the previous card is even, then play a card of same color. This is precisely the rule invented by the dealer of the game.

# SPARC/E–EXAMPLE: Game 3

A sequence of cards representing a snapshot of the game is shown below:



*What is the secret rule?*

INPUT:

- A description of each card in the sequence, together with an indication whether it is a correct or incorrect sequence continuation.

  [suit=diamond][rank=4] => correct
  [suit=heart][rank=5] => incorrect
  [suit=spade][rank=queen] => correct
  [suit=heart][rank=3] => correct

  • • •

  [suit=spade][rank=2] => incorrect
  [suit=diamond][rank=king] => correct

- Background knowledge containing a specification of attributes, their types and domains, and various constructive induction rules. Two initial attributes are: suit and rank. Suit of a card can be club, diamond, heart or spade. Its rank ranges from Ace to King. Constructive induction rules define new attributes as function of initial ones. For example, face card is a card whose rank is Jack, Queen and King. Color of the card is black if its suit is club or spade and red if diamond or heart. The parity of a card is even if its rank is even and odd if its rank is odd. The program is also given models and their preference criteria for candidate rules. Description models constrain the scope of legal syntactic structures of the rules.

OUTPUT:

The program discovered two rules based on the periodic and decompositional models. The first rule is based on periodic model:

Period( [color = red], [color = black])

which says that the sequence consists of two repeating phases: the first phase contains red cards and the second contains black cards.

The second rule discovered by the program is based on the decompositional model:

[color(card1)=red] => [color(card0)=black]
[color(card1)=black] => [color(card0)=red]

which says that if the previous card is black, then the next card must be red; if the previous card is red, then the next card must be black. This is precisely the rule invented by the dealer of the game.

These above periodic and decompositional rules are equivalent. The example shows that the program may discover equivalent rules using different description models.

| EXAMPLES OF THE DEALER'S SECRET RULE | THE DISCOVERED MACHINE RULE |
|---|---|
| 1. Each string of cards of the same suit have odd number of cards. | Same (7.48 sec,256 kbyte) |
| 2. If the value previous card is even, play a card of the same color, else play a card of different color. | Same (5.06 sec, 277 kbyte) |
| 3. Play alternating faced cards | Same (8.45 sec, 236 kbyte) |
| 4. If the value ofprevious card is even, play red card else play black card | Same (4.9 sec, 235 kbyte) |
| 5. If the number of previously played face cards is even, play an even card else play odd card. | Play cards of different value. (5.76 sec, 245 kbyte) |
| 6. Play cards of alternating color. | Same (6.4 sec, 237 kbyte) |

# ABACUS

## Qualitative Discovery

## BRIEF DESCRIPTION:

The program is an assistant for discovery of mathematical relationships in a given set of data. The data may contain numerical and symbolic information. The program formulates mathematical expressions binding numerical variables, and formulates conditions for which these expressions hold based on symbolic and numeric information.

## REFERENCES:

B. Falkenhainer, "ABACUS Adding Domain Constraints to Quantitative Scientific Discovery," ISG 84–7, UIUCDCS–F–84–927, Department of Computer Science, University of Illinois, Urbana, 1984.

Brian Falkenhainer, "Quantitative Empirical Learning: An Analysis and Methodology", UIUCDCS–F–85–947, ISG 85–16, 1985.

Brian Falkenhainer & Ryszard Michalski, "Integrating Quantitative and Qualitative Discovery: The ABACUS System", Machine Learning Journal No. 4, 1987.

## EXAMPLES:

- Kepler's Law
- Ideal Gas Law
- Falling Bodies

# ABACUS–EXAMPLE: KEPLER'S LAW

Discover a law from quantitative data.

## INPUT:

■ Quantitative data about planets. The table below shows the data given to the programs.

■ Background knowledge: rules for combining numerical quantities, types of given variables, legal ranges of variables and knowledge about units.

| Planet | Distance from the Sun (D) | Period of Rotation around the Sun (P) | Mass | Density |
|---|---|---|---|---|
| Mercury | 0.387 | 0.241 | 0.055 | 5.4 |
| Venus | 0.723 | 0.615 | 0.8 | 5.3 |
| Earth | 1.000 | 1.000 | 1.000 | 5.5 |
| Mars | 1.524 | 1.881 | 0.1 | 3.9 |
| Jupiter | 5.203 | 11.862 | 318.0 | 1.3 |
| Saturn | 9.539 | 29.458 | 95.0 | 0.7 |
| Uranus | 19.191 | 84.015 | 15.0 | 1.2 |
| Neptune | 30.071 | 164.788 | 17.0 | 1.6 |
| Pluto | 39.597 | 249.170 | 0.002 | 1.00 |

## OUTPUT:

A binding between the variable D (distance from the sun) and the variable P (period of rotation around the sun) is discovered by the program. This is precisely the same law Kepler derived.

$$\frac{D^3}{P^2}=1$$

# ABACUS–EXAMPLE: IDEAL GAS LAW

## Discover a law from quantitative properties of ideal gas.

## INPUT:

■ **Quantitative experimental data conerning ideal gas. The table below shows the data given to the program.**

| Volume (V) | Number of Moles (N) | Temperature (T) | Pressure (P) | State |
|---|---|---|---|---|
| 4992 | 42 | 170 | 0.7 | solid |
| 2496 | 1 | 300 | 1 | gas |
| 1664 | 1 | 300 | 1.5 | gas |
| 3328 | 1 | 300 | 0.75 | gas |
| 1996.8 | 1 | 300 | 1.25 | gas |
| 1680 | 20 | 220 | 2 | solid |
| 1690 | 20 | 100 | 2 | solid |
| 4992 | 1.2 | 250 | 0.5 | gas |
| 4992 | 0.8 | 375 | 0.5 | gas |
| 1690 | 20 | 150 | 2 | solid |
| 2496 | 1.1321 | 320 | 1.208 | gas |
| 4992 | 1.2 | 350 | 0.7 | gas |
| 1660 | 20 | 200 | 2 | solid |
| 4992 | 1.2 | 280 | 0.56 | gas |
| 1664 | 2 | 310 | 3.1 | gas |
| 1664 | 2 | 250 | 2.5 | gas |
| 4992 | 42 | 150 | 0.5 | solid |
| 4992 | 42 | 160 | 0.6 | solid |
| 1664 | 2 | 270 | 2.7 | gas |
| 1664 | 2 | 350 | 3.5 | gas |
| 1660 | 20 | 200 | 1 | solid |
| 1400 | 19 | 200 | 1 | solid |

■ Background knowledge: rules for combining numerical quantities, types of given variables, legal ranges of variables and knowledge about units.

OUTPUT:

Binding of the input variables: P (pressure), V (volume), N (number of moles), and T (temperature) in the following form:

$$[\text{state(liquid)}=\text{gas}] => P \cdot V = 8.3202 \cdot N \cdot T$$

This is the same as the Ideal Gas Law.

## ABACUS–EXAMPLE: Falling Bodies

Discover laws governing the motion of free–falling bodies.

INPUT:

■ Quantitative data of different balls falling through different media from rest. The balls dropped came in three sizes, for which there was a rubber ball and a clay ball in each size. The six balls were dropped through three different media, namely glycerol, caster oil, and a vacuum, once each for two different size containers. The following is an instance of the experiment:

| | |
|---|---|
| **Height:** | **1.0 m** |
| **Mass:** | **0.94 kg** |
| **Radius:** | **0.05 m** |
| **Duration:** | **0.055 sec** |
| **Velocity:** | **18.064 m/sec** |
| **Location:** | **Death Valley** |
| **Medium:** | **Glycerol** |

■ Background knowledge: the program is given the attributes of the height of the container, the mass of the ball, its radius, the duration of the fall, the velocity with which it strikes the bottom of the container, location of the experiment, and the medium through which the ball fell. Also given are rules for combining numerical quantities, types of given variables, legal ranges of variables and knowledge about units.

OUTPUT:

The following rules are discovered by the program:

Rule A: If [medium=Vacuum]
then velocity = 9.8175·t

Rule B: If [medium=Glycerol]
then velocity·radius = 0.9556·m

Rule C: If [medium=Castor Oil]
then velocity·radius = 0.7336·m

These rules are precisely specializations the following formula when Stoke's law is taken into consideration:

$$6\pi\eta r v_T = mg$$

where $\eta$ is the viscosity coefficient of the fluid, r is the radius of the ball, $v_T$ is the terminal velocity of the ball, m is the mass of the ball, and g is the gravitational acceleration.

# VARIABLE PRECISION LOGIC

## Reasoning with Incomplete Knowledge Under Time Constraints

## BRIEF DESCRIPTION:

The VPL system represents a significant step toward the development of systems capable of limited rationality. Limited rationality refers to the ability of an intelligent entity to make less than perfect decisions when the situation does not allow for full deliberation. The VPL system is capable of reasoning with incomplete and uncertain knowledge under time constraints. The certainty of inferences is varied to produce a decision with a given time limit. Censored production rules are used to represent both domain and control information. These have the general form

$$P -> D \lfloor C : \delta^+, \delta^-, \gamma^+, \gamma^-$$

Read: If P then D unless C.

The $\delta^+$ value is the lower bound on the probability of D given P and $\neg$C; the $\delta^-$ value is the lower bound on the probability of $\neg$D given P and C; the $\gamma^+$ value is the lower bound on the probability of D given P; and the $\gamma^-$ value is the lower bound on the probability of $\neg$D given P. This is summarized below:

$$\text{prob}(D|P \;\&\; \neg C) \in [\delta^+ \,..\, 1]$$
$$\text{prob}(\neg D|P \;\&\; C) \in [\delta^- \,..\, 1]$$
$$\text{prob}(D|P) \in [\gamma^+ \,..\, 1]$$
$$\text{prob}(\neg D|P) \in [\gamma^- \,..\, 1]$$

The $\gamma^+$ and $\gamma^-$ are constrained by the restriction that $\gamma^+ + \gamma^- \leq 1$, thus $\text{prob}(D|P) \in [\gamma^+ \,..\, (1 - \gamma^-)]$. The belief in each fact is represented as a probability range. Using these probability ranges and bounds, reasoning is performed with a scheme based on Dempster–Shafer theory.

For example, to express the fact that I read the paper before work unless I oversleep, which occurs once or twice a week, we might use the rule:

Weekday–morning –> Read–paper | Oversleep : 1, 1, .6, .2

where the belief factors are interpreted as follows:

- the first "1" states that on weekday mornings when I do not oversleep I read the paper for certain;
- the second "1" states that on weekday mornings when I oversleep I certainly do not read the paper;
- the ".6" states that I read the paper at least three out of five weekday mornings (because I oversleep at most twice a week);
- the ".2" states that I do not read the paper at least one out of five weekday mornings (because I oversleep at least once a week).

# REFERENCES:

Michalski, R.S., Winston,P.H., "Variable Precision Logic" MIT AI Memo 857, Artificial Intelligence Laboratory, MIT, August, 1985 (AI Journal, Vol. 29, No. 2, August 1986).

Haddawy, P., "Implementation of and Experiments with a Variable Precision Logic Inference System", Proceedings AAAI, August 1986.

# VPL–EXAMPLE: AUTONOMOUS CAR

Your autonomous car is taking you for a drive. Suddenly a truck pulls out into the road ahead of you. Your car has only a fraction of a second to decide what to do. This example shows how the VPL system is capable of determining the feasibility of one possible course of action within a given time limit.

INPUT:

■ Rules:
The first rule shown has multiple censors and the four belief factors described above. Some of the rules have no censors and thus they have only two belief factors, the $\gamma$ values.

(¬ speed–distance–ratio high) => (can–stop–in–time)
⌊ (tire–traction poor) ∨ (brake–condition poor) : 1.0 1.0 .7 0

(on ice road) => (tire–traction poor) ⌊ (using–chains) : 1.0 .8 .9 .05

(on gravel road) => (tire–traction poor) : .85 .1

(tire–type snow) => (tire–traction good) : .9 0

(tire–traction good) => (¬ tire–traction poor) : 1.0 0

(temperature below–freezing) & (road–appearance shiny)
=> (on ice road) : .9 .1

(construction–site) & (sound–of–pebbles–hitting–underside–of–car)
=> (on gravel road) : .9 .1

■ Facts:

(speed–distance–ratio high) : .05 .15
(temperature below–freezing) : .1 .2
(road–appearance shiny) : 0 0
(construction–site) : 1.0 1.0
(sound–of–pebbles–hitting–underside–of–car) : .1 .3
(brake–condition poor) : .1 .2
(using–chains) : 0 0
(tire–type snow) : 1.0 1.0

## OUTPUT:

Responses are shown to queries with decreasing time limits. As the time limit decreases, the amount of search performed decreases, and the certainty of the inference, defined as the width of the probability interval, correspondingly decreases.

The system is requested to determine if the car can stop in time within 0.15 second. It decides that it can chain on the censors to a rule depth of three, which corresponds to complete search, and finishes the inference in 0.138 second. The probability of stopping in time lies somewhere in the interval 0.61 to 0.91. The most likely range is 0.79 to 0.91, which indicates an uncertainty of (0.91 – 0.79) = 0.12. The results are in the form of probability ranges because the available information was insufficient to restrict the result to a point value. Based on the most likely range, we may conclude that the car problably can stop in time.

ENTER Command or make query of system
> (can–stop–in–time) in .15 seconds

using censor chaining depth of 3
0.138 seconds elapsed time

<RESULT>
Most Likely Range = [0.79 0.91]
Possible Range = [0.61 0.91]

Next, the system is requested to perform the same inference within 0.1 second. Using a chaining depth of one, the inference is completed in 0.061 second. The system again concludes that the car probably can stop in time but with an uncertainty of 0.37.

ENTER Command or make query of system
> (can–stop–in–time) in .1 seconds

using censor chaining depth of 1
0.061 seconds elapsed time

<RESULT>
Most Likely Range = [0.54 0.91]
Possible Range = [0.00 0.91]

Finally, when the time limit is set to 0.04 second a chaining depth of zero is used and the inference is completed in 0.020 second. No evidence for the censors is found and the system concludes that the car can probably stop in time but with very low certainty.

ENTER Command or make query of system
> (can–stop–in–time) in .04 seconds

using censor chaining depth of 0
0.020 seconds elapsed time

<RESULT>
Most Likely Range = [0.60 1.00]
Possible Range = [0.00 1.00]

The results show that the VPL system was able to make a rational decision about the ability of the car to stop even though both time and information were limited. The effect of the time limit on the certainty of the inference is transparent to the user, as reflected in the probability range of the answer. The "possible range" shows the effect of the immedicate evidence on the conclusion and indicates to the user whether the decision could possibly be reversed given more information. The "most likely range" augments the immediate evidence with past experience to produce an answer of varying default character.