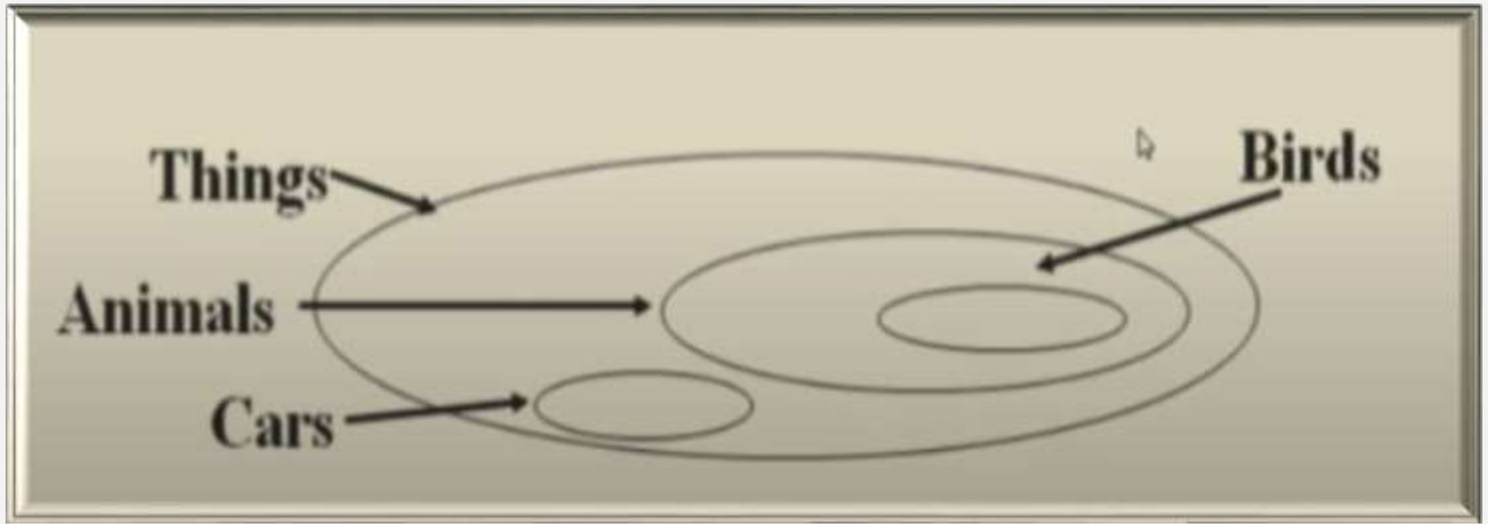
CONCEPT LEARNING AND THE GERNERAL-TO-SPECIFIC ORDERING

OVERVIEW

- Concept
- Concept Learning
- Concept Learning Task
- Concept Learning as a Search
- **Find-S:** Finding a maximally specific Hypothesis
- Version Spaces and the **CANDIDATE-ELIMINATION**
- Inductive Bias

WHAT IS A CONCEPT?

- A **Concept** is a subset of objects or events defined over a larger set.
- For example, we refer to the set of everything(i.e. all objects) as the set of things.
- Animals are a subset of things, and birds are subset of animals.
- In more technical terms, a Concept is a Boolean valued function defined over this larger set.
- For example, a function defined over all *animals* whose value is *true* for *birds* and *false* for every *other animal.*



CONCEPT LEARNING

- ***Concept learning:*** Acquiring/Learning the definition of a general category ***(concept)*** *from a* given a sample of positive and negative training examples of the category. *(Concept may be an event, an object …)*
- ***Concept learning:*** is a learning task in which we train a machine to learn some concept by giving some predefined examples.
- ***Concept learning*** can be formulated as a problem of searching through a predefined space of potential hypotheses for the **hypothesis** that best fits the training examples.
- Target Function (that we assume exists) that can best map inputs to outputs on all possible observations from the problem domain is called a **hypothesis** in machine learning.

- ***Formal Definition for Concept learning:*** **Inferring a boolean-valued function from training examples of its input and output.**

CONCEPT LEARNING-HYPOTHESIS

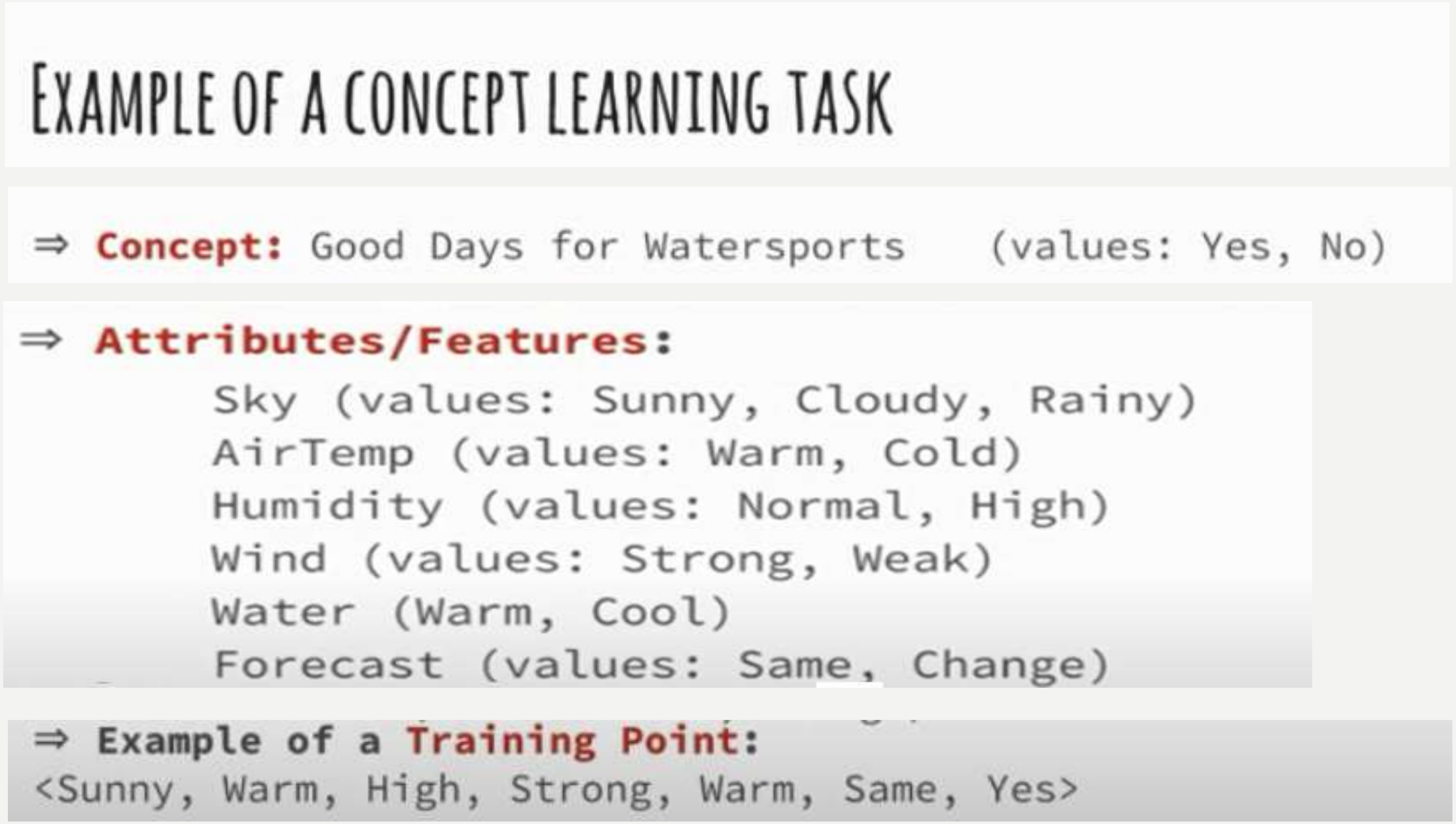
A learning task involves a set of input variables and a set of output variables.

The set of possible relationships (hypotheses) between input and output variables is known as the **hypothesis space**.

Hypothesis have representations such as numerical functions, symbolic rules, decision trees and artificial neural nets.

Most learning is performed in a closed world where the hypothesis space is predefined and finite.

A CONCEPT LEARNING TASK



Example of a concept learning task

⇒ **Concept:** Good Days for Watersports (values: Yes, No)

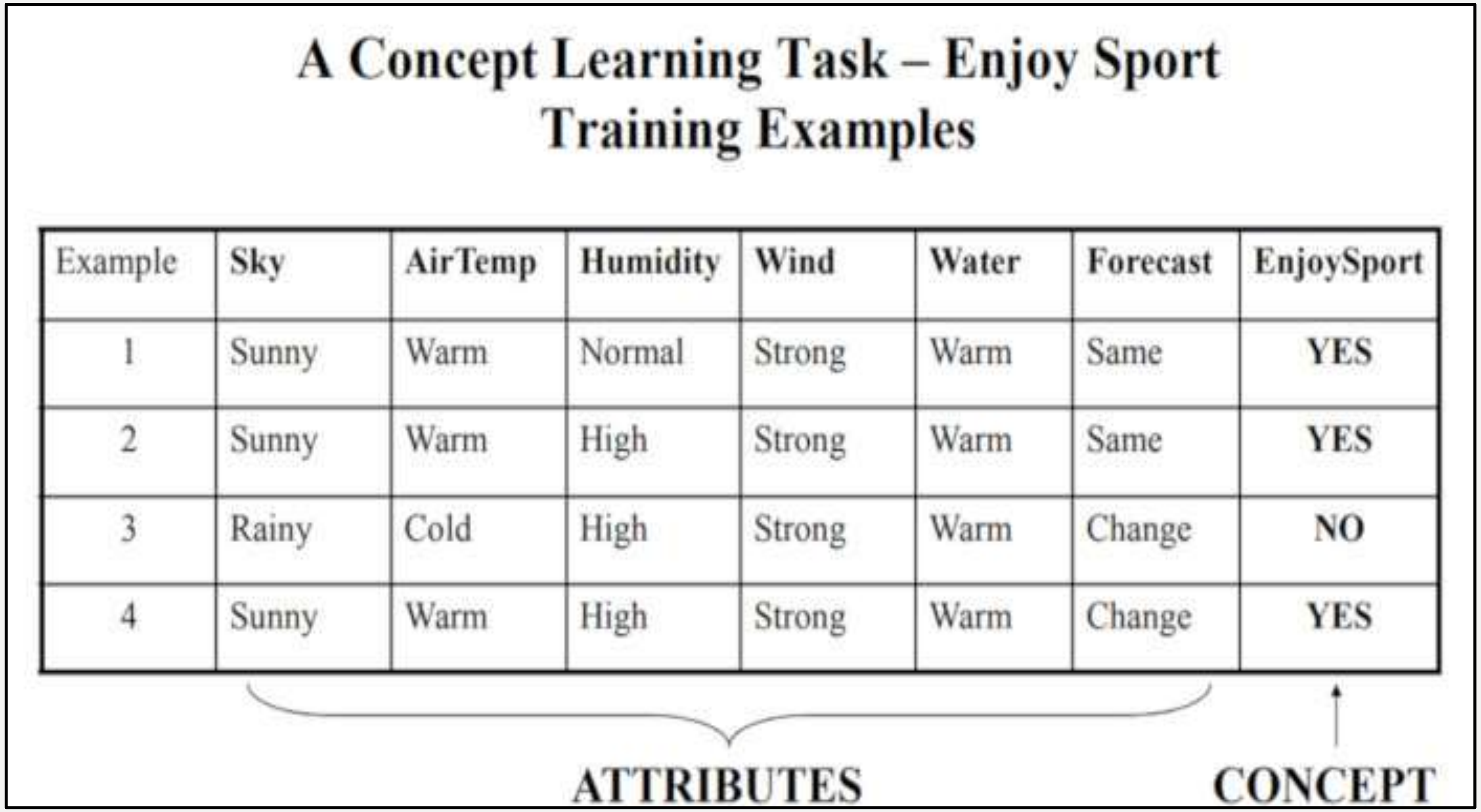
⇒ **Attributes/Features:**

```
Sky (values: Sunny, Cloudy, Rainy)
AirTemp (values: Warm, Cold)
Humidity (values: Normal, High)
Wind (values: Strong, Weak)
Water (Warm, Cool)
Forecast (values: Same, Change)
```

⇒ **Example of a Training Point:**

```
<Sunny, Warm, High, Strong, Warm, Same, Yes>
```

A CONCEPT LEARNING TASK



A Concept Learning Task – Enjoy Sport
Training Examples

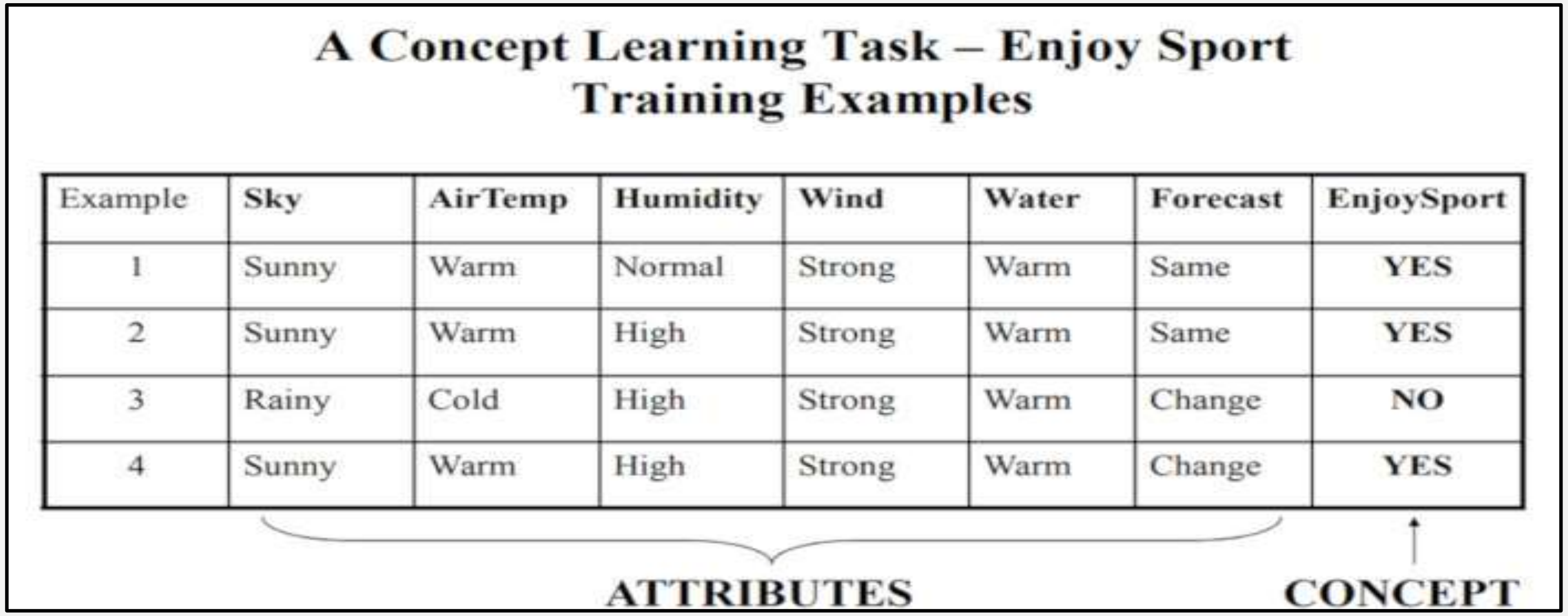
Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	YES
2	Sunny	Warm	High	Strong	Warm	Same	YES
3	Rainy	Cold	High	Strong	Warm	Change	NO
4	Sunny	Warm	High	Strong	Warm	Change	YES

ATTRIBUTES

CONCEPT

A CONCEPT LEARNING TASK

- Consider the example task of learning the target concept "**days on which my friend Beetha enjoys her favorite water sport.**"
- Table describes a set of example days, each represented by a set of ***attributes*** ***(Sky, AirTemp, Humidity, Wind, Water, and Forecast).***
- The attribute ***EnjoySport*** indicates whether or not Beetha enjoys her favorite water sport on this day.
- The task is to learn to predict the value of ***EnjoySport*** for an arbitrary day, based on the values of its other attributes.



A Concept Learning Task – Enjoy Sport Training Examples

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	YES
2	Sunny	Warm	High	Strong	Warm	Same	YES
3	Rainy	Cold	High	Strong	Warm	Change	NO
4	Sunny	Warm	High	Strong	Warm	Change	YES

ATTRIBUTES

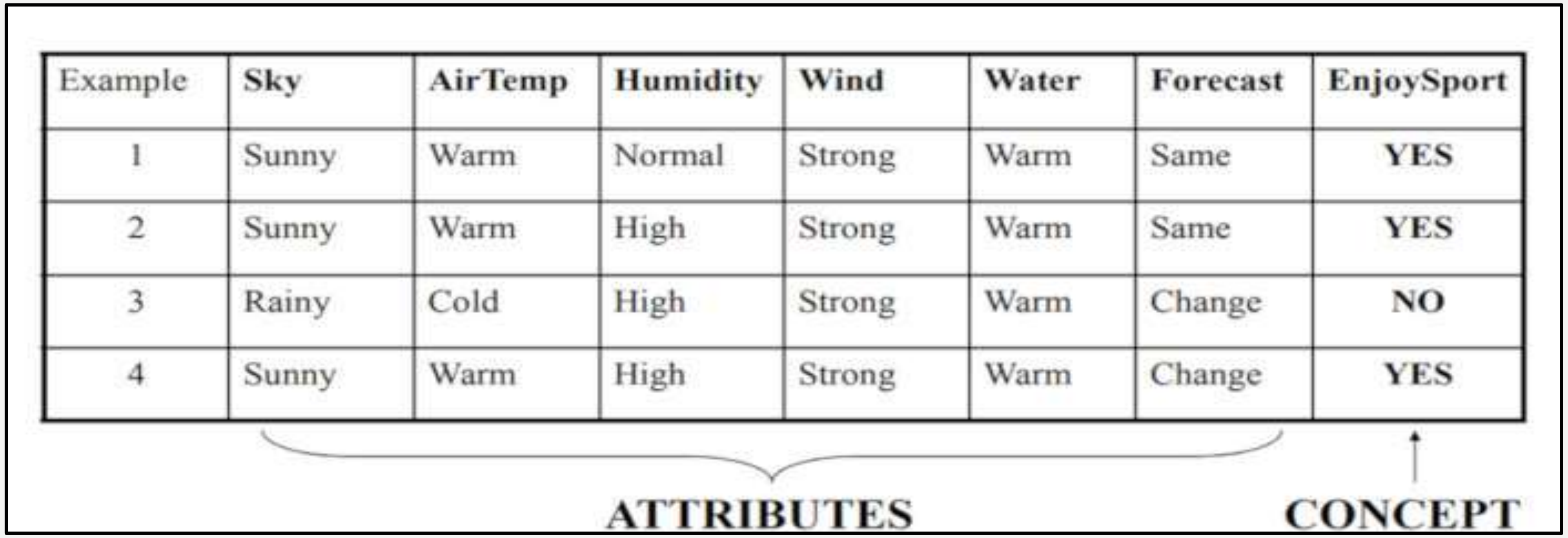
CONCEPT

A CONCEPT LEARNING TASK

What hypothesis representation shall we provide to the learner in this case?

- Let us begin by considering a simple representation in which each hypothesis consists of a conjunction of constraints on the instance attributes.
- In particular, let each hypothesis be a vector of six constraints, specifying the values of the six attributes
  - ***Sky – (values: Sunny, Cloudy, Rainy),***
  - ***AirTemp – (values: Warm, Cold),***
  - ***Humidity – (values: Normal, High),***
  - ***Wind – (values: Strong, Weak),***
  - ***Water – (values: Warm, Cold),***
  - ***Forecast – (values: Same, Change).***
- *For each attribute*, the hypothesis will either
  - ***indicate by a*** *"?'* ***that any value is acceptable for this attribute,***
  - ***specify a*** *single required value* ***(e.g., Warm) for the attribute, or***
  - ***indicate by a*** *"Ø"* ***that no value is acceptable.***
- If some instance x satisfies all the constraints of hypothesis h, then h classifies x as a positive example (**h(x) = 1**).

A CONCEPT LEARNING TASK



Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	YES
2	Sunny	Warm	High	Strong	Warm	Same	YES
3	Rainy	Cold	High	Strong	Warm	Change	NO
4	Sunny	Warm	High	Strong	Warm	Change	YES

$\underbrace{\text{Sky} \ldots \text{Forecast}}_{\text{ATTRIBUTES}}$ CONCEPT

- To illustrate, the hypothesis that
  - Beetha enjoys her favorite sport only on **warm days** with **high humidity** (independent of the values of the other attributes) is represented by the expression

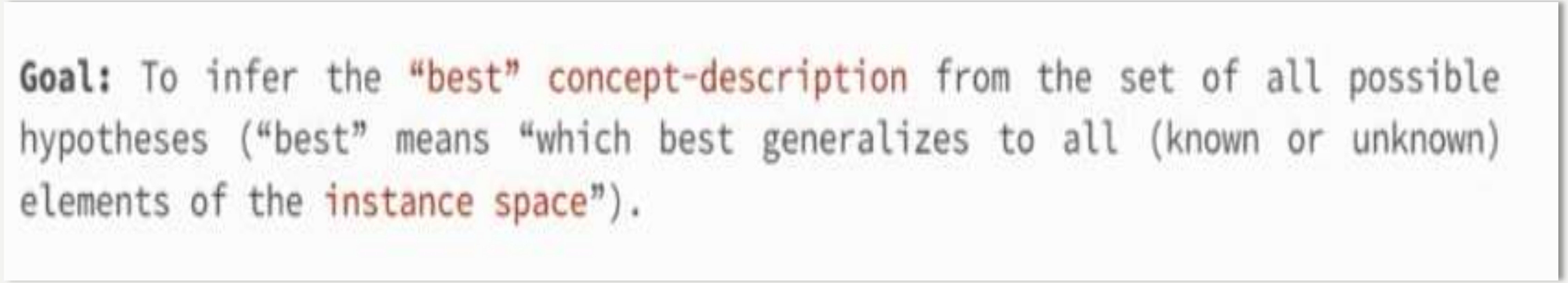
    <?, Warm, High, ?, ?, ?>
  - The most **general hypothesis** -Everyday is positive example-is represented by

    <?, ?,?, ?, ?, ? >
  - The **most specific possible hypothesis**- No day is a positive example-is represented by

    <Ø, Ø, Ø, Ø, Ø, Ø>

A CONCEPT LEARNING TASK

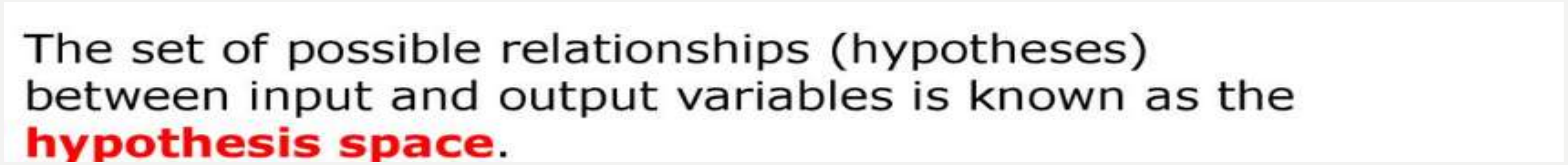
- The *EnjoySport* concept learning task requires learning the set of days for which ***EnjoySport = yes***, describing this set by a conjunction of constraints over the instance attributes.



Goal: To infer the "best" concept-description from the set of all possible hypotheses ("best" means "which best generalizes to all (known or unknown) elements of the instance space").

SO FAR.....

- Concept is a Boolean valued function defined over this larger set.
- ***Concept learning:*** **Inferring a boolean-valued function from training examples of its input and output.**
- Target Function (that we assume exists) that can best map inputs to outputs on all possible observations from the problem domain is called a **hypothesis** in machine learning.


The set of possible relationships (hypotheses) between input and output variables is known as the **hypothesis space**.

- **Concept Learning Task :**Concept Learning is a learning task in which we train the machine to learn some concept by giving some predefined examples.
- **General Hypothesis-** *<?, ?,?, ...?, ?, ? >* ***[accept all]***
- **Specific Hypothesis-*<?, Warm, High, ?, ?, ?>***
- **Most Specific Possible Hypothesis-<Ø, Ø, Ø, Ø, Ø, Ø>** ***[reject all]***

A CONCEPT LEARNING TASK- NOTATION

- The set of items over which the concept is defined is called the set of instances, which we denote by **X**.
- In the current example, **X** is the set of all possible days, each represented by the attributes ***Sky, AirTemp, Humidity, Wind, Water, and Forecast***.
- The concept or function to be learned is called the **target concept**, which we denote by **c**.
- In general, **c** can be any boolean valued function defined over the instances **X**; that is, ***Target function*****: a boolean function**

  $c: X \rightarrow \{0, 1\}$

- In the current example, the **target concept** corresponds to the value of the attribute ***EnjoySport.*** i.e.,
  - **c(x) = 1** if *EnjoySport* = Yes, and
  - **c(x) = 0** if *EnjoySport* = No**.**

A CONCEPT LEARNING TASK- NOTATION

When learning the target concept, the learner is presented a set of *training examples, each consisting of an instance x from X, along with its target concept* value ***c(x)****.*

- *Instances for which* ***c(x) = 1*** are called ***positive examples****, or members of the target concept.*
- *Instances for which* ***c(x) = 0*** *are called* ***negative examples****, or nonmembers of the target concept.*
- We will often write the ordered pair ***(x, c(x))*** *to describe the training example* consisting of the instance **x** and its target concept value ***c(x)****.*
- *We use the symbol* ***D*** *to denote the set of available training examples.*
- Given a set of training examples of the target concept ***c****, the problem faced* by the learner is to hypothesize, or estimate, ***c****.*
- *We use the symbol* ***H*** *to denote* the ***set of all possible hypotheses*** *that the learner may consider regarding the* identity of the target concept.
- In general, each hypothesis ***h*** *in* ***H*** *represents* a boolean-valued function defined over **X**;
  - that is, ***h : X → {0, 1}.***
- *The goal of the* learner is to find a hypothesis ***h*** *such that* ***h(x) = c(x)*** *for all x in X.*

FORMAL TASK DESCRIPTION

- Given:
  - ***X*** all possible days, as described by the attributes
  - A set of hypothesis ***H***, a conjunction of constraints on the attributes, representing a function

    [$h(x) = 1$ if x satisfies h; $h(x) = 0$ if x does not satisfy h]
  - A target concept: $c: X \rightarrow \{0, 1\}$ where

    $c(x) = 1$ iff *EnjoySport* = *Yes*;

    $c(x) = 0$ iff *EnjoySport* = *No*;
  - A training set of possible instances $D: \{\langle x, c(x)\rangle\}$
- Goal: find a hypothesis *h* in *H* such that

  $h(x) = c(x)$ **for all** x **in** D

  Hopefully ***h*** will be able to predict outside ***D***

HYPOTHESES REPRESENTATION

- *h* is a set of constraints on attributes:
  - a specific value: e.g. *Water* = *Warm*
  - any value allowed: e.g. *Water* = ?
  - no value allowed: e.g. *Water* = Ø
- Example hypothesis:

Sky	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*
⟨Sunny,	**?,**	**?,**	***Strong,***	**?,**	***Same⟩***

  Corresponding to boolean function:

  Sunny(Sky) ∧ Strong(Wind) ∧ Same(Forecast)

- ***H***, hypotheses space, all possible *h*

HYPOTHESIS SATISFACTION

- An instance ***x*** *satisfies* an hypothesis ***h*** iff all the constraints expressed by ***h*** are satisfied by the attribute values in ***x***.
- **Example 1:**

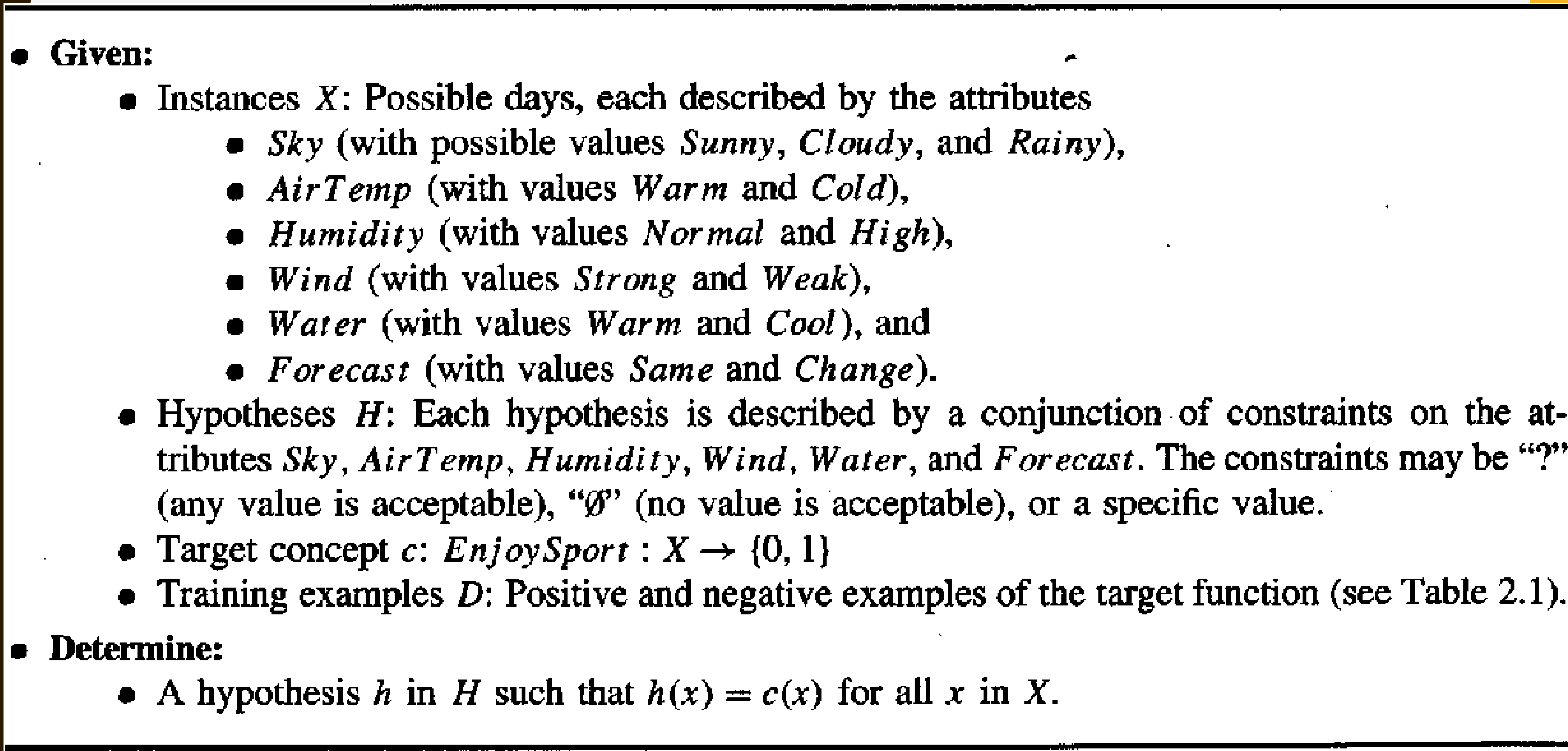
  x_1: ⟨*Sunny, Warm, Normal, Strong, Warm, Same*⟩

  h_1: ⟨*Sunny*, ?, ?, *Strong*, ?, *Same*⟩ **Satisfies? Yes**
- **Example 2**:

  x_2: ⟨*Sunny, Warm, Normal, Strong, Warm, Same*⟩

  h_2: ⟨*Sunny*, ?, ?, Ø, ?, *Same*⟩ **Satisfies? No**

THE *ENJOYSPORT* CONCEPT LEARNING TASK NOTATION



- **Given:**
  - Instances X: Possible days, each described by the attributes
    - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*),
    - *AirTemp* (with values *Warm* and *Cold*),
    - *Humidity* (with values *Normal* and *High*),
    - *Wind* (with values *Strong* and *Weak*),
    - *Water* (with values *Warm* and *Cool*), and
    - *Forecast* (with values *Same* and *Change*).
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "Ø" (no value is acceptable), or a specific value.
  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples D: Positive and negative examples of the target function (see Table 2.1).
- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

THE INDUCTIVE LEARNING HYPOTHESIS

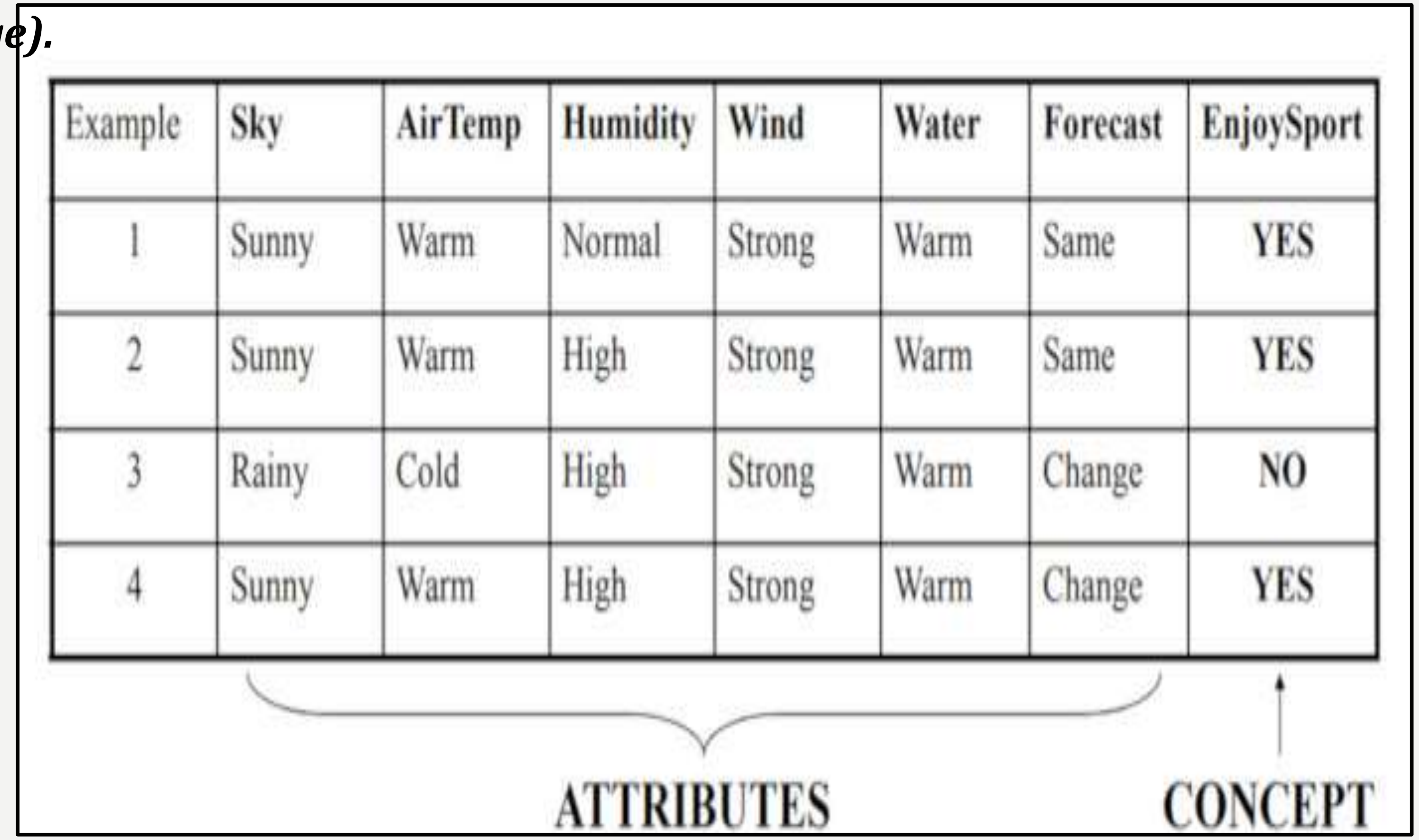
- **Inductive learning** is a learning approach in which rules(hypothesis) are inferred from the facts or data.
- **Inductive learning algorithms** can at best guarantee that the output hypothesis fits the target concept over the training data.
- ***Assumption*****:** an hypothesis that approximates well the training data will also approximate the target function over unobserved examples. i.e. given a significant training set, the output hypothesis is able to make predictions.
- **The inductive learning hypothesis**: Any hypothesis found to approximate the target function well over a sufficiently large set of training examples will also approximate the target function well over other unobserved examples.

CONCEPT LEARNING AS SEARCH

- Concept learning is a task of searching an hypotheses space.
- The goal of this search is to find the hypothesis that best fits the training examples.
- It is important to note that by selecting a hypothesis representation, the designer of the learning algorithm implicitly defines the space of all hypotheses that the program can ever represent and therefore can ever learn.

ENJOY SPORT – HYPOTHESIS SPACE

- *Sky – (values: Sunny, Cloudy, Rainy),*
- *AirTemp – (values: Warm, Cold),*
- *Humidity – (values: Normal, High),*
- *Wind – (values: Strong, Weak),*
- *Water – (values: Warm, Cold),*
- *Forecast – (values: Same, Change).*



Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	YES
2	Sunny	Warm	High	Strong	Warm	Same	YES
3	Rainy	Cold	High	Strong	Warm	Change	NO
4	Sunny	Warm	High	Strong	Warm	Change	YES

ATTRIBUTES
CONCEPT

ENJOY SPORT – HYPOTHESIS SPACE

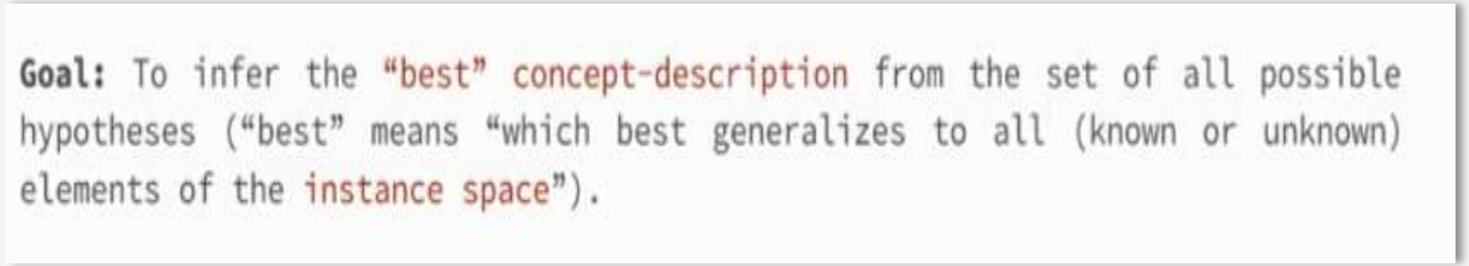
- Given that the attribute **Sky** has 3 possible values, and other 5 attributes have 2 possible values, thus, there are **3.2.2.2.2.2= 96** ***distinct instances.***
- There are **5.4.4.4.4.4=5120** ***syntactically distinct hypotheses*** within **H**(including '?' and 'Ø').
- Every hypothesis containing one or more " Ø " symbols represents the empty set of instances; that is, it classifies every instance as negative.
  - Thus there are **(1+4.3.3.3.3.3)=973** ***symantically distinct hypotheses*** within **H**(including '?' for each attribute and one 'Ø' common to all).
- Although **EnjoySport** example has a small, finite hypothesis space, most learning tasks have much larger(even infinite) hypothesis spaces.
  - We need efficient search algorithms on the hypothesis spaces.

SO FAR.....

- Concept is a Boolean valued function defined over this larger set.
- ***Concept learning:*** **Inferring a boolean-valued function from training examples of its input and output.**
- Target Function (that we assume exists) that can best map inputs to outputs on all possible observations from the problem domain is called a **hypothesis** in machine learning.
- **Concept Learning Task :**Concept Learning is a learning task in which we train the machine to learn some concept by giving some predefined examples.
- **General Hypothesis-** ***<?, ?,?, ...?, ?, ? > [accept all]***
- **Specific Hypothesis-<?, *Warm, High, ?, ?, ?*>**
- **Most Specific Possible Hypothesis-<Ø, Ø, Ø, Ø, Ø, Ø>** ***[reject all]***

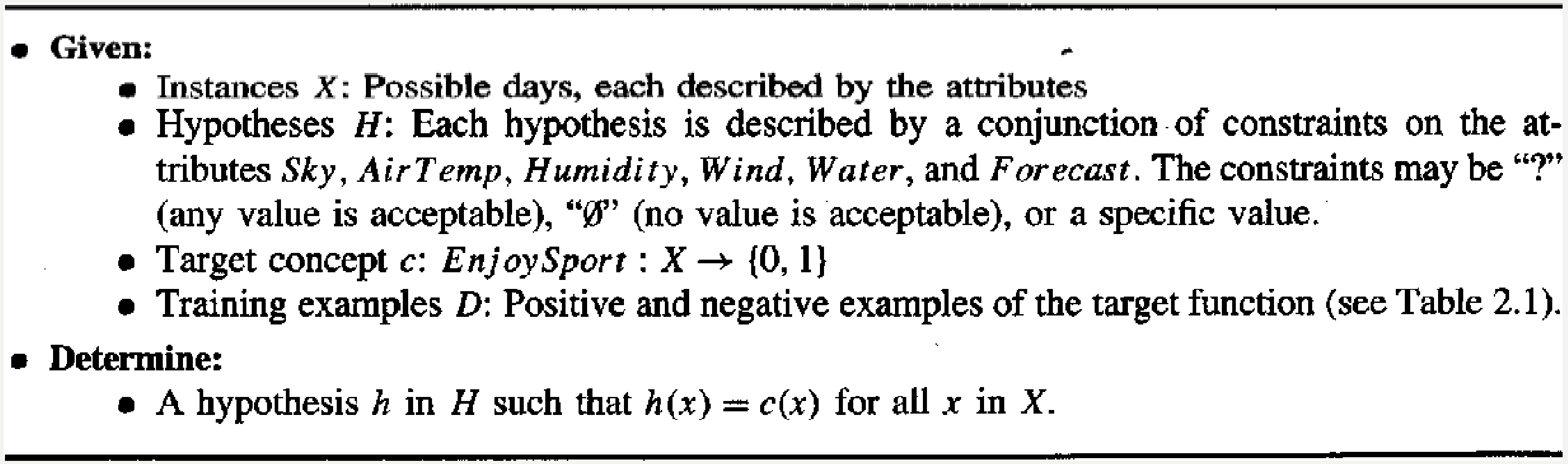
SO FAR..

- **Inductive Learning**
  - is a learning approach in which rules(hypothesis) are inferred from the facts or data.
- **Concept Learning as a Search**



Goal: To infer the "best" concept-description from the set of all possible hypotheses ("best" means "which best generalizes to all (known or unknown) elements of the instance space").

- **Concept Learning Task Notation-*Enjoysport***



- **Given:**
  - Instances X: Possible days, each described by the attributes
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "Ø" (no value is acceptable), or a specific value.
  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples D: Positive and negative examples of the target function (see Table 2.1).
- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

GENERAL TO SPECIFIC ORDERING

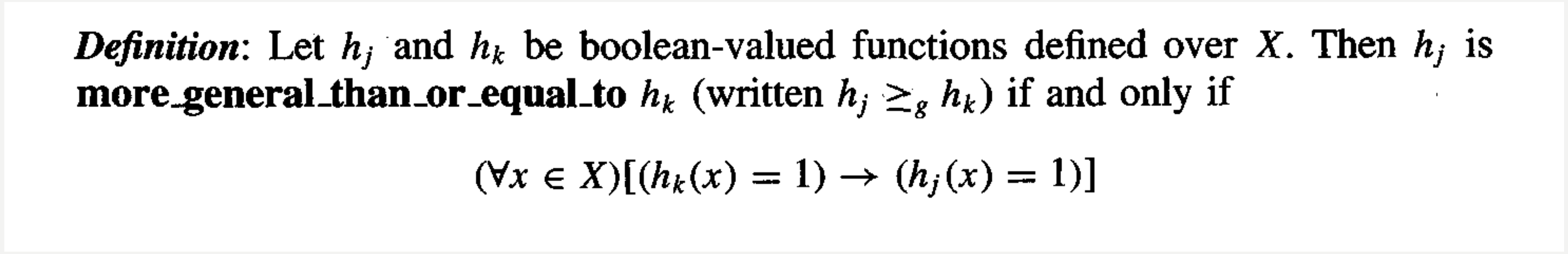
- Many algorithms for concept learning organize the search through the hypothesis space by relying on a very useful structure that exists for any concept learning problem: a **general-to-specific ordering** of hypotheses.
- By taking advantage of this naturally occurring structure over the hypothesis space, we can design learning algorithms that exhaustively search even infinite hypothesis spaces without explicitly enumerating every hypothesis.
- To illustrate the general-to-specific ordering, consider the two hypotheses

  h1 = (Sunny, ?, ?, Strong, ?, ?)

  h2 = (Sunny, ?, ?, ?, ?, ?)
- We can say that, h2 is more general than h1

GENERAL TO SPECIFIC ORDERING

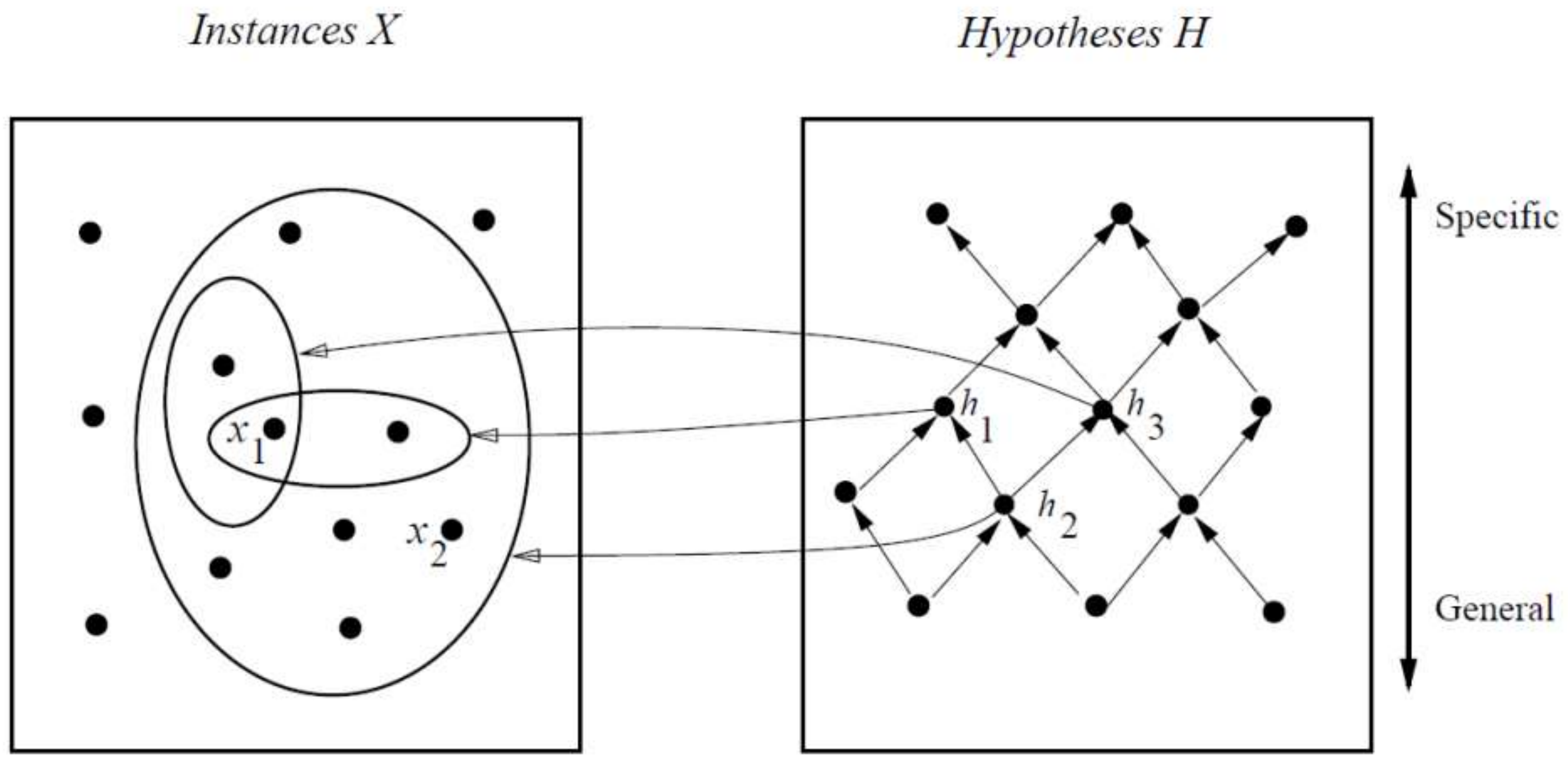
- We now **define the *more-general than or equal to*** a relation in terms of the sets of instances that satisfy the two hypotheses:
  - Given hypotheses $\mathbf{h_j}$ **and** $\mathbf{h_k}$, $\mathbf{h_j}$ is more-general-than or equal to $\mathbf{h_k}$ if and only if any instance that satisfies $\mathbf{h_k}$ also satisfies $\mathbf{h_j}$.
    - *More general than or equal to :* $\mathbf{h_j >= h_k}$
    - *More general than :* $\mathbf{h_j > h_k}$
  - *Sometimes find the inverse useful and will say that* $\mathbf{h_j}$ *is more specific than* $\mathbf{h_k}$ *when* $\mathbf{h_k}$ *is more_general-than* $\mathbf{h_j}$.



Definition: Let h_j and h_k be boolean-valued functions defined over X. Then h_j is **more_general_than_or_equal_to** h_k (written $h_j \geq_g h_k$) if and only if

$$(\forall x \in X)[(h_k(x) = 1) \rightarrow (h_j(x) = 1)]$$

MORE-GERNERAL-RELATION



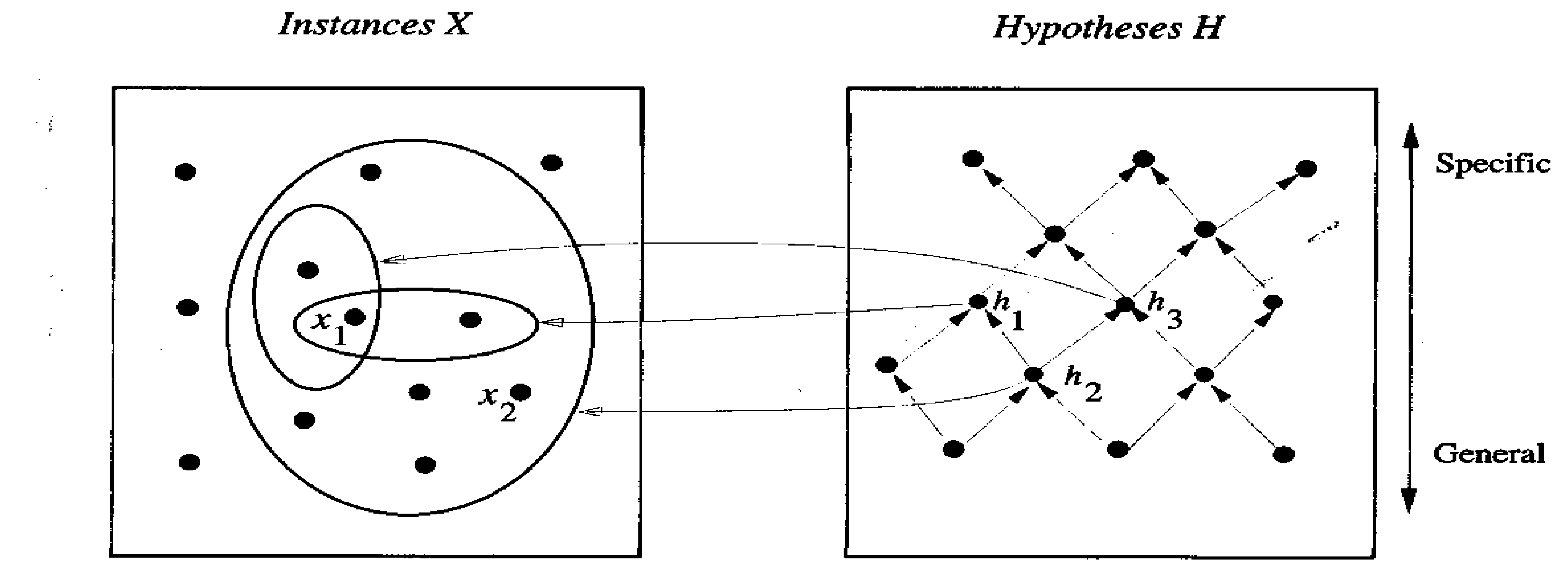
x_1= <Sunny, Warm, High, Strong, Cool, Same>
x_2= <Sunny, Warm, High, Light, Warm, Same>

h_1= <Sunny, ?, ?, Strong, ?, ?>
h_2= <Sunny, ?, ?, ?, ?, ?>
h_3= <Sunny, ?, ?, ?, Cool, ?>

- In the figure, the box on the left represents the set X of all instances, the box on the right the set H of all hypotheses.
- Each hypothesis corresponds to some subset of X-the subset of instances that it classifies positive.
- The arrows connecting hypotheses represent the ***more - general -than*** relation, with the arrow pointing toward the less general hypothesis.
- Note the subset of instances characterized by h_2 subsumes the subset characterized by h_1, hence h_2 is more - general– than h_1

MORE-GERNERAL-RELATION

- h2>h1   and   h2>h3
- But there is no more-general relation between h1 and h3

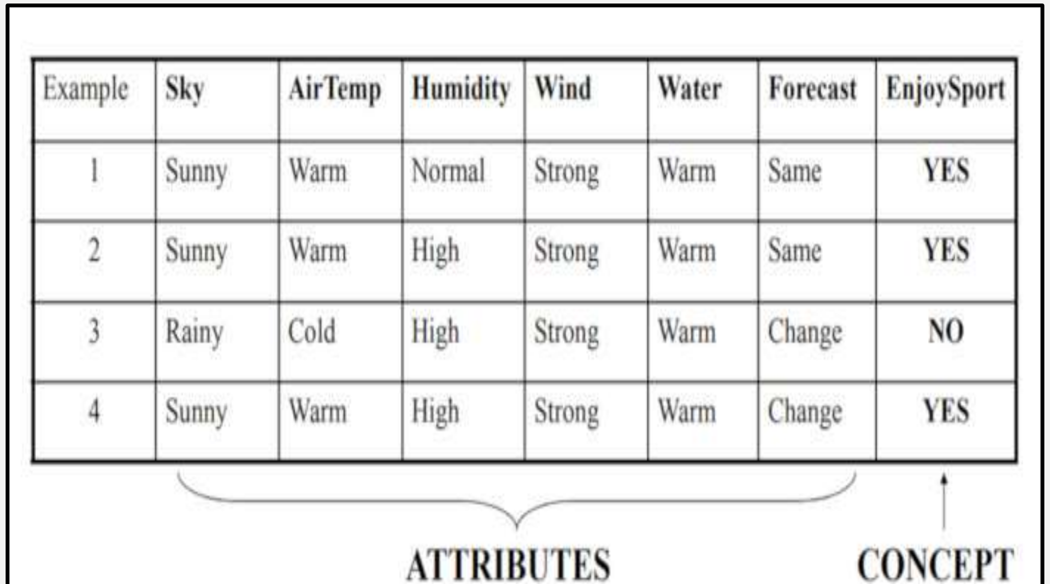


x_1= <Sunny, Warm, High, Strong, Cool, Same>
x_2= <Sunny, Warm, High, Light, Warm, Same>

h_1= <Sunny, ?, ?, Strong, ?, ?>
h_2= <Sunny, ?, ?, ?, ?, ?>
h_3= <Sunny, ?, ?, ?, Cool, ?>

FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS

- **FIND-S** Algorithm is used to find the **Maximally Specific Hypothesis.** Using the **Find-S** algorithm gives a single maximally specific hypothesis for the given set of training examples.
- **FIND-S** is guaranteed to output the most specific hypothesis within H that is consistent with **the positive training examples**.
- One way is to begin with the most specific possible hypothesis in H, then generalize this hypothesis each time it fails to cover an observed positive training example.
- To illustrate this algorithm, assume the learner is given the sequence of training examples from Table for the EnjoySport task.



Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	YES
2	Sunny	Warm	High	Strong	Warm	Same	YES
3	Rainy	Cold	High	Strong	Warm	Change	NO
4	Sunny	Warm	High	Strong	Warm	Change	YES

ATTRIBUTES
CONCEPT

STEPS INVOLVED IN FIND-S :

1. Start with the most specific hypothesis.

   h = <ϕ, ϕ, ϕ, ϕ, ϕ, ϕ>
2. Take the next example and if it is negative, then no changes occur to the hypothesis.
3. If the example is positive and we find that our initial hypothesis is too specific then we update our current hypothesis to general condition.
4. Keep repeating the above steps till all the training examples are complete.
5. After we have completed all the training examples we will have the final hypothesis when can used to classify the new examples.

IMPORTANT REPRESENTATION :

- **?** indicates that any value is acceptable for the attribute.
- specify a single required value (e.g., Cold) for the attribute.
- **Φ** indicates that no value is acceptable.
- The most **general hypothesis** is represented by: **<?, ?, ?, ?, ?, ?>**
- The most **specific hypothesis** is represented by : **<ϕ, ϕ, ϕ, ϕ, ϕ, ϕ>**

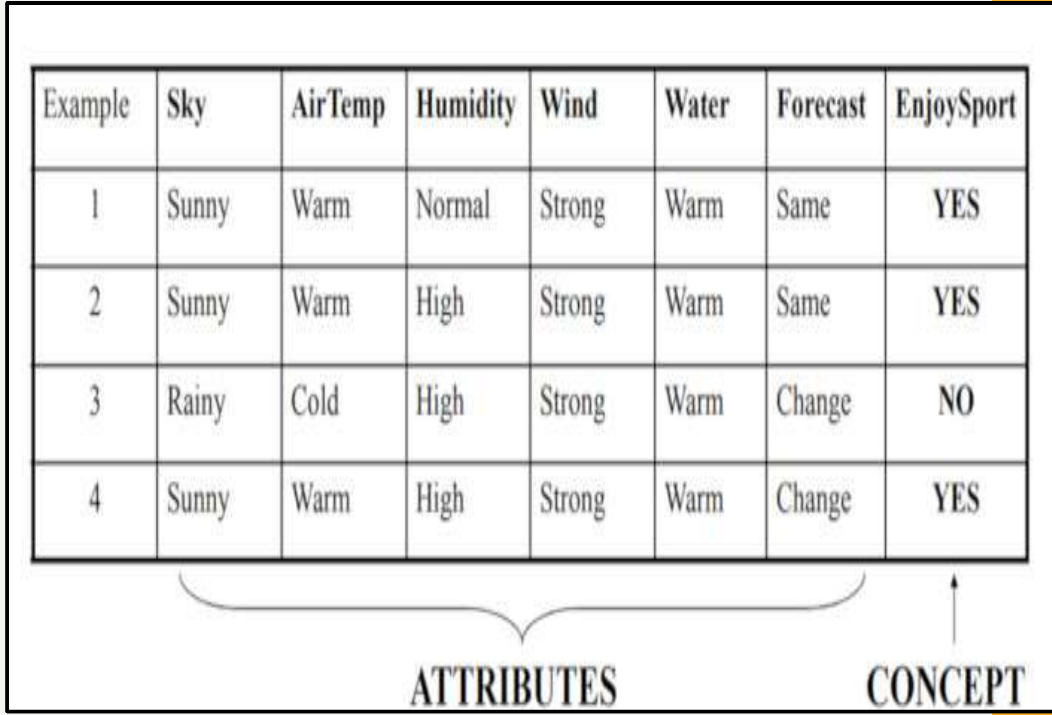
FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS

- The first step of FINDS is to initialize *h* to the most specific hypothesis in H.

  h0←(Ø, Ø, Ø, Ø, Ø, Ø)

- By observing the first training example which is positive, but it doesnot satisfy the hypothesis, so each Ø is replaced by the next more general constraint that fits the example.

  h1 → (Sunny, Warm, Normal, Strong, Warm, Same)



Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	YES
2	Sunny	Warm	High	Strong	Warm	Same	YES
3	Rainy	Cold	High	Strong	Warm	Change	NO
4	Sunny	Warm	High	Strong	Warm	Change	YES

ATTRIBUTES
CONCEPT

- Consider the next training example (also positive in this case) forces the algorithm to further generalize h, by substituting a "?' in place of any attribute value in h that is not satisfied by the new example.

  h2 → (Sunny, Warm, ?, Strong, Warm, Same)

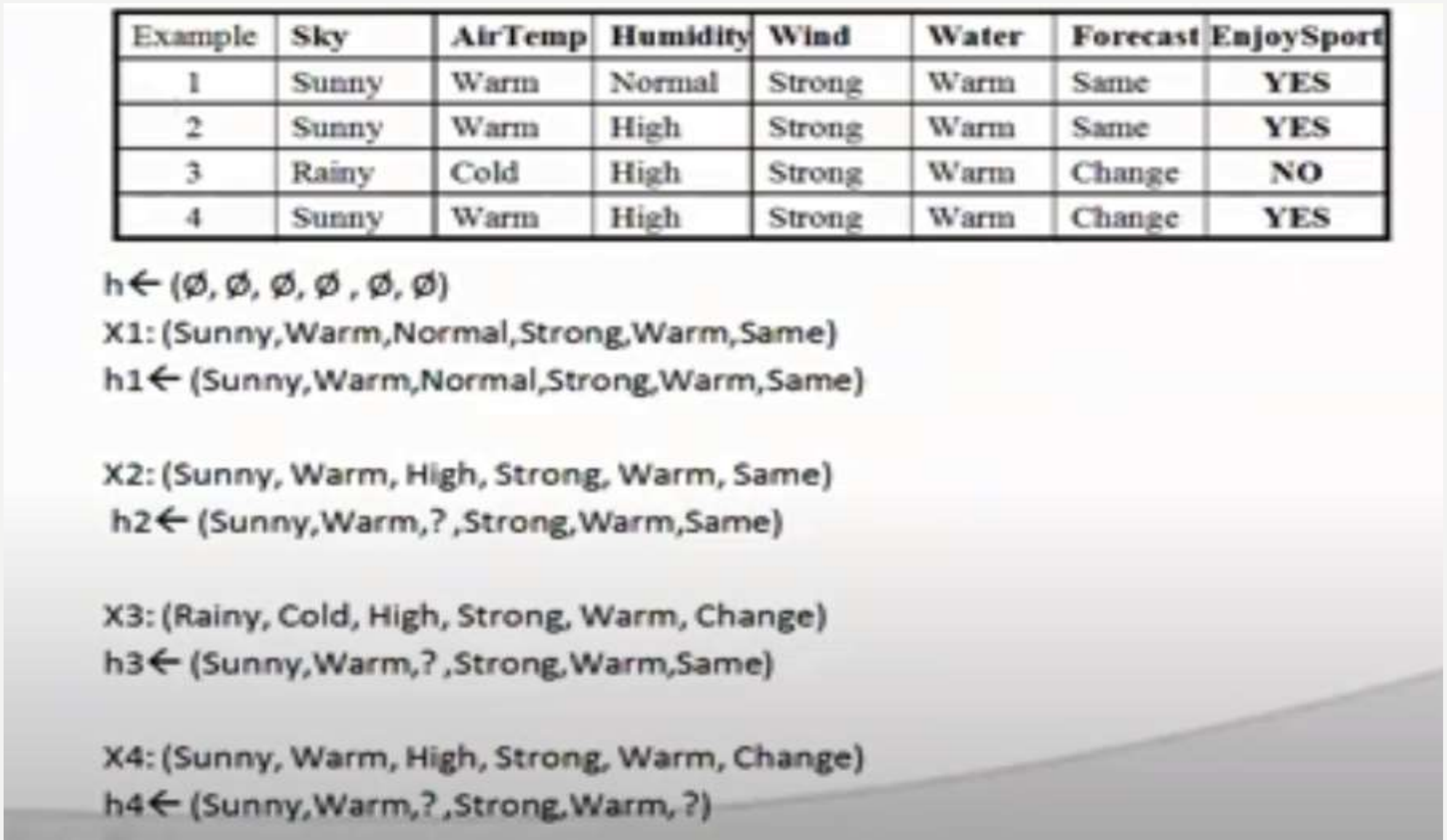
- Next is negative example, so need to consider.

  h3 → (Sunny, Warm, ?, Strong, Warm, Same)

- Next, the fourth example leads to further generalization of **h**

  h4 → (Sunny, Warm, ?, Strong, ?, ?)

EXAMPLE -1



Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	YES
2	Sunny	Warm	High	Strong	Warm	Same	YES
3	Rainy	Cold	High	Strong	Warm	Change	NO
4	Sunny	Warm	High	Strong	Warm	Change	YES

h← (Ø, Ø, Ø, Ø , Ø, Ø)

X1: (Sunny,Warm,Normal,Strong,Warm,Same)

h1← (Sunny,Warm,Normal,Strong,Warm,Same)

X2: (Sunny, Warm, High, Strong, Warm, Same)

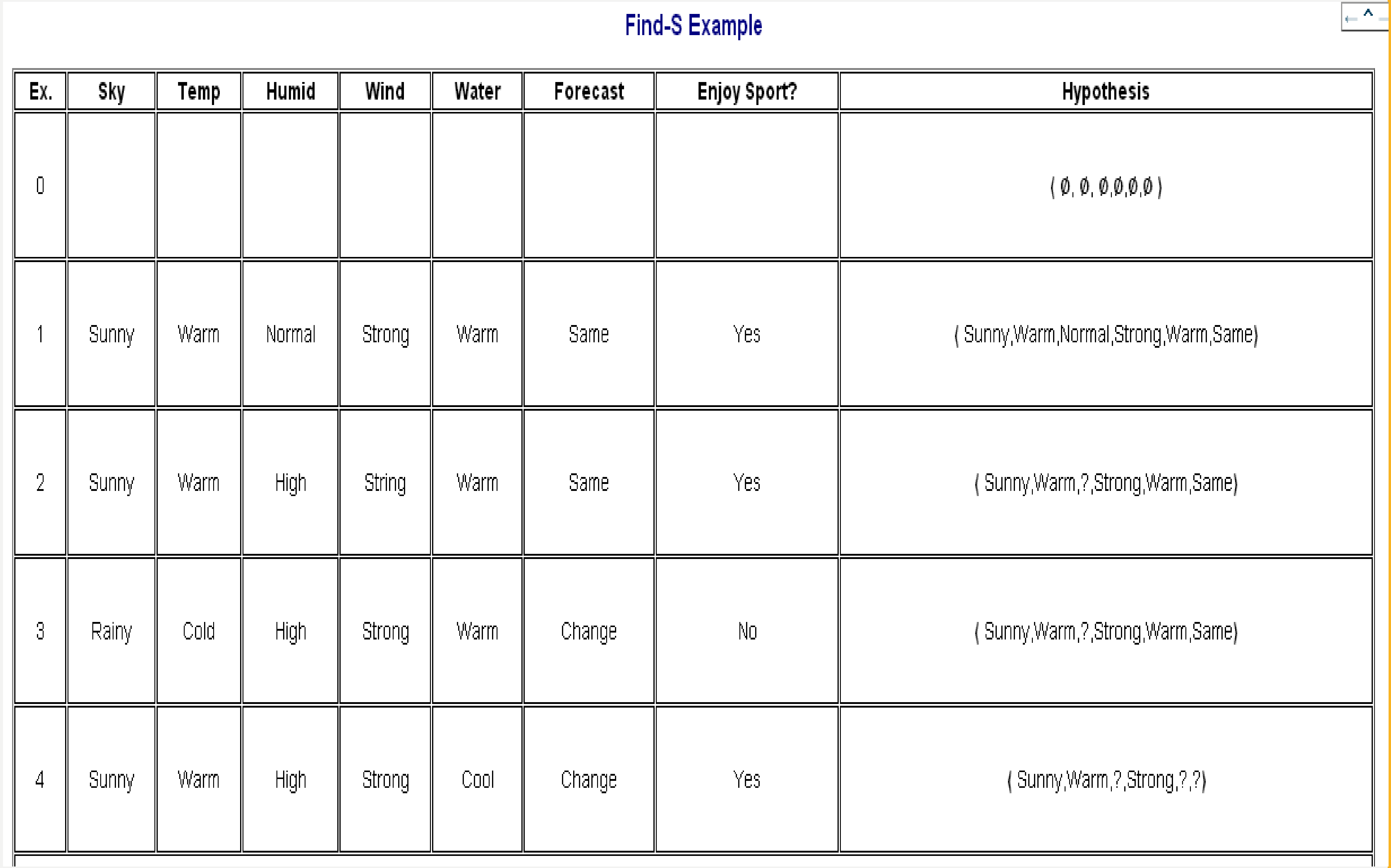
h2← (Sunny,Warm,? ,Strong,Warm,Same)

X3: (Rainy, Cold, High, Strong, Warm, Change)

h3← (Sunny,Warm,? ,Strong,Warm,Same)

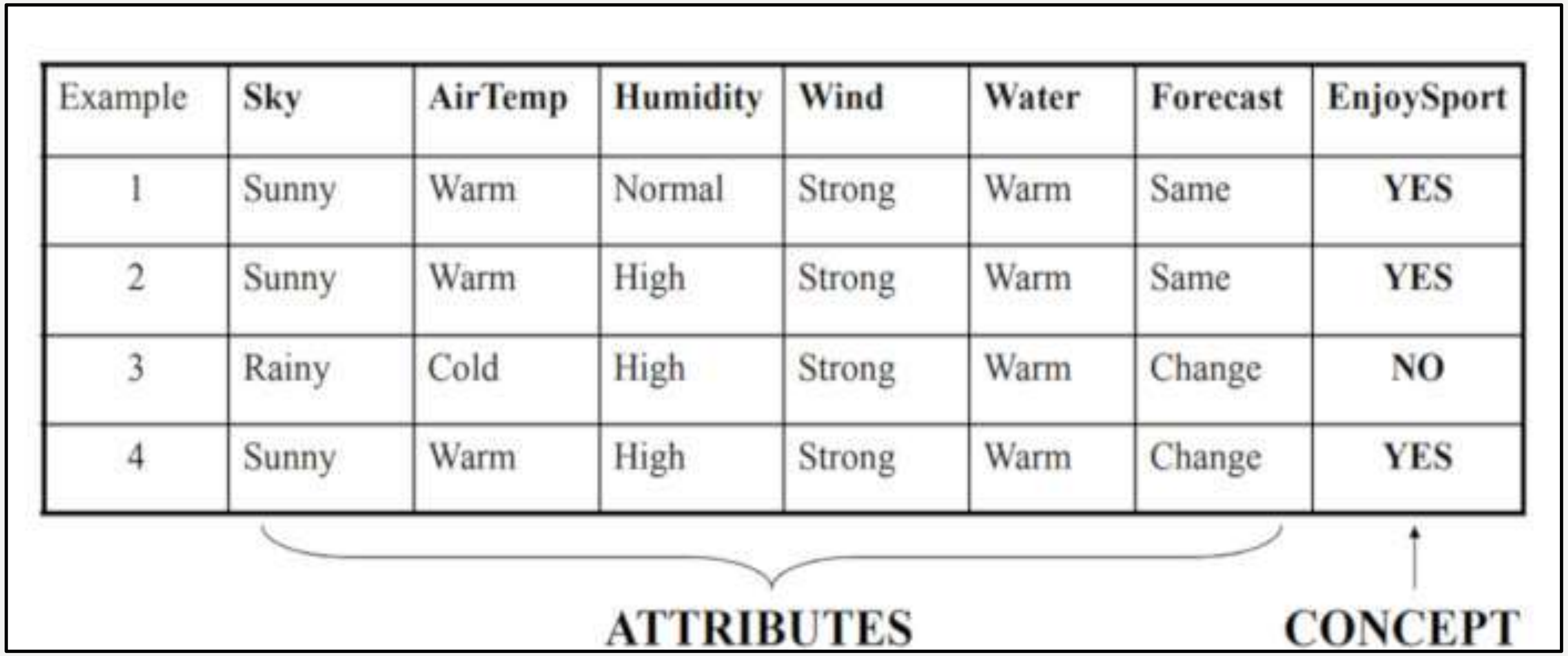
X4: (Sunny, Warm, High, Strong, Warm, Change)

h4← (Sunny,Warm,? ,Strong,Warm, ?)



Find-S Example

Ex.	Sky	Temp	Humid	Wind	Water	Forecast	Enjoy Sport?	Hypothesis
0								(Ø, Ø, Ø,Ø,Ø,Ø)
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes	(Sunny,Warm,Normal,Strong,Warm,Same)
2	Sunny	Warm	High	String	Warm	Same	Yes	(Sunny,Warm,?,Strong,Warm,Same)
3	Rainy	Cold	High	Strong	Warm	Change	No	(Sunny,Warm,?,Strong,Warm,Same)
4	Sunny	Warm	High	Strong	Cool	Change	Yes	(Sunny,Warm,?,Strong,?,?)



Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	YES
2	Sunny	Warm	High	Strong	Warm	Same	YES
3	Rainy	Cold	High	Strong	Warm	Change	NO
4	Sunny	Warm	High	Strong	Warm	Change	YES

ATTRIBUTES
CONCEPT

1. **How many concepts are possible for this instance space?**
2. **How many hypotheses can be expressed by the hypothesis language?**

ENJOY SPORT – HYPOTHESIS SPACE

1.Solution:

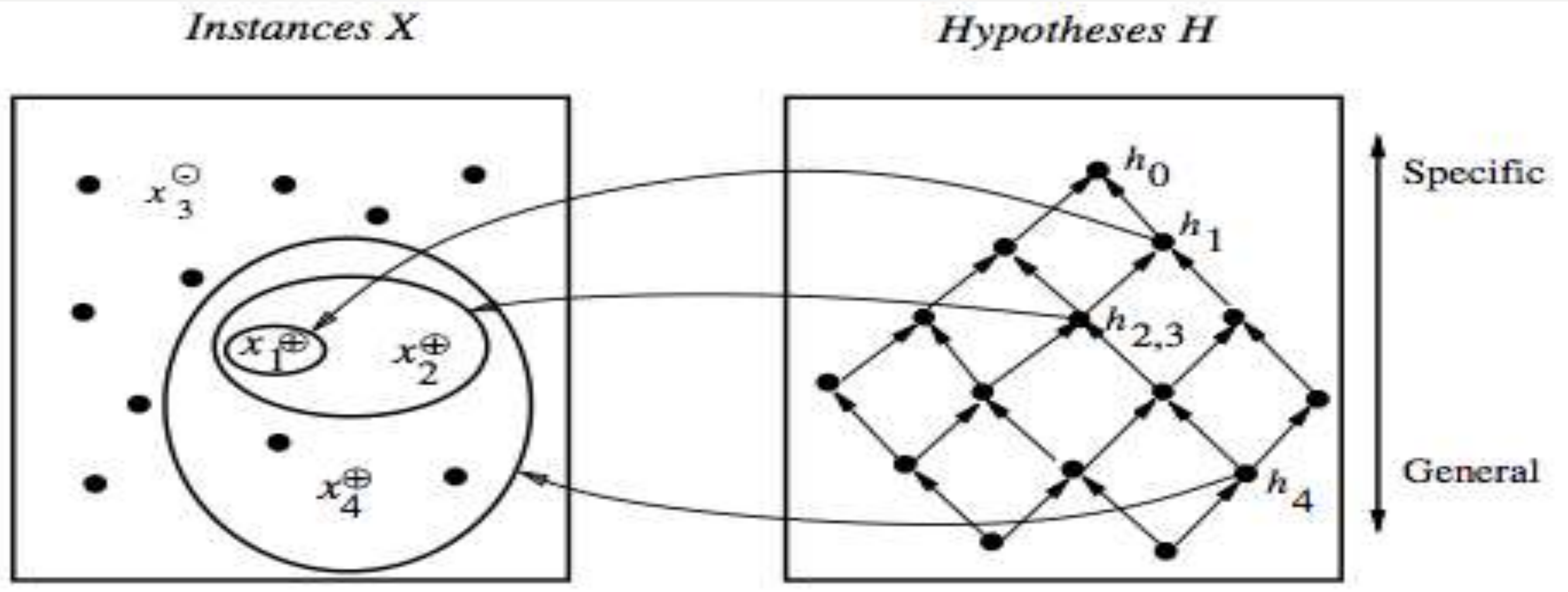
Given that the attribute **Sky** has 3 possible values, and other 5 attributes have 2 possible values, thus, there are **3.2.2.2.2.2= 96** ***distinct instances.***

2.Solution:

There are **5.4.4.4.4.4=5120** ***syntactically distinct hypotheses*** within **H** (including '?' and 'Ø').

There are **(1+4.3.3.3.3.3)=973** ***semantically distinct hypotheses*** within **H** (including '?' for each attribute and one 'Ø' common to all).

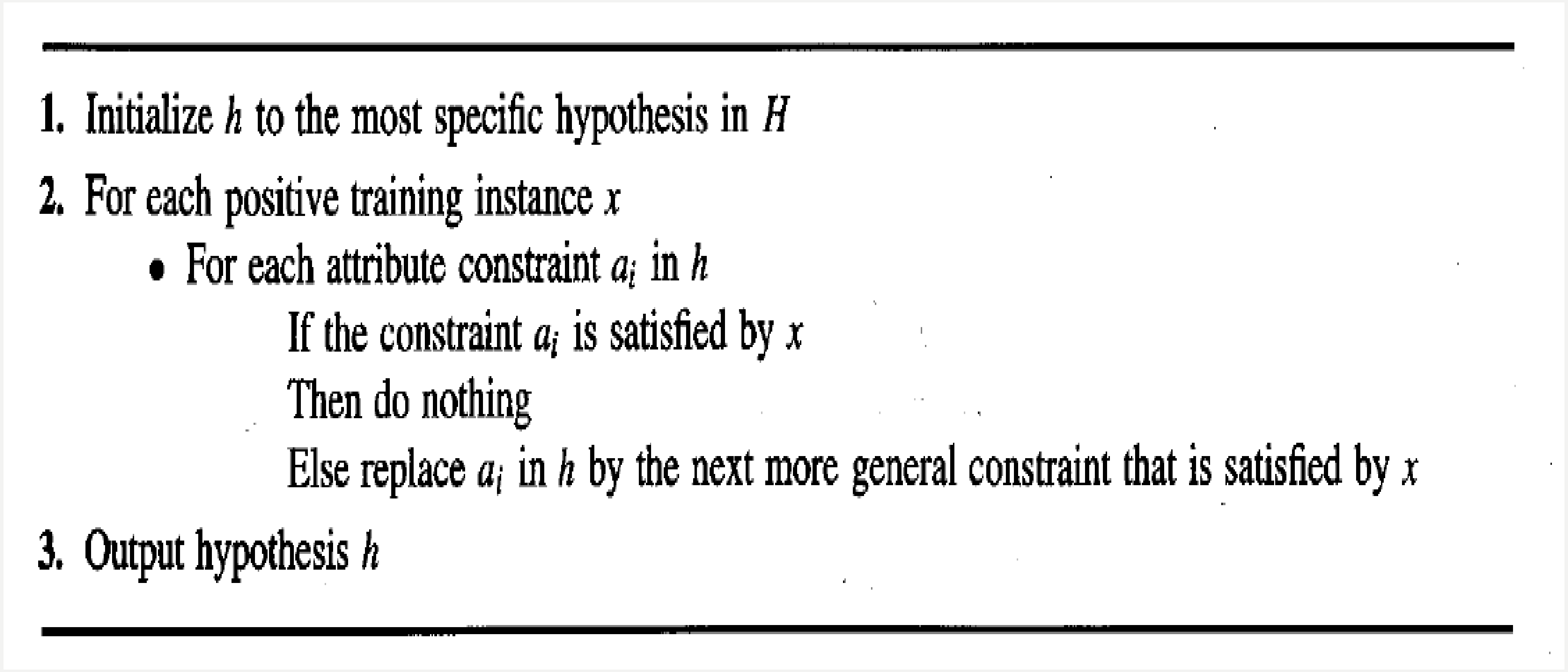
FIND-S



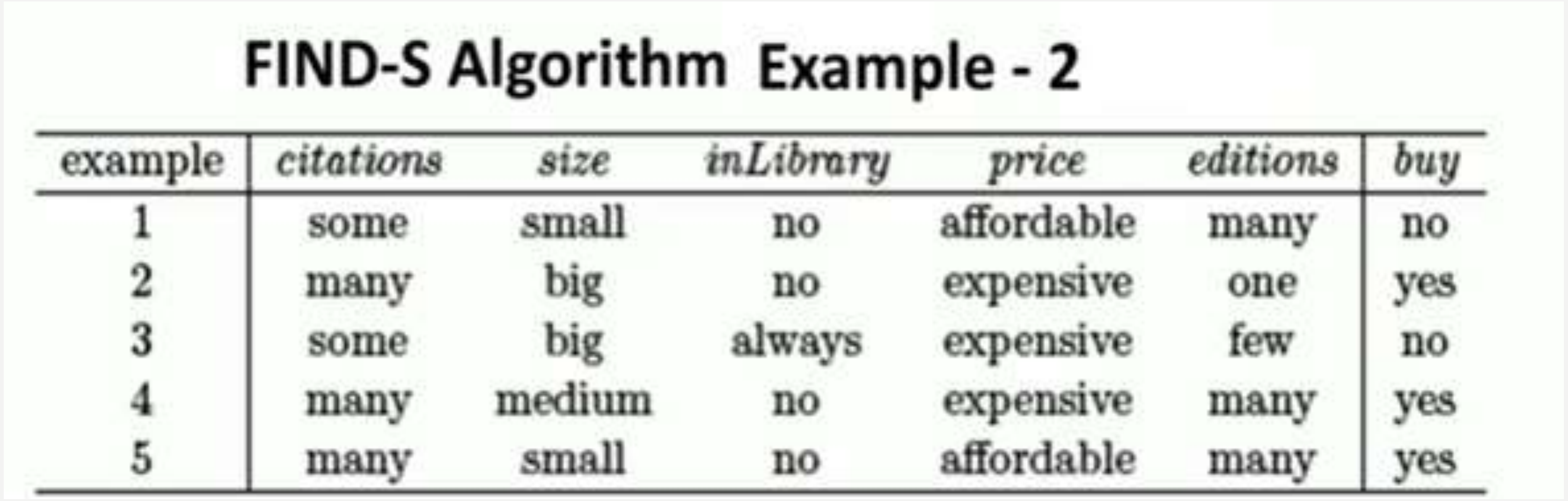
$h_0 = <\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing>$

x_1 = <*Sunny Warm Normal Strong Warm Same*>, +	h_1 = <*Sunny Warm Normal Strong Warm San*
x_2 = <*Sunny Warm High Strong Warm Same*>, +	h_2 = <*Sunny Warm ? Strong Warm Same*>
x_3 = <*Rainy Cold High Strong Warm Change*>, -	h_3 = <*Sunny Warm ? Strong Warm Same*>
x_4 = <*Sunny Warm High Strong Cool Change*>, +	h_4 = <*Sunny Warm ? Strong ? ?* >

FIND-S ALGORITHM



```
1. Initialize $h$ to the most specific hypothesis in $H$
2. For each positive training instance $x$
       • For each attribute constraint $a_i$ in $h$
             If the constraint $a_i$ is satisfied by $x$
             Then do nothing
             Else replace $a_i$ in $h$ by the next more general constraint that is satisfied by $x$
3. Output hypothesis $h$
```



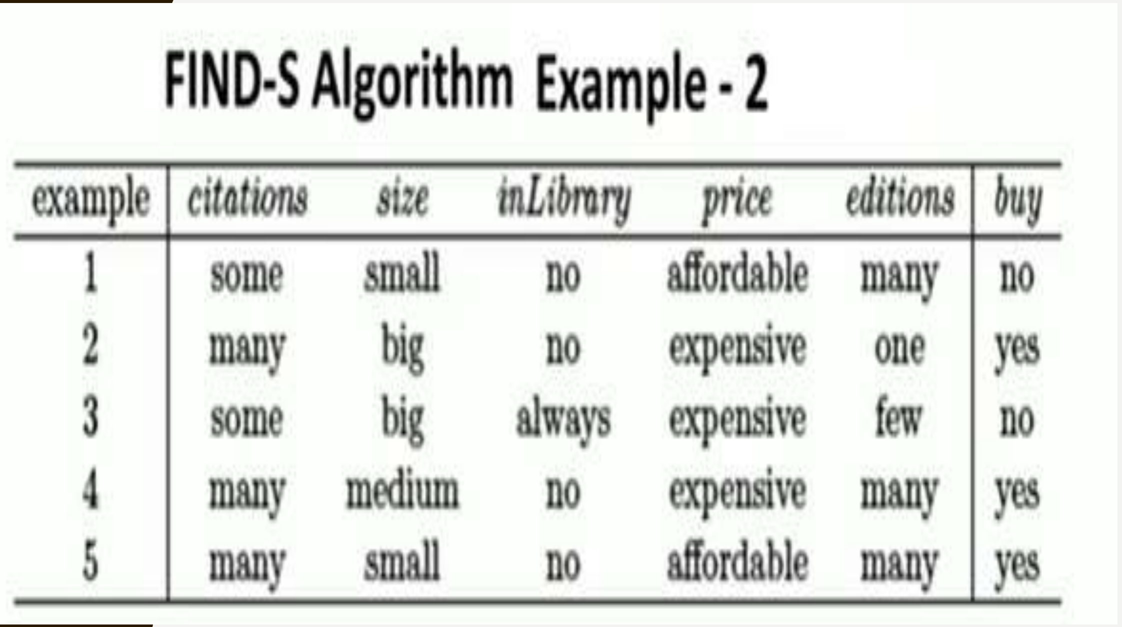
FIND-S Algorithm Example - 2

example	*citations*	*size*	*inLibrary*	*price*	*editions*	*buy*
1	some	small	no	affordable	many	no
2	many	big	no	expensive	one	yes
3	some	big	always	expensive	few	no
4	many	medium	no	expensive	many	yes
5	many	small	no	affordable	many	yes

1. How many concepts are possible for this instance space?

2. How many hypotheses can be expressed by the hypothesis language?

3. Apply the FIND-S algorithm by hand on the given training set. Consider the examples in the specified order and write down your hypothesis each time after observing an example.



FIND-S Algorithm Example - 2

example	citations	size	inLibrary	price	editions	buy
1	some	small	no	affordable	many	no
2	many	big	no	expensive	one	yes
3	some	big	always	expensive	few	no
4	many	medium	no	expensive	many	yes
5	many	small	no	affordable	many	yes

1. How many concepts are possible for this instance space?

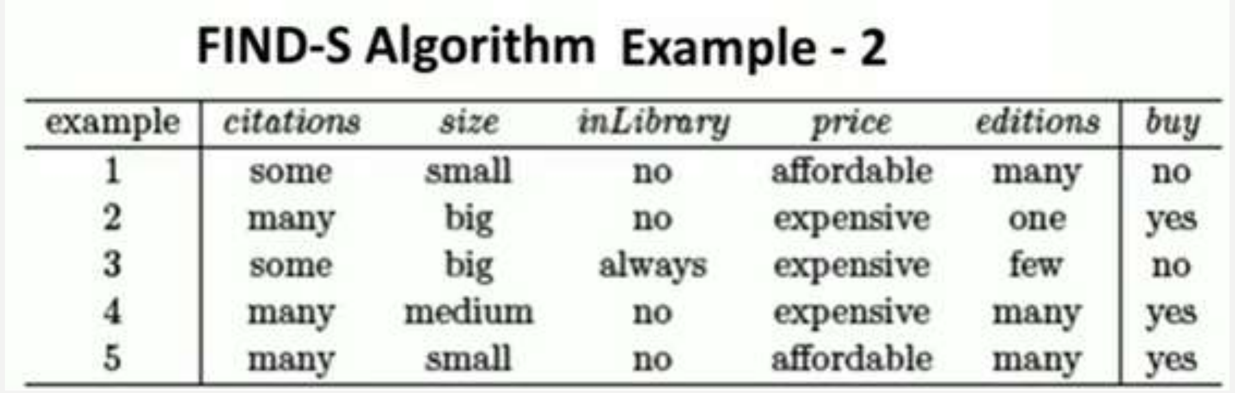
Solution:

Attibutes	Values
Citations	2
Size	3
inLibrary	2
Price	2
Editions	3

2*3*2*2*3= 72

2. How many hypotheses can be expressed by the hypothesis language?

Solution:
Syntatically Distinct Hypothesis=4*5*4*4*5=1600
Semantically Distinct Hypothesis = 1+(3*4*3*3*4) =433



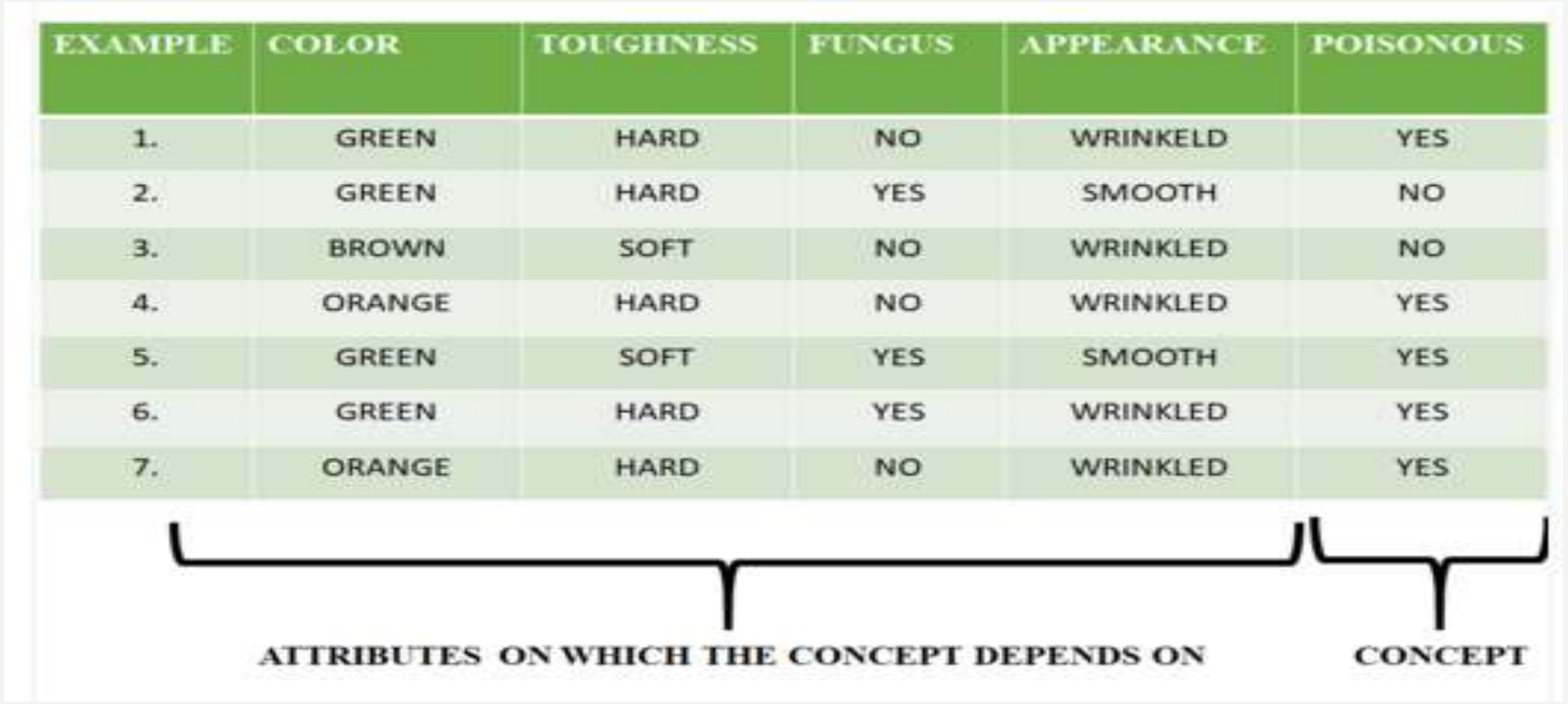
FIND-S Algorithm Example - 2

example	*citations*	*size*	*inLibrary*	*price*	*editions*	*buy*
1	some	small	no	affordable	many	no
2	many	big	no	expensive	one	yes
3	some	big	always	expensive	few	no
4	many	medium	no	expensive	many	yes
5	many	small	no	affordable	many	yes

3. Apply the FIND-S algorithm by hand on the given training set. Consider the examples in the specified order and write down your hypothesis each time after observing an example.

Eg.	citations	size	inlibrary	price	editions	buy	Hypothesis
0							h0=<Ø,Ø,Ø,Ø,Ø>
1	some	small	no	affordable	many	no	h1=<Ø,Ø,Ø,Ø,Ø>[-ve example]
2	many	big	no	expensive	one	yes	h2=<many,big,no,expensive,one >[+ve example]
3	some	big	always	expensive	few	no	h3=<many,big,no,expensive,one >[-ve example]
4	many	medium	no	expensive	many	yes	h4=<many,?,no, expensive,? >[+ve example]
5	many	small	no	affordable	many	yes	h5=<many,?,no,?,?>[+ve example]

EXAMPLE-3



EXAMPLE	COLOR	TOUGHNESS	FUNGUS	APPEARANCE	POISONOUS
1.	GREEN	HARD	NO	WRINKELD	YES
2.	GREEN	HARD	YES	SMOOTH	NO
3.	BROWN	SOFT	NO	WRINKLED	NO
4.	ORANGE	HARD	NO	WRINKLED	YES
5.	GREEN	SOFT	YES	SMOOTH	YES
6.	GREEN	HARD	YES	WRINKLED	YES
7.	ORANGE	HARD	NO	WRINKLED	YES

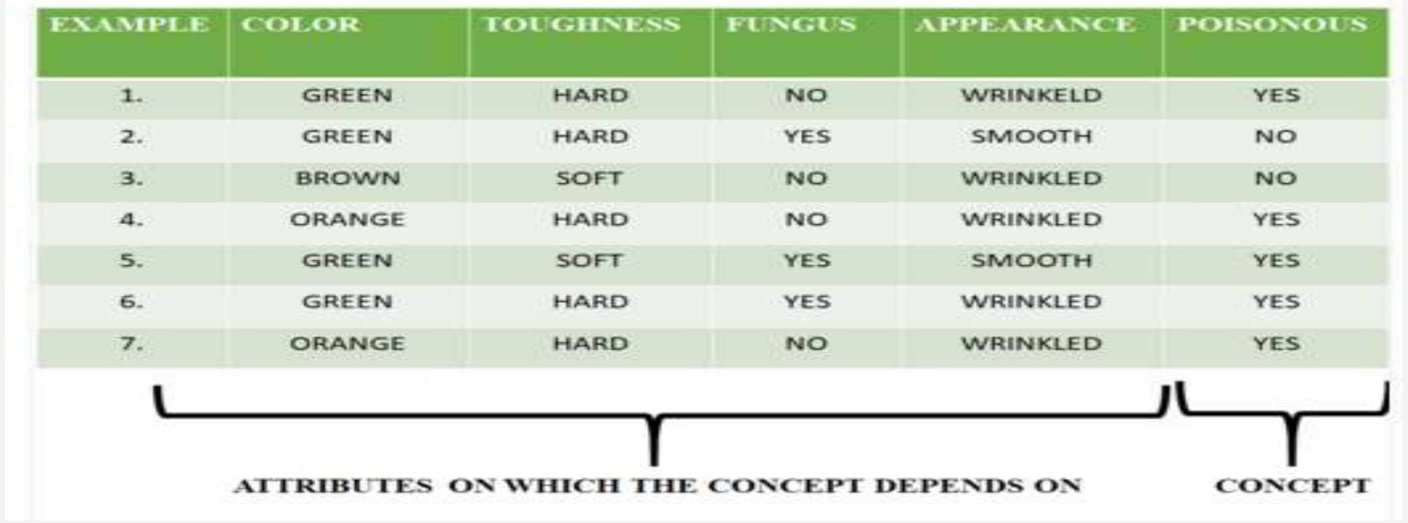
ATTRIBUTES ON WHICH THE CONCEPT DEPENDS ON

CONCEPT

1. How many concepts are possible for this instance space?

2. How many hypotheses can be expressed by the hypothesis language?

3. Apply the FIND-S algorithm by hand on the given training set. Consider the examples in the specified order and write down your hypothesis each time after observing an example.



EXAMPLE	COLOR	TOUGHINESS	FUNGUS	APPEARANCE	POISONOUS
1.	GREEN	HARD	NO	WRINKELD	YES
2.	GREEN	HARD	YES	SMOOTH	NO
3.	BROWN	SOFT	NO	WRINKLED	NO
4.	ORANGE	HARD	NO	WRINKLED	YES
5.	GREEN	SOFT	YES	SMOOTH	YES
6.	GREEN	HARD	YES	WRINKLED	YES
7.	ORANGE	HARD	NO	WRINKLED	YES

ATTRIBUTES ON WHICH THE CONCEPT DEPENDS ON

CONCEPT

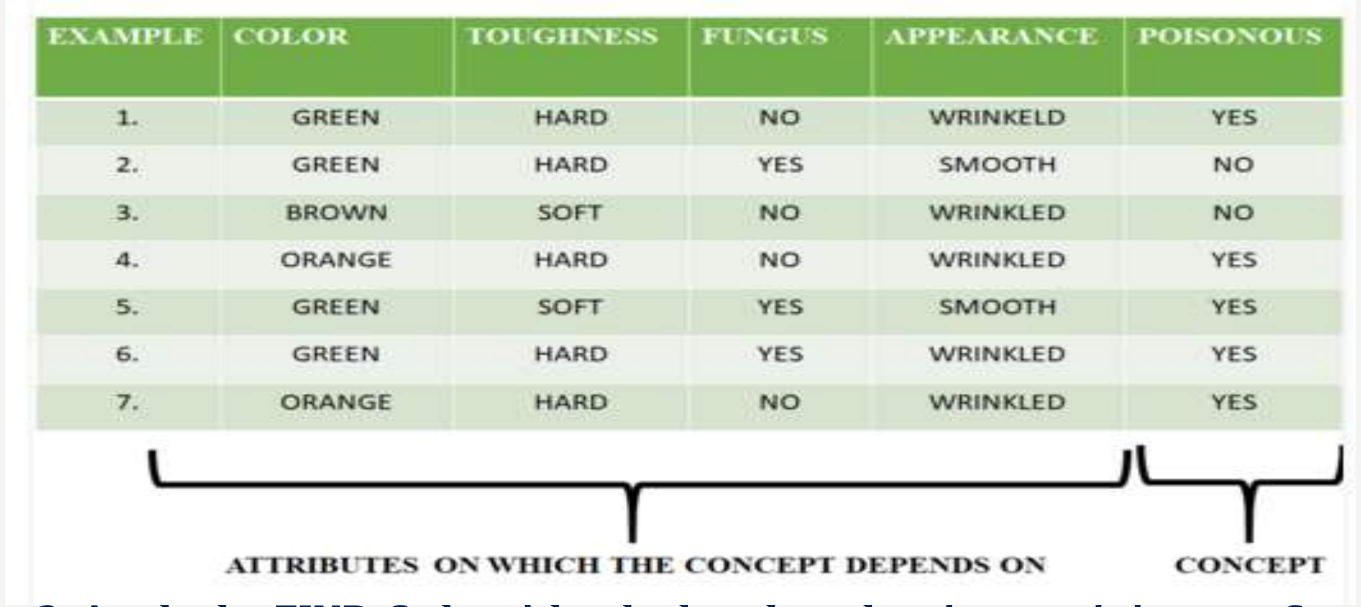
1.Solution:

3*2*2*2=24

2. Solution:

Syntactically Distinct Hypothesis=5*4*4*4=300

Semantically Distinct Hypothesis = 1+(4*3*3*3) =109



EXAMPLE	COLOR	TOUGHNESS	FUNGUS	APPEARANCE	POISONOUS
1.	GREEN	HARD	NO	WRINKELD	YES
2.	GREEN	HARD	YES	SMOOTH	NO
3.	BROWN	SOFT	NO	WRINKLED	NO
4.	ORANGE	HARD	NO	WRINKLED	YES
5.	GREEN	SOFT	YES	SMOOTH	YES
6.	GREEN	HARD	YES	WRINKLED	YES
7.	ORANGE	HARD	NO	WRINKLED	YES

ATTRIBUTES ON WHICH THE CONCEPT DEPENDS ON

CONCEPT

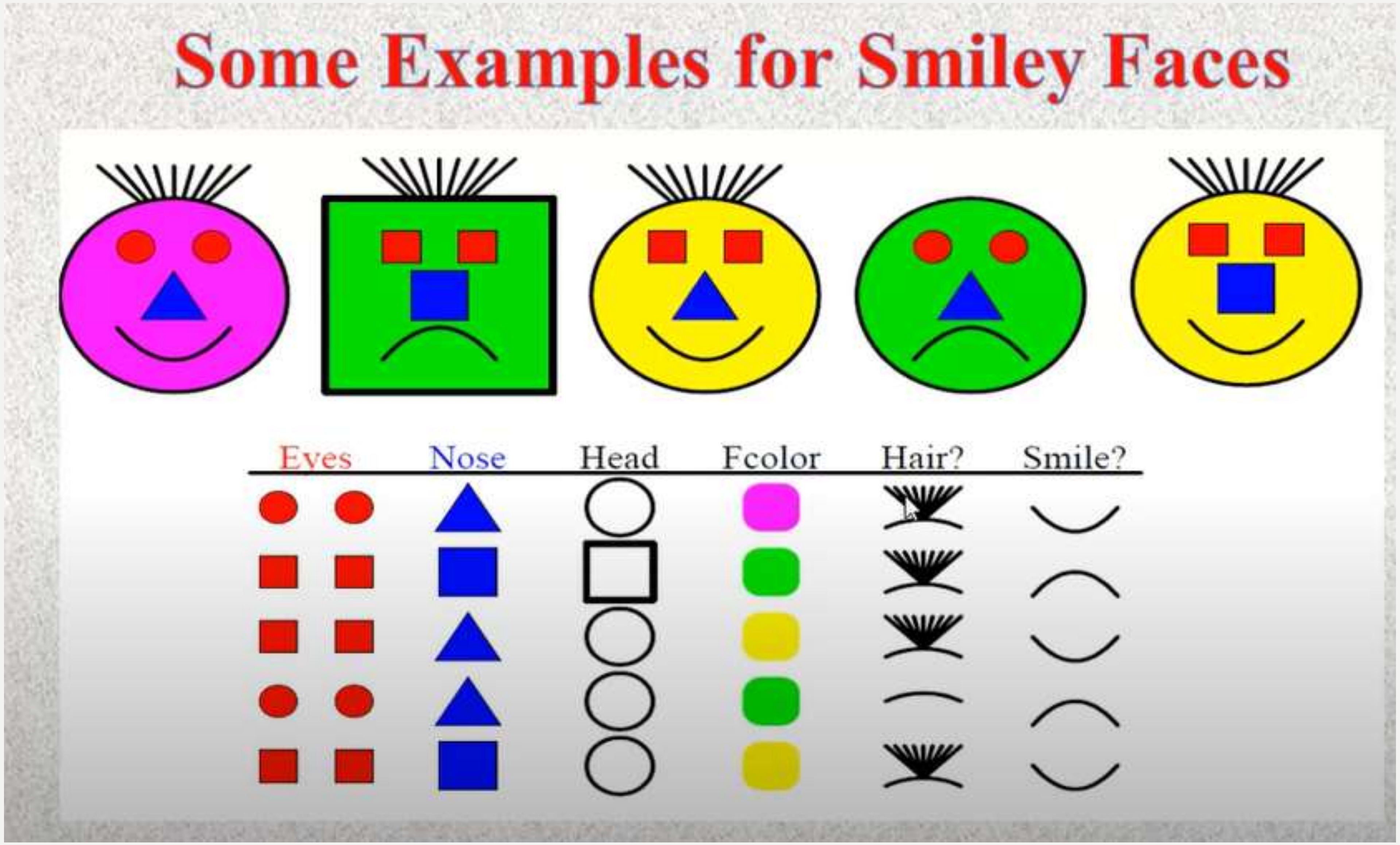
3. Apply the FIND-S algorithm by hand on the given training set. Consider the examples in the specified order and write down your hypothesis each time after observing an example.

Eg.	COLOR	TOUGHNESS	FUNGUS	APPEARANCE	POISONOUS	Hypothesis
0						h0=<Ø,Ø,Ø,Ø,Ø>
1	GREEN	HARD	NO	WRINKELD	YES	h1=<GREEN,HARD,NO,WRINKELD>[+ve example]
2	GREEN	HARD	YES	SMOOTH	NO	h2=<GREEN,HARD,NO,WRINKELD >[-ve example]
3	BROWN	SOFT	NO	WRINKELD	NO	h3=<GREEN,HARD,NO,WRINKELD >[-ve example]
4	ORANGE	HARD	NO	WRINKELD	YES	h4=<?,HARD,NO, WRINKELD>[+ve example]
5	GREEN	SOFT	YES	SMOOTH	YES	h5=<?,?,?,?>[+ve example]
6	GREEN	HARD	YES	WRINKELD	YES	h6=<?,?,?,?>[+ve example]
7	ORANGE	HARD	NO	WRINKELD	YES	h7=<?,?,?,?>[+ve example]

EXAMPLE-4

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive
Japan	Toyota	Green	1980	Economy	Positive
Japan	Honda	Red	1990	Economy	Negative

h1,h2 = {Japan, Honda, Blue, 1980, Economy}
h3,h4 = {Japan, ?, Blue, ?, Economy}
h5 = {Japan, ?, ?, ?, Economy}
h6,h7 = {Japan, ?, ?, ?, Economy}



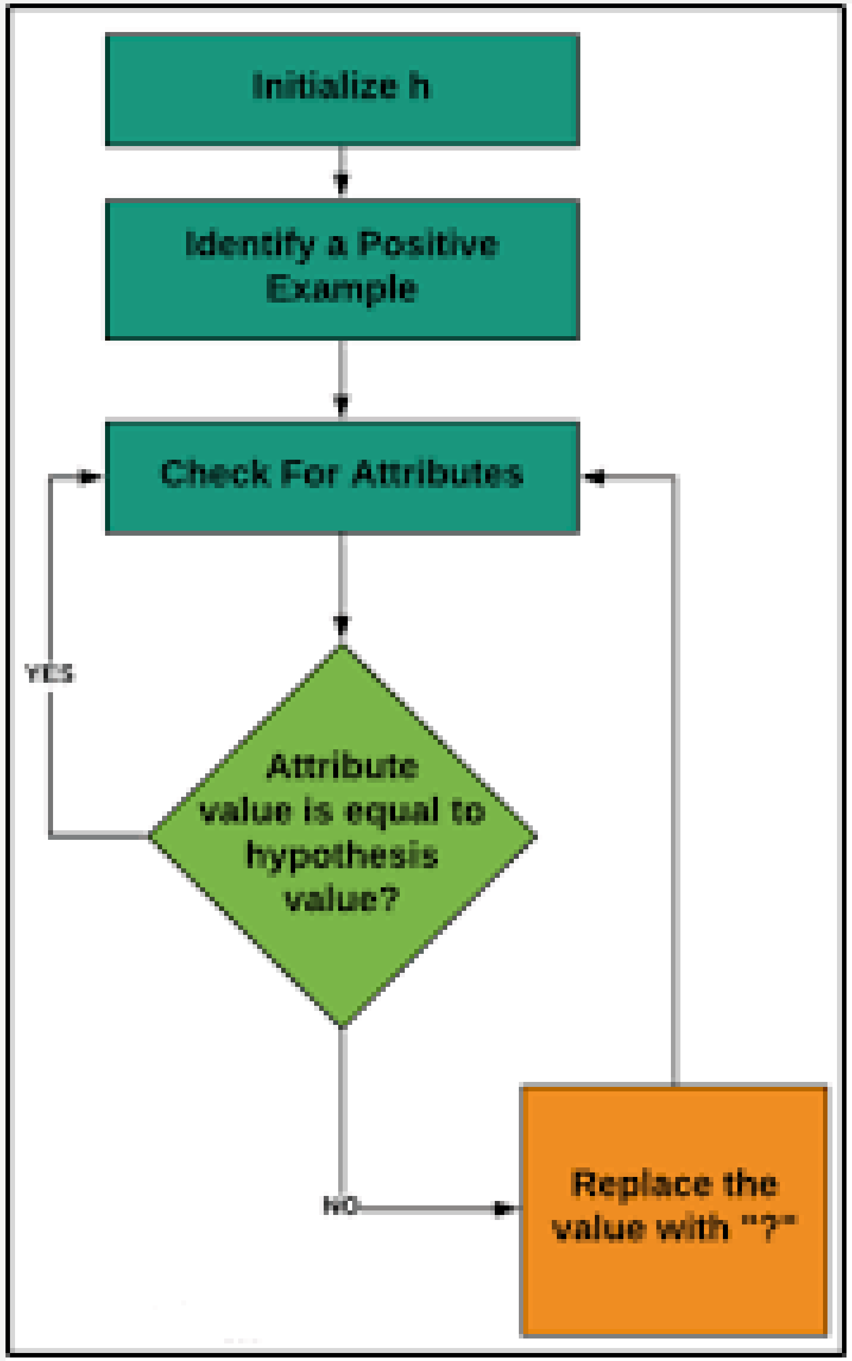
Some Examples for Smiley Faces

Eyes	Nose	Head	Fcolor	Hair?	Smile?
Round	Triangle	Round	Purple	Yes	Yes
Square	Square	Square	Green	Yes	No
Square	Triangle	Round	Yellow	Yes	Yes
Round	Triangle	Round	Green	No	No
Square	Square	Round	Yellow	Yes	Yes

1. h0=<Ø,Ø,Ø,Ø,Ø>
2. x1→+ h1=<round, triangle, round, purple, yes>
3. x2=→- h2=<round, triangle, round, purple, yes>
4. x3→+ h3=<?, triangle, round, ?, yes>
5. x4→- h4 =<?, triangle, round, ?, yes>
6. **x5→+ h5 =<?, ?, round, ?, yes>**

PROPERTIES OF FIND-S

- ***Find-S*** is guaranteed to output the *most specific hypothesis* within *H* that is consistent with the *positive* training examples
- ***Find-S*** algorithm's final hypothesis will also be consistent with the negative examples provided the correct target concept is contained in H, and provided the training examples are correct.
- Problems:
  - There can be more than one "most specific hypotheses"
  - We cannot say if the learner converged to the correct target
  - Why choose the most specific?
  - If the training examples are inconsistent, the algorithm can be mislead: no tolerance to rumor.
  - Negative examples are not considered


Initialize h
Identify a Positive Example
Check For Attributes
YES
Attribute value is equal to hypothesis value?
NO
Replace the value with "?"

MACHINE LEARNING LAB- PROGRAM-1

- Implement and demonstrate the **FIND-S ALGORITHM** for finding the most specific hypothesis based on a given set of training data samples.
- Read the training data from a **.CSV** file.
- Using python
- Use Jupiter notebook or google colab

CSV-WHAT,WHY?

- **CSV** stands for ***C****omma* ***S****eparated* ***V****alues.*
- It is a plain text file that contain list of data
- These database fields have been exported into a format that contains ***a single line where a comma separates each database record.***
- Files with the **. csv** extension are similar to plain text files.
- Since they're plain text, they're easier to import into a spreadsheet or another storage database, regardless of the specific software you're using. To better organize large amounts of data.

READING CSV FILES IN PYTHON

- We are going to exclusively use the **csv** module built into Python for this task. For this, we will have to import the module as : **import csv**
- Read CSV files with **csv.reader( )**
  - We can read the contents of the file with the following program:

```
with open('E://weather.csv', 'r') as f:
            reader = csv.reader(f)
            for row in reader:
                print(row)
```

- The csv.reader() is used to read the file, which returns an iterable reader object.
- The reader object is then iterated using a for loop to print the contents of each row.

1. Implement and demonstrate the **FIND-S ALGORITHM** for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

```
#read training data from csv file
import csv
with open('E://weather.csv', 'r') as f:
    reader = csv.reader(f)
    data = list(reader) #convert data into list of rows
```

1. Implement and demonstrate the **FIND-S ALGORITHM** for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

```
#Training data from CSV file
print("Training data")
for row in data:
    print(row)


attribute_length=len(data[0])-1       # finding no. of attributes
h = ['0']*attribute_length       # Initialize h to the most specific hypothesis in H
                                 #h=<0,0,0,0,0,0>
```

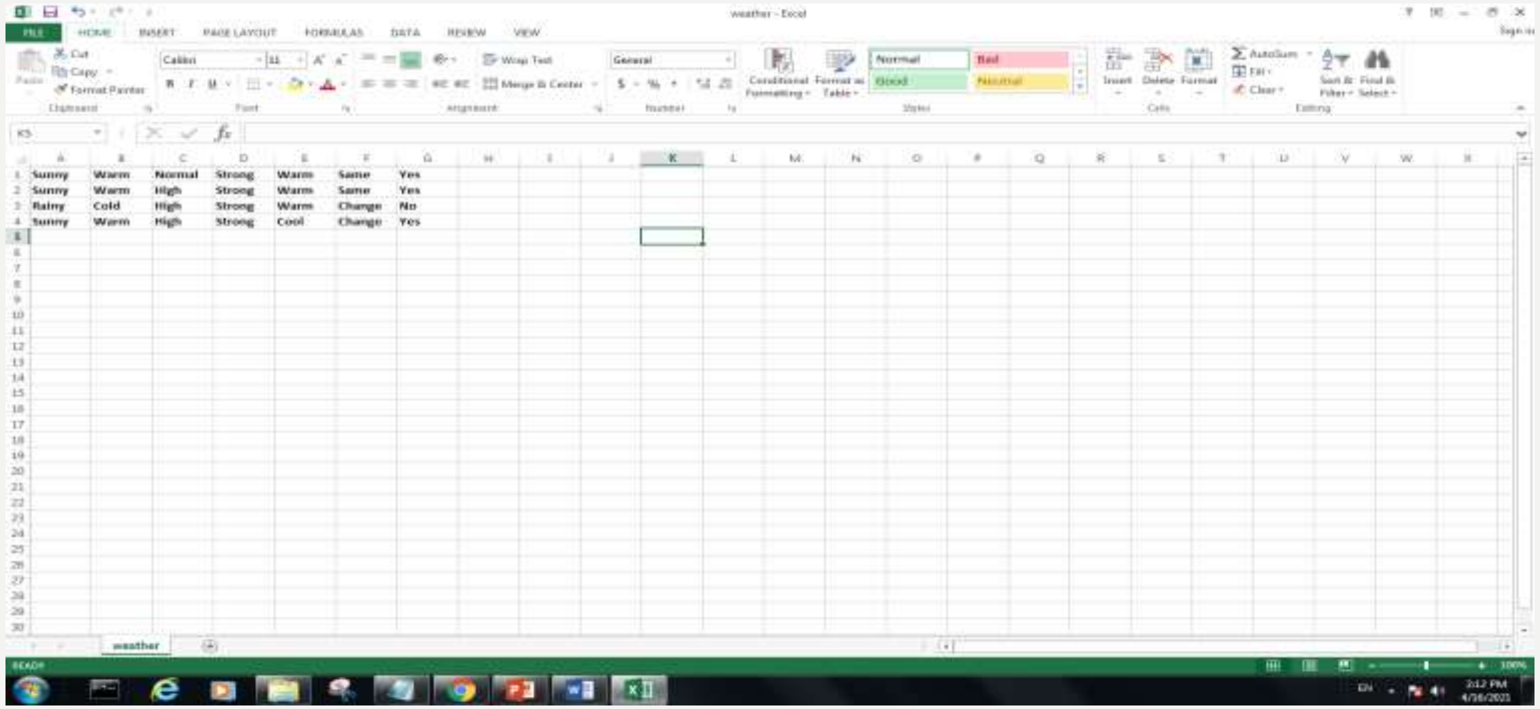
1. Implement and demonstrate the **FIND-S ALGORITHM** for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

```
print("The Hypothesis are")
for row in data:
  if row[-1] == 'Yes': #For each positive training instance x
    j = 0
    for col in row: #For each attribute constraint a, in h
      if col != 'Yes': #replace a, in h by the next more general constraint that is satisfied by x
        if col != h[j] and h[j] == '0':
          h[j] = col
        elif col != h[j] and h[j] != '0':
          h[j] = '?'
      j = j + 1
  print(h)          #print all Hypothesis
print('Maximally Specific Hypothesis: ', h)          #print final hypothesis
```

Input- WEATHER.CSV

Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes



A	B	C	D	E	F	G
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Ouput

Training data

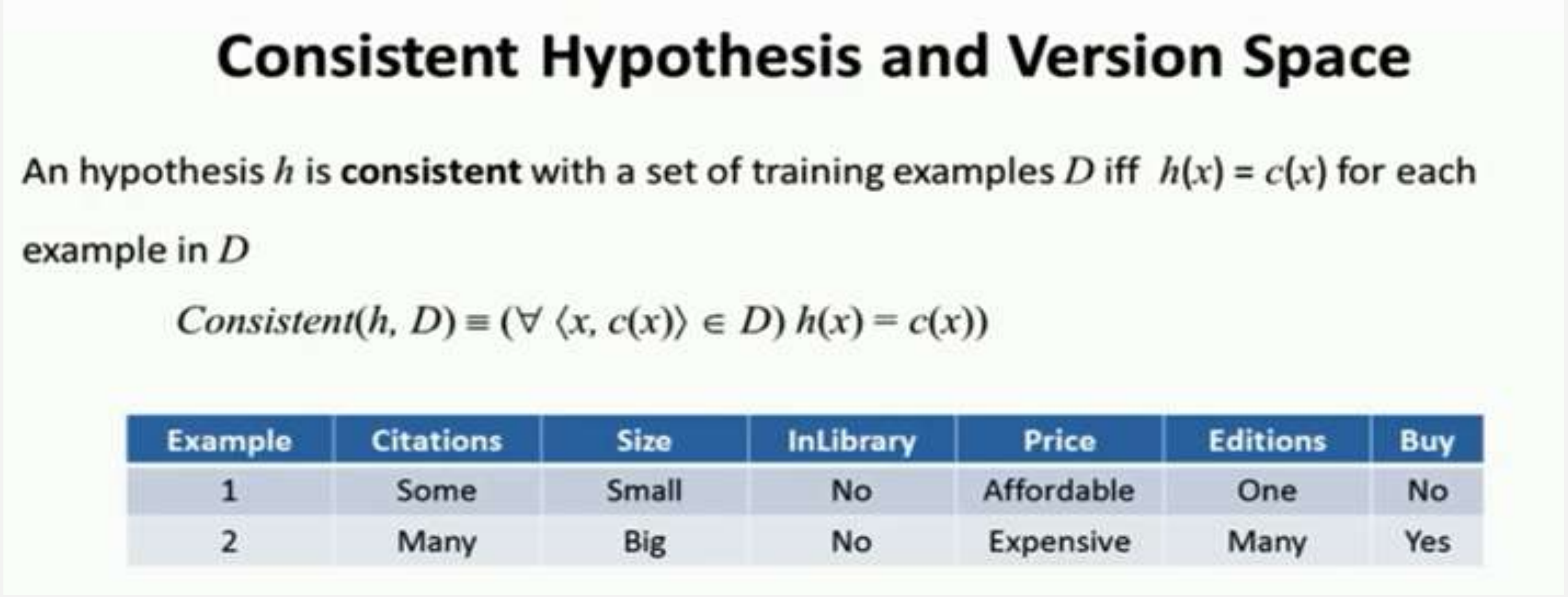
```
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes']
['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes']
['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'No']
['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes']
```

The Hypothesis are

```
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
['Sunny', 'Warm', '?', 'Strong', '?', '?']
```

Maximally Specific Hypothesis:

```
 ['Sunny', 'Warm', '?', 'Strong', '?', '?']
```



Consistent Hypothesis and Version Space

An hypothesis h is **consistent** with a set of training examples D iff $h(x) = c(x)$ for each example in D

$$Consistent(h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x))$$

Example	Citations	Size	InLibrary	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes

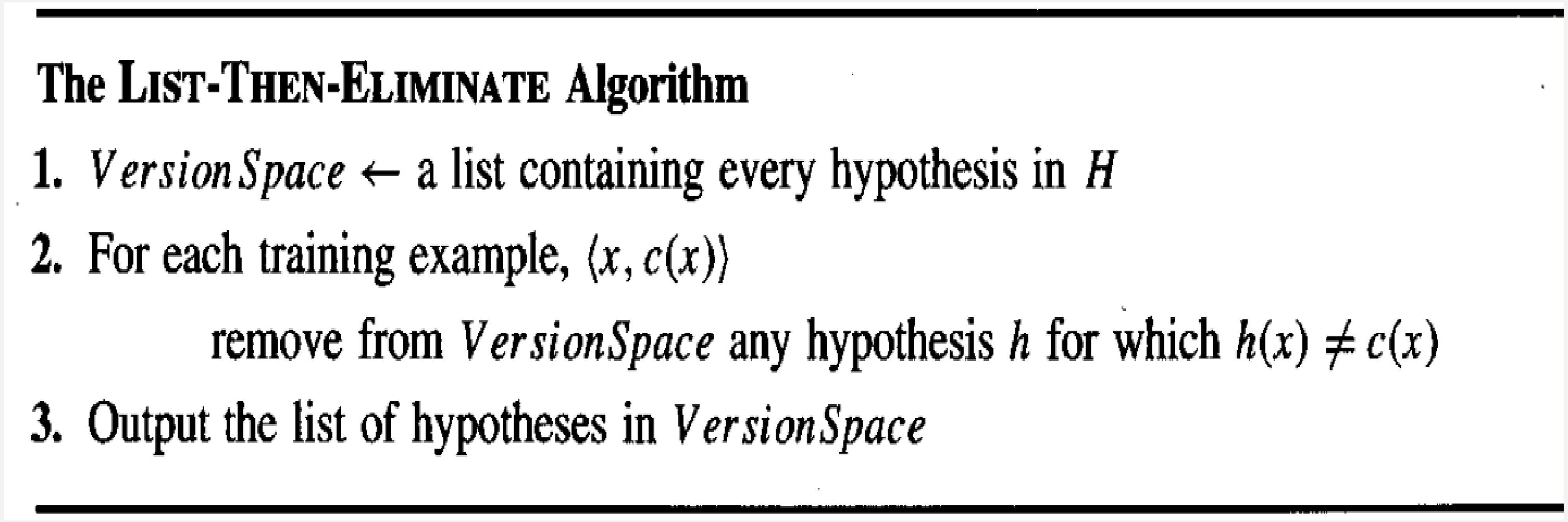
- **h1=<?,?, No, ?, Many>**
  -Consistent
- **h2==<?,?, No, ?, Many>**
  -Not Consistent

VERSION SPACE

- The version space $VS_{H,D}$ is the subset of the hypothesis from H *consistent* with the training example in D,

$$VS_{H,D} \equiv \{h \in H \mid Consistent(h, D)\}$$

LIST-THEN-ELIMINATE ALGORITHM



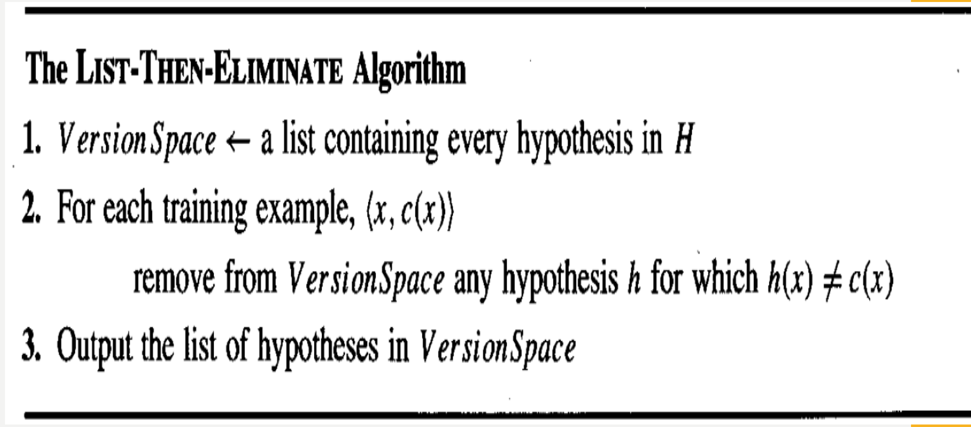
The LIST-THEN-ELIMINATE Algorithm

```
1. $VersionSpace \leftarrow$ a list containing every hypothesis in $H$
2. For each training example, $\langle x, c(x) \rangle$
        remove from $VersionSpace$ any hypothesis $h$ for which $h(x) \neq c(x)$
3. Output the list of hypotheses in $VersionSpace$
```

CONSISTENT HYPOTHESIS AND VERSION SPACE

- **Attrib1→**A,B
- **Attrib2→**P,Q
- Here Attrib1 and Attrib2 are two features (attributes) with two possible values for each feature or attribute.
- **Instance Space:** (A, P), (A, Q), (B, P), (B, Q) – **4 Examples**
- **Hypothesis Space:**
  - (A, P), (A, Q), (A, **ø**), (A, ?), (B, P), (B, Q), (B, **ø**), (B, ?), (**ø**, P), (**ø**, Q), (**ø**, **ø**), (**ø**, ?), (**?**, P), (**?**, Q), (**?**, **ø**), (**?**, ?) **– 16 Hypothesis**
- **Semantically Distinct Hypothesis :**
  - (A, P), (A, Q), (A, ?), (B, P), (B, Q), (B, ?), (**?**, P), (**?**, Q), (**?**, ?), (**ø**, **ø**) **– 10**

CONSISTENT HYPOTHESIS AND VERSION SPACE LIST-THEN-ELIMINATE ALGORITHM


The LIST-THEN-ELIMINATE Algorithm

```
1. VersionSpace ← a list containing every hypothesis in H
2. For each training example, ⟨x, c(x)⟩
       remove from VersionSpace any hypothesis h for which $h(x) \neq c(x)$
3. Output the list of hypotheses in VersionSpace
```

- **Version Space:**
  - (A, P), (A, Q), (A, ?), (B, P), (B, Q), (B, ?), (**?**, P), (**?**, Q), (**?**, ?), (**ø**, **ø**)
- **Training instances:**

Attrib1	Attrib1	Target
A	P	YES
A	Q	YES

- **Consistent Hypothesis are (Version Space):**
  - (A, ?), (?, ?)

VERSION SPACES AND THE CANDIDATE-ELIMINATION ALGORITHM

- **Version Space:** It is intermediate of **general hypothesis** and **Specific hypothesis**. It not only just written one hypothesis but a **set of all possible hypothesis** based on training data-set.
- The candidate elimination algorithm incrementally **builds the version space** given a hypothesis space H and a set D of examples.
- The examples are added one by one; each example possibly shrinks the **version space** by removing the hypotheses that are inconsistent with the example.
- The candidate elimination algorithm does this by updating the ***general*** and ***specific boundary*** for each new example.
  - You can consider this as an ***extended form of Find-S algorithm.***
  - Consider ***both positive and negative*** examples.
  - Actually, **positive examples are used here as Find-S algorithm** (Basically they are generalizing from the specification).
  - While the ***negative example is specified from generalize form***.

THE CANDIDATE-ELIMINATION ALGORITHM

- Initialize **G** and **S** as most general and specific hypothesis.
  - ***G=<?,?,?,?,?,?>***
  - ***S=< Ø, Ø, Ø, Ø, Ø, Ø>***
- *For each example e:*

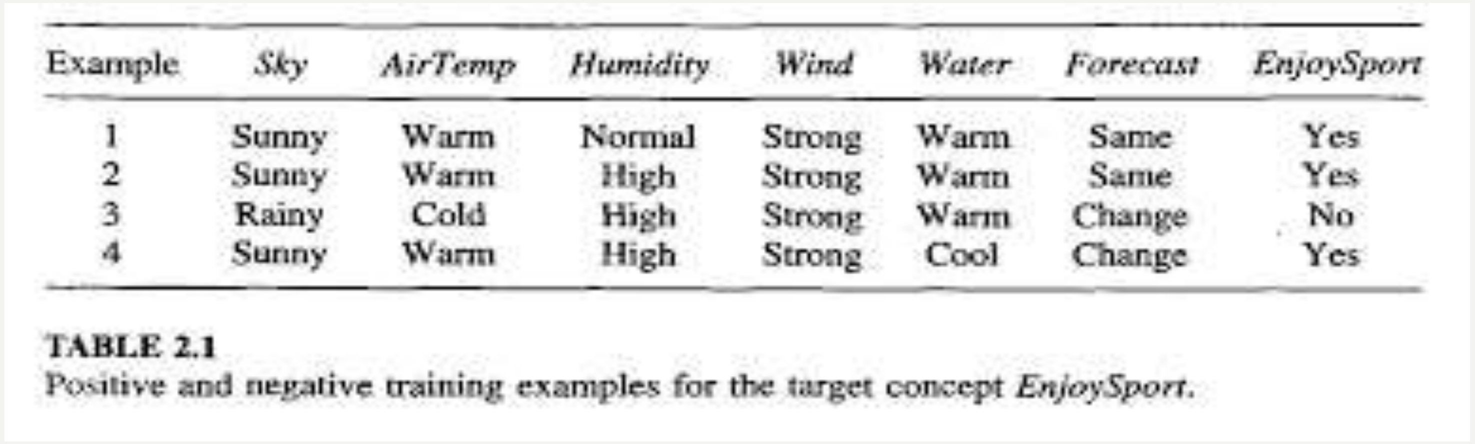
  *If e is **+ve**:*

  → *Make Specific Hypothesis more general [Find-S]*

  else:

  → *Make general Hypothesis more specific*

THE CANDIDATE-ELIMINATION ALGORITHM EXAMPLE



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

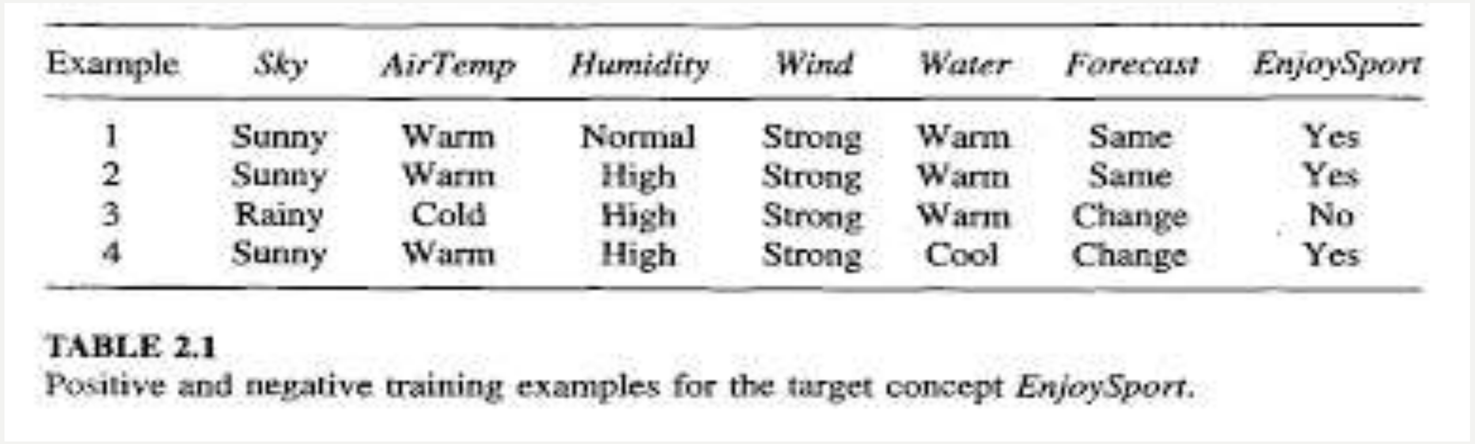
TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

Step-1: ***S0=< Ø, Ø, Ø, Ø, Ø, Ø> ,*** ***G0=<?,?,?,?,?,?>***
Step-2:

1. *+ve example*
   S1=< Sunny, Warm, Normal, Strong, Warm, Same>
   G1=<?,?,?,?,?,?>
2. *+ve example*
   S2=< Sunny, Warm, ?, Strong, Warm, Same>
   G2=<?,?,?,?,?,?>
3. *-ve example*
   S3=< Sunny, Warm, ?, Strong, Warm, Same>
   G3={<Sunny,?,?,?,?,?>, <?,Warm,?,?,?,?>, <?,?,Normal,?,?,?><?,?,?,?,Cool,?><?,?,?,?,?,Same>}

4. *+ve example*
   S4=< Sunny, Warm, ?, Strong, ?, ?>
   G4={<Sunny,?,?,?,?,?>, <?,Warm,?,?,?,?>}

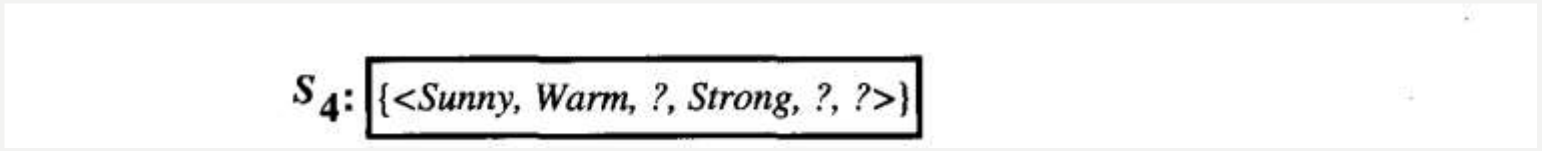
THE CANDIDATE-ELIMINATION ALGORITHM EXAMPLE



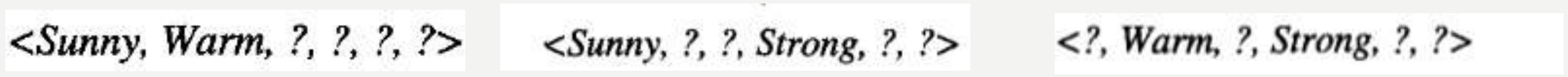
Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

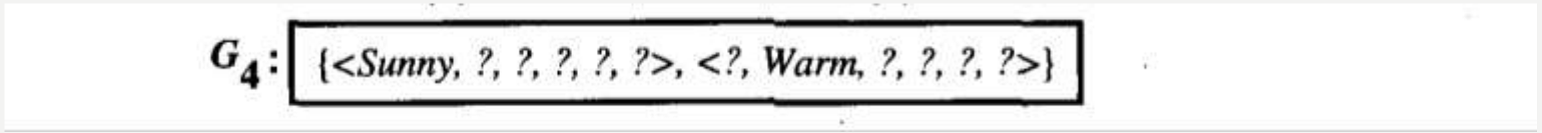
- **The final version space for the *EnjoySport concept learning problem and training examples are:***



S_4: {<*Sunny, Warm, ?, Strong, ?, ?*>}

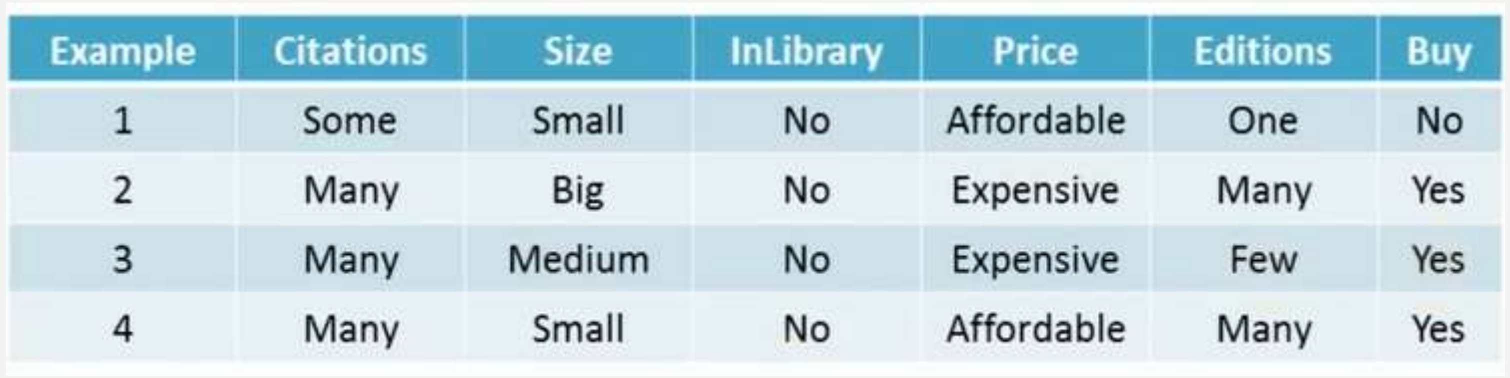


<*Sunny, Warm, ?, ?, ?, ?*>   <*Sunny, ?, ?, Strong, ?, ?*>   <*?, Warm, ?, Strong, ?, ?*>



G_4: {<*Sunny, ?, ?, ?, ?, ?*>, <*?, Warm, ?, ?, ?, ?*>}

Consider the Concept and instances given below, identify the hypothesis using candidate-Elimination learning algorithm.



Example	Citations	Size	InLibrary	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes
3	Many	Medium	No	Expensive	Few	Yes
4	Many	Small	No	Affordable	Many	Yes

Step-1: ***S0=< Ø, Ø, Ø, Ø, Ø>*** **,** ***G0=<?,?,?,?,?>***
Step-2:

1. -ve example

S1=< Ø, Ø, Ø, Ø, Ø >
G1=<many,?,?,?,?>, <?,big,?,?,?>, <?,medium,?,?,?>, <?,?,?,expensive,?>, <?,?,?,?,many>, <?,?,?,?,few>

2. +ve example

S2 =< many, big, no, expensive, many>
G2 =<many,?,?,?,?>, <?,big,?,?,?>, <?,?,?,expensive,?>, <?,?,?,?,many>

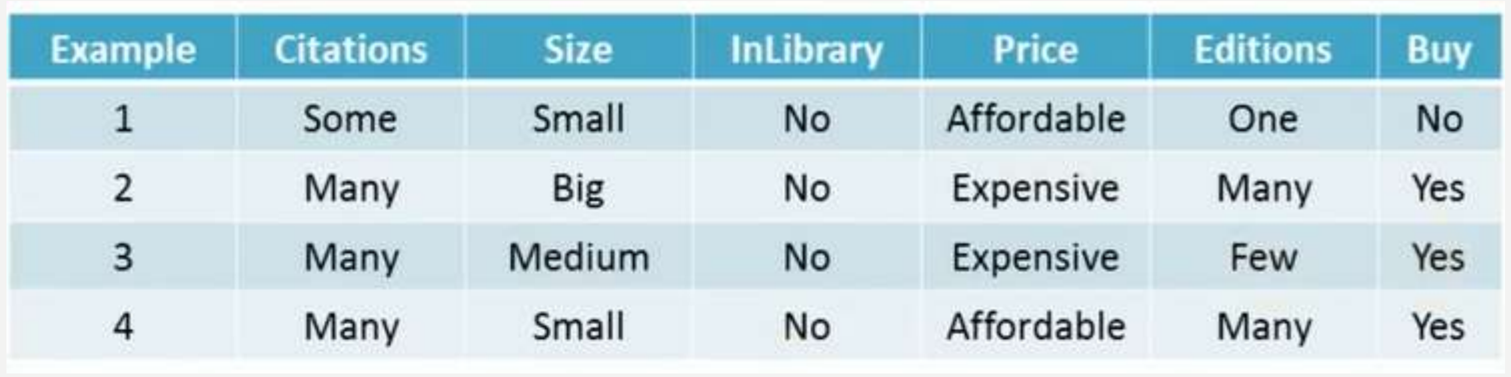
3. +ve example

S3 =< many, ?, no, expensive, ?>
G3 =<many,?,?,?,?>, <?,?,?,expensive,?>

4. +ve example

S4 =< many, ?, no, ?, ?>
G4 =<many,?,?,?,?>

THE CANDIDATE-ELIMINATION ALGORITHM EXAMPLE-2



Example	Citations	Size	InLibrary	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes
3	Many	Medium	No	Expensive	Few	Yes
4	Many	Small	No	Affordable	Many	Yes

- **The final version space:*****< many, ?, no, ?, ?>, <many,?,?,?,?>***

$$\begin{tikzcd} \textit{S4 =< many, ?, no, ?, ?>} \arrow[d] \\ \textit{< many, ?, no, ?, ?>, <many,?,?,?,?>} \\ \textit{G4 =<many,?,?,?,?>} \arrow[u] \end{tikzcd}$$

Consider the "Japanese Economy Car" Concept and instances given below, identify the hypothesis using candidate-Elimination learning algorithm.

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive

Step-1: ***S0=< Ø, Ø, Ø, Ø, Ø>*** , ***G0=<?,?,?,?,?>***

Step-2:

1. +ve example

S1=< Japan, Honda, Blue, 1980, Economy>

G1=<?,?,?,?,?>

2. -ve example

S2 =< Japan, Honda, Blue, 1980, Economy>

G2={ <?,Honda,?,?,?>,<?,?,Blue,?,?>, <?,?,?,1980,?>, <?,?,?,?,Economy>}

3. +ve example

S3 =< Japan, ?, Blue, ?, Economy>

G3 ={<?,?,Blue,?,?>, <?,?,?,?,Economy>}

4. -ve example

S4 =< Japan, ?, Blue, ?, Economy>

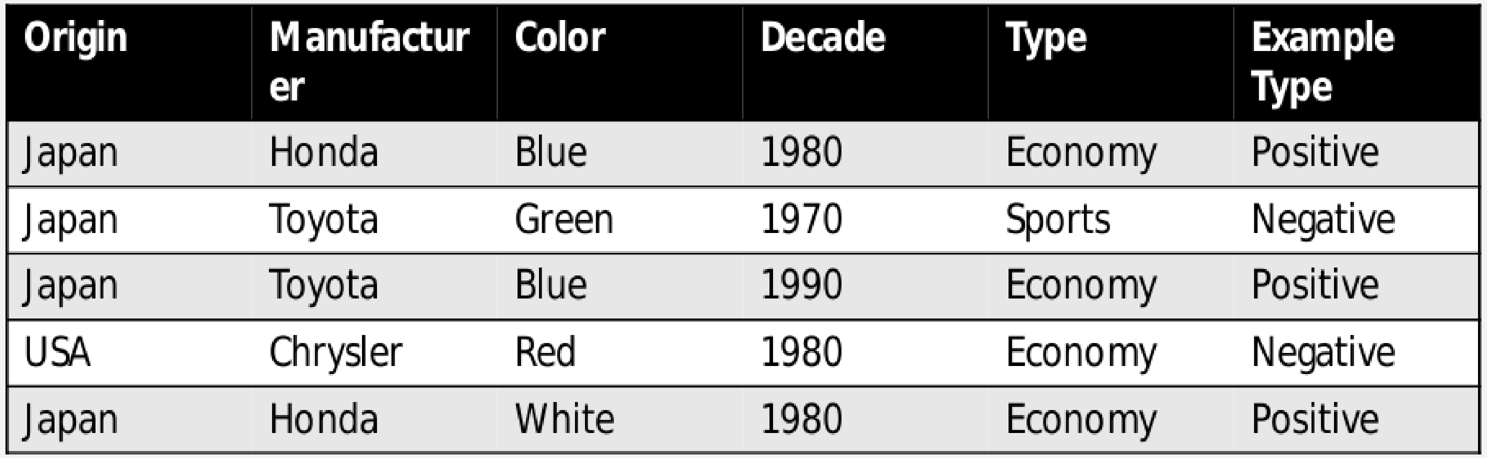
G4 ={<Japan,?,?,?, Economy > ,<?,?,Blue,?,?>}

5. +ve example

S5 =< Japan, ?, ?,?, Economy>

G5 ={<Japan,?,?,?, Economy > }

THE CANDIDATE-ELIMINATION ALGORITHM EXAMPLE-3

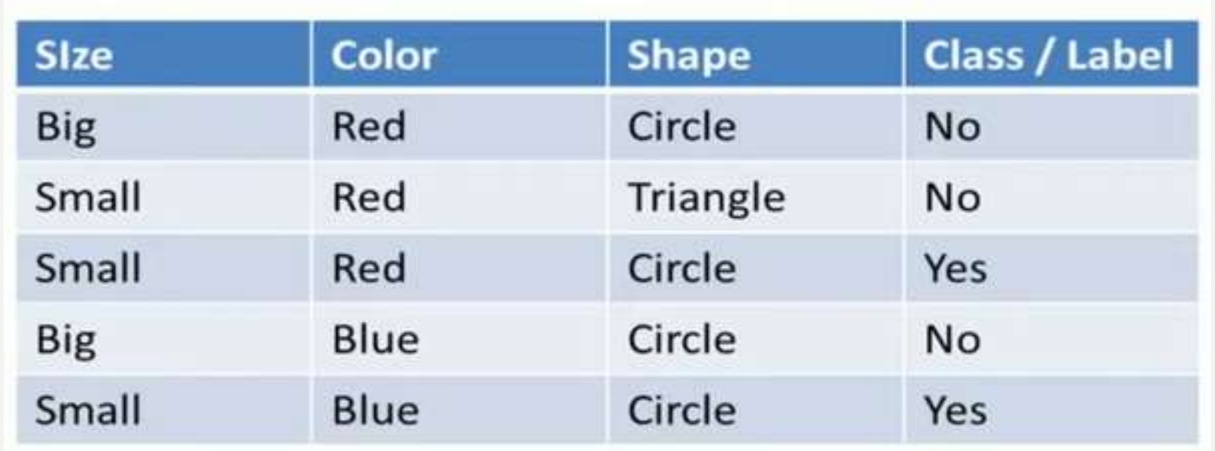


Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive

- **The final version space :*<Japan, ?,?,?, Economy>***

$$\begin{tikzcd}
\textit{S5 =< Japan, ?, ?,?, Economy>} \arrow[d] \\
\textit{<Japan, ?,?,?, Economy>} \\
\textit{G4 =\{ <Japan,?,?,?, Economy>\}} \arrow[u]
\end{tikzcd}$$

THE CANDIDATE-ELIMINATION ALGORITHM EXAMPLE-4



SIze	Color	Shape	Class / Label
Big	Red	Circle	No
Small	Red	Triangle	No
Small	Red	Circle	Yes
Big	Blue	Circle	No
Small	Blue	Circle	Yes

Step-1: ***S0=< Ø, Ø, Ø> ,*** ***G0=<?,?,? >***
Step-2:

1. -ve example
S1=< Ø, Ø, Ø>
G1=<small,?,? >, <?,blue,? >, <?,?,triangle>

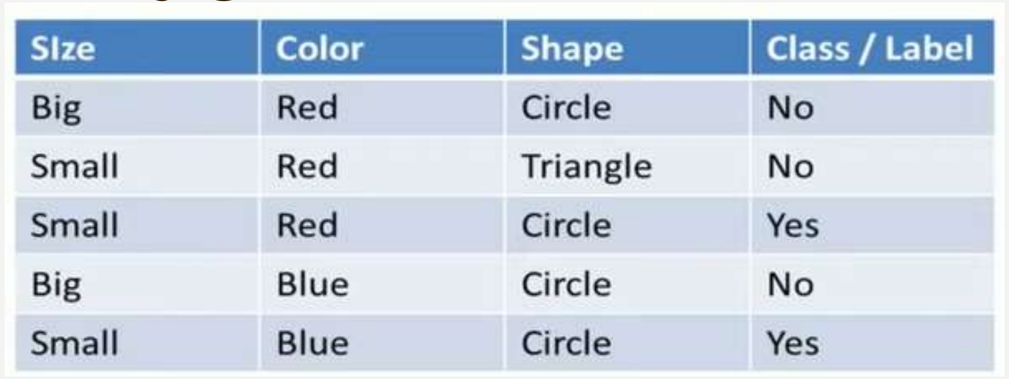
2. -ve example
S2 =< Ø, Ø, Ø>
G2 =<small, blue,? >, <small,?,circle>, <?,blue,? >, <big,?,triangle>, <?,blue, triangle>

3. +ve example
S3 =< small, red, circle>
G3 = <small,?,circle>

4. -ve example
S4 =< small, red, circle>
G4 = <small,?,circle>

5. +ve example
S5 =< small, ?, circle>
G5 = <small,?,circle>

THE CANDIDATE-ELIMINATION ALGORITHM EXAMPLE-4

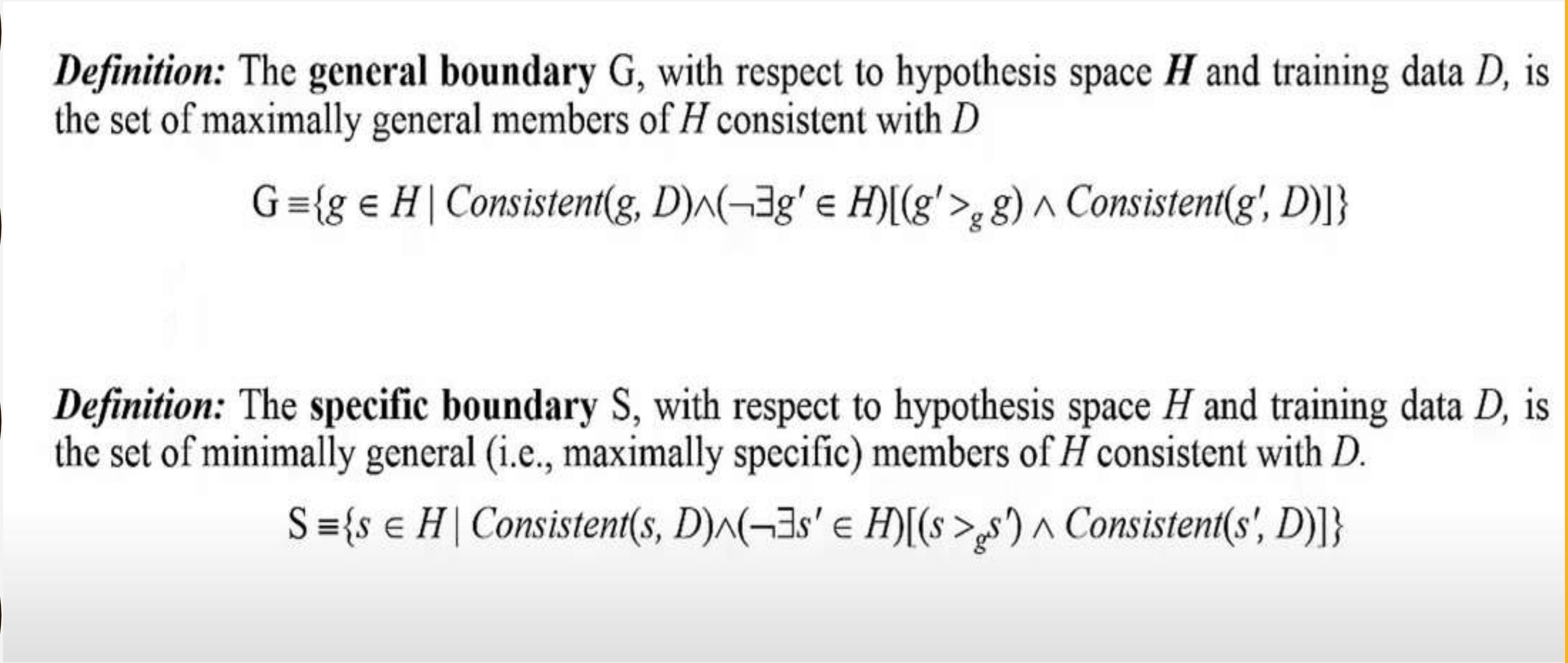


SIze	Color	Shape	Class / Label
Big	Red	Circle	No
Small	Red	Triangle	No
Small	Red	Circle	Yes
Big	Blue	Circle	No
Small	Blue	Circle	Yes

- **The final version space :*<small, ?,circle>***

$$\begin{tikzcd}
S5 =< small, ?, circle> \arrow[d] \\
<small, ?,circle> \\
G4 =< small, ?, circle> \arrow[u]
\end{tikzcd}$$

GENERAL BOUNDARY AND SPECIFIC BOUNDARY



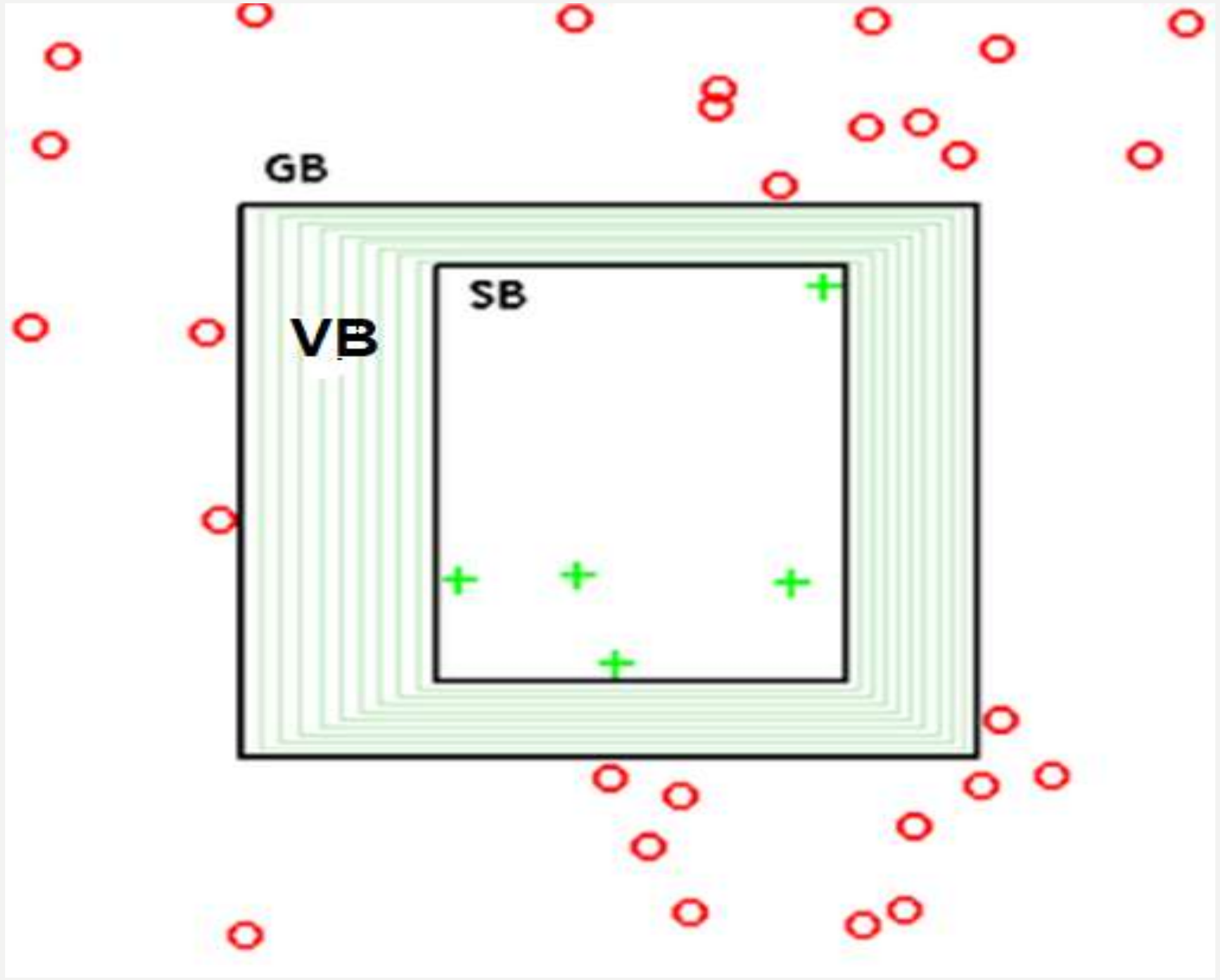
Definition: The **general boundary** G, with respect to hypothesis space ***H*** and training data *D*, is the set of maximally general members of *H* consistent with *D*

$$G \equiv \{g \in H \mid Consistent(g, D) \wedge (\neg \exists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$$

Definition: The **specific boundary** S, with respect to hypothesis space *H* and training data *D*, is the set of minimally general (i.e., maximally specific) members of *H* consistent with *D*.

$$S \equiv \{s \in H \mid Consistent(s, D) \wedge (\neg \exists s' \in H)[(s >_g s') \wedge Consistent(s', D)]\}$$

SB, GB, VS



VERSION SPACE REPRESENTATION THEOREM

- ***Theorem:*** Let X be an arbitrary set of instances and Let H be a set of Boolean-valued hypotheses defined over X. Let c : X →{O, 1} be an arbitrary target concept defined over X, and let D be an arbitrary set of training examples {(x, c(x))). For all X, H, c, and D such that S and G are well defined,

$$VS_{H,D}=\{h \in H \mid (\exists s \in S)\ (\exists g \in G)\ (g \geq_g h \geq_g s)\}$$

$$VS_{H,D}=\{h \in H \mid (\exists s \in S)\ (\exists g \in G)\ (g \geq_g h \geq_g s)\}$$

To Prove:

1. Every h satisfying the right hand side of the above expression is in $VS_{H,D}$
2. Every member of $VS_{H,D}$ satisfies the right-hand side of the expression

Sketch of proof:

1. Let **g, h, s** be arbitrary members of G, H, S respectively with $g \geq_g h \geq_g s$
   - By the definition of ***S***, **s** must be satisfied by all positive examples in D. Because $h \geq_g s$, h must also be satisfied by all positive examples in D.
   - By the definition of ***G***, g cannot be satisfied by any negative example in D, and because $g \geq_g h$, h cannot be satisfied by any negative example in D. Because h is satisfied by all positive examples in D and by no negative examples in D, h is consistent with D, and therefore h is a member of VSH,D

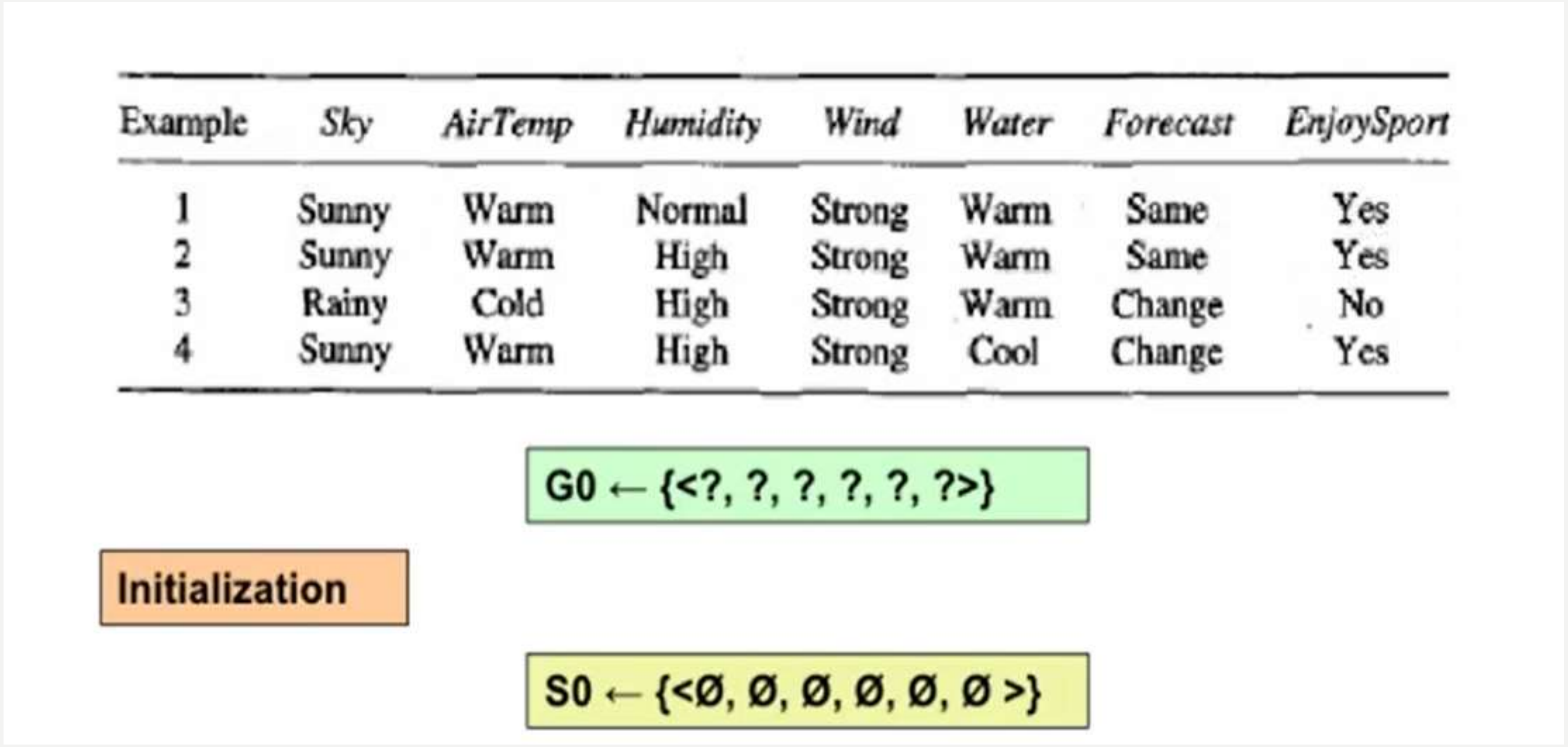
2. It can be proven by assuming some h in $VS_{H,D}$,that does not satisfy the right-hand side of the expression, then showing that this leads to an inconsistency.

THE CANDIDATE-ELIMINATION ALGORITHM

```
Initialize $G$ to the set of maximally general hypotheses in $H$
Initialize $S$ to the set of maximally specific hypotheses in $H$
For each training example $d$, do
• If $d$ is a positive example
    • Remove from $G$ any hypothesis inconsistent with $d$
    • For each hypothesis $s$ in $S$ that is not consistent with $d$
        • Remove $s$ from $S$
        • Add to $S$ all minimal generalizations $h$ of $s$ such that
            • $h$ is consistent with $d$, and some member of $G$ is more general than $h$
        • Remove from $S$ any hypothesis that is more general than another hypothesis in $S$
• If $d$ is a negative example
    • Remove from $S$ any hypothesis inconsistent with $d$
    • For each hypothesis $g$ in $G$ that is not consistent with $d$
        • Remove $g$ from $G$
        • Add to $G$ all minimal specializations $h$ of $g$ such that
            • $h$ is consistent with $d$, and some member of $S$ is more specific than $h$
        • Remove from $G$ any hypothesis that is less general than another hypothesis in $G$
```

TABLE 2.5
CANDIDATE-ELIMINATION algorithm using version spaces. Notice the duality in how positive and negative examples influence S and G.

ILLUSTRATION



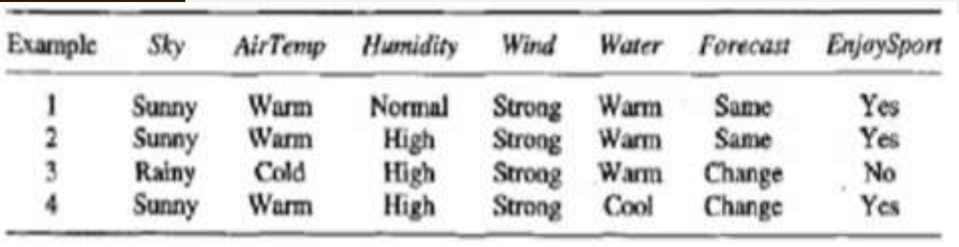
Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

G0 ← {<?, ?, ?, ?, ?, ?>}

Initialization

S0 ← {<Ø, Ø, Ø, Ø, Ø, Ø >}

ILLUSTRATION



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

G0 ← {<?, ?, ?, ?, ?, ?>}

S0 ← {<Ø, Ø, Ø, Ø, Ø, Ø >}

Iteration 1

x1 = <Sunny, Warm, Normal, Strong, Warm, Same>

G1 ← {<?, ?, ?, ?, ?, ?>}

S1 ← {< Sunny, Warm, Normal, Strong, Warm, Same >}

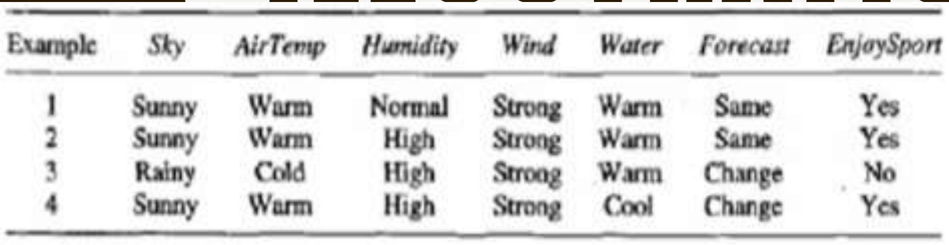
Iteration 2

x2 = <Sunny, Warm, High, Strong, Warm, Same>

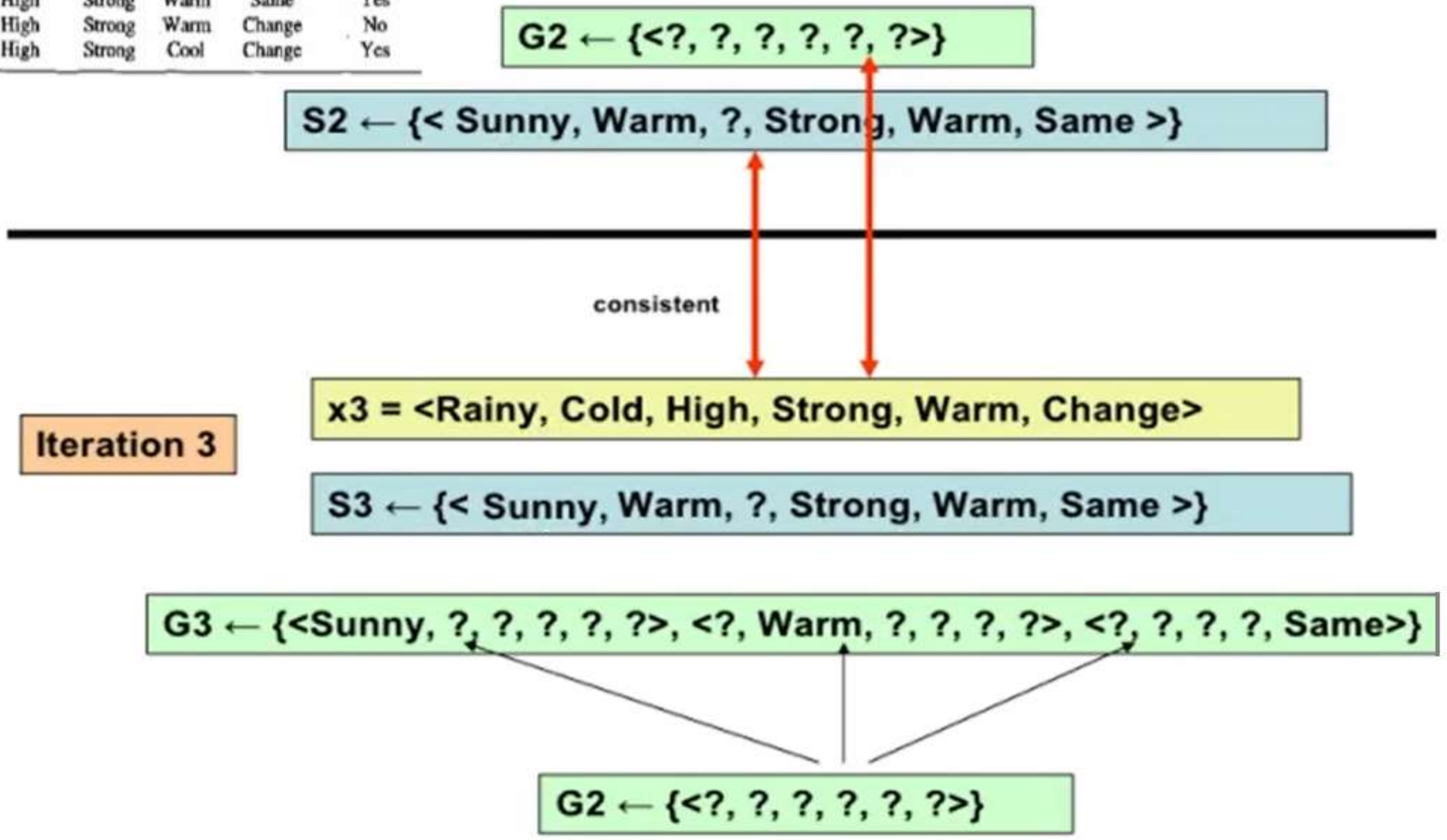
G2 ← {<?, ?, ?, ?, ?, ?>}

S2 ← {< Sunny, Warm, ?, Strong, Warm, Same >}

ILLUSTRATION



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



G2 ← {<?, ?, ?, ?, ?, ?>}

S2 ← {< Sunny, Warm, ?, Strong, Warm, Same >}

consistent

Iteration 3

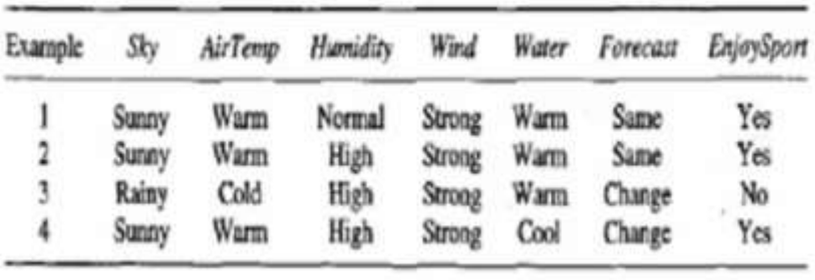
x3 = <Rainy, Cold, High, Strong, Warm, Change>

S3 ← {< Sunny, Warm, ?, Strong, Warm, Same >}

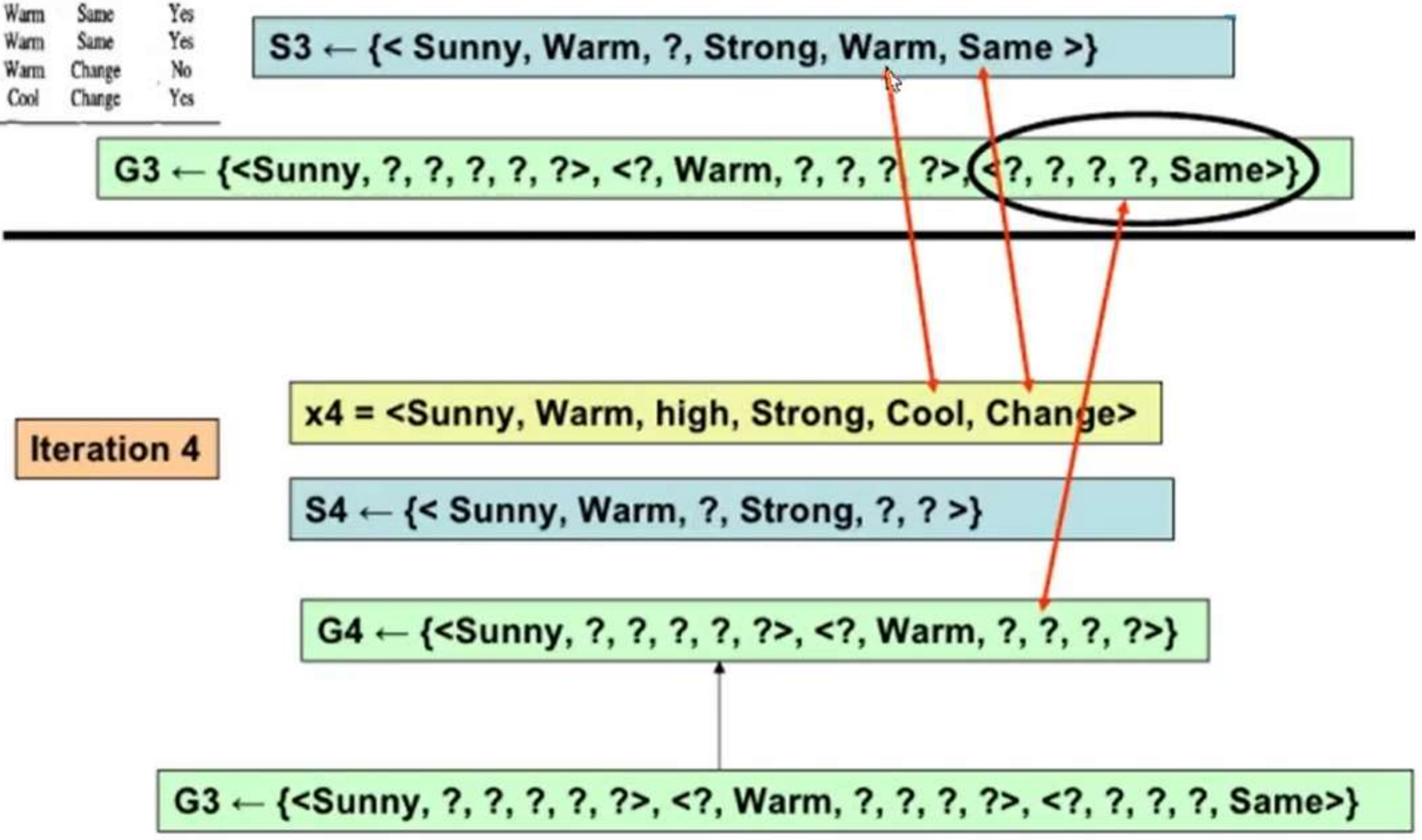
G3 ← {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, ?, Same>}

G2 ← {<?, ?, ?, ?, ?, ?>}

ILLUSTRATION



Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



S3 ← {< Sunny, Warm, ?, Strong, Warm, Same >}

G3 ← {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, Same>}

Iteration 4

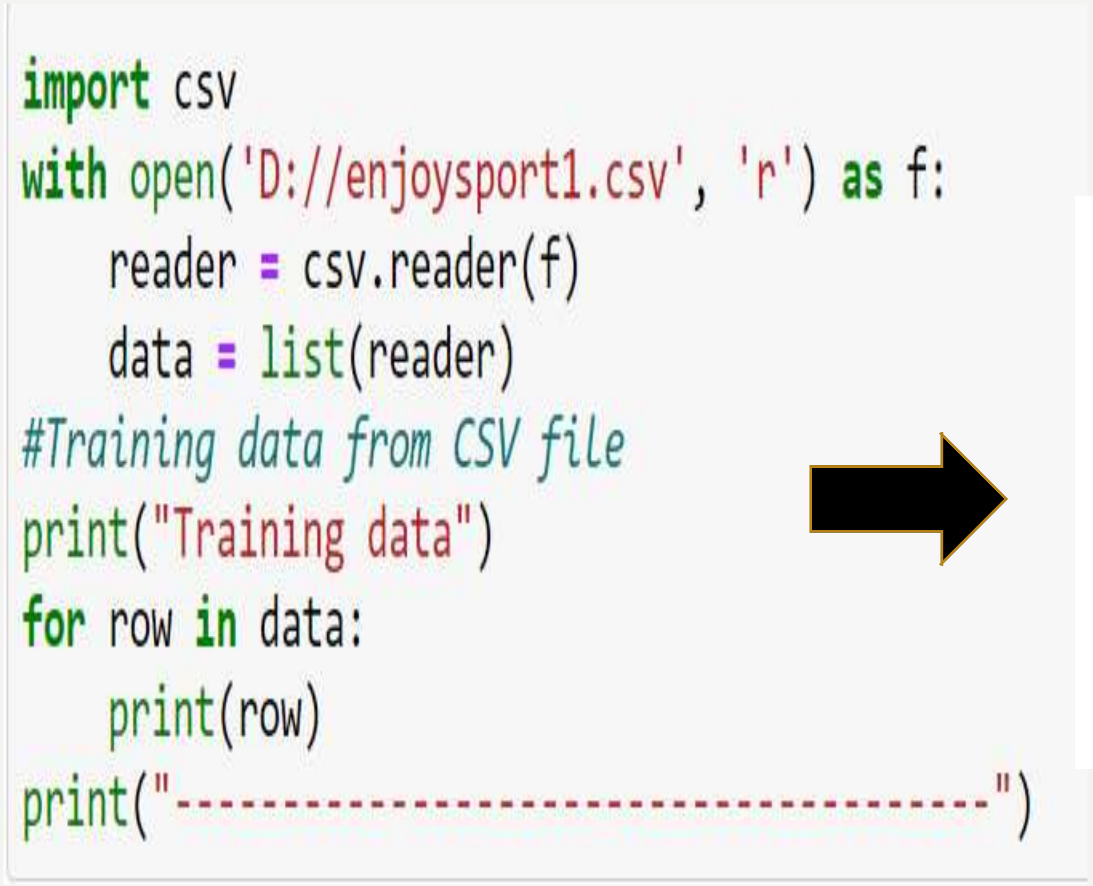
x4 = <Sunny, Warm, high, Strong, Cool, Change>

S4 ← {< Sunny, Warm, ?, Strong, ?, ? >}

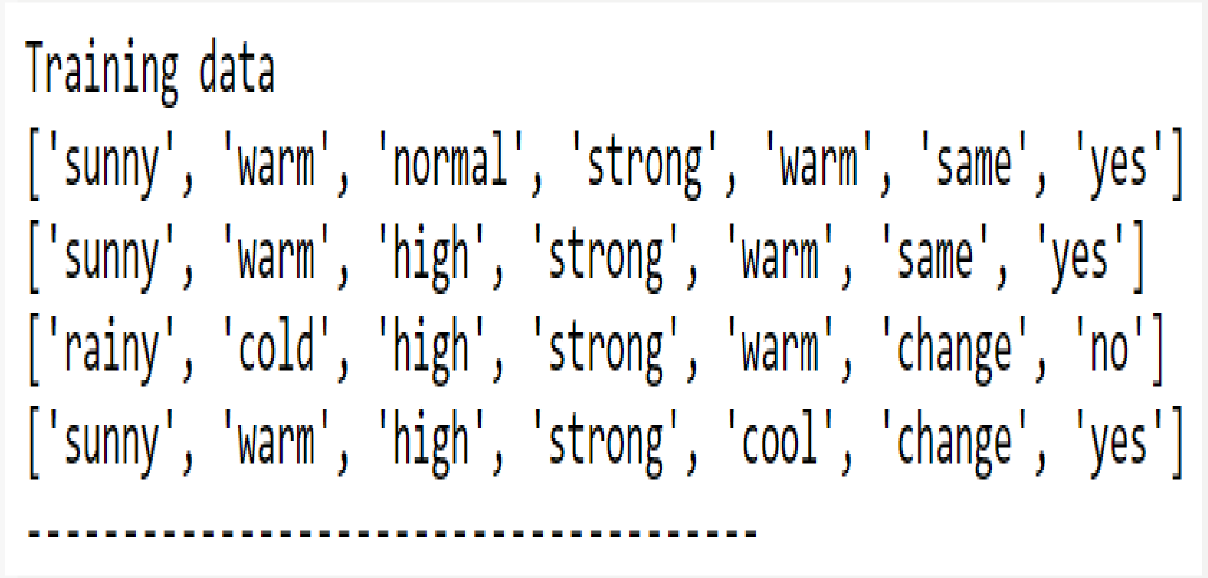
G4 ← {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>}

G3 ← {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, Same>}

IMPLEMENTATION/PROGRAM

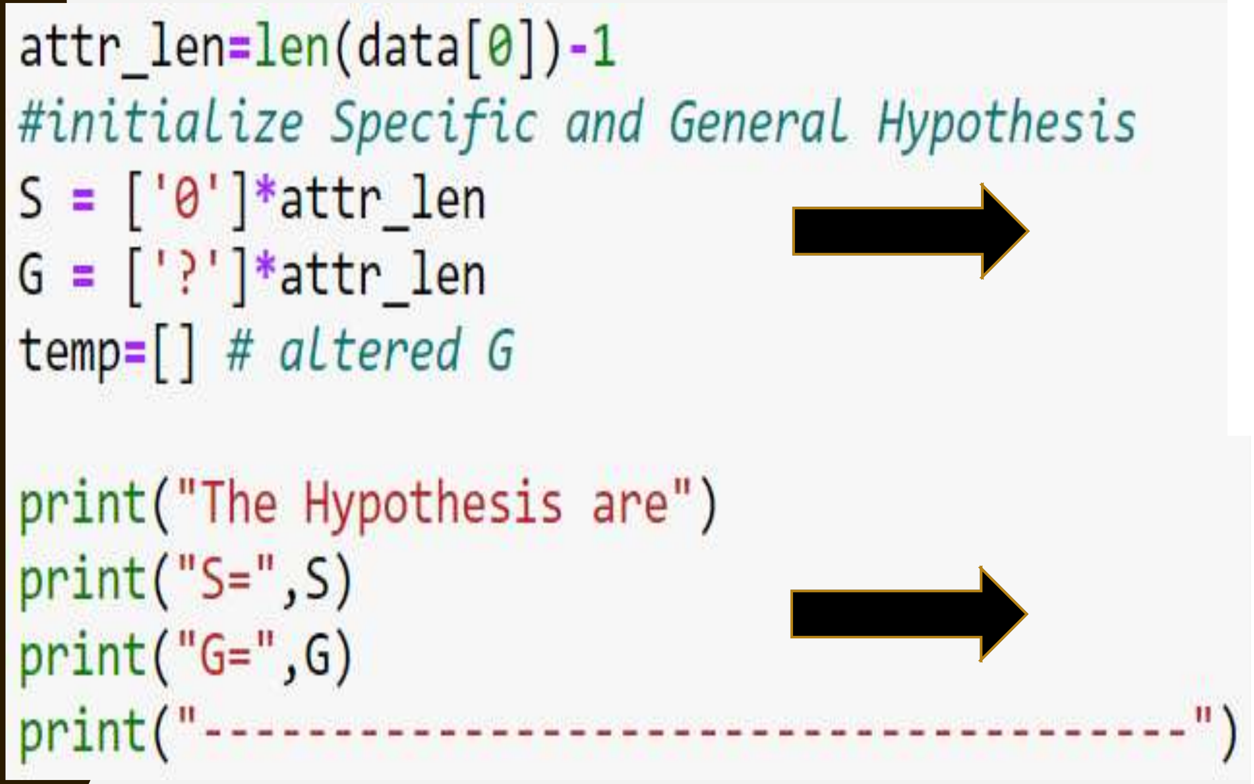


```python
import csv
with open('D://enjoysport1.csv', 'r') as f:
    reader = csv.reader(f)
    data = list(reader)
#Training data from CSV file
print("Training data")
for row in data:
    print(row)
print("-----------------------------------")
```

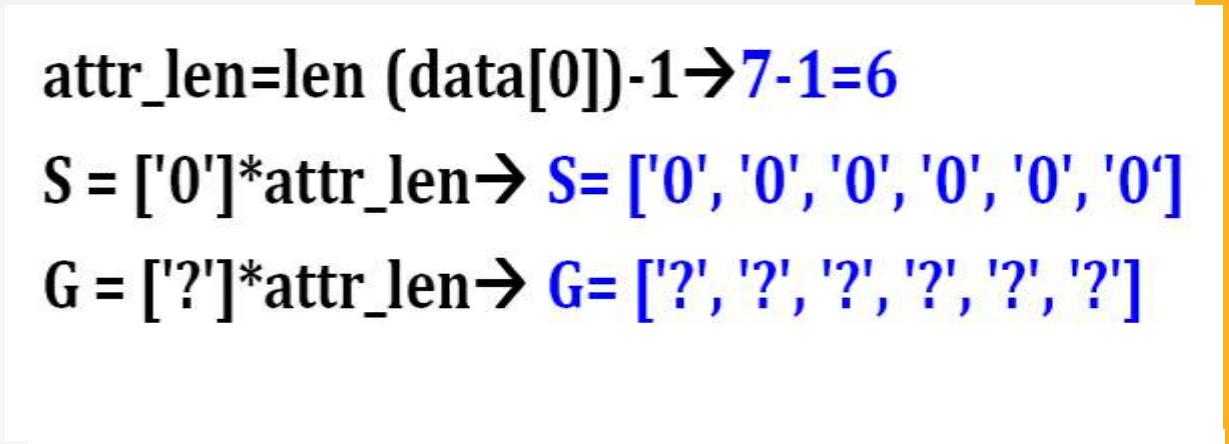


```
Training data
['sunny', 'warm', 'normal', 'strong', 'warm', 'same', 'yes']
['sunny', 'warm', 'high', 'strong', 'warm', 'same', 'yes']
['rainy', 'cold', 'high', 'strong', 'warm', 'change', 'no']
['sunny', 'warm', 'high', 'strong', 'cool', 'change', 'yes']
-----------------------------------
```

IMPLEMENTATION/PROGRAM



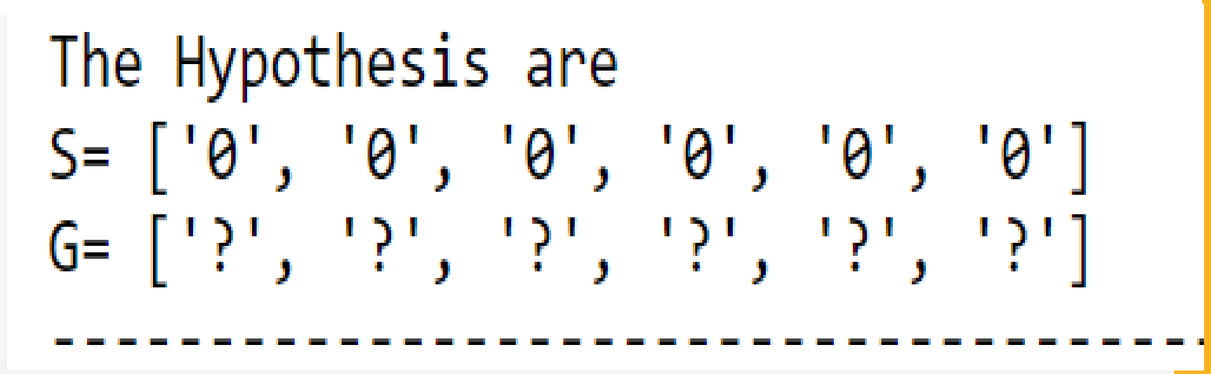
```python
attr_len=len(data[0])-1
#initialize Specific and General Hypothesis
S = ['0']*attr_len
G = ['?']*attr_len
temp=[] # altered G

print("The Hypothesis are")
print("S=",S)
print("G=",G)
print("-----------------------------------")
```



attr_len=len (data[0])-1→7-1=6

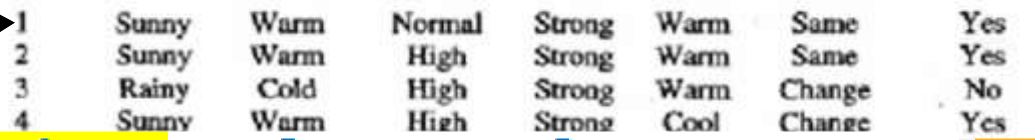
S = ['0']*attr_len→ S= ['0', '0', '0', '0', '0', '0']

G = ['?']*attr_len→ G= ['?', '?', '?', '?', '?', '?']

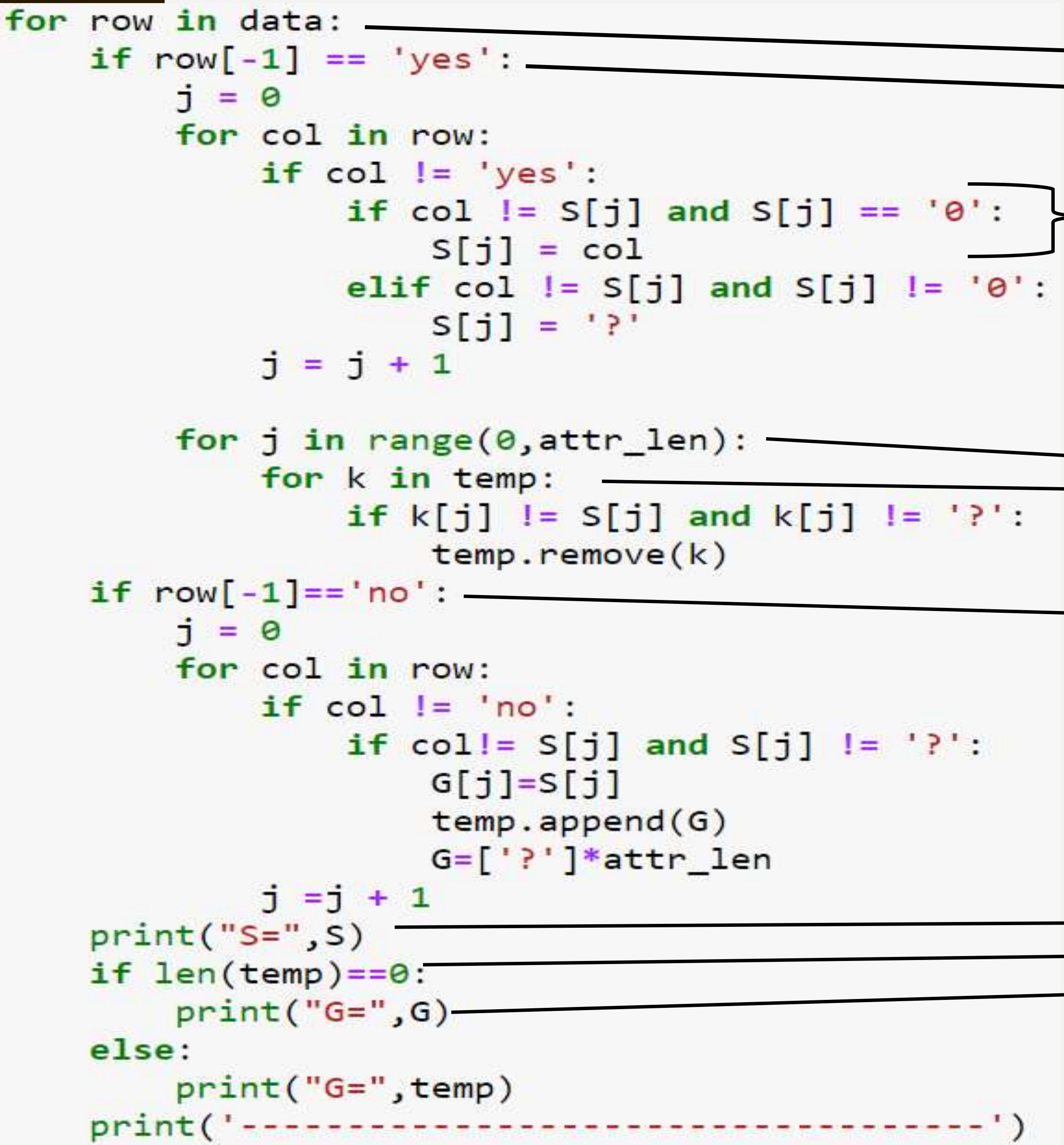


```
The Hypothesis are
S= ['0', '0', '0', '0', '0', '0']
G= ['?', '?', '?', '?', '?', '?']
-----------------------------------
```

IMPLEMENTATION/PROGRAM



1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



```
for row in data:
    if row[-1] == 'yes':
        j = 0
        for col in row:
            if col != 'yes':
                if col != S[j] and S[j] == '0':
                    S[j] = col
                elif col != S[j] and S[j] != '0':
                    S[j] = '?'
            j = j + 1

        for j in range(0,attr_len):
            for k in temp:
                if k[j] != S[j] and k[j] != '?':
                    temp.remove(k)
    if row[-1]=='no':
        j = 0
        for col in row:
            if col != 'no':
                if col!= S[j] and S[j] != '?':
                    G[j]=S[j]
                    temp.append(G)
                    G=['?']*attr_len
            j =j + 1
    print("S=",S)
    if len(temp)==0:
        print("G=",G)
    else:
        print("G=",temp)
    print('-----------------------------------')
```

Iteration-1 S=[0.0.0.0.0.0]

Sunny	Warm	Normal	Strong	Warm	Same	Yes

True

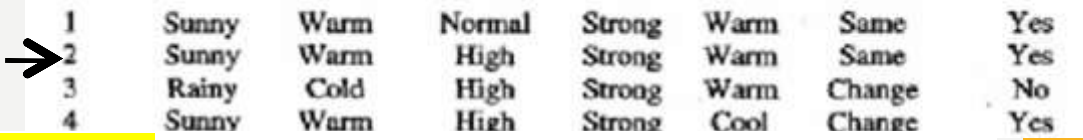
S[0]!='Sunny' and S[0]=='0'→T→S[0]='Sunny'
S[1]!='Warm' and S[1]=='0' →T →S[1]=' Warm'
S[2]!='Normal' and S[2]=='0' →T →S[2]=' Normal'
S[3]!='Strong' and S[3]=='0' →T →S[3]=' Strong'
S[4]!='Warm' and S[4]=='0' →T →S[4]=' Warm'
S[5]!='Same' and S[5]=='0' →T →S[5]='Same'

(0,5)
Temp=[ ]→false

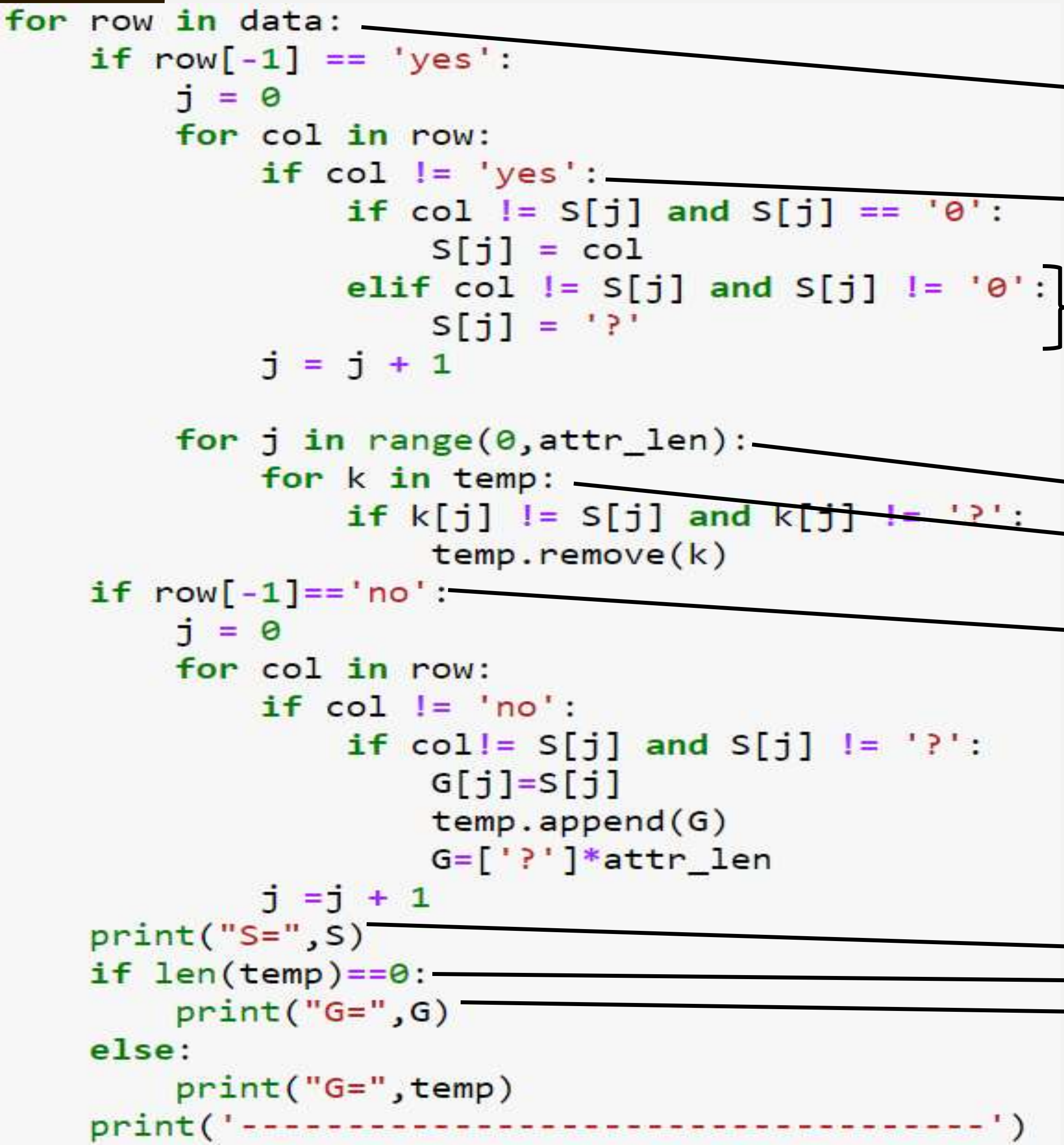
False

S=['Sunny','Warm','Normal','Strong','Warm','Same']
len(temp)==0→True
G=['?', '?', '?', '?', '?', '?']

IMPLEMENTATION/PROGRAM



1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



```
for row in data:
    if row[-1] == 'yes':
        j = 0
        for col in row:
            if col != 'yes':
                if col != S[j] and S[j] == '0':
                    S[j] = col
                elif col != S[j] and S[j] != '0':
                    S[j] = '?'
            j = j + 1

        for j in range(0,attr_len):
            for k in temp:
                if k[j] != S[j] and k[j] != '?':
                    temp.remove(k)
    if row[-1]=='no':
        j = 0
        for col in row:
            if col != 'no':
                if col!= S[j] and S[j] != '?':
                    G[j]=S[j]
                    temp.append(G)
                    G=['?']*attr_len
            j =j + 1
    print("S=",S)
    if len(temp)==0:
        print("G=",G)
    else:
        print("G=",temp)
    print('--------------------------------')
```

Iteration-2

S=['Sunny','Warm','Normal','Strong','Warm','Same']

Sunny	Warm	High	Strong	Warm	Same	Yes

True

S[0]!='Sunny' and S[0]!='0'→F
S[1]!='Warm' and S[1]!='0' →F
S[2]!='High' and S[2]!='0' →T →S[2]=' ?'
S[3]!='Strong' and S[3]!='0' →F
S[4]!='Warm' and S[4]!='0' →F
S[5]!='Same' and S[5]!='0' →F

(0,5)
Temp=[ ]→false

False

S=['Sunny','Warm','Normal','Strong','Warm','Same']
len(temp)==0→True
G=['?', '?', '?', '?', '?', '?']

IMPLEMENTATION/PROGRAM

3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

```
for row in data:
    if row[-1] == 'yes':
        j = 0
        for col in row:
            if col != 'yes':
                if col != S[j] and S[j] == '0':
                    S[j] = col
                elif col != S[j] and S[j] != '0':
                    S[j] = '?'
            j = j + 1

        for j in range(0,attr_len):
            for k in temp:
                if k[j] != S[j] and k[j] != '?':
                    temp.remove(k)
    if row[-1]=='no':
        j = 0
        for col in row:
            if col != 'no':
                if col!= S[j] and S[j] != '?':
                    G[j]=S[j]
                    temp.append(G)
                    G=['?']*attr_len
            j =j + 1
    print("S=",S)
    if len(temp)==0:
        print("G=",G)
    else:
        print("G=",temp)
    print('----------------------------------')
```

Iteration-3

S=['Sunny','Warm','?','Strong','Warm','Same']

Rainy	Cold	High	Strong	Warm	Change	No

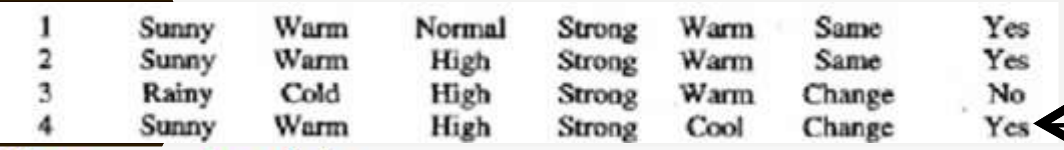
False

True

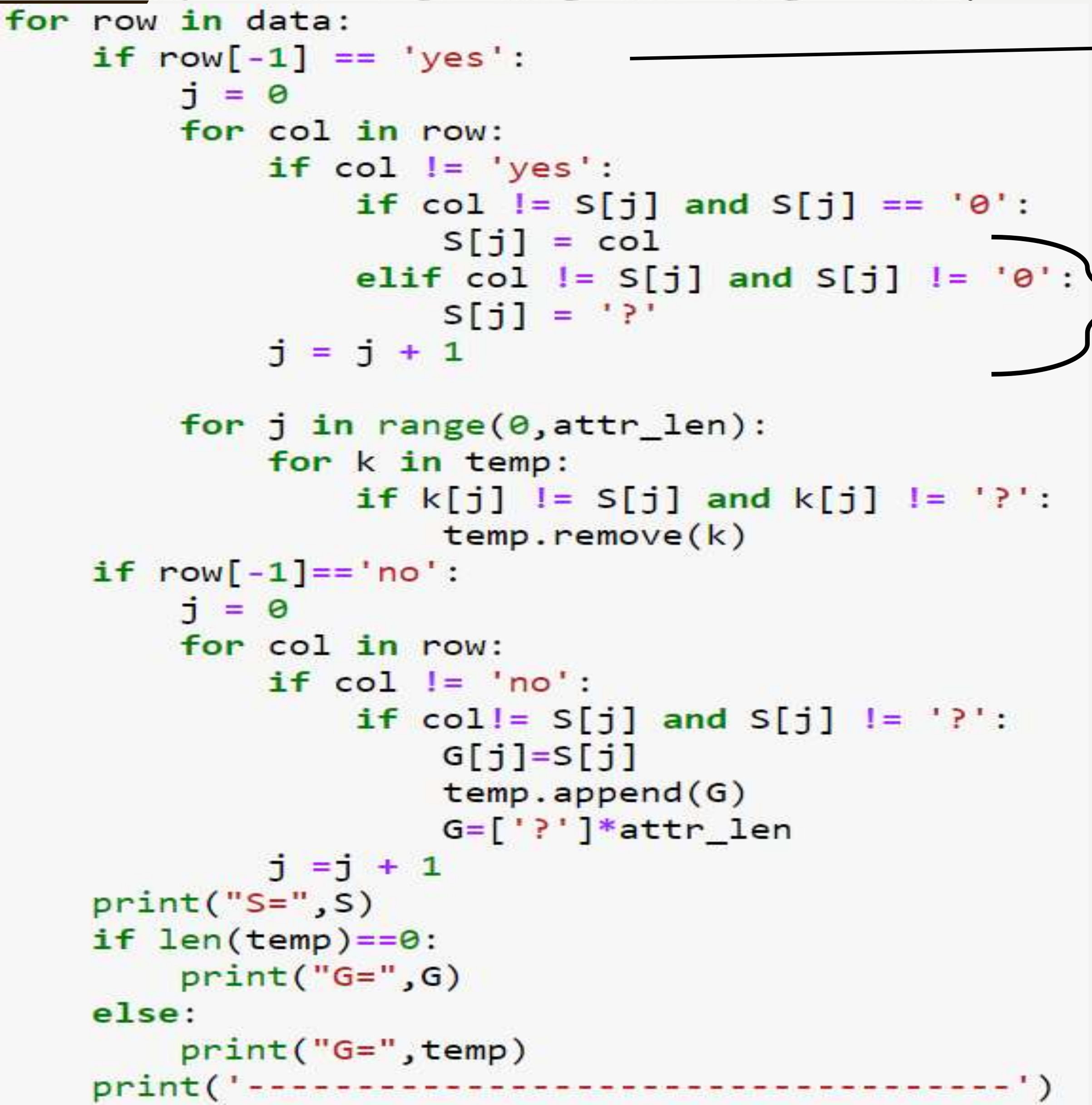
'Rainy' !=S[0] and S[0]!='?'→T
G[0]='Sunny'
temp=['Sunny','?', '?', '?', '?', '?']
G=['?', '?' '?', '?', '?', '?']
'Cold' != S[1]and S[1]!='?' →T
G[1]='Warm'
temp=[['Sunny','?', '?', '?', '?', '?'], ['?','Warm', '?', '?', '?', '?']]
G=['?', '?' '?', '?', '?', '?']
'High' !=S[2]and S[2]!='?' →F
'Strong' != S[3] and S[3]!='?' →F
'Warm' !=S[4] and S[4]!='?' →F
'Change' != S[5] and S[5]!='?' →T
G[5]='Same'
temp=[['Sunny','?', '?', '?', '?', '?'], ['?','Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?','Same']]
G=['?', '?' '?', '?', '?', '?']

S=['Sunny','Warm'.'?'.'Strong','Warm','Same']
len(temp)==0→ False→G=temp
G=[['Sunny','?', '?', '?', '?', '?'], ['?','Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?','Same']]

IMPLEMENTATION/PROGRAM



1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



```
for row in data:
    if row[-1] == 'yes':
        j = 0
        for col in row:
            if col != 'yes':
                if col != S[j] and S[j] == '0':
                    S[j] = col
                elif col != S[j] and S[j] != '0':
                    S[j] = '?'
            j = j + 1

        for j in range(0,attr_len):
            for k in temp:
                if k[j] != S[j] and k[j] != '?':
                    temp.remove(k)
    if row[-1]=='no':
        j = 0
        for col in row:
            if col != 'no':
                if col!= S[j] and S[j] != '?':
                    G[j]=S[j]
                    temp.append(G)
                    G=['?']*attr_len
            j =j + 1
    print("S=",S)
    if len(temp)==0:
        print("G=",G)
    else:
        print("G=",temp)
    print('----------------------------------')
```

Iteration-4

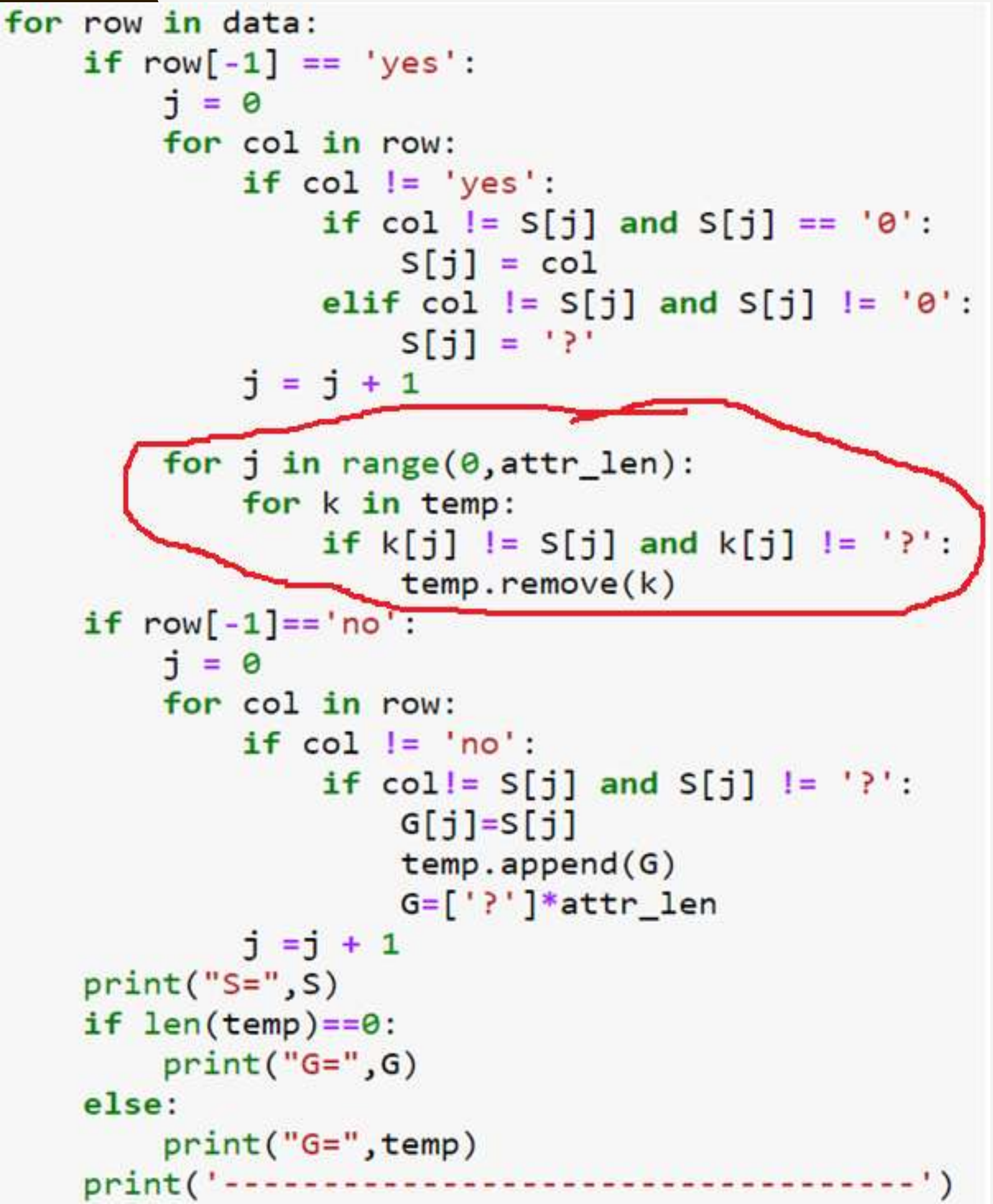
S=['Sunny','Warm','?','Strong','Warm','Same']

Sunny	Warm	High	Strong	Cool	Change	Yes

True

S[0]!='Sunny' and S[0]!='0'→F
S[1]!='Warm' and S[1]!='0' →F
S[2]!='High' and S[2]!='0' →T →S[2]=' ?'
S[3]!='Strong' and S[3]!='0' →F
S[4]!='Warm' and S[4]!='0' →T→S[4]='?'
S[5]!='Same' and S[5]!='0' → T→S[5]='?'

IMPLEMENTATION/PROGRAM



```
for row in data:
    if row[-1] == 'yes':
        j = 0
        for col in row:
            if col != 'yes':
                if col != S[j] and S[j] == '0':
                    S[j] = col
                elif col != S[j] and S[j] != '0':
                    S[j] = '?'
            j = j + 1

        for j in range(0,attr_len):
            for k in temp:
                if k[j] != S[j] and k[j] != '?':
                    temp.remove(k)
    if row[-1]=='no':
        j = 0
        for col in row:
            if col != 'no':
                if col!= S[j] and S[j] != '?':
                    G[j]=S[j]
                    temp.append(G)
                    G=['?']*attr_len
            j =j + 1
    print("S=",S)
    if len(temp)==0:
        print("G=",G)
    else:
        print("G=",temp)
    print('-----------------------------')
```

Iteration-4

S=['Sunny','Warm','?','Strong','Warm','Same']

Sunny	Warm	High	Strong	Cool	Change	Yes

S=[Sunny, Warm, ?, Strong, ?, ?]

j=(0,5)→

j=0

k in temp=[['Sunny','?', '?', '?','?', '?'], ['?','Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?','Same']]

iter1→k=['Sunny','?', '?', '?', '?', '?']

if k[0]!=s[0] and k[0]!='?'→'Sunny'!='Sunny' and 'Sunny'!='?' →F & T→F

iter2→k=['?','Warm', '?', '?', '?', '?']

if k[0]!=s[0] and k[0]!='?'→'?'!='Sunny' and '?'!='?' →T&F→F

iter3→k=['?','?', '?', '?', '?', 'Same']

if k[0]!=s[0] and k[0]!='?'→'?'!= 'Sunny' and '?'!='?' →T&F→F

j=1

k in temp=[['Sunny','?', '?', '?','?', '?'], ['?','Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?','Same']]

iter1→k=['Sunny','?', '?', '?', '?', '?']

if k[1]!=s[1] and k[1]!='?'→'?'!='Warm' and '?'!='?' →T & F→F

iter2→k=['?','Warm', '?', '?', '?', '?']

if k[1]!=s[1] and k[1]!='?'→'Warm'!='Warm' and 'Warm'!='?'→F&T→F

iter3→k=['?','?', '?', '?', '?', 'Same']

if k[1]!=s[1] and k[1]!='?'→'?'!= 'Sunny' and '?'!='?' →F&T→F

j=2

k in temp=[['Sunny','?', '?', '?','?', '?'], ['?','Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?','Same']]

iter1→k=['Sunny','?', '?', '?', '?', '?']

if k[2]!=s[2] and k[2]!='?'→'?'!='?'and '?'!='?' →F&F→F

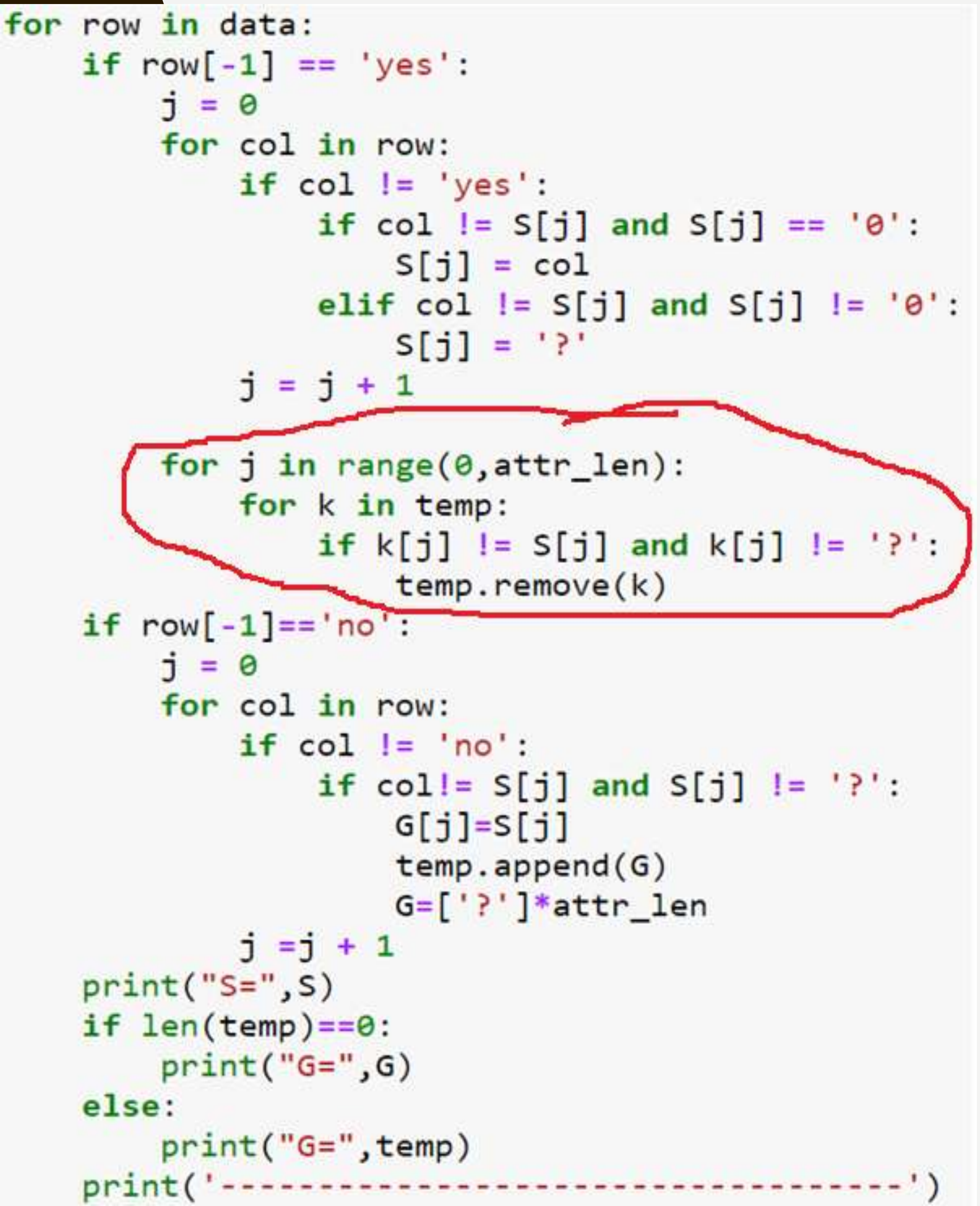
iter2→k=['?','Warm', '?', '?', '?', '?']

if k[2]!=s[2] and k[2]!='?'→'?'!='?' and '?'!='?'→F&F→F

iter3→k=['?','?', '?', '?', '?', 'Same']

if k[2]!=s[2] and k[2]!='?'→'?'!= '?'and '?'!='?'→F&F→F

IMPLEMENTATION/PROGRAM



```
for row in data:
    if row[-1] == 'yes':
        j = 0
        for col in row:
            if col != 'yes':
                if col != S[j] and S[j] == '0':
                    S[j] = col
                elif col != S[j] and S[j] != '0':
                    S[j] = '?'
            j = j + 1

        for j in range(0,attr_len):
            for k in temp:
                if k[j] != S[j] and k[j] != '?':
                    temp.remove(k)
    if row[-1]=='no':
        j = 0
        for col in row:
            if col != 'no':
                if col!= S[j] and S[j] != '?':
                    G[j]=S[j]
                    temp.append(G)
                    G=['?']*attr_len
            j =j + 1
    print("S=",S)
    if len(temp)==0:
        print("G=",G)
    else:
        print("G=",temp)
    print('-----------------------------------')
```

j=3

k in temp=[[‘Sunny’,’?’, ’?’, ’?’,’?’, ’?’], [‘?’,’Warm’, ’?’,’?’, ’?’, ’?’], [‘?’, ‘?’, ‘?’, ‘?’, ’?’,’Same’]]

iter1→k=[‘Sunny’,’?’, ’?’, ’?’, ’?’, ’?’]

if k[3]!=s[3] and k[3]!=’?’→’?’!=’Strong’and ‘?’!=’?’ →F&F→F

iter2→k=[‘?’,’Warm’, ’?’, ’?’, ’?’, ’?’]

if k[3]!=s[3] and k[3]!=’?’→’?’!=’ Strong’ and ‘?’!=’?’→F&F→F

iter3→k=[‘?’,’?’, ’?’, ’?’, ’?’, ’Same’]

if k[3]!=s[3] and k[3]!=’?’→’?’!= ’ Strong’and ‘?’!=’?’→F&F→F

j=4

k in temp=[[‘Sunny’,’?’, ’?’, ’?’,’?’, ’?’], [‘?’,’Warm’, ’?’,’?’, ’?’, ’?’], [‘?’, ‘?’, ‘?’, ‘?’, ’?’,’Same’]]

iter1→k=[‘Sunny’,’?’, ’?’, ’?’, ’?’, ’?’]

if k[4]!=s[4] and k[4]!=’?’→’?’!=’?’and ‘?’!=’?’ →F&F→F

iter2→k=[‘?’,’Warm’, ’?’, ’?’, ’?’, ’?’]

if k[4]!=s[4] and k[4]!=’?’→’?’!=’?’ and ‘?’!=’?’→F&F→F

iter3→k=[‘?’,’?’, ’?’, ’?’, ’?’, ’Same’]

if k[4]!=s[4] and k[4]!=’?’→’?’!= ’?’and ‘?’!=’?’→F&F→F

j=5

k in temp=[[‘Sunny’,’?’, ’?’, ’?’,’?’, ’?’], [‘?’,’Warm’, ’?’,’?’, ’?’, ’?’], [‘?’, ‘?’, ‘?’, ‘?’, ’?’,’Same’]]

iter1→k=[‘Sunny’,’?’, ’?’, ’?’, ’?’, ’?’]

if k[5]!=s[5] and k[5]!=’?’→’?’!=’?’and ‘Same’!=’?’ →F&F→F

iter2→k=[‘?’,’Warm’, ’?’, ’?’, ’?’, ’?’]

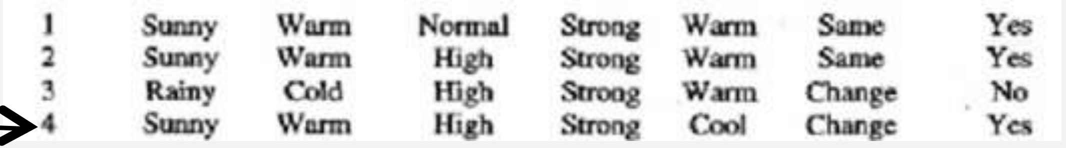
if k[5]!=s[5] and k[5]!=’ ?’→’?’!=’?’ and ‘Same’!=’?’→F&F→F

iter3→k=[‘?’,’?’, ’?’, ’?’, ’?’, ’Same’]

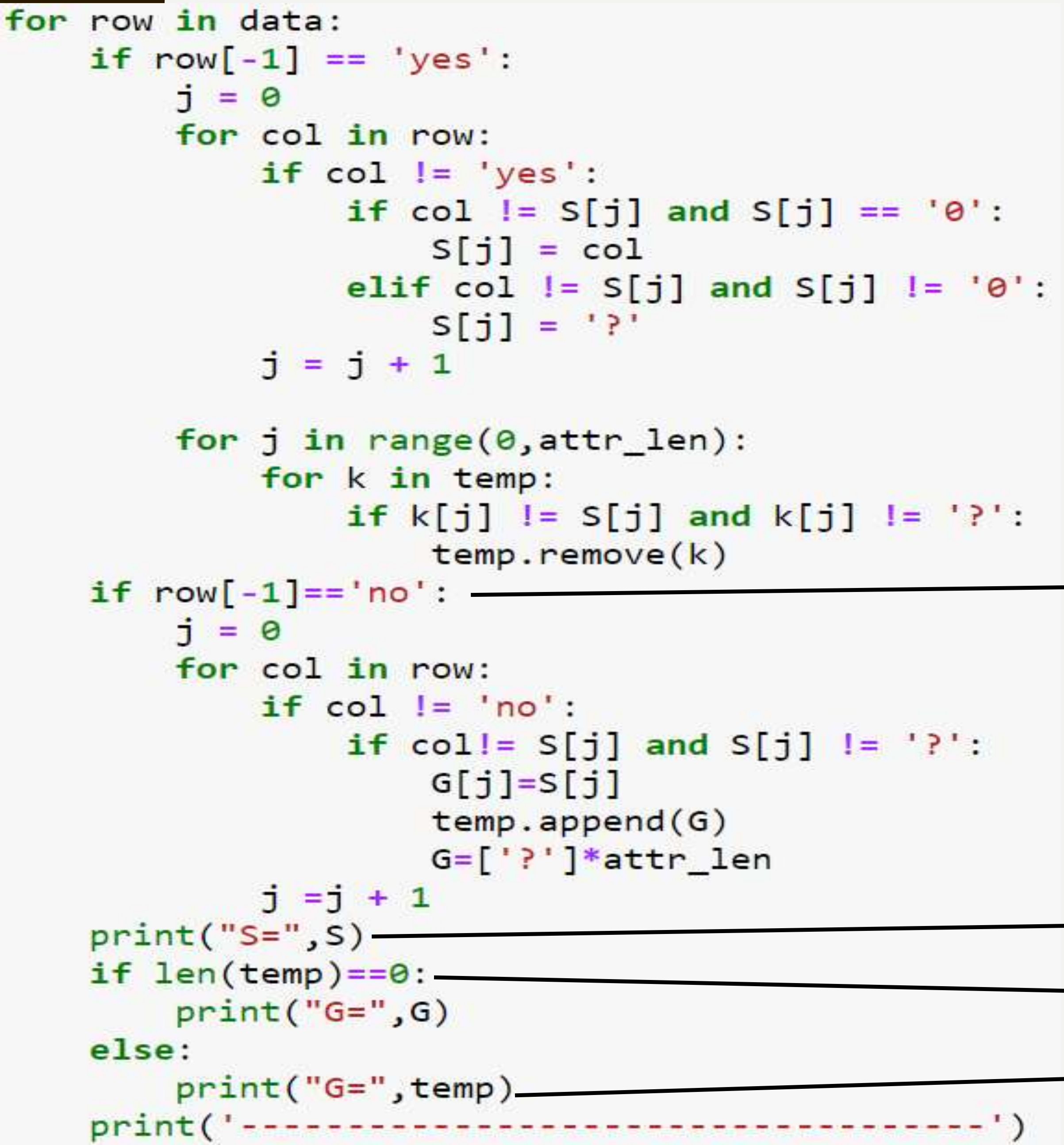
if k[5]!=s[5] and k[5]!=’?’→’ Same’!= ’?’and ‘Same’!=’?’→T&T→True

temp.remove(k)→remove([‘?’, ‘?’, ‘?’, ‘?’, ‘?’,’Same’])

IMPLEMENTATION/PROGRAM



1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes



```
for row in data:
    if row[-1] == 'yes':
        j = 0
        for col in row:
            if col != 'yes':
                if col != S[j] and S[j] == '0':
                    S[j] = col
                elif col != S[j] and S[j] != '0':
                    S[j] = '?'
            j = j + 1

        for j in range(0,attr_len):
            for k in temp:
                if k[j] != S[j] and k[j] != '?':
                    temp.remove(k)
    if row[-1]=='no':
        j = 0
        for col in row:
            if col != 'no':
                if col!= S[j] and S[j] != '?':
                    G[j]=S[j]
                    temp.append(G)
                    G=['?']*attr_len
            j =j + 1
    print("S=",S)
    if len(temp)==0:
        print("G=",G)
    else:
        print("G=",temp)
    print('---------------------------------')
```

False

S=['Sunny','Warm','?','Strong','?','?']

len(temp)==0→False →G=temp

G=[['Sunny','?', '?', '?', '?', '?'], ['?','Warm', '?', '?', '?', '?']]

REMARKS ON VERSION SPACES AND CANDIDATE-ELIMINATION

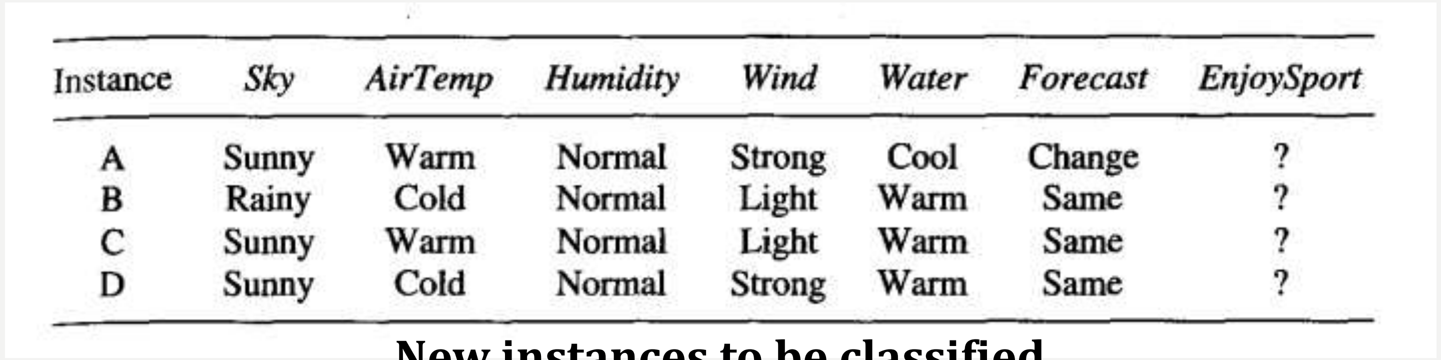
1. **Will the CANDIDATE-ELIMINATION algorithm converge to the correct hypothesis?**
   - The version space learned by the candidate-elimination algorithm will converge toward the hypothesis that correctly describes the target concept, provided

     (a) there are no errors in the training examples, and

     (b) there is some hypothesis in H that correctly describes the target concept.
2. **What Training Example Should the Learner Request Next?**
   - Examples provided by external
   - Take an example on learner's own
   - Version space size is reduced with each new training example.
   - $\log_2|VS|$ training examples are required to get correct target concept if by each example version space size is reduced by half.
   - Otherwise more than $\log_2|VS|$ training examples are required.

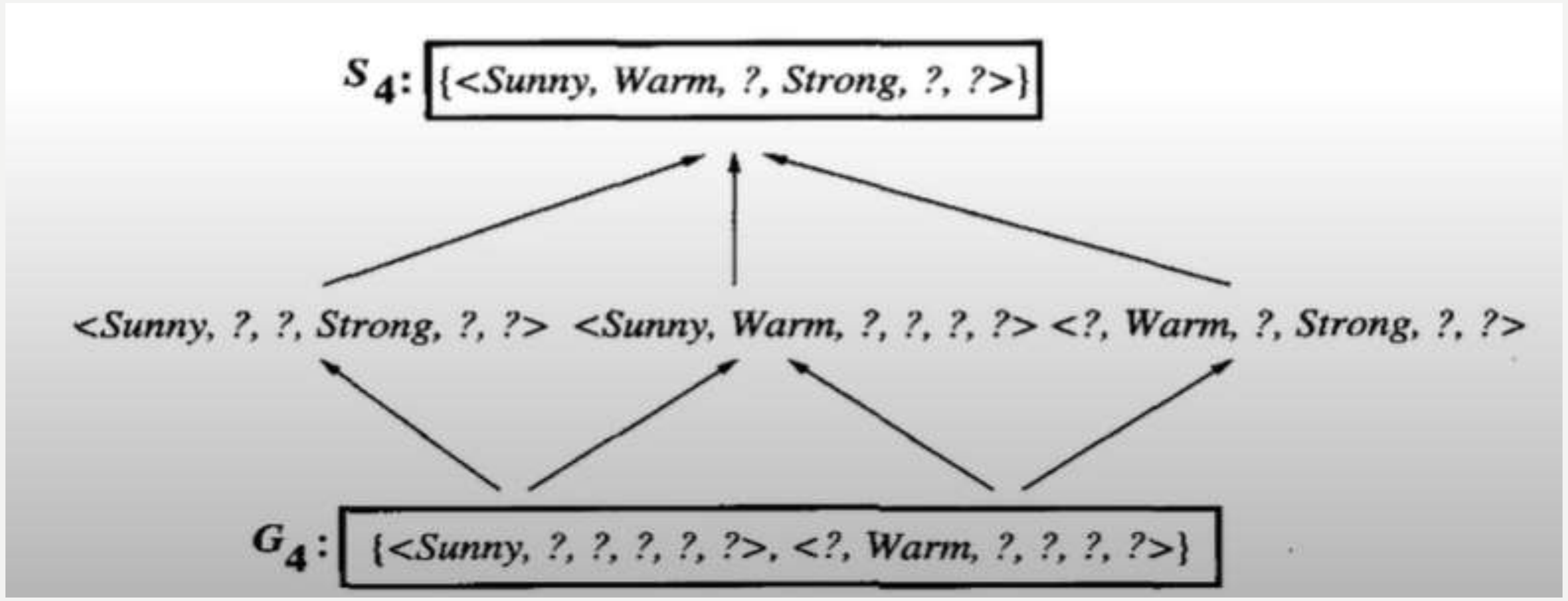
REMARKS ON VERSION SPACES AND CANDIDATE-ELIMINATION

3. **How Can Partially Learned Concepts Be Used?**



Instance	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
A	Sunny	Warm	Normal	Strong	Cool	Change	?
B	Rainy	Cold	Normal	Light	Warm	Same	?
C	Sunny	Warm	Normal	Light	Warm	Same	?
D	Sunny	Cold	Normal	Strong	Warm	Same	?

New instances to be classified.



INDUCTIVE BIAS

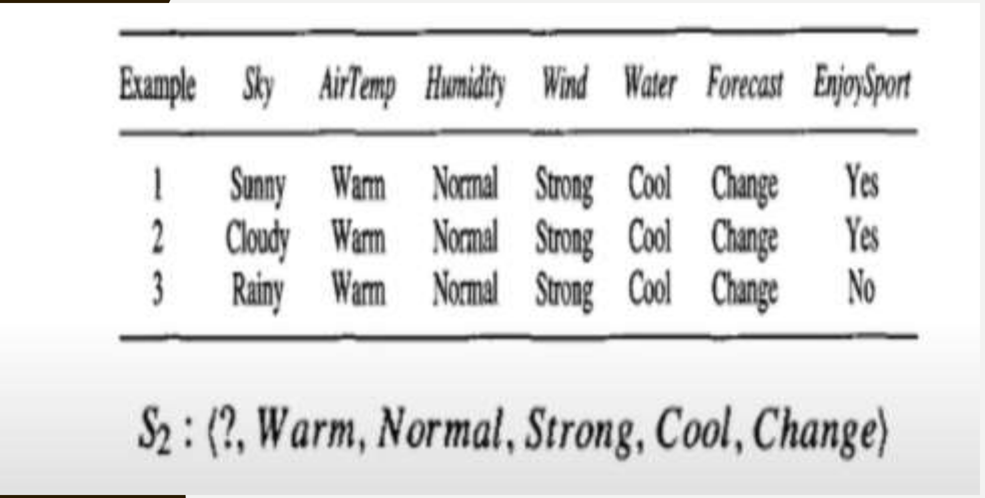
The fundamental questions for inductive inference :

1. What if the target concept is not contained in the hypothesis space?
2. Can we avoid this difficulty by using a hypothesis space that includes every possible hypothesis?
3. How does the size of this hypothesis space influence the ability of the algorithm to generalize to unobserved instances?
4. How does the size of the hypothesis space influence the number of training examples that must be observed?

- These fundamental questions are examined in the context of the **CANDIDATE- ELIMINTION** algorithm.
- As we shall see, though, the conclusions we draw from this analysis will apply to ***any concept learning system that outputs any hypothesis consistent with the training data.***

A BIASED HYPOTHESIS SPACE

- **Suppose the target concept c(x) is not contained in the hypothesis space H**, then none of the hypothesis of H will be consistent with a set of training examples D.
- The obvious solution is to enrich the hypothesis space to include every possible hypothesis.
- To illustrate, consider again the ***EnjoySport*** example in which we restricted the hypothesis space to include only conjunctions of attribute values.
- Because of this restriction, the hypothesis space is unable to represent even simple disjunctive target concepts such as "Sky = Sunny or Sky = Cloudy".



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Cool	Change	Yes
2	Cloudy	Warm	Normal	Strong	Cool	Change	Yes
3	Rainy	Warm	Normal	Strong	Cool	Change	No

$S_2 : \langle ?, Warm, Normal, Strong, Cool, Change \rangle$

- This hypothesis, although it is the maximally specific hypothesis from H that is consistent with the first two examples, is already overly general.
- It incorrectly covers the third (negative) training example.
- The problem is that we have biased the learner to consider only conjunctive hypotheses. In this case we require a more expressive hypothesis space.

AN UNBIASED LEARNER

- The obvious solution to the problem of assuring that the target concept is in the hypothesis space H is to provide a hypothesis space capable of representing every teachable concept.
    - Every possible subset of the instances X → ***the power set of X.***
- What is the size of the hypothesis space H (the power set of X) ?
    - In ***EnjoySport***, the size of the instance space X is 96.
    - The size of the power set of X is 2^X → The size of H is 2^{96}
    - Our conjunctive hypothesis space is able to represent only 973 of these hypotheses.

        →a very biased hypothesis space

AN UNBIASED LEARNER: PROBLEM

- Let the hypothesis space H to be the power set of X.
  - A hypothesis can be represented with disjunctions, conjunctions, and negations of our earlier hypotheses.
  - The target concept "Sky = Sunny or Sky = Cloudy" could then be described as

<Sunny, ?, ?, ?, ?, ?> V <Cloudy, ?, ?, ?, ?, ?>

NEW PROBLEM:

- **Our concept learning algorithm is now completely unable to generalize beyond the observed examples.**
  - Three positive examples (xl,x2,x3) and two negative examples (x4,x5) to the learner.
  - **S :** { x1 V x2 V x3 } and **G :** {⌐(x4 V x5) } →**NO GENERALIZATION**
  - Therefore, the only examples that will be unambiguously classified by **S** and **G** are the observed training examples themselves.

THE FUTILITY OF BIAS-FREE LEARNING

- The fundamental property of inductive inference —

  ***"a learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen* instances"**

- Candidate elimination algorithm was able to generalize beyond the observed training examples in our original formulation of the EnjoySport task is that it was biased by the implicit assumption that the target concept could be represented by a conjunction of attribute values.

- In cases where this assumption is correct (and the training examples are error-free), its classification of new instances will also be correct.

- If this assumption is incorrect, however, it is certain that the candidate elimination algorithm will miss-classify at least some instances from X.

THE IDEA OF INDUCTIVE BIAS

- The key idea we wish to capture here is the policy by which the learner generalizes beyond the observed training data, to infer the classification of new instances.

- Need to make assumptions
  - Experience alone doesn't allow us to make conclusions about unseen data instances.

- 2 types of bias:
  - **Restriction** : limit the hypothesis space
  - **Preference:** impose ordering on hypothesis space.

- A **preference bias** is more desirable than a **restriction bias**, because it allows the learner to work within a complete hypothesis space that is assured to contain the unknown target function.

INDUCTIVE BIAS

- The **inductive bias** (also known as learning **bias**) of a learning algorithm is the set of assumptions that the learner uses to predict outputs of given inputs that it has not encountered.

- In machine learning, the term **inductive bias** refers to a set of (explicit or implicit) assumptions made by a learning algorithm in order to perform induction, that is, to generalize a finite set of observation (training data) into a general model of the domain.

- Inductive reasoning is the process of learning general principles on the basis of specific instances – in other words, it's what any machine learning algorithm does when it produces a prediction for *any* unseen test instance on the basis of a *finite* number of training instances. Inductive bias describes the tendency for a system to prefer a certain set of generalizations over others that are equally consistent with the observed data.

- Without inductive bias, a learner can't generalize from observed examples to new examples better than random guessing.

INDUCTIVE BIAS

•Consider a concept learning algorithm L for the set of instances X. Let c be an arbitrary concept defined over X, and let $D_c = \{<x,c(x)>\}$ be an arbitrary set of training examples of c.

•After training, L is asked to classify a new instance x_i.

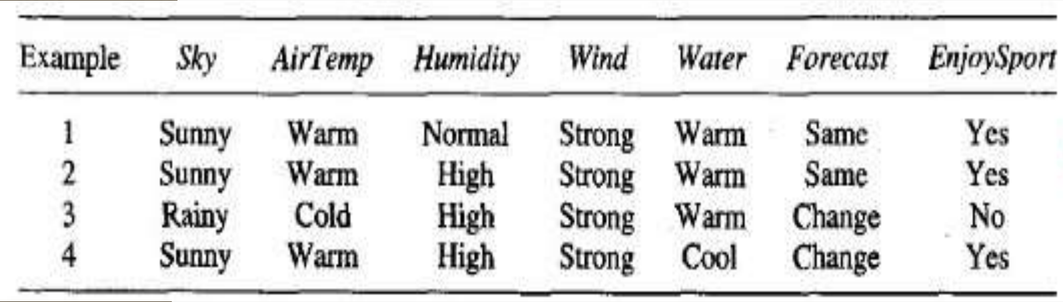
•Let $L(x_i, D_c)$ denote the classification (e.g., positive or negative) that L assigns to x_i after learning from the training data D_c.

•We can describe this inductive inference step performed by L as :

$$(D_c \wedge x_i) \succ L(x_i \wedge D_c)$$

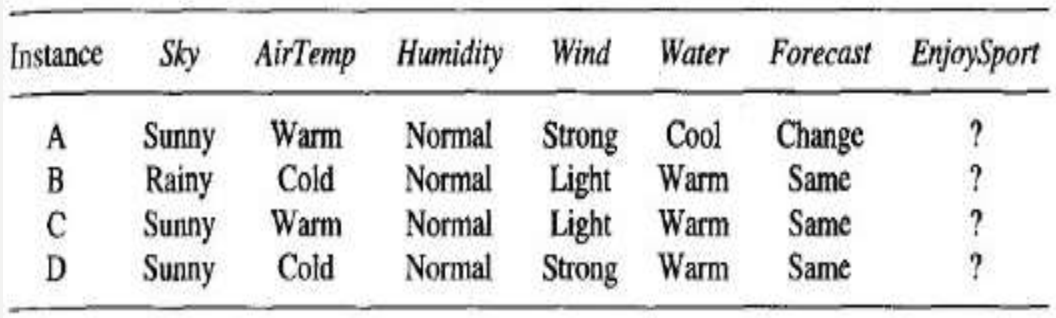
•The symbol $\succ$ is called "inductively inferred from"

•For example, if we take L to be the candidate elimination algorithm, Dc, to be the training data as below, and x_i to be the fist instance from test data as below, then the inductive inference performed in this case concludes that $L(x_i, Dc)$ =(EnjoySport = yes).



Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Training Data D



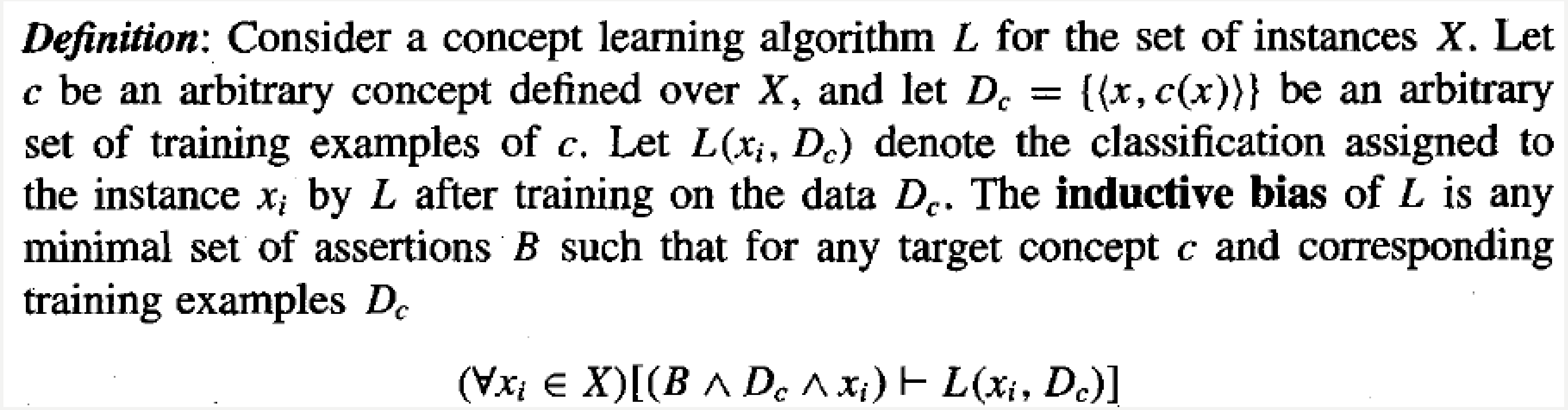
Instance	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
A	Sunny	Warm	Normal	Strong	Cool	Change	?
B	Rainy	Cold	Normal	Light	Warm	Same	?
C	Sunny	Warm	Normal	Light	Warm	Same	?
D	Sunny	Cold	Normal	Strong	Warm	Same	?

Test Data

INDUCTIVE BIAS-DEFINITION

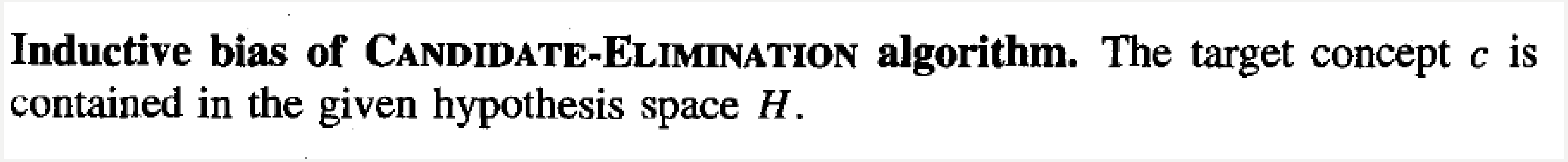
- We define inductive bias of L to be the set of assumptions **B** such that for all new instances x_i $(\forall x_i \in X)\ [(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$
- $\vdash$ this symbol is called as "follows deductively"
- Thus, we define the inductive bias of a learner as the set of additional assumptions B sufficient to justify its inductive inferences as deductive inferences.

INDUCTIVE BIAS-DEFINITION



Definition: Consider a concept learning algorithm L for the set of instances X. Let c be an arbitrary concept defined over X, and let $D_c = \{\langle x, c(x)\rangle\}$ be an arbitrary set of training examples of c. Let $L(x_i, D_c)$ denote the classification assigned to the instance x_i by L after training on the data D_c. The **inductive bias** of L is any minimal set of assertions B such that for any target concept c and corresponding training examples D_c

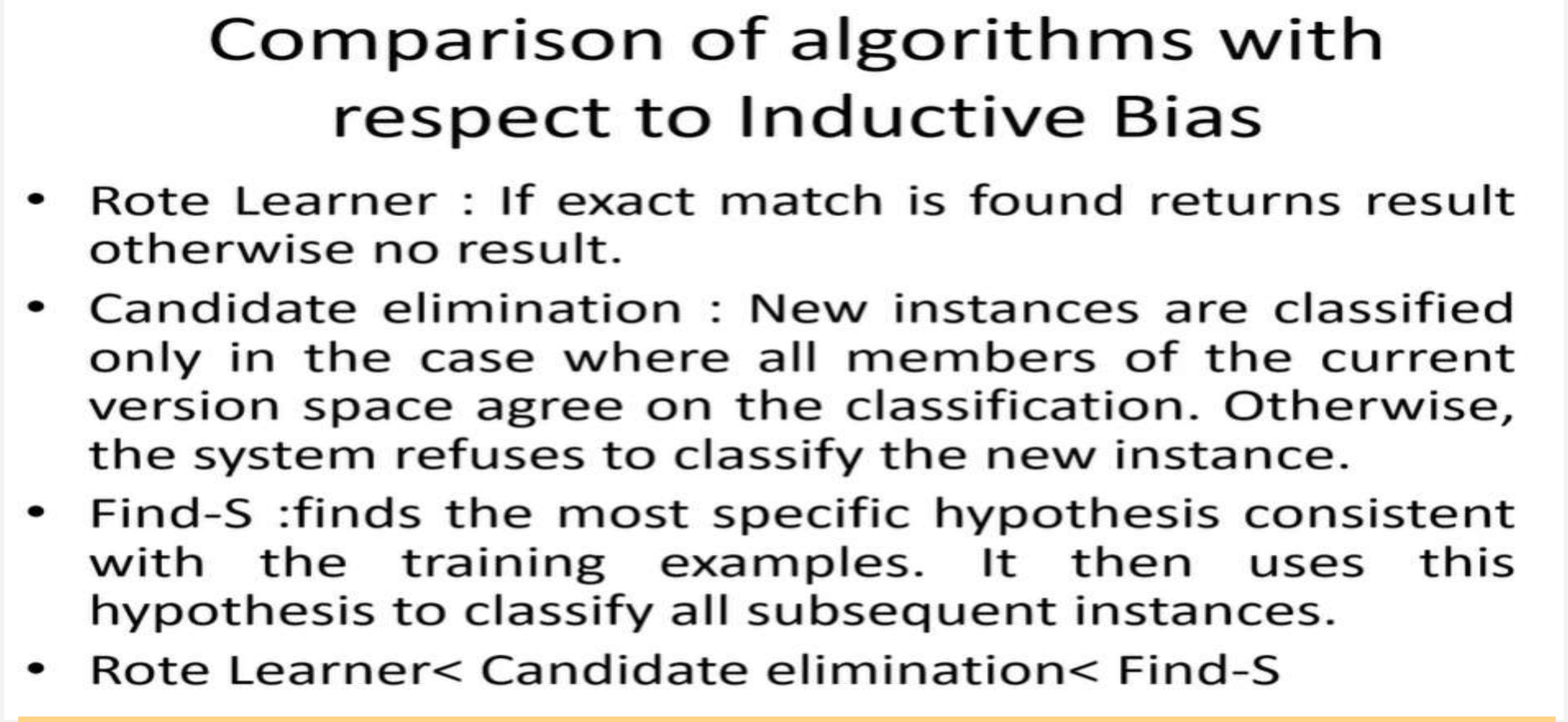
$$(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)]$$



Inductive bias of CANDIDATE-ELIMINATION algorithm. The target concept c is contained in the given hypothesis space H.

INDUCTIVE BIAS-ADVANTAGE

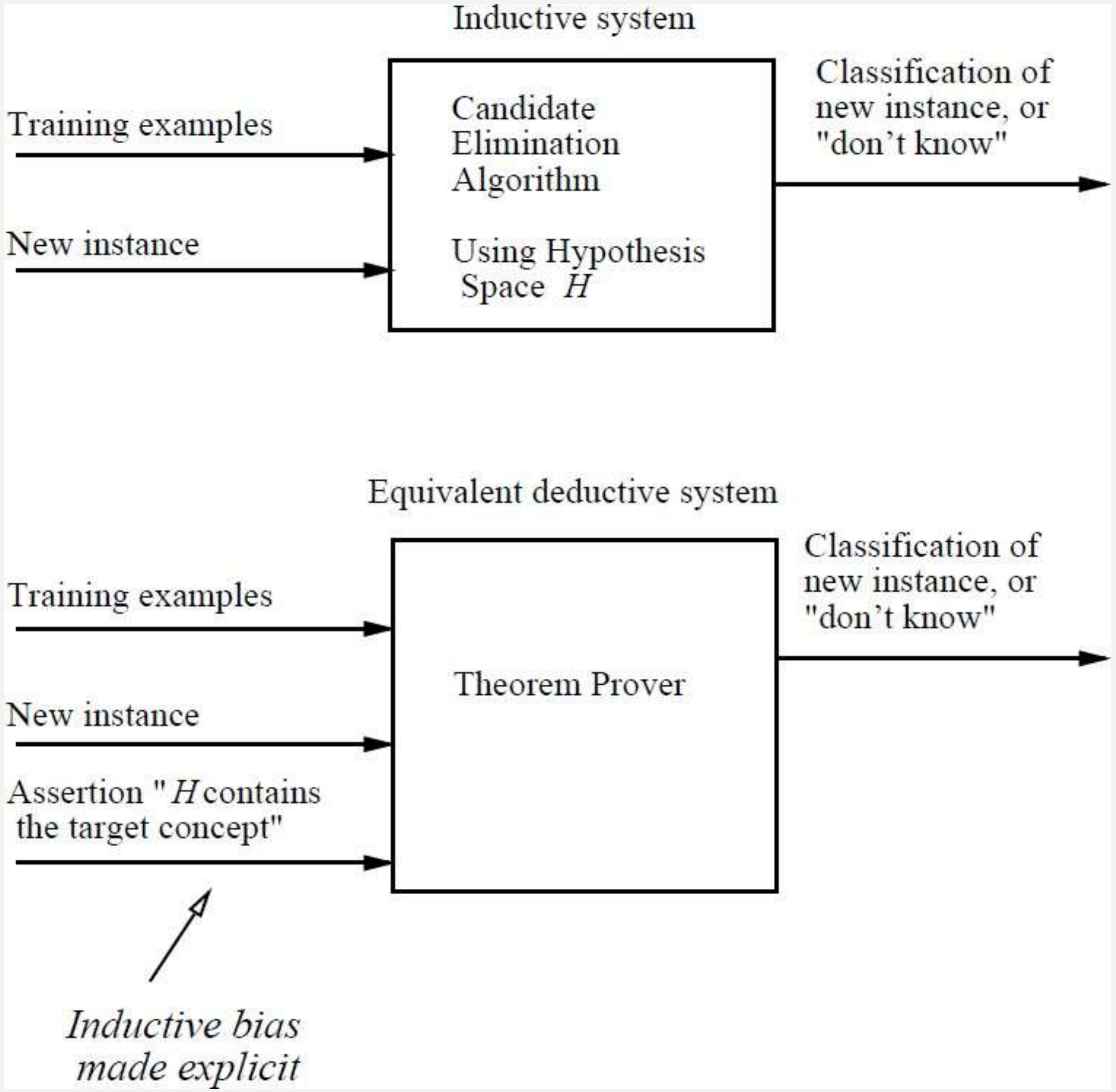
- It provides a nonprocedural means of characterizing their policy for generalizing beyond the observed data.
- It allows comparison of different learners according to the strength of the inductive bias they employ.



Comparison of algorithms with respect to Inductive Bias

- Rote Learner : If exact match is found returns result otherwise no result.
- Candidate elimination : New instances are classified only in the case where all members of the current version space agree on the classification. Otherwise, the system refuses to classify the new instance.
- Find-S :finds the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances.
- Rote Learner< Candidate elimination< Find-S

Algorithm	**Inductive Bias**
Rote-Learner	None
Candidate-Elimination	The target concept c is contained in the **hypothesis** space H.
Find-S	The target concept can be described in its **hypothesis** space. All instances are negative instances unless demonstrated otherwise.


Inductive system
Training examples
New instance
Candidate Elimination Algorithm
Using Hypothesis Space H
Classification of new instance, or "don't know"
Equivalent deductive system
Training examples
New instance
Assertion "H contains the target concept"
Theorem Prover
Classification of new instance, or "don't know"
Inductive bias made explicit

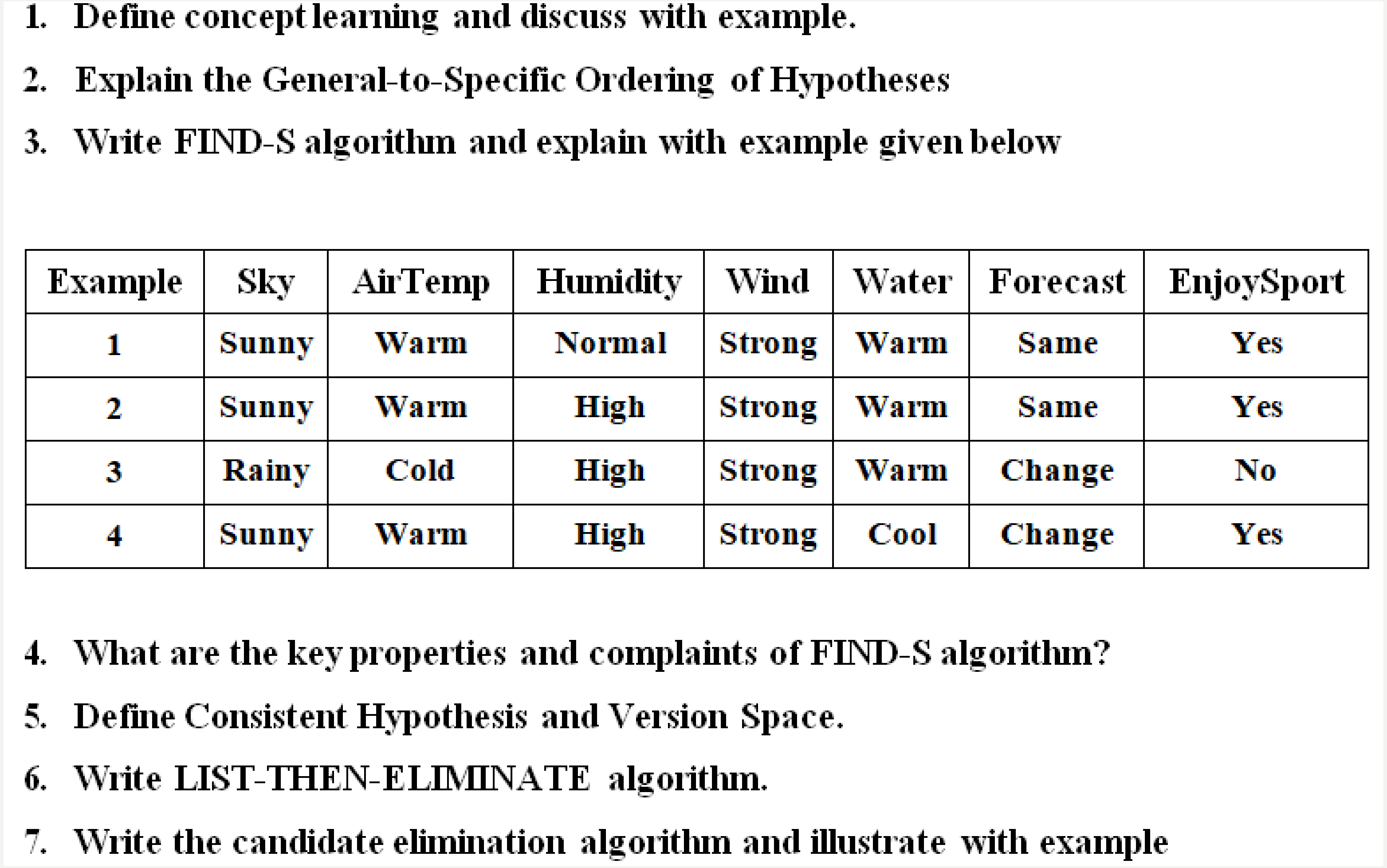
Modeling inductive systems by Equivalent deductive systems.

- The input-output behavior of the CANDIDATE-ELIMINATION algorithm using a hypothesis space H is identical to that of a deductive theorem prover utilizing the assertion "H contains the target concept." This assertion is therefore called the inductive bias of the CANDIDATE-ELIMINATION algorithm.
- Characterizing inductive systems by their inductive bias allows modelling them by their equivalent deductive systems. This provides a way to compare inductive systems according to their policies for generalizing beyond the observed training data.

SUMMARY

- Concept
- Concept Learning
- Concept Learning Task
- Inductive Learning
- Concept Learning as a Search
- Find-S: Finding a maximally specific Hypothesis
- List-Then-Eliminate algorithm
- General Boundary, Specific Boundary
- Version Spaces and the CANDIDATE-ELIMINATION
- Inductive Bias

QUESTIONS

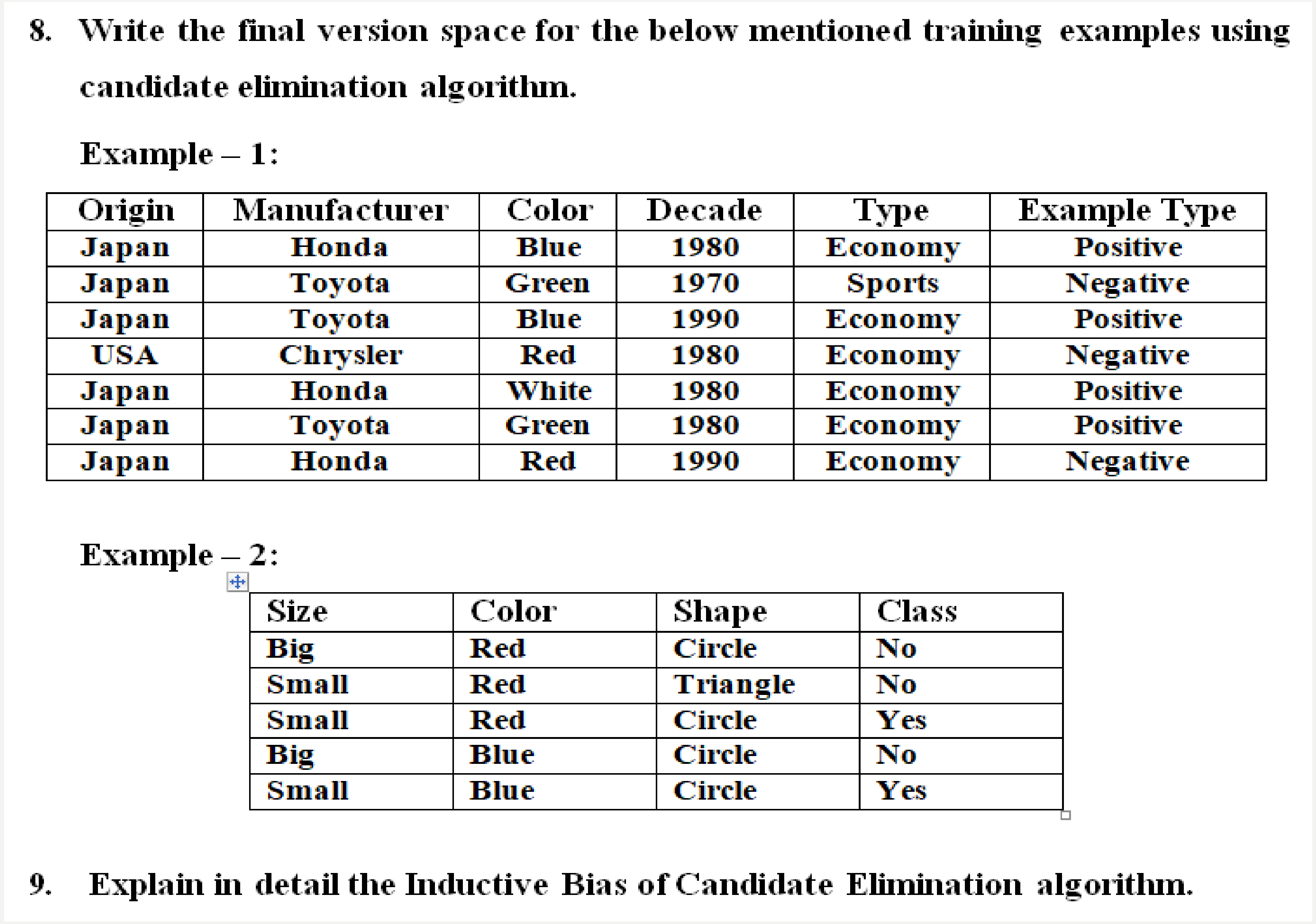


1. Define concept learning and discuss with example.
2. Explain the General-to-Specific Ordering of Hypotheses
3. Write FIND-S algorithm and explain with example given below

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

4. What are the key properties and complaints of FIND-S algorithm?
5. Define Consistent Hypothesis and Version Space.
6. Write LIST-THEN-ELIMINATE algorithm.
7. Write the candidate elimination algorithm and illustrate with example

QUESTIONS



8. Write the final version space for the below mentioned training examples using candidate elimination algorithm.

Example – 1:

Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive
Japan	Toyota	Green	1980	Economy	Positive
Japan	Honda	Red	1990	Economy	Negative

Example – 2:

Size	Color	Shape	Class
Big	Red	Circle	No
Small	Red	Triangle	No
Small	Red	Circle	Yes
Big	Blue	Circle	No
Small	Blue	Circle	Yes

9. Explain in detail the Inductive Bias of Candidate Elimination algorithm.