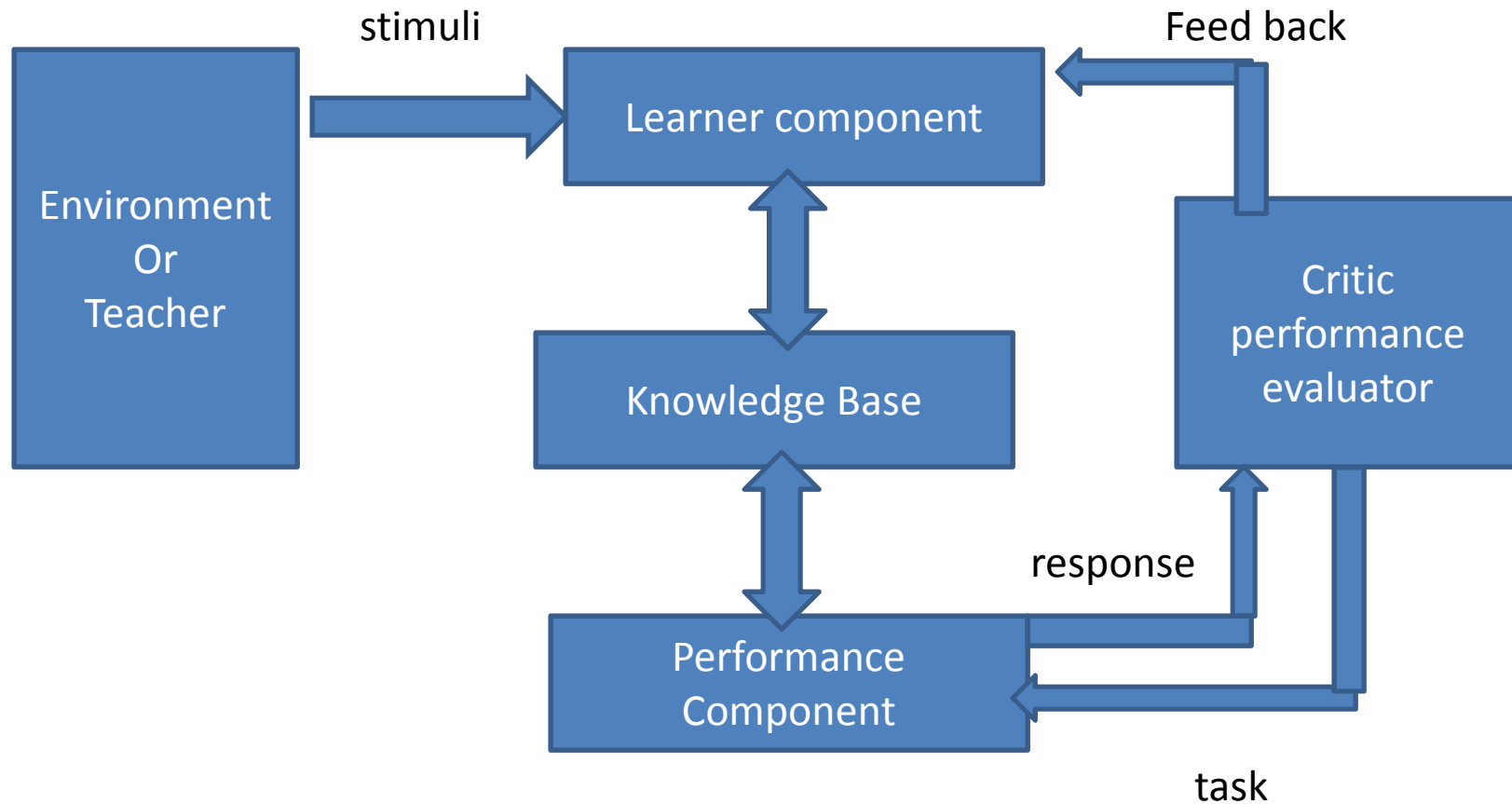


# Learning

# learning

- Machine cannot be called intelligent unless they learn new things and adapt to changes.
- Learning covers wide range of phenomenon
  - *Skill refinement* : People get better at task simply by practicing
  - *Knowledge acquisition*: knowledge is source of power in AI and is acquired through experience

# Model of learning



This cycle may be repeated number of times until the performance of the system has reached acceptable level.

# Components of learning

- Environment: form of nature that produces random stimuli
  - Example
    - Input from keyboard
    - Sensors
- Learner component
  - input in form of stimuli is taken and this information is used to create a new knowledge or modify the existing one

# Components of learning

- Performance component
  - It uses the knowledge structure to carry out the task like game playing, classification problem
  - It produces the response describing the action
- Critic evaluation
  - Evaluates the response produced by performance component relative to optimal response
- Feed back :
  - is sent by Critic module to learner component indicating whether performance was acceptable

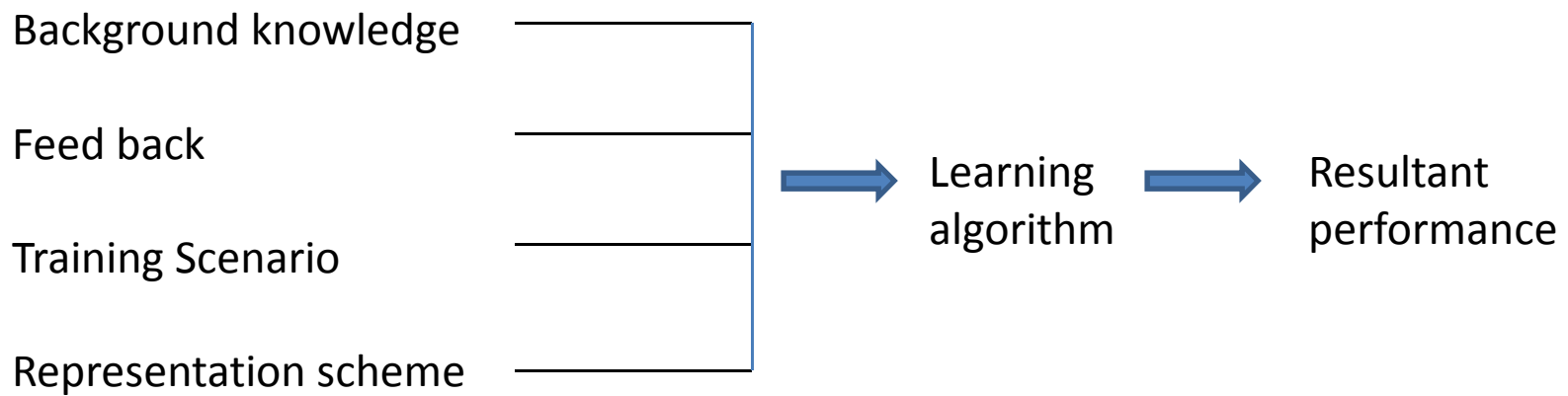
# Factors effecting learning

- Training Scenarios:
  - Sample may be positive or negative. System get affected by training sample sequence.
- Background knowledge
  - As learning is also a search through possible hypothesis. One may reduce this search space if proper background knowledge is available
- Feedback
  - It allows system to know whether change in knowledge structure is improving learning system or not

# Factors effecting learning

- Representation scheme
  - Representation scheme of environment and learning system may be same or different. If different conversions are required.
- Learning algorithm
  - Must be efficient to control search and build knowledge structure. They consider training examples and background knowledge for better performance

# Factors effecting learning





# Performance measure

- Generality:
  - Ease with which method can be adapted to different domains of applications, More general algorithm is the one which can adapt in any environment
- Efficiency:
  - Measure of average time required to construct the target knowledge structures from some initial structures

# Performance measure

- Robustness
  - Ability of a system to work with unreliable feedback and variety of training program and samples
- Efficacy
  - Measure of overall power of the system
  - Generality + Efficiency+ Robustness

# Types of learning

- Rote learning
  - Store lots of information in form of DBMS or knowledge base
- learning by taking advice
- Learning by problem solving
- Learning by example (Induction)
- Explanation based learning

# Rote learning

- When computer stores a piece of information it is performing a rudimentary form of learning
- It allows program to perform better in future
- Learning by memorization ( data is cached)
- In data caching computed values are stored and these values are recalled when a same problem state occurs

# Rote learning

- It is simple and must involve some capabilities like
  - Organized storage of information:
    - To avoid recomputing, caching is used hence stored value must be retrieved as fast as possible so that cost of retrieving is less than cost of recomputing
    - Organized storage structure like : indexing and hashing is used
  - Generalization
    - Number of stored object is very large
    - To keep this number manageable some kind of generalization is needed

# Learning from example-Induction

- Classification problem is one of the example of learning by induction
- Classification is the process of assigning a particular input the name of the class to which it belongs
- Classes must be defines
  - Statistical information
  - Structural representation

# Learning from example-Induction

- Statistical method
  - Identify the feature of task domain and then define each class as weighted sum of these features
  - It will have the form  
 $c_1t_1 + c_2t_1 + \dots$  where  $c_i$  is the class and  $t_i$  is the weight
  - Example: weather prediction feature may be
    - Measurement of rainfall
    - Temperature
    - Location of cold areas
    - Different features are combined to predict the weather as cloudy or sunny

# Learning from example-Induction

- Structural method
  - Identify and isolate features of task domain and represent each class as set of structures
  - Idea of producing a classification program that can evolve its own class definition is called concept learning or **induction**
- Example
  - Winston's learning program



- Winston's described structural learning program
- Program operated on simple block world domain
- Goal was to construct representation of definition of concept in block domain
- It learned the concept *house, tent and arch*

# Winston's learning program







	Concept	Near Miss
House		
Tent		
Arch		

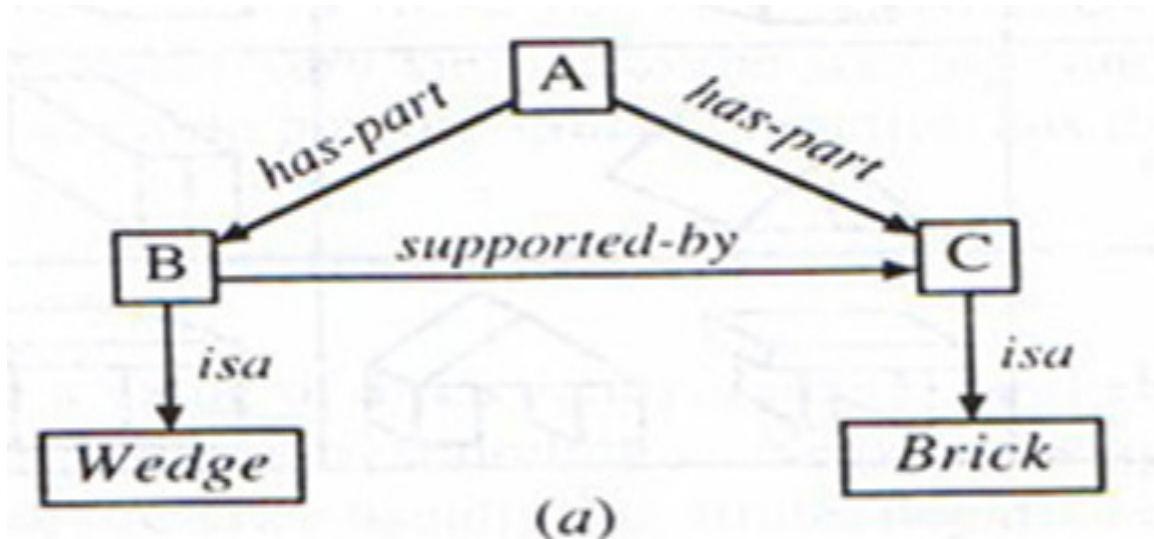
Figure 17.2: Some Blocks World Concepts

Near miss is an object that is not an instance of the concept but that is similar to such instances

# Winston's learning program

- Started with line drawing of block structure
- Analyzed the drawing and constructed semantic net representation of structural description of the object
- Structural description was then provided as an input to the learning program

# Structural description of *house*



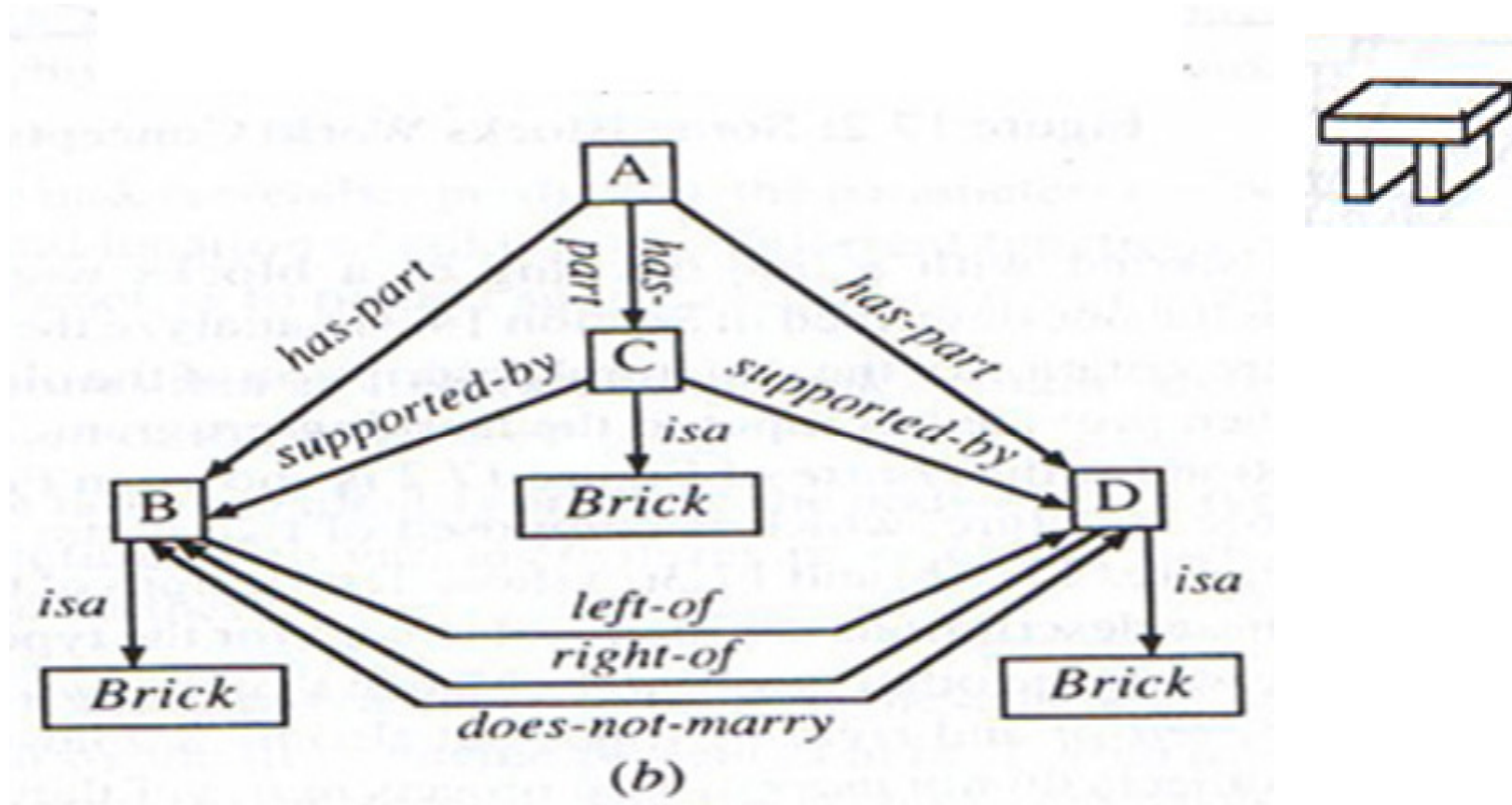
Node A- entire house structure and is composed of two parts

Node B- a wedge

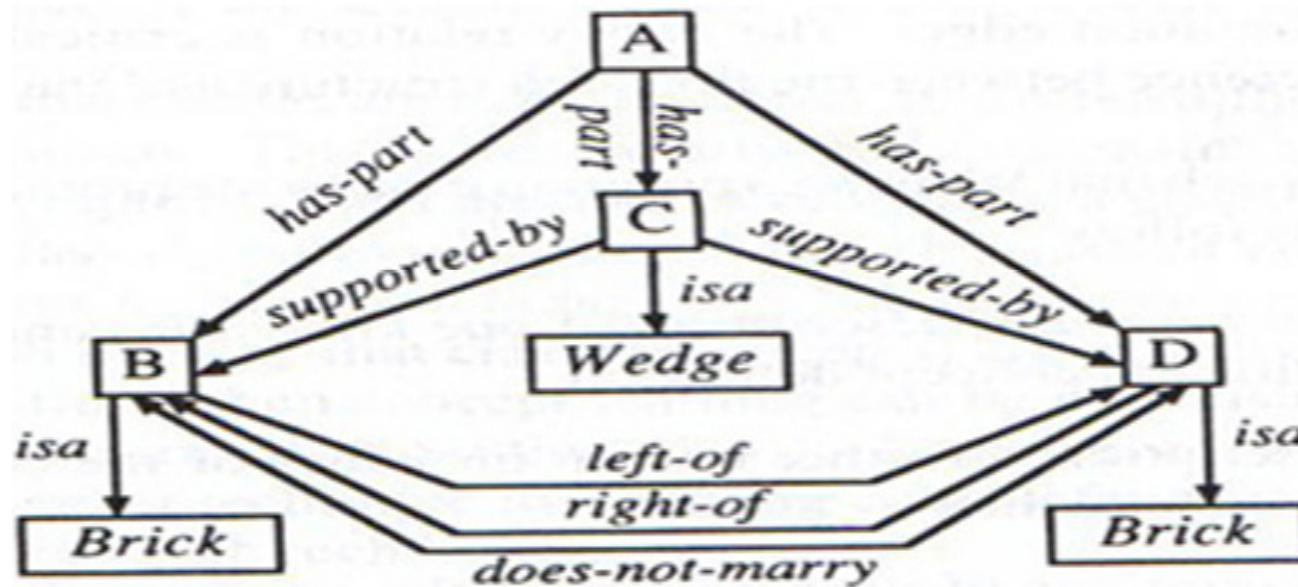
Node C -a Brick

Relations- left -of, right-of

# Structural description of *Arch-1*



# Structural description of *Arch-2*



# Algorithm

1. Begin with a structural description of one known instance of a concept. Call that description **concept definition**
2. Examine the description of other known instances of the concept **Generalize** the definition to include them
3. Examine the description of near miss .Restrict the definition to exclude them

Step 2 and 3 rely heavily on a comparison process by which similarities and differences can be detected

Output of the comparison process is a skeleton structure describing commonalities between the two input structures

It is annotated with a set of comparison note that describe specific similarities and difference between inputs

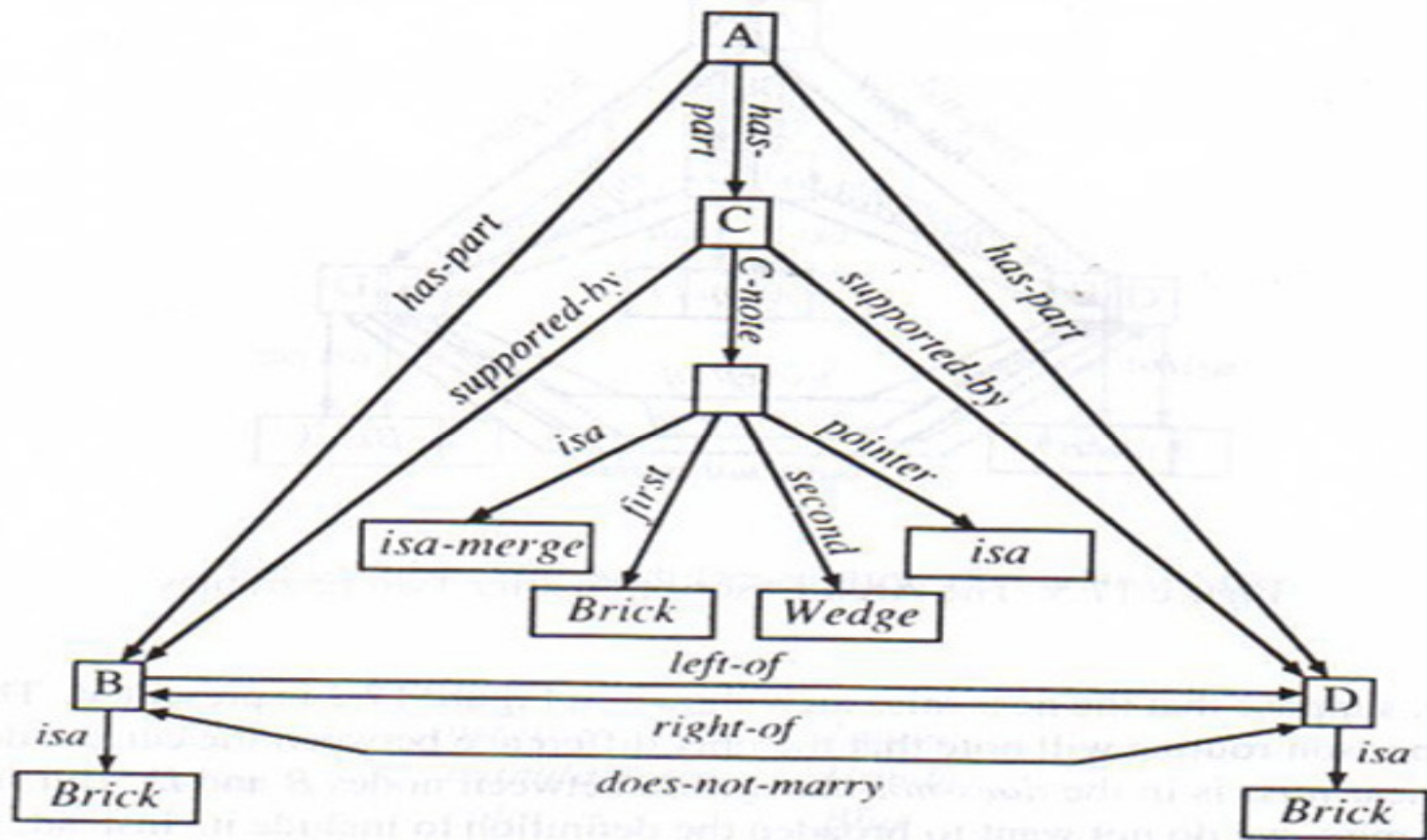


Figure 17.4: The Comparison of Two Arches



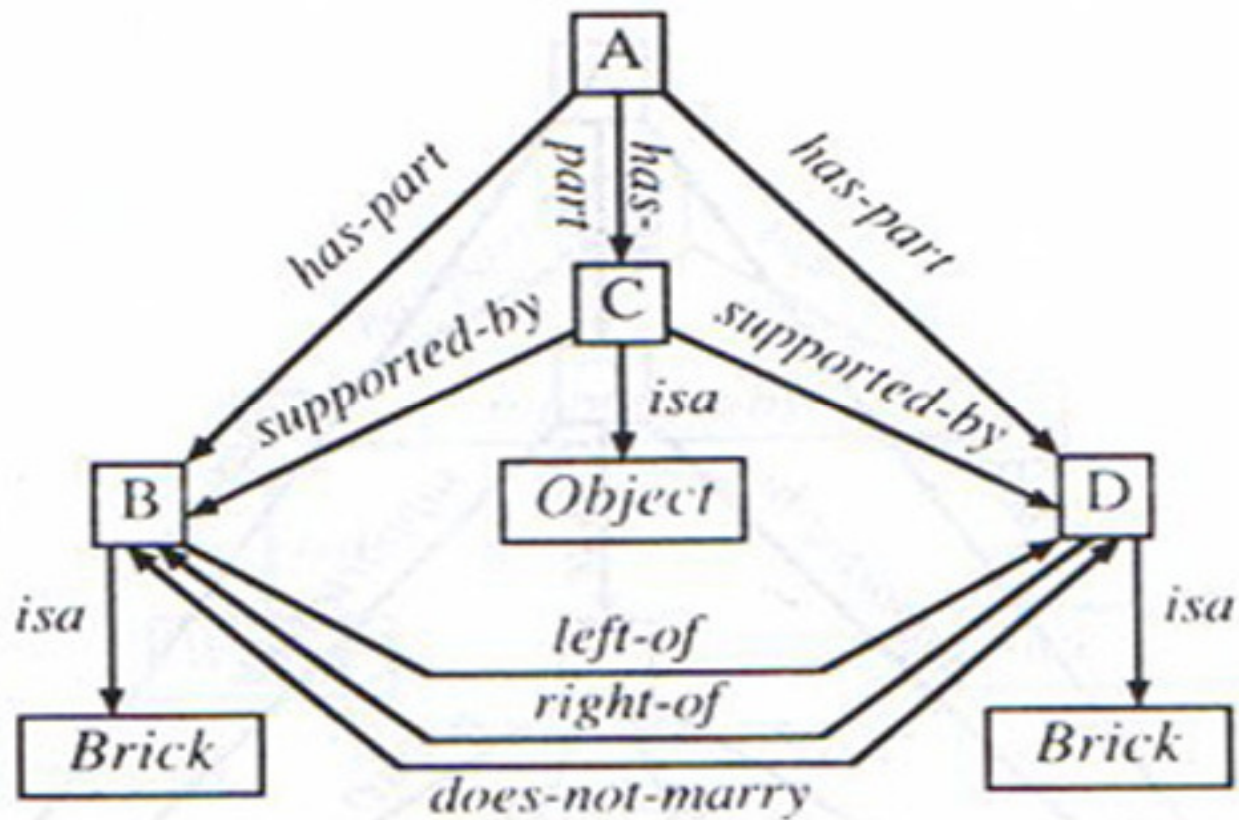


Figure 17.5: The Arch Description after Two Examples

# Version spaces

- Another approach to concept learning is version spaces
- Winston's evolved the concept by single concept description
- Version space does it by maintaining set of possible description and evolving that set as new example and near misses are presented
- Uses frame based language

# Version Spaces

The goal : to produce a description that is consistent with all positive examples but no negative examples in the training set.

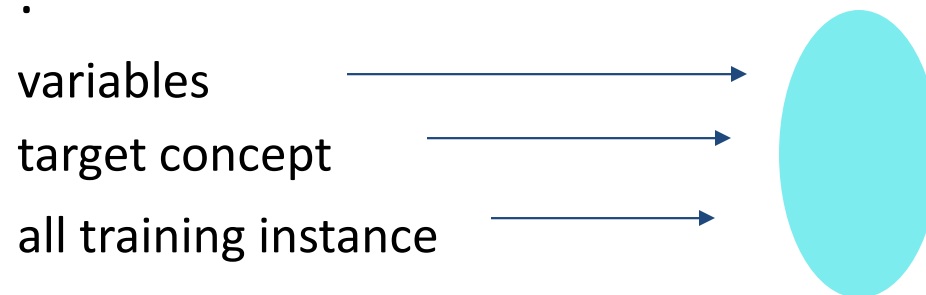
use frame representing concept for car

Features/Slots : { value1, value2,...,valueN }

origin : { Japan, USA, Britain }

Variables : X1, X2, X3

concept space :



# Version Spaces

<i>Car023</i>	
<i>origin :</i>	<i>Japan</i>
<i>manufacturer :</i>	<i>Honda</i>
<i>color :</i>	<i>Blue</i>
<i>decade :</i>	<i>1970</i>
<i>type :</i>	<i>Economy</i>

Figure 17.7: An Example of the Concept *Car*

# Version Spaces

- version space = current hypothesis = subset of **concept space** = largest collection of descriptions that is consistent with all the training examples seen so far.
- **concept space** =  $G$  or  $S$
- $G$  = contain the most general descriptions consistent with the training example seen so far.
- $S$  = contain the most specific descriptions consistent with training examples
- positive example (+)  $\rightarrow$  move  $S$  to more specific
- negative example (-)  $\rightarrow$  move  $G$  to more specific
- if  $G$  and  $S$  sets converge  $\rightarrow$  the hypothesis is a single concept description

# Version Spaces

## CANDIDATE ELIMINATE ALGORITHM

algorithm that use to narrow down the version space

by remove any descriptions that are inconsistent with set G and set S

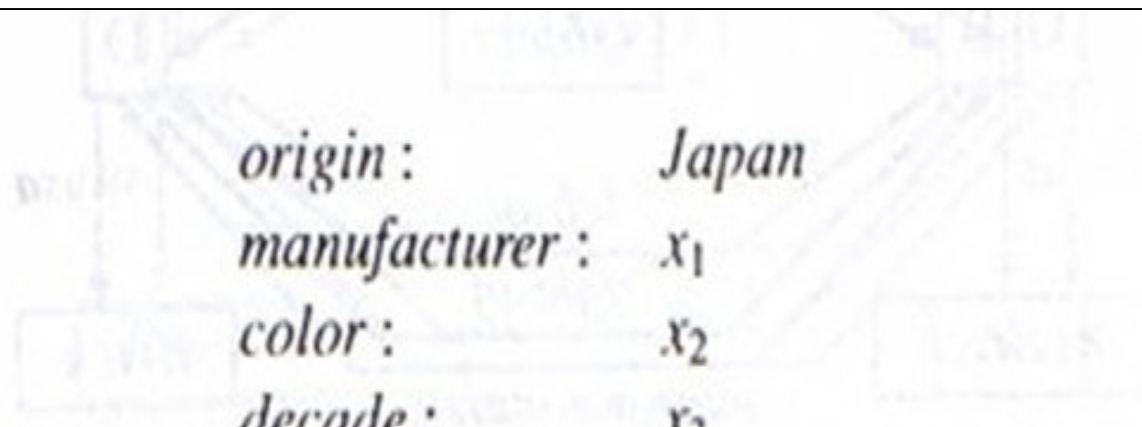
- Algorithm
- Initialize G to contain one element: Null description all feature are available
- Initialize S to contain one element: first positive example
- Accept a new training example
  - If positive, remove from G any description that do not cover the example
  - Then update S to contain most specific set of description in version space that cover the example
  - Generalize element of S
  - If negative remove from S any description that cover the example
  - Update G to contain most general set of description in version space that do not cover the example
  - Specialize the elements of G
- If G and S are both singleton set , then they are identical , output their values and halt. If both sets are different but singleton then training examples were not consistent.

# Version Spaces

<i>origin</i>	∈	{ <i>Japan, USA, Britain, Germany, Italy</i> }
<i>manufacturer</i>	∈	{ <i>Honda, Toyota, Ford, Chrysler, Jaguar, BMW, Fiat</i> }
<i>color</i>	∈	{ <i>Blue, Green, Red, White</i> }
<i>decade</i>	∈	{ <i>1950, 1960, 1970, 1980, 1990, 2000</i> }
<i>type</i>	∈	{ <i>Economy, Luxury, Sports</i> }

Figure 17.8: Representation Language for Cars

# Version Spaces



*origin :* *Japan*  
*manufacturer :*  $x_1$   
*color :*  $x_2$   
*decade :*  $x_3$   
*type :* *Economy*

Figure 17.9: The Concept “Japanese economy car”



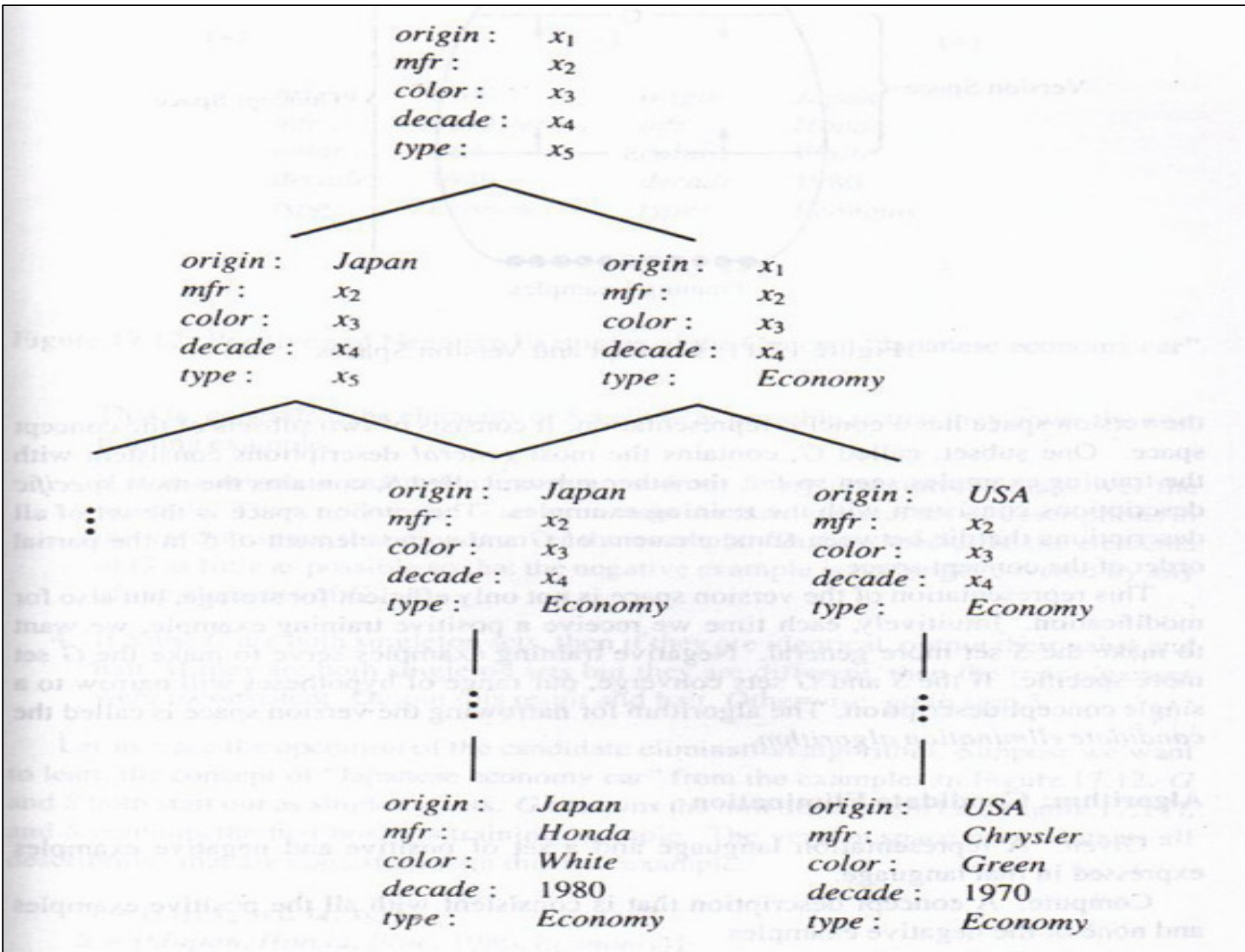
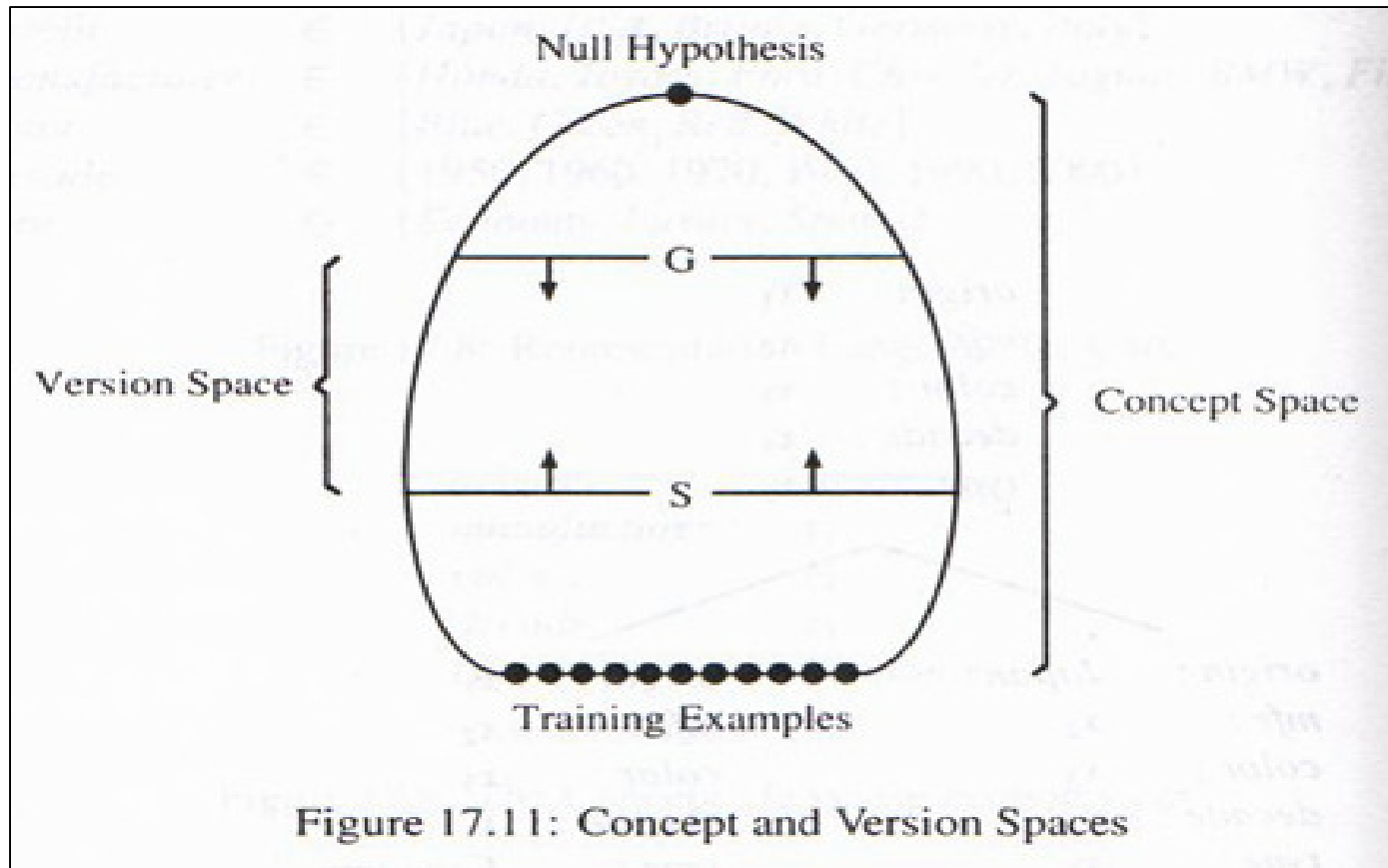


Figure 17.10: Partial Ordering of Concepts Specified by the Representation Language

# Version Spaces



# Version Spaces

<i>origin :</i> Japan	<i>origin :</i> Japan	<i>origin :</i> Japan
<i>mfr :</i> Honda	<i>mfr :</i> Toyota	<i>mfr :</i> Toyota
<i>color :</i> Blue	<i>color :</i> Green	<i>color :</i> Blue
<i>decade :</i> 1980	<i>decade :</i> 1970	<i>decade :</i> 1990
<i>type :</i> Economy	<i>type :</i> Sports	<i>type :</i> Economy
(+)	(-)	(+)
<i>origin :</i> USA	<i>origin :</i> Japan	
<i>mfr :</i> Chrysler	<i>mfr :</i> Honda	
<i>color :</i> Red	<i>color :</i> White	
<i>decade :</i> 1980	<i>decade :</i> 1980	
<i>type :</i> Economy	<i>type :</i> Economy	
(-)	(+)	

Figure 17.12: Positive and Negative Examples of the Concept “Japanese economy car”

# Candidate Eliminate Algorithm

## Algorithm: Candidate Elimination

Given: A representation language and a set of positive and negative examples expressed in that language.

Compute: A concept description that is consistent with all the positive examples and none of the negative examples.

1. Initialize  $G$  to contain one element: the null description (all features are variables).
2. Initialize  $S$  to contain one element: the first positive example.
3. Accept a new training example.

If it is a *positive example*, first remove from  $G$  any descriptions that do not cover the example. Then, update the  $S$  set to contain the most specific set of descriptions in the version space that cover the example and the current elements of the  $S$  set.

# Candidate Eliminate Algorithm

That is, generalize the elements of  $S$  as little as possible so that they cover the new training example.

If it is a *negative* example, first remove from  $S$  any descriptions that cover the example. Then, update the  $G$  set to contain the most general set of descriptions in the version space that *do not* cover the example. That is, specialize the elements of  $G$  as little as possible so that the negative example is no longer covered by any of the elements of  $G$ .

4. If  $S$  and  $G$  are both singleton sets, then if they are identical, output their value and halt. If they are both singleton sets but they are different, then the training cases were inconsistent. Output this result and halt. Otherwise, go to step 3.

We want “*Japanese economy car*”

From Figure 17.12 Positive and negative examples of car : p. 467

[origin = X1, manufacture = X2, color = X3, decade = X4, type = X5]

- GET EX1 (+)       $G = \{(X1, X2, X3, X4, X5)\}$   
                           $S = \{(Japan, Honda, Blue, 1980, Economy)\}$  = Figure 17.12 in EX1
- GET EX2 (-)       $G = \{(X1, Honda, X3, X4, X5), (X1, X2, Blue, X4, X5),$   
                           $(X1, X2, X3, 1980, X5), (X1, X2, X3, X4, Economy)\}$   
                           $S = \{(Japan, Honda, Blue, 1980, Economy)\}$   
                          \*\* the same because (-) example
- GET EX3 (+)      check G first,  $G = \{(X1, X2, Blue, X4, X5), (X1, X2, X3, X4, Economy)\}$   
                           $S = \{(Japan, X2, Blue, X4, Economy)\}$
- GET EX4 (-)      check G first,  $G = \{(Japan, X2, Blue, X4, X5),$   
                           $(Japan, X2, X3, X4, Economy)\}$   
                           $S = \{(Japan, X2, Blue, X4, Economy)\}$   
                          \*\* the same because (-) example
- GET EX5 (+)      check G first,  $G = \{(Japan, X2, X3, X4, Economy)\}$   
                           $S = \{(Japan, X2, X3, X4, Economy)\}$

# Version Spaces

- **Note :** The algorithm is least commitment algorithm : produce as little as possible at each step
- **Problems**
  - 1.) S and G may not converge to a single hypothesis
  2. ) if there is a noise (inconsistent data) → the algorithm will be premature, we may prune the target concept too fast
    - \* **For example** if the data number three given the negative sign (-) instance of positive sign (+) ... no matter how much the data is we can not find the concept....
    - \* **How to fix this problem** is to maintain several G and S sets
      - BUT** it is costly and may have the bounded inconsistency problem
  - 3.) We can not use OR in the questions ask
    - \* **For example :** Italian sport car or German luxury car”

# Explanation based learning

- In previous algorithms substantial number of positive and negative training sample were provided to learn a concept.
- But, at times, people can learn from single example or sample
- What makes single example learning possible-knowledge base
- Machine learning algorithm are shifting from empirical data intensive approach to more analytical , knowledge intensive approach
- One of such approach is *explanation based learning* (EBL)



# EBL

- System attempts to learn from a single example  $x$  by explaining why  $x$  is example of target concept
- Explanation is then generalized
- System performance is improved through availability of knowledge
- DeJong and Mooney (1986) described a general framework for EBL programs

# EBL general framework

- EBL program accepts the following input
  - *A training example*: when learning programs “sees” the concept. Example--- Car
  - *A goal concept*: A high level description of what the program is supposed to learn
  - *An operational criterion*: A description of which concepts are usable
  - *A domain theory*: a set of rules that describe relationship between objects and action in a domain

# EBL

- From this input, EBL computes a generalization of training example, that is sufficient to define a goal concept
- It also satisfies operatinality criterion
- Explanation based generalization (EBG) is an algorithm for EBL which has two steps
  - Explain
    - Domain theory is used to remove all unimportant aspect of training example with respect to goal concept
    - What is left is an explanation of why training instance is an example of goal concept
  - Generalize
    - Genralize explanation as far as possible

# Example of EBL- concept of *cup*

- Training example
  - *Owner(object23, Ralph)  $\wedge$  has-part(object23, concavity)  $\wedge$  is (object23, light)  $\wedge$  color(object23, brown).....*
  - Some features are more relevant to its being cup than others
  - How to find relevant feature--- use domain knowledge.
  - Previous learning algorithm used many positive and negative samples

# Example of EBL (contd)....

- Domain knowledge
  - $Is(x, light) \wedge has-part(x, y) \wedge isa(y, handle) \rightarrow liftable(x)$
  - $Has-part(x, y) \wedge isa(y, bottom) \wedge is(y, flat) \rightarrow stable(x)$
  - $Has-part(x, y) \wedge isa(y, concavity) \wedge is(y, upward-pointing) \rightarrow open-vessel(x)$
- Goal concept: cup
  - x is cup if it is liftable, stable and open-vessel
- Operationality criterion
  - Given a training sample
  - Build a structural description of cup
  - Explain why object23 is a cup
  - Use

# example

