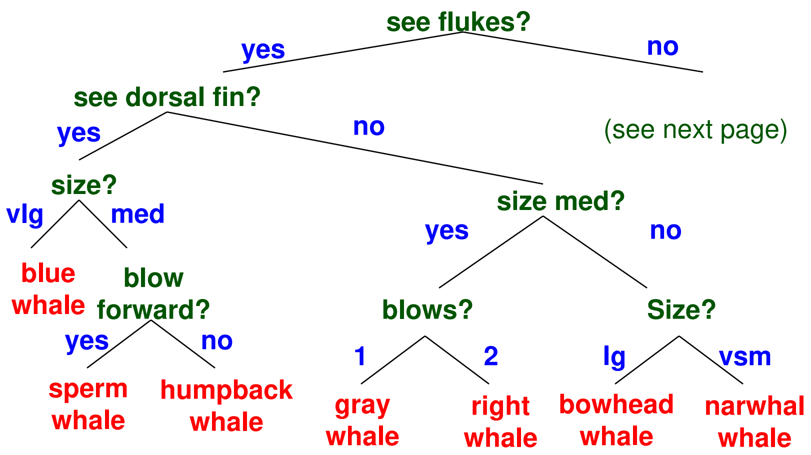
10b Machine Learning: Symbol-based

Additional references for the slides:
Jean-Claude Latombe's CS121 slides:
robotics.stanford.edu/~latombe/cs121

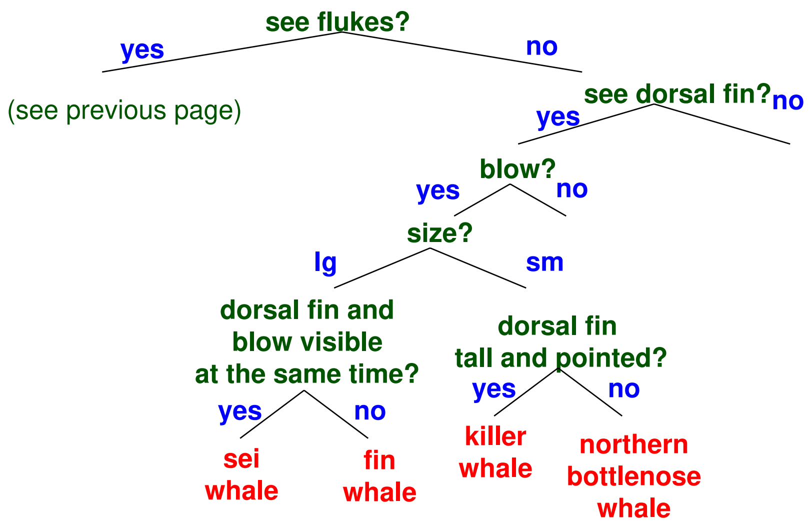
Decision Trees

- A ***decision tree*** allows a classification of an object by testing its values for certain properties
- check out the example at:
  www.aiinc.ca/demos/whale.html
- The learning problem is similar to concept learning using version spaces in the sense that we are trying to identify a class using the observable properties.
- It is different in the sense that we are trying to learn a structure that determines class membership after a sequence of questions. This structure is a decision tree.

Reverse engineered decision tree of the whale watcher expert system



Reverse engineered decision tree of the whale watcher expert system (cont'd)



What might the original data look like?

Place	Time	Group	Fluke	Dorsal fin	Dorsal shape	Size	Blow	…	Blow fwd	Type
Kaikora	17:00	Yes	Yes	Yes	small triang.	Very large	Yes		No	Blue whale
Kaikora	7:00	No	Yes	Yes	small triang.	Very large	Yes		No	Blue whale
Kaikora	8:00	Yes	Yes	Yes	small triang.	Very large	Yes		No	Blue whale
Kaikora	9:00	Yes	Yes	Yes	squat triang.	Medium	Yes		Yes	Sperm whale
Cape Cod	18:00	Yes	Yes	Yes	Irregu-lar	Medium	Yes		No	Hump-back whale
Cape Cod	20:00	No	Yes	Yes	Irregu-lar	Medium	Yes		No	Hump-back whale
Newb. Port	18:00	No	No	No	Curved	Large	Yes		No	Fin whale
Cape Cod	6:00	Yes	Yes	No	None	Medium	Yes		No	Right whale
…										

The search problem

Given a table of observable properties, search for a decision tree that

- **correctly represents the data (assuming that the data is noise-free), and**
- **is as small as possible.**

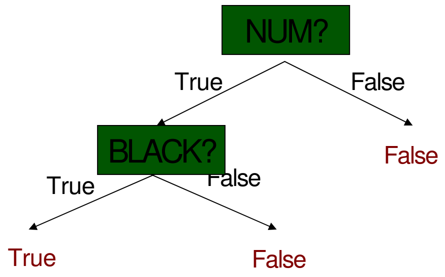
What does the search tree look like?

Comparing VSL and learning DTs

A hypothesis learned in VSL can be represented as a decision tree.

Consider the predicate that we used as a VSL example:
NUM(r) ∧ BLACK(s) ⇔ REWARD([r,s])

The decision tree on the right represents it:

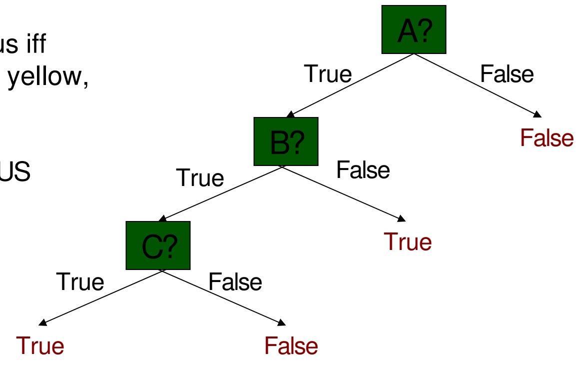


Predicate as a Decision Tree

The predicate CONCEPT(x) ⇔ A(x) ∧ (¬B(x) v C(x)) can be represented by the following decision tree:

Example:
A mushroom is poisonous iff it is yellow and small, or yellow, big and spotted

- x is a mushroom
- CONCEPT = POISONOUS
- A = YELLOW
- B = BIG
- C = SPOTTED
- D = FUNNEL-CAP
- E = BULKY

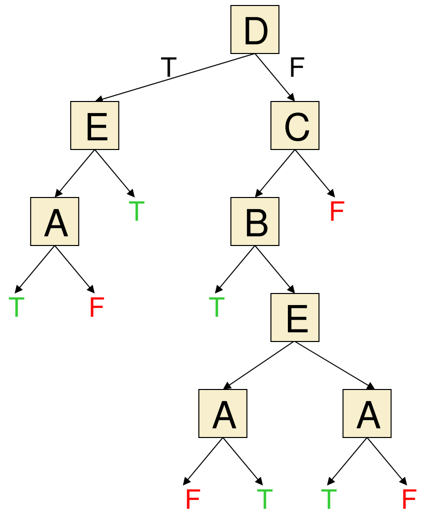


Training Set

Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True

Possible Decision Tree

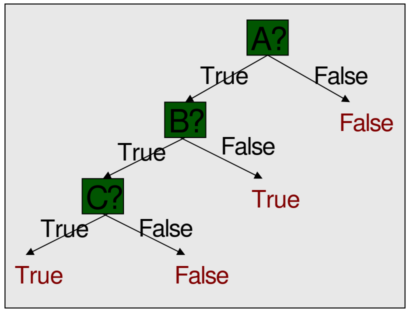
Ex. #	A	B	C	D	E	CONCEPT
1	False	False	True	False	True	False
2	False	True	False	False	False	False
3	False	True	True	True	True	False
4	False	False	True	False	False	False
5	False	False	False	True	True	False
6	True	False	True	False	False	True
7	True	False	False	True	False	True
8	True	False	True	False	True	True
9	True	True	True	False	True	True
10	True	True	True	True	True	True
11	True	True	False	False	False	False
12	True	True	False	False	True	False
13	True	False	True	True	True	True

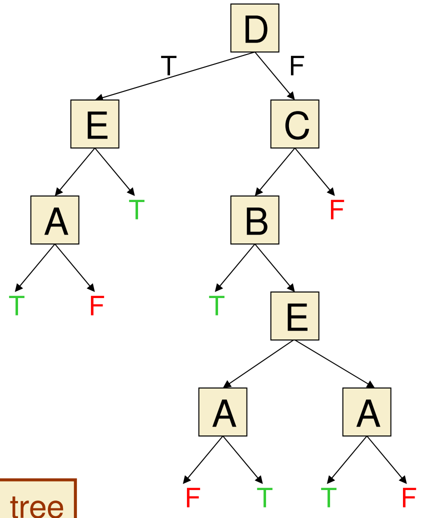


Possible Decision Tree

CONCEPT $\Leftrightarrow$
$(D \wedge (\neg E \vee A)) \vee$
$(C \wedge (B \vee ((E \wedge \neg A) \vee A)))$

CONCEPT $\Leftrightarrow A \wedge (\neg B \vee C)$





KIS bias → Build smallest decision tree

Computationally intractable problem→ greedy algorithm

Getting Started

The distribution of the training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Getting Started

The distribution of training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could report that CONCEPT is False (majority rule) with an estimated probability of error P(E) = 6/13

Getting Started

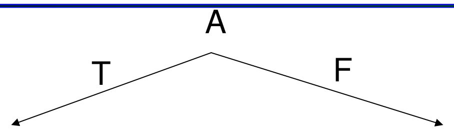
The distribution of training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could report that CONCEPT is False (majority rule) with an estimated probability of error P(E) = 6/13

Assuming that we will only include one observable predicate in the decision tree, which predicate should we test to minimize the probability of error?

How to compute the probability of error



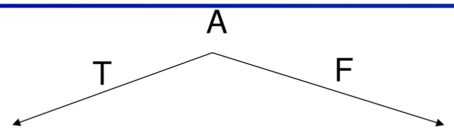
True:	6, 7, 8, 9, 10, 13	
False:	11, 12	1, 2, 3, 4, 5

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise.

The estimated probability of error is:
Pr(E) = (8/13)x(2/8) + (5/8)x0 = 2/13

8/13 is the probability of getting True for A, and
2/8 is the probability that the report was incorrect
(we are always reporting True for the concept).

How to compute the probability of error



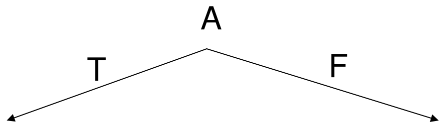
True:	6, 7, 8, 9, 10, 13	
False:	11, 12	1, 2, 3, 4, 5

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise.

The estimated probability of error is:
Pr(E) = (8/13)x(2/8) + (5/8)x0 = 2/13

5/8 is the probability of getting False for A, and
0 is the probability that the report was incorrect
(we are always reporting False for the concept).

Assume It's A

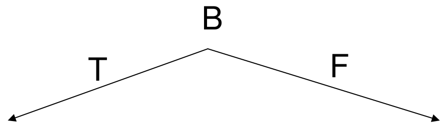


True:	6, 7, 8, 9, 10, 13	
False:	11, 12	1, 2, 3, 4, 5

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise

The estimated probability of error is:
Pr(E) = (8/13)x(2/8) + (5/8)x0 = 2/13

Assume It's B

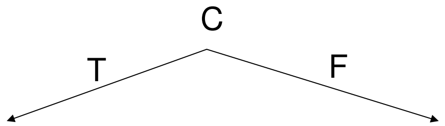


True:	9, 10	6, 7, 8, 13
False:	2, 3, 11, 12	1, 4, 5

If we test only B, we will report that CONCEPT is False if B is True and True otherwise

The estimated probability of error is:
Pr(E) = (6/13)x(2/6) + (7/13)x(3/7) = 5/13

Assume It's C

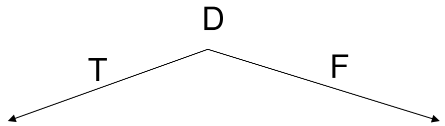


True:	6, 8, 9, 10, 13	7
False:	1, 3, 4	1, 5, 11, 12

If we test only C, we will report that CONCEPT is True if C is True and False otherwise

The estimated probability of error is:
Pr(E) = (8/13)x(3/8) + (5/13)x(1/5) = 4/13

Assume It's D

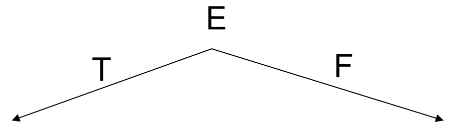


True:	7, 10, 13	6, 8, 9
False:	3, 5	1, 2, 4, 11, 12

If we test only D, we will report that CONCEPT is True if D is True and False otherwise

The estimated probability of error is:
Pr(E) = (5/13)x(2/5) + (8/13)x(3/8) = 5/13

Assume It's E



True:	8, 9, 10, 13	6, 7
False:	1, 3, 5, 12	2, 4, 11

If we test only E we will report that CONCEPT is False, independent of the outcome

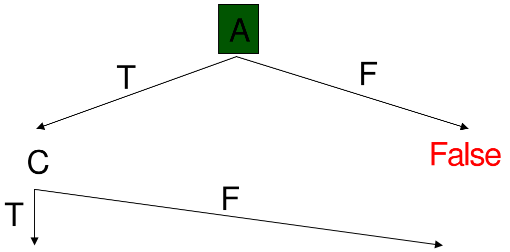
The estimated probability of error is:
Pr(E) = (8/13)x(4/8) + (5/13)x(2/5) = 6/13

Pr(error) for each

- **If A: 2/13**
- **If B: 5/13**
- **If C: 4/13**
- **If D: 5/13**
- **If E: 6/13**

So, the best predicate to test is A

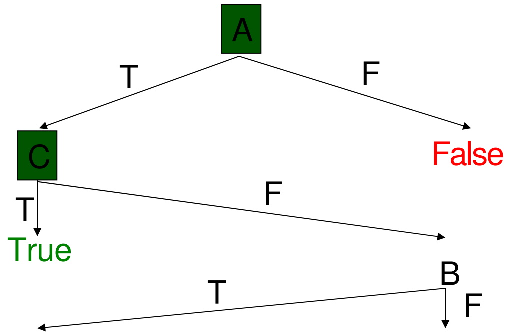
Choice of Second Predicate



True:	6, 8, 9, 10, 13	7
False:		11, 12

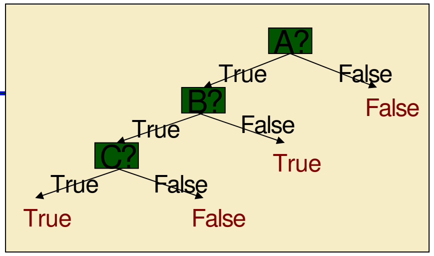
The majority rule gives the probability of error Pr(E|A) = 1/8 and Pr(E) = 1/13

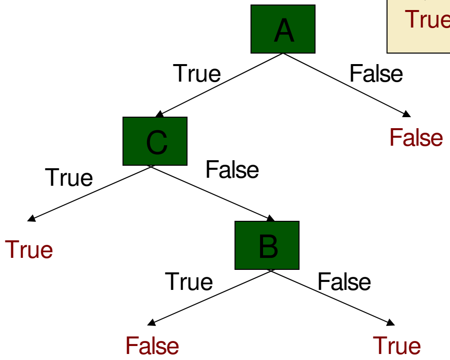
Choice of Third Predicate



True:		7
False:	11,12	

Final Tree





$L \equiv CONCEPT \Leftrightarrow A \wedge (C \vee \neg B)$

Learning a decision tree

Function induce_tree (example_set, properties)

```
begin
if all entries in example_set are in the same class
  then return a leaf node labeled with that class
else if properties is empty
      then return leaf node labeled with disjunction of all
        classes in example_set
    else begin
        select a property, P, and make it the root of the current tree;
        delete P from properties;
        for each value, V, of P
          begin
            create a branch of the tree labeled with V;
            let partition_v be elements of example_set with values V
              for property P;
            call induce_tree (partition_v, properties), attach result to
              branch V
          end
    end
end
```

If property V is Boolean: the partition will contain two sets, one with property V true and one with false

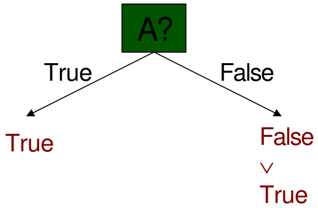
What happens if there is noise in the training set?

The part of the algorithm shown below handles this:

if properties is empty
then return leaf node labeled with disjunction of all classes in example_set

Consider a very small (but inconsistent) training set:

A	classification
T	T
F	F
F	T

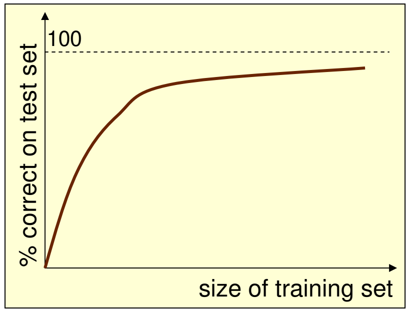


Using Information Theory

Rather than minimizing the probability of error, most existing learning procedures try to minimize the expected number of questions needed to decide if an object x satisfies CONCEPT.

This minimization is based on a measure of the "quantity of information" that is contained in the truth value of an observable predicate and is explained in Section 9.3.2. We will skip the technique given there and use the "probability of error" approach.

Assessing performance



Typical learning curve

The evaluation of ID3 in chess endgame

Size of Training Set	Percentage of Whole Universe	Errors in 10,000 trials	Predicted Maximum Errors
200	0.01	199	728
1,000	0.07	33	146
5,000	0.36	8	29
25,000	1.79	6	7
125,000	8.93	2	1

Other issues in learning decision trees

- **If data for some attribute is missing and is hard to obtain, it might be possible to *extrapolate* or use "*unknown*."**
- **If some attributes have continuous values, *groupings* might be used.**
- **If the data set is too large, one might use *bagging* to select a sample from the training set. Or, one can use *boosting* to assign a weight showing importance to each instance. Or, one can *divide the sample set into subsets* and train on one, and test on others.**

Inductive bias

- **Usually the space of learning algorithms is very large**
- **Consider learning a classification of bit strings**
  - **A classification is simply a subset of all possible bit strings**
  - **If there are n bits there are 2^n possible bit strings**
  - **If a set has m elements, it has 2^m possible subsets**
  - **Therefore there are 2^(2^n) possible classifications (if n=50, larger than the number of molecules in the universe)**
- **We need additional heuristics (assumptions) to restrict the search space**

Inductive bias (cont'd)

- *Inductive bias* refers to the assumptions that a machine learning algorithm will use during the learning process
- One kind of inductive bias is *Occams Razor*: assume that the simplest consistent hypothesis about the target function is actually the best
- Another kind is *syntactic bias*: assume a pattern defines the class of all matching strings
  - "nr" for the cards
  - {0, 1, #} for bit strings

Inductive bias (cont'd)

- Note that syntactic bias restricts the concepts that can be learned
  - If we use "nr" for card subsets, "all red cards except King of Diamonds" cannot be learned
  - If we use {0, 1, #} for bit strings "1##0" represents {1110, 1100, 1010, 1000} but a single pattern cannot represent all strings of even parity (the number of 1s is even, including zero)
- The tradeoff between expressiveness and efficiency is typical

Inductive bias (cont'd)

- Some representational biases include
  - Conjunctive bias: restrict learned knowledge to conjunction of literals
  - Limitations on the number of disjuncts
  - Feature vectors: tables of observable features
  - Decision trees
  - Horn clauses
  - BBNs
- There is also work on programs that change their bias in response to data, but most programs assume a fixed inductive bias

Explanation based learning

- Idea: can learn better when the background theory is known
- Use the domain theory to explain the instances taught
- Generalize the explanation to come up with a "learned rule"

Example

- We would like the system to learn what a cup is, i.e., we would like it to learn a rule of the form: premise(X) $\rightarrow$ cup(X)

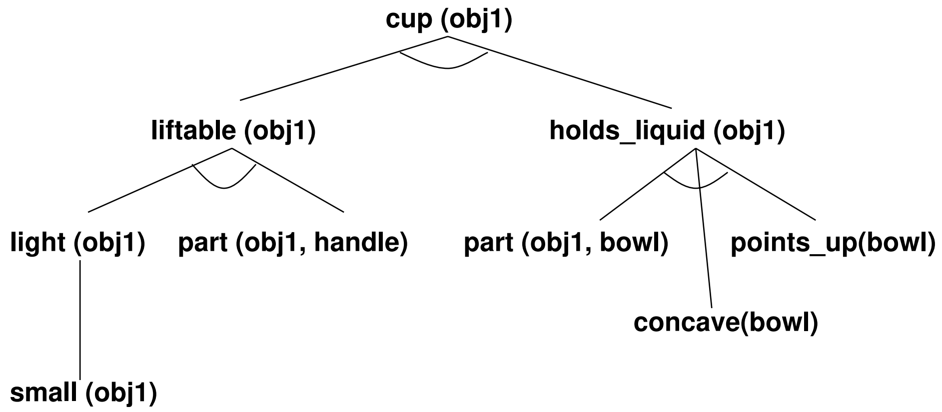
- Assume that we have a domain theory:

liftable(X) $\wedge$ holds_liquid(X) $\rightarrow$ cup(X)
part (Z,W) $\wedge$ concave(W) $\wedge$ points_up $\rightarrow$ holds_liquid (Z)
light(Y) $\wedge$ part(Y,handle) $\rightarrow$ liftable (Y)
small(A) $\rightarrow$ light(A)
made_of(A,feathers) $\rightarrow$ light(A)

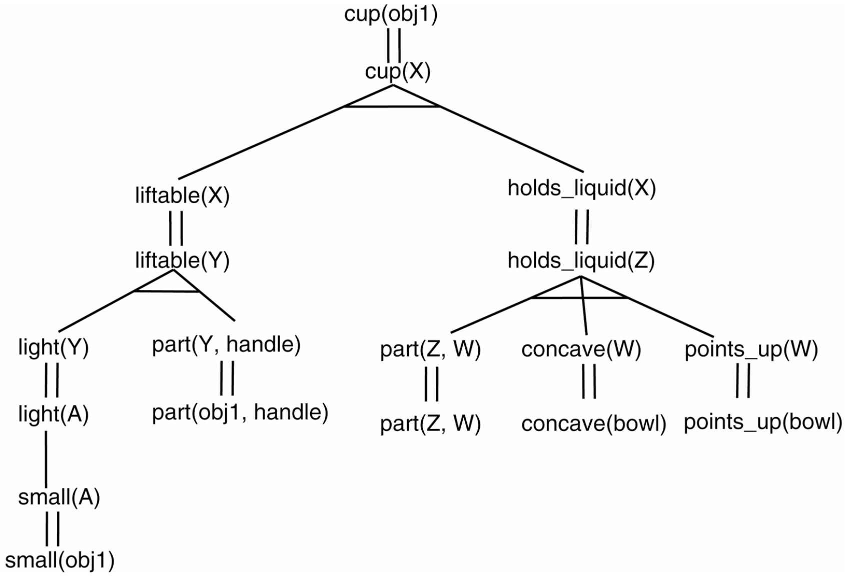
- The training example is the following:

cup (obj1)	small(obj1)
small(obj1)	part(obj1,handle)
owns(bob,obj1)	part(obj1,bottom)
part(obj1, bowl)	points_up(bowl)
concave(bowl)	color(obj1,red)

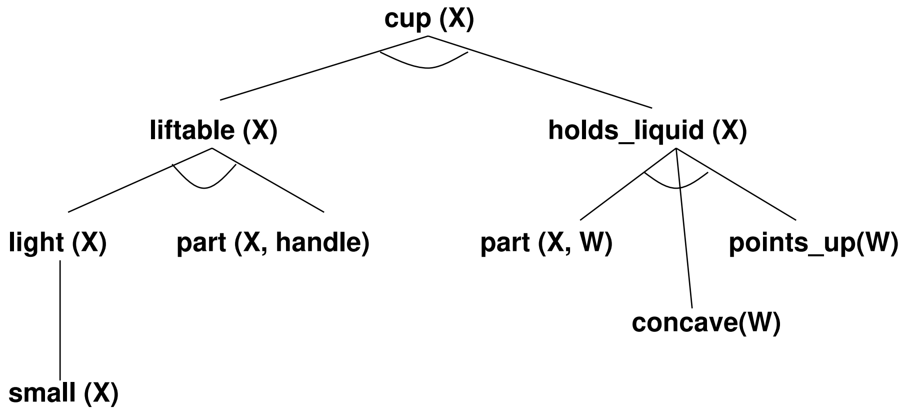
First, form a specific proof that obj1 is a cup



Second, analyze the explanation structure to generalize it



Third, adopt the generalized the proof



The EBL algorithm

Initialize hypothesis = { }

For each positive training example not covered by hypothesis:

1. ***Explain*** **how training example satisfies target concept, in terms of domain theory**
2. ***Analyze*** **the explanation to determine the most general conditions under which this explanation (proof) holds**
3. ***Refine*** **the hypothesis by adding a new rule, whose premises are the above conditions, and whose consequent asserts the target concept**

Wait a minute!

- **Isn't this "just a restatement of what the learner already knows?"**
- **Not really**
  - **a theory-guided generalization from examples**
  - **an example-guided operationalization of theories**
- **Even if you know all the rules of chess you get better if you play more**
- **Even if you know the basic axioms of probability, you get better as you solve more probability problems**

Comments on EBL

- **Note that the "irrelevant" properties of obj1 were disregarded (e.g., color is red, it has a bottom)**
- **Also note that "irrelevant" generalizations were sorted out due to its goal-directed nature**
- **Allows justified generalization from a single example**
- **Generality of result depends on domain theory**
- **Still requires multiple examples**
- **Assumes that the domain theory is correct (error-free)---as opposed to *approximate domain theories* which we will not cover.**
  - **This assumption holds in chess and other search problems.**
  - **It allows us to assume explanation = proof.**

Two formulations for learning

Inductive	***Analytical***
Given:	**Given:**
• **Instances**	• **Instances**
• **Hypotheses**	• **Hypotheses**
• **Target concept**	• **Target concept**
• **Training examples of the target concept**	• **Training examples of the target concept**
	• **<u>Domain theory for explaining examples</u>**
Determine:	**Determine:**
• **Hypotheses consistent with the training examples**	• **Hypotheses consistent with the training examples <u>and the domain theory</u>**

Two formulations for learning (cont'd)

Inductive	*Analytical*
Hypothesis fits data	Hypothesis fits domain theory
Statistical inference	Deductive inference
Requires little prior knowledge	Learns from scarce data
Syntactic inductive bias	Bias is domain theory

DT and VS learners are "similarity-based"

Prior knowledge is important. It might be one of the reasons for humans' ability to generalize from as few as a single training instance.

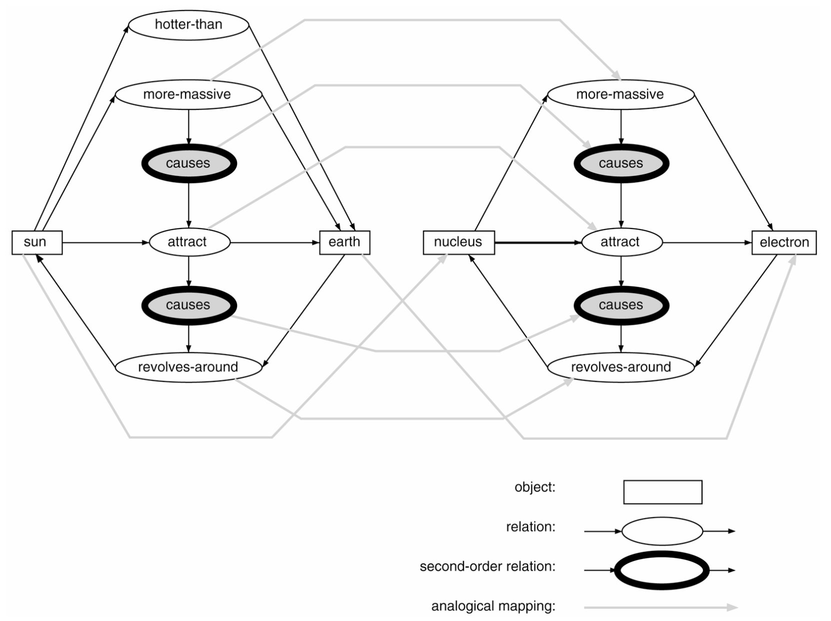
Prior knowledge can guide in a space of an unlimited number of generalizations that can be produced by training examples.

An example: META-DENDRAL

- Learns rules for DENDRAL
- Remember that DENDRAL infers structure of organic molecules from their chemical formula and mass spectrographic data.
- Meta-DENDRAL constructs an explanation of the site of a cleavage using
  - structure of a known compound
  - mass and relative abundance of the fragments produced by spectrography
  - a "half-order" theory (e.g., double and triple bonds do not break; only fragments larger than two carbon atoms show up in the data)
- These explanations are used as examples for constructing general rules

Analogical reasoning

- Idea: if two situations are similar in some respects, then they will probably be in others
- Define the ***source*** of an analogy to be a problem solution. It is a theory that is relatively well understood.
- The ***target*** of an analogy is a theory that is not completely understood.
- Analogy constructs a ***mapping*** between corresponding elements of the target and the source.



Example: atom/solar system analogy

- The source domain contains:
  yellow(sun)
  blue(earth)
  hotter-than(sun,earth)
  causes(more-massive(sun,earth), attract(sun,earth))
  causes(attract(sun,earth), revolves-around(earth,sun))
- The target domain that the analogy is intended to explain includes:
  more-massive(nucleus, electron)
  revolves-around(electron, nucleus)
- The mapping is: sun $\rightarrow$ nucleus and earth $\rightarrow$ electron
- The extension of the mapping leads to the inference:
  causes(more-massive(nucleus,electron), attract(nucleus,electron))
  causes(attract(nucleus,electron), revolves-around(electron,nucleus))

A typical framework

- ***Retrieval***: Given a target problem, select a potential source analog.
- ***Elaboration***: Derive additional features and relations of the source.
- ***Mapping and inference***: Mapping of source attributes into the target domain.
- ***Justification***: Show that the mapping is valid.