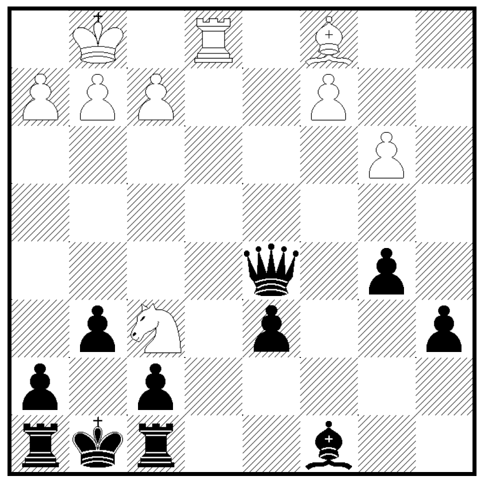
# Outline

- Two formulations for learning: Inductive and Analytical
- Perfect domain theories and Prolog-EBG

# A Positive Example



# The Inductive Generalization Problem

Given:

- Instances
- Hypotheses
- Target Concept
- Training examples of target concept

Determine:

- Hypotheses consistent with the training examples

# The Analytical Generalization Problem

Given:

- Instances
- Hypotheses
- Target Concept
- Training examples of target concept
- *Domain theory for explaining examples*

Determine:

- Hypotheses consistent with the training examples *and the domain theory*

# An Analytical Generalization Problem

Given:

- Instances: pairs of objects
- Hypotheses: sets of horn clause rules
- Target Concept: Safe-to-stack(x,y)
- Training Example: Safe-to-stack(OBJ1,OBJ2)

  On(OBJ1,OBJ2)
  Isa(OBJ1,BOX)
  Isa(OBJ2,ENDTABLE)
  Color(OBJ1,RED)
  Color(OBJ2,BLUE)
  Volume(OBJ1,.1)
  Density(OBJ1,.1)
  ...

- Domain Theory:

  Safe-To-Stack(x,y) :- Not(Fragile(y))
  Safe-To-Stack(x,y) :- Lighter(x,y)
  Lighter(x,y) :- Weight(x,wx), Weight(y,wy),
  Less(wx,wy)
  Weight(x,w) :- Volume(x,v), Density(x,d),
  Equal(w, v*d)
  Weight(x,5) :- Isa(x, ENDTABLE)
  ...

Determine:

- Hypotheses consistent with training examples and domain theory

# Learning from Perfect Domain Theories

Assumes domain theory is *correct* (error-free)

- Prolog-EBG is algorithm that works under this assumption
- This assumption holds in chess and other search problems
- Allows us to assume explanation = proof
- Later we'll discuss methods that assume *approximate* domain theories

# Prolog EBG

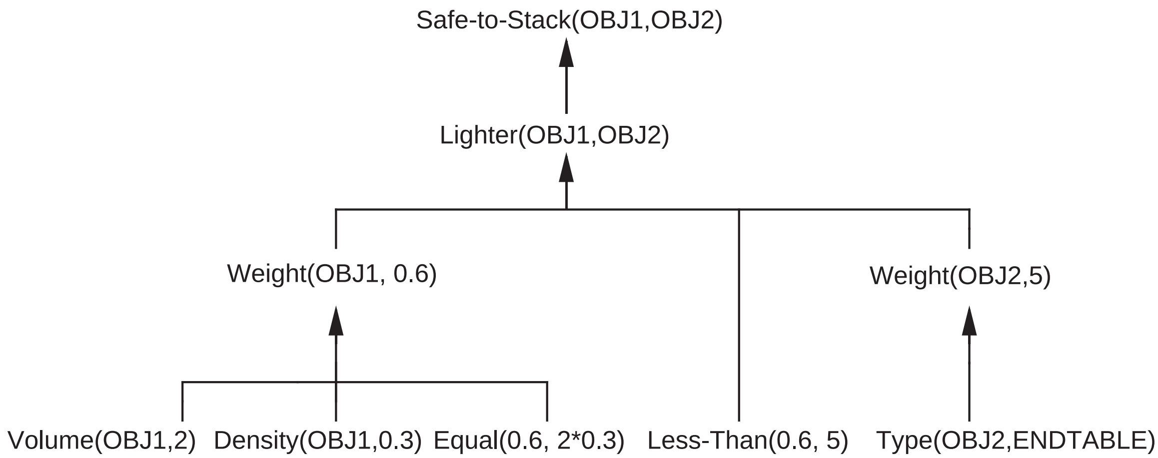
Initialize hypothesis = {}

For each positive training example not covered by hypothesis:

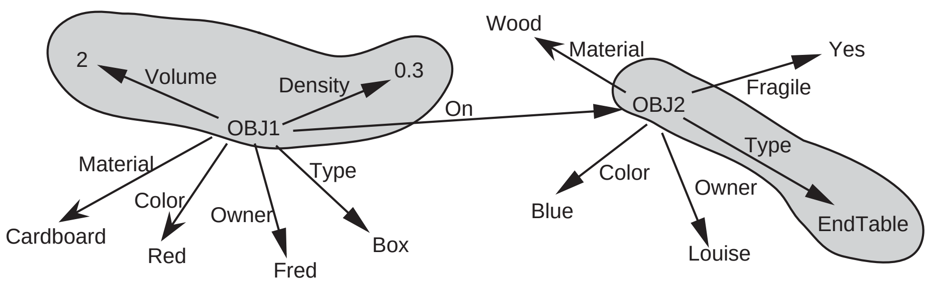
1. **Explain** how training example satisfies target concept, in terms of domain theory
2. **Analyze** the explanation to determine the most general conditions under which this explanation (proof) holds
3. **Refine** the hypothesis by adding a new rule, whose preconditions are the above conditions, and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

# Explanation of a Training Example

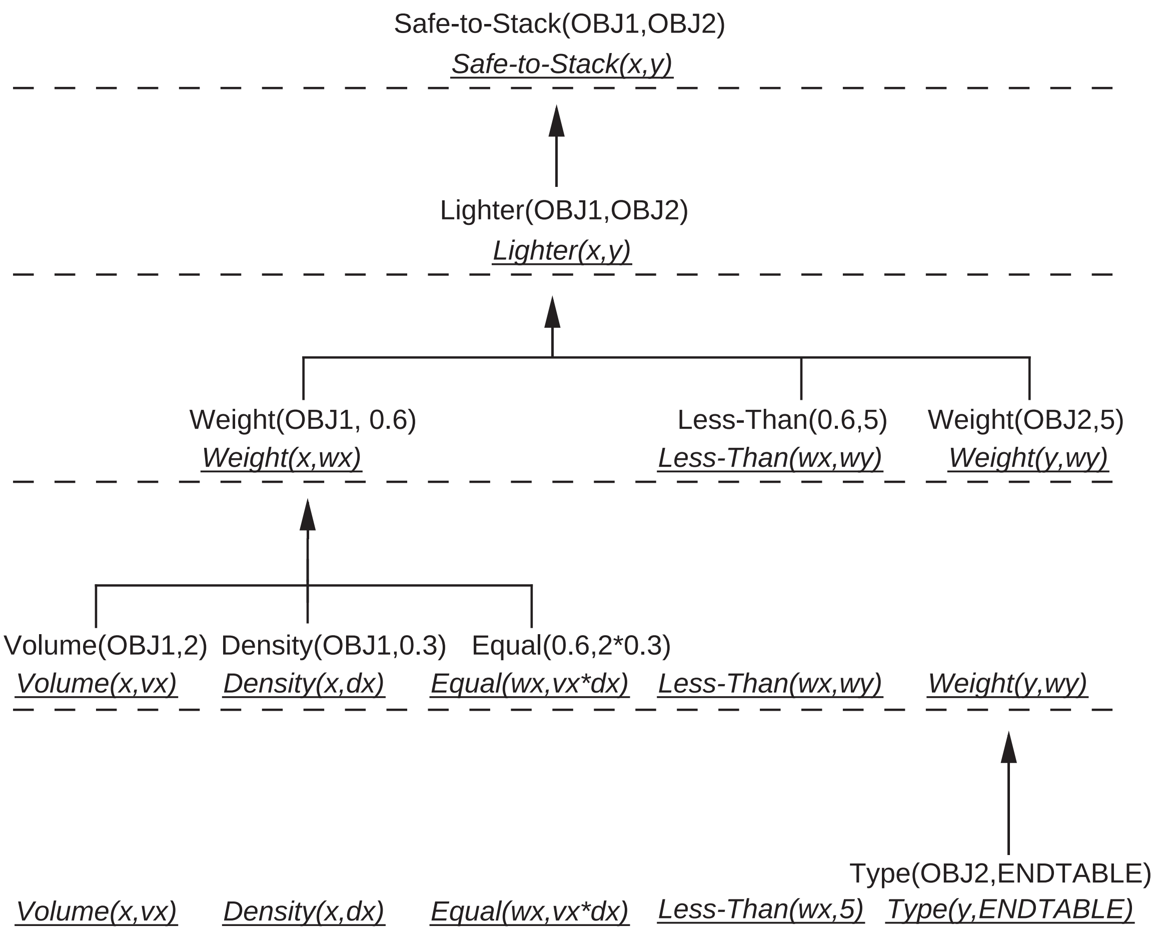
***Explanation:***




***Training Example:***



# Computing the Weakest Preimage of Explanation



# Regression Algorithm

---

Regress(*Frontier, Rule, Expression,* $U_{I,R}$)

*Frontier*: the set of expressions to be regressed through *Rule*
*Rule*: a horn clause.
*Expression*: the member of *Frontier* that is inferred by *Rule* in the explanation.
$U_{I,R}$: the substitution that unifies *Rule* to the training example in the explanation

Returns the list of expressions forming the weakest preimage of *Frontier* with respect to *Rule*

let Consequent ← *Rule* consequent

let Antecedents ← *Rule* antecedents

1. $U_{E,R}$ ← most general unifier of *Expression* with Consequent such that there exists a substitution S for which

$$S(U_{E,R}(\text{Consequent})) = U_{I,R}(\text{Consequent})$$

2. Return $U_{E,R}$({*Frontier* -Consequent+Antecedent})

---

**Example:**

Regress({Volume(x,vs), Density(x,dx), Equal(wx,vx*dx),
Less-Than(wx,wy), Weight(y,wy)},
Weight(z,5) :- Type(z,ENDTABLE),
Weight(y,wy),
{OBJ2/z})

Consequent ← Weight(z,5)

Antecedents ← Type(z,ENDTABLE)

$U_{E,R}$ ← {y/z, 5/wy}, (S = {OBJ2/y})

Result ← {Volume(x,vs), Density(x,dx), Equal(wx,vx*dx),
Less-Than(wx,5), Type(y,ENDTABLE)}

# Lessons from Safe-to-Stack Example

- Justified generalization from single example
- Explanation determines feature <u>relevance</u>
- Regression determines needed feature <u>constraints</u>
- Generality of result depends on domain theory
- Still require multiple examples

# Perspectives on Prolog-EBG

- Theory-guided generalization from examples
- Example-guided operationalization of theories
- "Just" restating what learner already "knows"

Is it learning?

- Are you learning when you get better over time at chess?
    - Even though you already know everything in principle, once you know rules of the game...
- Are you learning when you sit in a mathematics class?
    - Even though those theorems follow deductively from the axioms you've already learned...