Planning and Learning: Explanation-Based Learning

Manuela Veloso

Carnegie Mellon University

Planning, Execution, and Learning
Fall 2016

Thanks to Daniel Borrajo

Learning in Planning

Opportunities and improvements along several dimensions:

• **Search Efficiency**: Learn control knowledge to guide the planner through its search space.

• **Domain Specification**: Learn the preconditions and effects of the planning actions.

• **Quality**: Learn control knowledge for high quality plans.
Choices... The Need for Learning!

• Inductive methods
  – Data-intensive
  – Extract a general description of a concept from many examples

• Deductive methods
  – Knowledge-intensive
  – Explain and analyze an example
  – Identify the explanation as the sufficient conditions for describing the concept
  – Generalize instantiated explanation to apply to other instances

Explanation-Based Generalization – EBG, (Mitchell ’80s)

Inputs:
• Target concept definition
• Training example
• Domain theory
• Operationality criterion

Output:
Generalization of the training example that is
• sufficient to describe the target concept, and
• satisfies the operationality criterion.
The SAFE-TO-STACK Example

Input:

• **target concept**: SAFE-TO-STACK(x,y)

• **training example**:
  ON(OBJ1,OBJ2)
  ISA(OBJ1, BOX)  ISA(OBJ2, ENDTABLE)
  COLOR(OBJ1, RED)  COLOR(OBJ2, BLUE)
  VOLUME(OBJ1,1)  DENSITY(OBJ1,0.1) ...

The SAFE-TO-STACK Example

Input:

• **domain theory**:
  1. NOT(FRAGILE(y)) or LIGHTER(x,y) → SAFE-TO-STACK(x,y)
  2. VOLUME(x,v) and DENSITY(x,d) → WEIGHT(x,v*d)
  3. WEIGHT(x1,w1) and WEIGHT(x2,w2) and LESS(w1,w2)
     → LIGHTER(x1,x2)
  4. ISA(x,ENDTABLE) → WEIGHT(x,5)
  5. LESS(0,1,5) ...

• **operationality criterion**: 
  learned description should be built of terms used to describe examples directly, or other “easily” evaluated, such as LESS.
The SAFE-TO-STACK Example

- Explain why \texttt{obj1} is SAFE-TO-STACK on \texttt{obj2}.
  - Construct a proof.
  - Do \textbf{Goal regression}: regress target concept through proof structure.
  - Proof isolates \textit{relevant} features.

\begin{center}
\begin{tikzpicture}
  \node (goal) at (0,0) {SAFE-TO-STACK(obj1,obj2)};
  \node (lighter) at (-1,-1) {LIGHTER(obj1,obj2)};
  \node (weight1) at (0,-2) {WEIGHT(obj1)};
  \node (less1) at (1,-2) {LESS-THERE(1,5)};
  \node (weight2) at (2,-2) {WEIGHT(obj2)};
  \node (volume1) at (-1,-3) {VOLUME(obj1)};
  \node (density1) at (0,-3) {DENSITY(obj1)};
  \node (isa) at (1,-3) {ISA(obj2,ENDTABLE)};
  \draw (lighter) -- (weight1);
  \draw (weight1) -- (volume1);
  \draw (weight1) -- (density1);
  \draw (lighter) -- (less1);
  \draw (less1) -- (weight2);
  \draw (weight2) -- (isa);
\end{tikzpicture}
\end{center}

Generating Operational Knowledge

- Generalize proof:
  - Sometimes simply replace constants by variables.
  - Prove that all identified relevant features are necessary in general (hard! -- may need a lot of “extra” knowledge, \textit{domain axioms}).

Output:
\texttt{VOLUME(x,v1) and DENSITY(x,d1) and ISA(y,ENDTABLE) and}
\texttt{and LESS(v1*d1,5) \rightarrow SAFE-TO-STACK(x,y)}
**EBL: A Deductive Learning Method**

**Why are examples needed?**
- Domain theory contains all the information: simply operationalize target concept.
- Examples focus on the relevant operationalizations: characterize only examples that actually occur.

**Actual purpose of EBL:**
- *not* to “learn” more about target concept,
- *but* to “re-express” target concept in a more operational manner (=efficiency).
- control learning.

---

**EBL in PRODIGY (Minton 87)**

**Goal:** -- improve the efficiency of the planner
- -- learn control rules.

**Control rules:**
- Apply at individual decisions.
- Antecedent matches the state of the planner at decision making time.
- Antecedent is operational -- planner can match its state using control rule language.
- Consequent *selects, rejects or prefers* particular alternatives.
Target Concepts

Identify the choices of the particular planner:

• Select goal $goal$
• Select operator $op$ for achieving $goal$
• Select bindings for operator $op$ and goal $goal$
• Decide subgoal if $op$ is applicable
• Decide apply $op$

Examples of Control Rules in PRODIGY

(CONTROL-RULE SELECT-OP-UNSTACK-FOR-HOLDING
  (if (and (current-goal (holding $<x>$))
           (true-in-state (on $<x>$ $<y>$)))
      (then select operator UNSTACK))

(CONTROL-RULE SELECT-BINDINGS-UNSTACK-HOLDING
  (if (and (current-goal (holding $<x>$))
           (current-ops (UNSTACK))
           (true-in-state (on $<x>$ $<y>$)))
      (then select bindings ((<ob> . $<x>$) (<underob> . $<y>$))))

(CONTROL-RULE SELECT-OP-PUTDOWN-FOR-ARMEMPTY
  (if (and (current-goal (arm-empty))
           (true-in-state (holding $<ob>$)))
      (then select operator PUT-DOWN))

(CONTROL-RULE SELECT-BINDINGS-PUTDOWN
  (if (and (current-ops (PUT-DOWN))
           (true-in-state (holding $<x>$)))
      (then select bindings ((<ob> . $<x>$))))
Discussion

• Very successful in a variety of domains.
• Learned rules are applied as other rules, i.e. if their antecedent *totally* matches planning situation.
• Utility problem: The more rules learned, the slower the deliberation.
  – Matching cost (cost of utilization)
  – Frequency of application
  – Savings every time it is applied
  – Organization of learned rules!
• If EBL system is eager to learn provably correct, the explanation effort is really large, requiring a *complete* domain theory for generalization.
  – Incremental refinement of learned rules

HAMLET: Deduction and Induction
(Borrajo & Veloso 94)

• Extend the basic EBL approach developed for linear problem solving
  – Define new learning opportunities
  – Consider solution quality
• Reduce the explanation effort
  – No need to acquire extra domain knowledge
• Incrementally refine control knowledge
  – Converges towards an experience-supported correct set of rules
A Typical Search Tree

What are the learning opportunities?

HAMLET’s Architecture
HAMLET's Algorithm

Let L refer to the set of learned control rules. Let ST, ST' refer to search trees. Let P be a problem to be solved. Let Q be a quality measure. Initially L is empty. For all P in training problems

ST = Result of solving P without any rules.
ST' = Result of solving P with current set of rules L.
If positive-examples-p(ST, ST',Q)
Then L' = Bounded-Explanation(ST, ST',Q)
       L'' = Induce(L,L')
If negative-examples-p(ST, ST',Q)
Then L=Refine(ST, ST',L'')

Induction Module

• Why induction?
  – Bounded explanation generates possibly over-specific rules
• Inductive operators
  – Deletion of rules that subsume others
  – Intersection of preconditions. state
  – Refinement of subgoaling dependencies. prior goal
  – Relaxing the subgoaling dependencies. prior goal
  – Refinement of the set of interacting goals. other goals
  – Find common superclass. type of object
Rule Learned by HAMLET

(control-rule select-bind-fly-airplane-1
  (if  (current-operator fly-airplane)
    (current-goal (at-airplane <plane1> <airport3>))
    (true-in-state (at-airplane <plane1> <airport2>))
    (true-in-state (at-object <package4> <airport1>))
    (other-goals ((at-object <package4> <airport3>))))
  (then select bindings ((<plane> . <plane1>)
                        (<loc-from> . <airport1>)
                        (<loc-to> . <airport3>))))

Inducing Over Two Rules

- Old rule:
  (control-rule select-unload-airplane-1
    (if  (current-goal (at-object <object1> <airport2>))
      (true-in-state (at-airplane <plane4> <airport3>))
      (true-in-state (at-object <object1> <airport3>))
      (then select operators unload-airplane))

- New rule:
  (control-rule select-unload-airplane-2
    (if  (current-goal (at-object <object1> <airport2>))
      (true-in-state (at-airplane <plane4> <airport5>))
      (true-in-state (at-object <object1> <airport3>))
      (then select operators unload-airplane))

- Induced rule:
  (control-rule induced-select-unload-airplane-3
    (if  (current-goal (at-object <object1> <airport2>))
      (true-in-state (at-object <object1> <airport3>))
      (then select operators unload-airplane))
Refining

• Why refinement?
  – HAMLET may produce over-general rules

• Negative examples: occasions in which control rules have been applied and should have not

Overgeneralization

• Induced rule
  (control-rule induced-select-unload-airplane-3
   (if (current-goal (at-object <object1> <airport2>))
    (true-in-state (at-object <object1> <airport3>)))
   (then select operators unload-airplane))

• New rule
  (control-rule induced-select-unload-airplane-4
   (if (current-goal (at-object <object1> <airport2>))
    (true-in-state (inside-airplane <object1> <plane4>)))
   (then select operators unload-airplane))

• Overgeneral rule
  (control-rule induced-select-unload-airplane-5
   (if (current-goal (at-object <object1> <airport2>)))
   (then select operators unload-airplane))
Empirical Results

<table>
<thead>
<tr>
<th>Goals</th>
<th>Problems</th>
<th>Unsolved problems</th>
<th>Solved by both (279 problems, 53.14%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>without rules with rules</td>
<td>without rules with rules</td>
</tr>
<tr>
<td>Better solutions</td>
<td>Solution length</td>
<td>Nodes explored</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>5 0</td>
<td>0 11</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>15 6</td>
<td>0 25</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>44 18</td>
<td>1 33</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>68 32</td>
<td>1 24</td>
</tr>
<tr>
<td>20</td>
<td>75</td>
<td>62 36</td>
<td>0 10</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>49 40</td>
<td>0 0</td>
</tr>
<tr>
<td>totals</td>
<td></td>
<td>525 243</td>
<td>2 103</td>
</tr>
<tr>
<td>%</td>
<td></td>
<td>46.3% 25.1%</td>
<td>0.7% 36.9%</td>
</tr>
</tbody>
</table>

Training problems

<table>
<thead>
<tr>
<th>Problems</th>
<th>Unsolved problems</th>
<th>Better solutions</th>
<th>Solved by both</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>without rules with rules</td>
<td>without rules with rules</td>
<td>without rules with rules</td>
<td>without/ with rules</td>
</tr>
<tr>
<td></td>
<td>without rules with rules</td>
<td>without rules with rules</td>
<td>without rules with rules</td>
<td>without/ with rules</td>
</tr>
<tr>
<td>75</td>
<td>46.29% 25.14%</td>
<td>0.35% 36.9%</td>
<td>1.11% 1.34%</td>
<td>1.3</td>
</tr>
<tr>
<td>150</td>
<td>46.29% 34.29%</td>
<td>0.72% 31.9%</td>
<td>1.06% 0.33%</td>
<td>1.25</td>
</tr>
<tr>
<td>400</td>
<td>46.29% 25.14%</td>
<td>0.72% 36.92%</td>
<td>1.08% 0.32%</td>
<td>1.34</td>
</tr>
</tbody>
</table>

Summary – EBL in Planning

- Long-term goal of automating planning efficiency.
- Knowledge in domain theory is not usually effective.
- Explain examples to produce operational control knowledge for decisions.
- Provably correct explanations that generalize to new situations are hard to learn.
- Difficult goal and operator choice interactions can be learned through a combined deductive and inductive approach.
- User's quality metrics can be cast in the learned knowledge.