Programming by Example

- Synthesis from Examples
- Inductive Synthesis
  (sometimes: inductive learning)
Ancient Synthesis: Zendo

What is the secret of the Buddha-Nature?

- Zen master provides a set of positive (white) and negative (black) examples.
- Students have to guess the general rule.

Programming by Example

- Synthesize a program whose behavior satisfies a set of examples
  - Wait... doesn't machine learning do that?

Programming by Example

<table>
<thead>
<tr>
<th>Traditional Machine Learning</th>
<th>Inductive Synthesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learn a function from a set of examples</td>
<td>Learn a function from a set of examples</td>
</tr>
<tr>
<td>Millions of data points</td>
<td>Small numbers of examples</td>
</tr>
<tr>
<td>Scalability is a challenge</td>
<td>Analogity is a challenge</td>
</tr>
<tr>
<td>Data is noisy</td>
<td>Data is the spec</td>
</tr>
<tr>
<td>Need to avoid over-fitting, but representative solutions are still good enough</td>
<td>It would be annoying if user said f(x)=5 and the system assumed the user is wrong and decided f(x)=6</td>
</tr>
<tr>
<td>Search space is parameterized</td>
<td>Search space has complex structure</td>
</tr>
</tbody>
</table>

Optimization-based search (fast)  
Combinatorial search (slow)

Programming by Example

- Motivation
  - Algorithm Designers
  - Software Developers
  - End-Users (including: students and teachers)

Inductive Learning: Key Issues

1. How do you find a program that matches the observations?
2. How do you know it is the program you are looking for?
Inductive Learning: Key Issues

1. How do you find a program that matches the observations?
2. How do you know it is the program you are looking for?

Modern research focuses more on the space of programs.

Easiest PBE Ever

- **Task**: find a polynomial \( P(x) \).

\[
\begin{align*}
6 & = a \cdot 2^2 + b \cdot 3^2 + c \cdot 4 + d \\
-30 & = a \cdot 2^2 + b \cdot 3^2 + c \cdot 4 + d \\
60 & = a \cdot (-2)^2 + b \cdot (-3)^2 + c \cdot (-4) + d \\
50 & = a \cdot (-9)^2 + b \cdot (-9)^2 + c \cdot (-9) + d \\
\end{align*}
\]

Another Form of PBE

- **Task**: find a regular expression.

\[
\begin{align*}
\epsilon & \in L \\
a & \not\in L \\
b & \in L \\
aa & \not\in L \\
ab & \in L \\
\end{align*}
\]

Introduction to Sketching

- **Idea**: write the program but leave holes.
  - Simple hole construct: "??"
    - Sketch will replace it with an integer constant.

\[
\text{harness } \quad \text{void } \text{doubledSketch}(\text{int } x) \{ \\
\text{int } t = x \times ??; \\
\text{assert } t == x + x; \\
\} \\
\]

Theseus and the Minotaur

- **Rules**:
  - Theseus can move in any direction (but not through walls).
  - Minotaur obeys the holy laws of King Minos:
    - Only moves if it can get closer to Theseus
    - Horizontal takes precedence over vertical
    - Makes two moves for every move of Theseus
Introduction to Sketching
• Control — conditional
  
  ```
  if (??)
    t = doOneThing(??);
  else
    t = doAnotherThing(??);
  ```

  ```
  int sel = ??;
  if (sel == 0)
    t = doOneThing(??);
  else if (sel == 1)
    t = doAnotherThing(??);
  else
    assert false;
  ```

Introduction to Sketching
• Control – repeat
  
  ```
  repeat (??) {
    t = someFunc(t, ??);
    t = t + 1;
  }
  ```

  ```
  for (int i = 0; i < ??; i++) {
    t = someFunc(t, ??);
    t = t + 1;
  }
  ```

Introduction to Sketching
• Generators
  
  ```
  generator int affine(int a, int b) {
    return ?? + a * ?? + b * ??;
  }
  ```

  ```
  generator int affine(int a, int b) {
    return ?? + a * ?? + b * ??;
  }
  ```

• Without `generator`: same value for all invocations (like a normal function)

Lab #8
Create a list of moves for Theseus using Sketch.

Q2. Ignore Minotaur, find the exit.

Q2. Model Minotaur moves as well, avoid capture.

```java
// THESEUS = new At(x=1, y=2);
// MINOTAUR = new At(x=1, y=0);
// EXIT = new At(x=2, y=1);
harness escape() {
  At theseus = copy(THESEUS),
  minotaur = copy(MINOTAUR);
  ⋯
  left, right, right, up
  ⋯
  assert (success)
}
```