Programming by Example
Synthesis!

MIT & NASA, 1957

"Code" (≈165cm)
• Automatic programming?
  - but I have to tell the computer what I want...

Synthesis!

level of abstraction

```plaintext
???
Python, Haskell, ...
C
assembly
machine code
```

Synthesis = an unusually concise / intuitive programming language + a compiler that sometimes doesn’t work 😊
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?

MIT/LCS/TR-76
LEARNING STRUCTURAL DESCRIPTIONS FROM EXAMPLES
Patrick H. Winston
September 1970

Patrick Winston [1970]
Programming by Example

Traditional Machine Learning
- Learn a function from a set of examples
- Millions of data points
  Scalability is a challenge
- Data is noisy
  Need to avoid overfitting, but approximate solutions are also good enough
- Search space is parametrized
  (vector)

Optimization-based search
  (fast)

Inductive Synthesis
- Learn a function from a set of examples
- Small numbers of examples
  Ambiguity is a challenge
- Data is the spec
  It would be annoying if user said $f(x)=5$ and the system assumed the user is wrong and decided that $f(x)=6$
- Search space has complex structure
  (syntax tree)

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 1.
Easiest PBE Ever

• **Task**: find a polynomial \( \mathbb{R}(x) \).

\[
\begin{array}{|c|c|}
\hline
x & y \\
\hline
2 & 6 \\
1 & -30 \\
-4 & 60 \\
-9 & 50 \\
\hline
\end{array}
\]

\[ y = ax^3 + bx^2 + cx + d \]

\[ 6 = a \cdot 2^3 + b \cdot 2^2 + c \cdot 2 + d \]
\[ -30 = a \cdot 1^3 + b \cdot 1^2 + c \cdot 1 + d \]
\[ 60 = a \cdot (-4)^3 + b \cdot (-4)^2 + c \cdot (-4) + d \]
\[ 50 = a \cdot (-9)^3 + b \cdot (-9)^2 + c \cdot (-9) + d \]

\[
\begin{array}{|c|c|c|c|}
\hline
a & b & c & d \\
\hline
1 & -10 & -1 & -40 \\
\hline
\end{array}
\]
Another Form of PBE

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

<table>
<thead>
<tr>
<th>$w$</th>
<th>$w \in L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$a$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$b$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>$aa$</td>
<td>$\times$</td>
</tr>
<tr>
<td>$ab$</td>
<td>$\checkmark$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$w$</th>
<th>$\varepsilon$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>$aa$</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$ab$</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

$= \{\varepsilon, aa\} = q_0 \quad = \{a, b, ab\} = q_1\quad a, b$
Introduction to Sketching

• **Idea**: write the program but leave **holes**.
  
  ‣ Simple hole construct: “??”
  Sketch will replace it with an **integer constant**.

```c
harness void doubleSketch(int x) {
    int t = x * ??;
    assert t == x + x;
}
```

**entry point**

**objective**

2
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 can get closer to Theseus
    ◦ Horizontal takes precedence over vertical
    ◦ Makes two moves for every move of Theseus
Introduction to Sketching

- Control — conditional

```java
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

\[
\text{repeat ( ?? ) }
\{
  t = \text{someFunc}(t, ??);
  t = t + 1;
\}
\]

\[
\neq
\]

\[
\text{for (int i = 0; i < ?? ; i++) }
\{
  t = \text{someFunc}(t, ??);
  t = t + 1;
\}
\]
Introduction to Sketching

• Generators

```c
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.

Q1. Ignore Minotaur, find the exit.

Q2. Model Minotaur moves as well, avoid capture.

Use: ??, assert, repeat