if (justMetYou) {
    crazy = true;
    cout << number << endl;
    int x = rand()%100;
    if (x>=50)
        callMe();
}

"Call Me Maybe"

private function bad() {
    break;
}

"Breaking Bad"

class StarWars(int episode) {
    if (episode == 6)
        return Jedi;
}

"Star Wars: Episode VI – Return of the Jedi"
try
{
    Assert(Life.Real);
    Assert(Life.Fantasy);
}
catch(LandSlideException ex)
{
    #region Reality
    while(true)
    {
        character.Eyes.ForEach(eye => eye.Open().Orient(Direction.Sky).See());
        self.Wealth = null;
        self.Sex = Sex.Male;

        if(self.ComeDifficulty == Difficulty.Easy && self.GoDifficulty == Difficulty.Easy && self.High < 0.1 && self.Low < 0.1)
        {
            self.Sympathies.Clear();

            switch(wind.Direction)
            {
                case Direction.North:
                case Direction.East:
                case Direction.South:
                case Direction.West:
                    default:
                        piano.Play();
                        break;
                }
        }
    }
    #endregion
}
Section 1: Code Reasoning

Alex Mariakakis

cse331-staff@cs.washington.edu (staff-wide)
INTRO + STORY TIME !!!
Reasoning About Code

- Two purposes
  - Prove our code is correct
  - Understand why code is correct
- Forward reasoning: determine what follows from initial conditions
- Backward reasoning: determine sufficient conditions to obtain a certain result
Forward Reasoning

\{x \geq 0, y \geq 0\}

y = 16;

\{x \geq 0, y = 16\}

x = x + y

\{x \geq 16, y = 16\}

x = \text{sqrt}(x)

\{x \geq 4, y = 16\}

y = y - x

\{x \geq 4, y \leq 12\}
Forward Reasoning

\{true\}

if \( (x > 0) \) {
    \{x > 0\}
    abs = x
    \{x > 0, abs = x\}
}
else {
    \{x <= 0\}
    abs = -x
    \{x <= 0, abs = -x\}
}
\{x > 0, abs = x OR x <= 0, abs = -x\}
\{abs = |x|\}
Backward Reasoning

\{x + 3b - 4 > 0\}

a = x + b;

\{a + 2b - 4 > 0\}

c = 2b - 4

\{a + c > 0\}

x = a + c

\{x > 0\}
Backward Reasoning

\{y > 15 \mid \mid (y \leq 5 \land \land y + z > 17)\}\n
if (y > 5) {
    \{y > 15\}
    x = y + 2
    \{x > 17\}
}
else {
    \{y + z > 17\}
    x = y + z;
    \{x > 17\}
}
\{x > 17\}
Implication

- Hoare triples are just an extension of logical implication
  - Hoare triple: \{P\} S \{Q\}
  - P → Q after statement S
- Everything implies true
- False implies everything

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P → Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
Weaker vs. Stronger

• If $P_1 \rightarrow P_2$, then
  o $P_1$ is stronger than $P_2$
  o $P_2$ is weaker than $P_1$

• Weaker statements are more general
• Stronger statements are more restrictive
Weaker vs. Stronger

\[
y \geq 16 \quad y = 16
\]

\[
x \text{ is even, } y = x + 1 \quad x \text{ is even, } y \text{ is odd}
\]

“Alex is an awesome TA” “Alex is a TA”
Weakest Precondition

• The most lenient assumptions such that a postcondition will be satisfied
• If $P^*$ is the weakest precondition for $\{P\} S \{Q\}$, then $P \rightarrow P^*$ for all $P$ that make the Hoare triple valid
• Notation: $WP = \text{wp}(S, Q)$
Weakest Precondition

\[ \text{wp}(x = y \cdot y, x > 4) \]
\[ |y| > 2 \]

\[ \text{wp}(y = x+1; z = y-3, z = 10) \]
\[ \text{wp}(y = x+1, \text{wp}(z = y-3, z = 10)) \]
\[ \text{wp}(y = x+1, y-3 = 10) \]
\[ \text{wp}(y = x+1, y = 13) \]
\[ x = 12 \]