A novel working scheme for program repair

**Generate and Validate**

- Generate using a SAT solver
- Validate using an SMT solver

**Analyze error using error generalization**

\[
\text{int } \text{max1}(\text{int } x) \{ \\
\quad r = 1; \\
\quad \text{if}(x > 0) \\
\quad \quad x > 0, \quad x \leq 0, \quad x \geq r, \quad x + r \geq 0 \\
\quad \quad r = x + 1, \quad r = x - 1, \quad x = r, \quad r = x \% 2 \\
\quad \text{assert}(r > 0); \\
\}\n\]

1. \(x \leftarrow 0, r \leftarrow 0\), \(r = 0\)
2. \(x \leftarrow 0, r \leftarrow 0\), \(x > 0, \ r \leftarrow 0\), \(r = x\)
3. \(x \leftarrow 0, r \leftarrow 0\), \(x \leftarrow 0, r \leftarrow 0\), \(x > 0, \ r \leftarrow 0\), \(r = x \% 2\)

**Analyze error in other ways**

- Prune the search space differently:
  - Remove all programs that fail on same input
  - Remove all programs that fail on same path
  - Remove some (not all) programs
  - Remove programs that only might fail
- Update search priority:
  - Prioritize all locations along the path
  - Use fault localization for the trace, and prioritize resulting locations
  - Prioritize certain changes over others, based on observations in error traces

**Learn from your mistakes**

- Bat-Chen Rothenberg and Orna Grumberg
- Technion – Israel Institute of Technology