Learn Your Program
Hadar Frenkel*, Orna Grumberg, and Sarai Sheinvald
Computer Science Department, Technion — Israel Institute of Technology

Goal
• Synthesis framework for first-order LTL formulas over program variables
• Infer program states using automata learning
• Infer program statements using abduction

Specifications
Quantifier-free first-order LTL formulas:

\[ \text{spec} = (x = 0) \land \text{Globally}((x = 0) \rightarrow \text{Finally}(x > 0)) \]

\(x\) is the program variable

Program Alphabet
We infer program statements in two ways:
1. Syntactic inference of program statements out of the specification:
   \( x := 0 \) from \( x = 0 \)
   \( x := 1 \) from \( x > 0 \)
   \( \text{if} \ (x > 0) \ldots \text{else} \) from \( x > 0 \)
2. Semantic inference using abduction, in case the statements obtained in (1) are not enough.
   Q: Are there cases in which (1) is not enough?

L* Algorithm [Angluin 1987]
Learns an automaton for a regular language \( L \) using membership queries: is \( w \in L \) ?
and equivalence queries: does \( L(C) = L \) for candidate \( C \)?

Membership queries:
Is the predicates sequence \(<p_1><p_2>\ldots<p_n>\) in \( T_{\text{spec}} \) ?

| \(<\bot>\) | No |
| \(<\bot> x:=0 <x=0>\) | No |
| \(<\bot> x:=0 <x=0>\) | Yes |
| \(<x=0> x:=x+1 <x>0>\) |

Equivalence queries:
For a candidate program \( P \), we check if \( L(P) \subseteq L(\varphi) \)

Q: How do we obtain “interesting” programs?
Q: How do we avoid vacuous results?

Termination: when does the process converge into a candidate automaton?
Hoare triplets inference: how do we infer predicates?

* This research was partially supported by the Technion Hiroshi Fujiwara Cyber Security Research Center and the Israel National Cyber Directorate.