MOTIVATION

Limitations of automated software verification
- restricted expressiveness
- imprecision (due to approximations)
- requiring expert knowledge for
  - finding invariants or
  - generating proofs

GOAL

Automated Theorem Proving (ATP) for Software Verification

- Full First-Order Logic with Theories: Superposition-Based ATP
- Program semantics that allow expressing program variables over program locations and timepoints: RAPID Framework.
- Automating Inductive Reasoning over Loops with Trace Lemmas: necessary instantiations of the induction axiom to automate proof search

Verification Task

- Imperative Program Reachability or Relational Property
- Custom Program Semantics
- Automatically generated Trace Lemmas

1. func main() {
2.   Int[] a;
3.   const Int alength;
4.   const Int v;
5.   Int i = 0;
6.   while (i < alength) {
7.     a[i] = v;
8.     i = i+1;
9.   }
10. }

Let's prove that the array \( a \) is initialized with some integer value \( v \)...

\[ \forall \text{pos} \in \mathbb{N} (0 \leq \text{pos} \land \text{pos} < \text{alength} \land \text{alength} > 0) \rightarrow a(\text{end}, \text{pos}) = v \]

What the prover doesn't know is that due to the incrementation of \( i \) from 0 to \( \text{alength} - 1 \), every position of \( a \) in this range will be affected by exactly one of the loop iterations.

What the prover needs to know to show is that \( i \) will have the value of every integer in this range at least once, formalized by the following trace lemma:

\[ \forall i_1 \cdot ((i(l_7(\text{zero})) \leq x \land x < i(l_7(\text{lastIt})) \land \forall i_1 \cdot i(l_7(s(i_1))) = i(l_7(i_1)) + 1)) \rightarrow \exists i_1 \cdot i(l_7(i_1)) = x \land x < \text{lastIt} \]

CONCLUSION

Program semantics with timepoints are powerful: loop nesting, relational verification.

Allow for automated inductive reasoning in combination with trace lemmas.

Trace lemmas: automatically generated inductive properties for program variables instead of program-specific loop invariants.