2LS: Memory Safety and Non-Termination (Competition contribution)

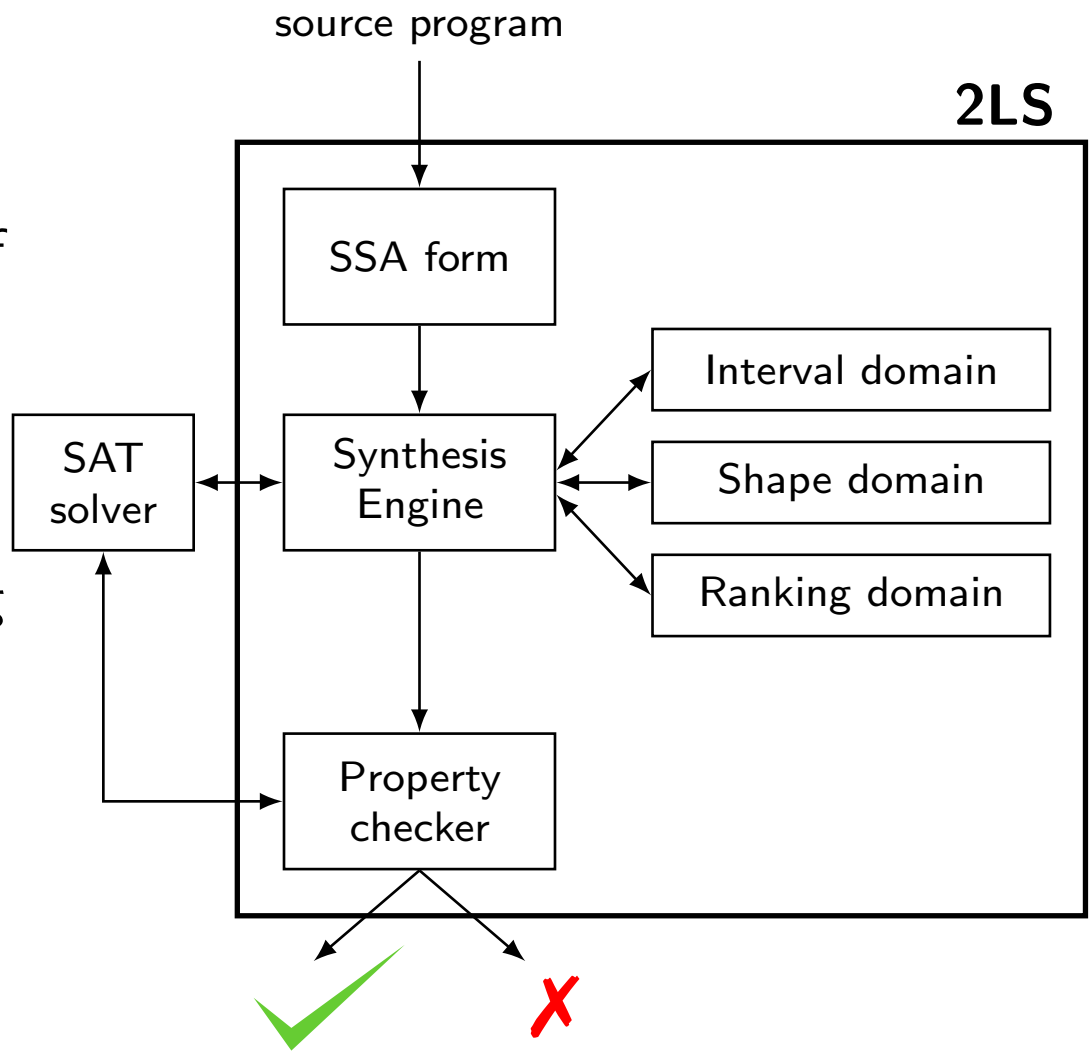
Martin Hruška[3], Viktor Malík[1,3],
Peter Schrammel[1,2], and Tomáš Vojnar[3]

[1]DiffBlue Ltd, Oxford, UK [2]University of Sussex, Brighton, UK
[3]Brno University of Technology, Brno, CZ

April 6, 2019

Program Verification in 2LS

- Framework combining multiple analysis techniques together
- Uses template-based synthesis of invariants and ranking functions
- Competition version performs monolithic analysis
- Capable of verifying and refuting assertions and termination
- New features: **domain combinations** and improved **memory safety**
- Based on CPROVER



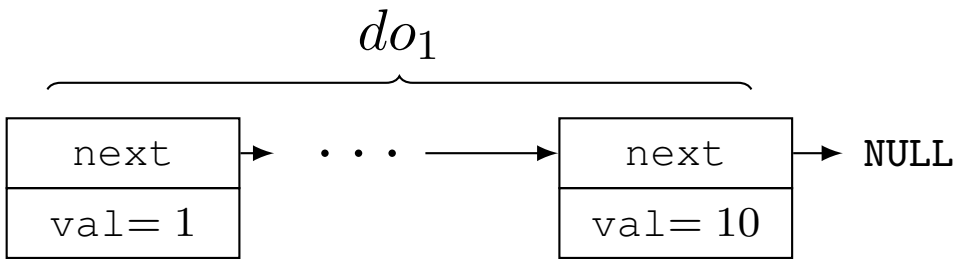
Improved Shape Analysis

- We implemented an improved:
  - **Memory model** – all objects allocated by a single `malloc` are abstracted by a finite number of *abstract dynamic objects*
  - Representation of **memory operations** in the SSA form
  - Abstract **shape domain** based on computing the *points-to relation* between memory objects
- These allowed us to use **combinations of abstract domains**
- Based on our recently published work:

  Template-Based Verification of Heap-Manipulating Programs (FMCAD'19)

Domain Combinations

- Each abstract domain has a form of a template – a formula in first order logic over bitvectors.
- We combine **shape** and **interval** abstract domains – the resulting formula is a conjunction.
- For example, an unbounded linked list:



can be described by a formula:

$$\underbrace{(do_1.next = \&do_1 \vee do_1.next = \texttt{NULL})}_{\text{shape part}} \wedge \underbrace{do_1.val \in [1, 10]}_{\text{interval part}}$$