Automated N-Version Programming with LLMs

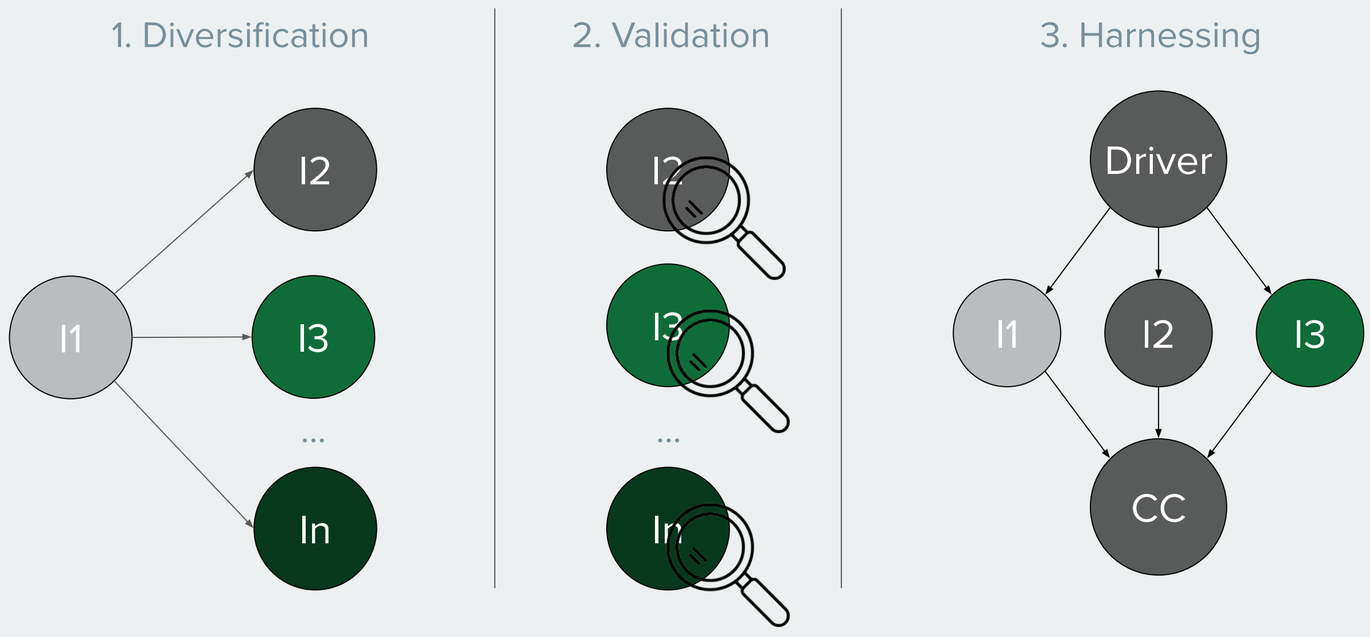
Javier Ron Arteaga <javierro@kth.se>
School of Electrical Engineering and Computer Science, KTH

1 Context

N-Version programming is a software development approach, which consists of creating N implementations or versions of a specific program [1]. When these versions are simultaneously executed, errors can be timely detected and mitigated by comparing their outputs. are avoided. [2].

2 Contribution

To facilitate automatic generation of N-Version programs, we design, implement and evaluate Galapagos: a tool for automa ed, verified N-Version programming. Galapagos consists of a pipeline with 3 passes: *Diversification*, *validation* and *harnessing*.



First, the Diversification pass generates different variants of the reference (I1) by calling an LLM API. Second, the collection of generated variants (In) is passed to the Validation pass, which filters out non-viable variants. The Validation pass proceeds through a sequence of validation steps: compilation, testing, and formal equivalence check. Last, the Harnessing pass uses the resulting variants to assemble an executable, where the original function is replaced by an N-Version implementation of that function with corresponding driver and cross-check (CC) points.

3 Evaluation

Galapagos is evaluated in 4 dimensions: (1) to what extent is it able to produce equivalent code variants with LLMs; (2) to what extent is it able to produce code variants which are diverse both on disk and at execution-time; (3) can it harden critical sections of software against miscompilation bugs, and; (4) how much performance overhead does it introduce.

4 Results

Equivalent variant generation	***Behavioral diversity of variants***	***Miscompilation mitigation***	***Performance Overhead***
600 variants generated from 30 reference programs	170 equivalent variants analized from Diversification pass	3 miscompilation bugs selected from the Clang compiler	23 N-Version programs from Harnessing pass
170 equivalent variants, at least 1 equivalent variant for 23 programs	127 unique variants, at least 1 unique variant for 23 programs	3 Miscompilation bugs mitigated	Performance overhead close to 1x per additional version in average

WASP WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM