SafeRiver
SAFETY & SECURITY FORGE

SAFEPROVER
FORMAL PROOF FOR C LANGUAGE

SafeRiver has developed an efficient formal verification tool named **SafeProver**, which performs both:

- **reachability analysis** using **Bounded Model Checking (BMC)** as a refutation technique for counterexample detection, and
- **invariant satisfaction analysis** using a set of algorithms derived from the **K-Induction principle.**

SafeProver enables the **formal and exhaustive analysis** of any **discrete-time, bounded system.**
A bounded system is defined as follows:

1. the system must be specified using **fixed-size data types** (e.g., fixed-width integers and bounded arrays);
2. the system's variables **cannot change size between execution cycles**;
3. loops must be **statically bounded** with a maximum number of iterations;
4. recursive functions, if any, must be **bounded and guaranteed to terminate** (i.e., no cyclic dependency).

SafeProver was **T2 qualified for SIL4** in 2020.
It verifies properties on the input model after detecting **runtime errors (RTEs)** such as:

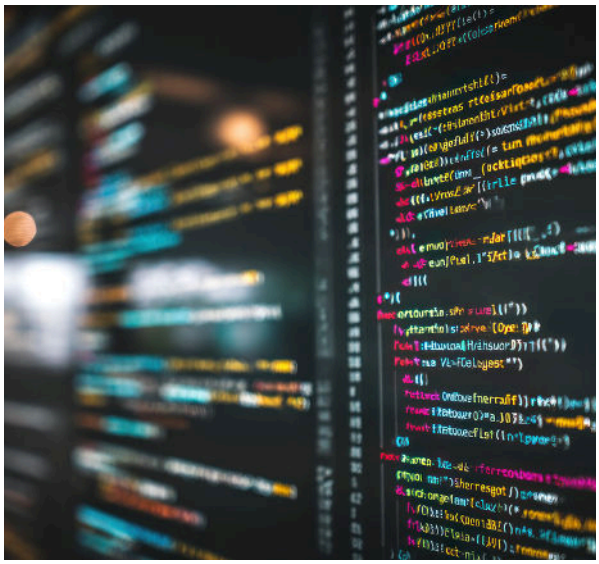
- out-of-bounds access,
- division by zero,
- and contradictions between the model and the properties.

If an RTE or contradiction is detected, the SafeProver engine does not run the proof, ensuring that properties are not proven on a poorly constructed model.
If the model is coherent, SafeProver proves valid properties, detects falsified properties and shows the corresponding counter example.

SAFEPROVER FOR C LANGUAGE

SafeProver has recently been extended to allow **C language property proofs** on C models, which may be generated from higher-level models.
The low-level (C) model must comply with the **bounded system constraints** imposed by SafeProver.
Some C constructions, such as **assembly code** or **function pointers**, are not supported.

C model checkers such as **BLAST** (University of Berkeley) and **CPAChecker** (Cornell University) support C as a **modeling language**, but not as a **property language.**
SafeProver supports **C for both the target model and property descriptions.**



SAFEPROVER IN A MODEL-BASED DESIGN (MBD) CONTEXT

SafeProver for C is currently under evaluation in an **industrial Model-Based Design (MBD)** context.
MBD is an approach that starts with a paper description of a system and ends with a **digital model.**
It requires teams to maintain a **single central model** covering all development aspects: requirements tracking, design specifications, implementation, verification, and validation (V&V).

MBD is used for complex systems in domains such as **aerospace, defense, energy, railway, and automotive**.
It **increases engineering productivity, reduces costs**, and **shortens development cycles.**

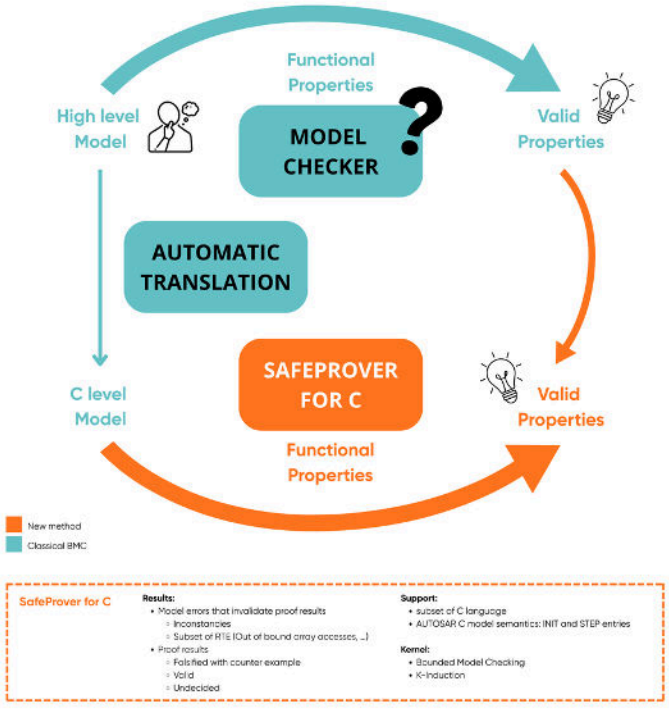
Operationally, MBD separates the **functional view** from the **software view:**

- the functional view is handled by domain experts,
- while the implementation is **automatically generated from the model.**

Advantages include:

1. domain expertise is concentrated at the model level,
2. software can be automatically generated in multiple languages without modifying the model-only the code generator needs to be rerun.

If the **code generator is qualified or certified,** the generated source code is **proven** equivalent to the model, so V&V performed at the model level applies directly to the code.
If the code generator is not qualified or certified, equivalence must be proven at the code level by verifying the same functional properties on the low-level (generated) code.



INDUSTRIAL APPLICATION AND RESULTS

SafeProver for C has been applied to **industrial-size AUTOSAR C source codes** (2,500–8,000 lines) generated from **Matlab/Simulink models.**

The experiments confirmed that the generated models are **bounded systems:**

- they do not use unsupported C constructs,
- AUTOSAR semantics is preserved: non-generated Rte functions are **semantically stubbed,** and calls to initialization (init) and task computation (task) comply with C AUTOSAR model semantics.

Verification of industrial complexity properties revealed:

1. **behavioral discrepancies** between generated code and the original model,
2. **differences in execution context** between the environments where the code is tested and the model is verified.

SafeProver for C is currently being **industrialized** and will be available for further industrial experiments next year.

MODEL-TO-CODE EQUIVALENCE VERIFICATION

When the **code generator is qualified or certified,** the generated code is **proven equivalent to the model,** and model-level V&V applies directly to the code.

When the generator is **not qualified**, equivalence must be **proven.**
Here, the **high-level model** refers to the modeling language description, and the **low-level model** refers to the generated source code.

The low-level model must be **RTE-free** and meet system requirements.
One way to guarantee equivalence is to **verify all functional properties proven at the high-level model on the low-level model.**
Proof correctness is only guaranteed if each property is generated using the **same code generator** that translated the model. Variables must be coherent between the property and the model.
Properties can also be **handwritten from the model-level properties,** but equivalence between model-level and code-level properties must be verified.

Christèle FAURE | Tools expert | c.faure-SAFERIVER@serma.com
SafeRiver - A SERMA Safety and Security Company – www.serma-safety-security.com