

Automated Parameterized Verification of a Railway Protection System with Dafny

Roberto Cavada, Alessandro Cimatti, Alberto Griggio,
Christian Lidström, **Gianluca Redondi**, Giuseppe Scaglione,
Matteo Tessi, Dylan Trenti

July 24th, Zagreb, CAV 2025



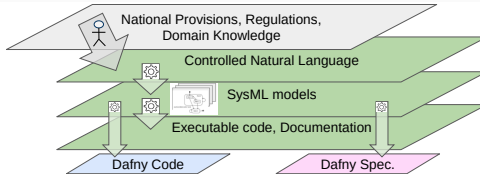
AIDA: Model-Based Design of Railway Logics

Inputs

- DSL (domain specific language)
- Classes
- Transitions (guards and effects)

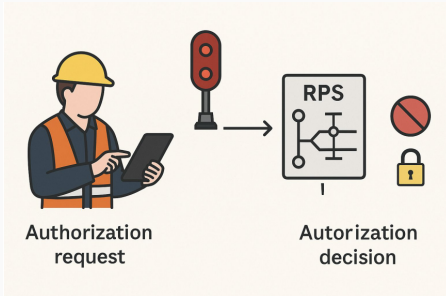
Outputs

- Documentation
- C for execution
- Python for debugging
- **Dafny for verification**



- **Dafny** = imperative/O-O language *with contracts*
 - `requires/ensures` (pre/post)
 - `modifies/reads` frames
 - Loop invariants, quantifiers, ...
- Verification via VC generation $\rightarrow$ Boogie $\rightarrow$ SMT (Z3)
- **AIDA auto-generates** all contracts for the Dafny implementation of a logic

Railway Protection System (RPS)



- 12 classes
- 150 variables, 138 transitions
- ~11K lines of C
- ~17K lines of Dafny
- Specifications = 52% of code

The RPS logic is **parameterized**: it can be deployed in several configurations.

Generated Dafny Code Example

DSL: *'Check that all inputs from the signals are safe'*

Transition Guard Method (auto-generated)

```
method from_Free_to_Authorize_guard() returns (b : bool)
ensures (∀ c :: c in Signals ⇒ c.input = Safe) ⇔ b {
  b := true;
  for i:= 0 to |Signals|
    invariant (∀ c :: c in Signals[..i] ⇒
c.input = Safe)) ⇔ b {
    if (Safe[i].input != Safe) {
      b := false; break;
    } } }
```

Contracts and loop invariants are automatically generated from the AIDA description of the RPS.

Goal: Verify safety properties over *all* valid configurations.

Configuration Axiom: Track has a Signal

```
predicate ValidConfiguration() {  
   $\forall t :: t \text{ in Tracks} \Rightarrow$   
     $\exists s :: s \text{ in Signals} \ \&\& \ s \text{ in } t.\text{Signals}$   
}
```

Property: No Unsafe Access

```
predicate NoUnsafeAccess() {  
   $\forall t : t \text{ in Tracks} ::$   
    ( $\exists s :: s \text{ in } t.\text{signals} \ \&\& \ s.\text{input} = \text{Unsafe}$ )  $\Rightarrow$   
    ( $\forall r :: r \text{ in } t.\text{requests} \Rightarrow !r.\text{granted}$ )  
}
```

The verification consists of two main stages:

1. Local Verification:

- Check that each method (transition) satisfies its auto-generated contract

2. Global Verification:

- Prove that each safety property is an inductive invariant
- I.e., the property holds initially and is preserved by every transition

Proof Obligation (Hoare-style):

$$\{\text{Prop} \wedge \text{Guard} \text{ Contract} \wedge \text{Configuration Axioms}\} \text{ Effect } \{\text{Prop}\}$$

1. Opaque Blocks

- Ignore non-relevant bodies during verification
- 2–5× speed-up on affected methods (via VC simplification)

2. Singleton Optimization

- Replace 1-element sequences with single objects
- Eliminates quantifiers → up to 10× faster
- Applied to 17/26 lists in the RPS logic

Verification Results

- 21 safety properties proved, extracted from regulations
- 4 required manual strengthening (initially not inductive)

Category	Avg. time (s)	Avg. time – optimised (s)
Local contracts (for a class)	1 699	493
Global props. (inductive, 17/21)	1 591	147
Global props. (non-inductive, 4/21) *	22 069	496

* after lemma strengthening

Conclusions & Future Work

Achievements

- Formal proofs for parameterized RPS
- 21 safety properties verified
- Optimisations lead to significant speed-ups

Next Steps

- Extend to other logics
- Automatic invariant generation
- Multi-backend co-simulation

Thank you!

Questions?

Per-Class Verification Stats

Table 1. Verification statistics for embedded contracts

Class	Methods	Base				W/ Opaque				W/ Singleton			
		Time	Ass.	T/O	Loops	Time	Ass.	T/O	Loops	Time	Ass.	T/O	Loops
1	22	14s	79	0	16	14s	79	0	16	12s	79	0	6
2	11	1,068s	40	0	21	984s	40	0	21	914s	40	0	21
3	30	110s	123	0	22	68s	123	0	22	36s	123	0	0
4	34	919s	141	0	25	98s	141	0	25	34s	141	0	0
5	50	3,615s	213	0	25	161s	213	0	25	43s	213	0	0
6	26	13s	104	0	8	13s	104	0	8	13s	104	0	0
7	22	24s	79	0	11	15s	79	0	11	14s	79	0	0
8	12	9s	41	0	0	9s	41	0	0	9s	41	0	0
9	52	24s	175	0	0	25s	175	0	0	23s	175	0	0
10	126	799s	531	0	166	636s	531	0	166	642s	543	0	154
11	12	6,733s	45	2	47	6,935s	45	2	47	563s	57	0	0
12	12	7,059s	45	2	40	7,096s	45	2	40	3,612s	53	0	4

Table 2. Verification times for safety properties (non-inductive ones marked with *)

Properties 1–7			Properties 8–14			Properties 15–21		
ID	Base	Opt.	ID	Base	Opt.	ID	Base	Opt.
1*	2,144s	462s	8	98s	77s	15	675s	178s
2	551s	188s	9	104s	78s	16*	1,394s	542s
3	7,150s	193s	10	101s	68s	17	3,278s	189s
4*	23,474s	433s	11	108s	65s	18	4,004s	189s
5	106s	68s	12	110s	64s	19*	61,265s	547s
6	101s	77s	13	110s	63s	20	558s	188s
7	101s	77s	14	9,385s	180s	21	513s	558s