Verification of SCADE models with S3 model-checker

Marielle Petit-Doche, Matthias Güdemann, Roméo Courbis
Systerel

November 16, 2015

Abstract

This document describes the verification and validation processes applicable to SCADE models using the S3 model-checker.

Contents

1 Introduction

This document gives a description of the VnV process applied on a Scade design model. The Scade model covers two functions of the ETCS on-board unit:

Level Management function , described in SRS-26 §5.10

Mode Management function , described in SRS-26, §4.6, §5.4, 5.6, 5.7, 5.9, 5.11, 5.13, 5.19

2 Verification processes applicable to a SCADE model

The principe of verification consists in the definition of properties in textual languages, and verification of these properties by model-checking techniques on a textual translation of the Scade Model.

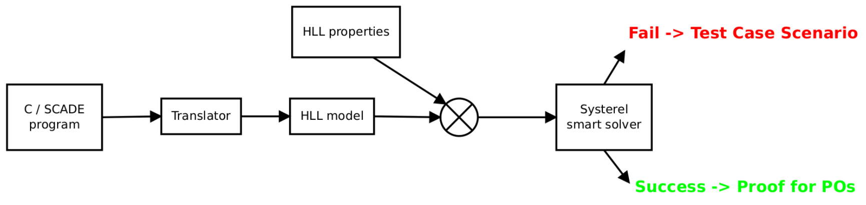

Figure 1: Procedure of verification of Scade model

Figure 1 describes the process:

- the main input is a Scade model to verify (same approach and tool can be applied to a C program) which is translated in a High Level Language (a textual description of the model)
- the second input is the properties to verify, written in HLL
- these both inputs are merged in a unique HLL file, used directly as input of the Systerel Smart Solver (S3 model-checker) for verification
- result of the S3 tool is Success or Failure; in case of failure counter-example is provided for analysis.

The S3 tool is a model-checker which manages as well an internal SAT solver as external SAT solvers.

This process can be apply to cover three purposes:

Properties of an HLL Model: The prover may be used to prove or disprove properties of an HLL model. Those properties are modeled as proof obligations.

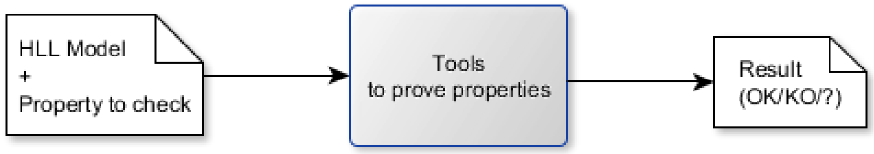

Figure 2: Properties of an HLL Model

Solving Properties of an HLL Model: The prover may be used as a solver, by finding values for the streams that comply with some property P. To do so, just put the negation of P as a proof obligation. If the prover succeeds to disprove the proof obligation, it will provide a solution solving P. If the prover succeeds to prove the proof obligation, then this proves that the property cannot be solved. This solution can be used to defined test cases too.

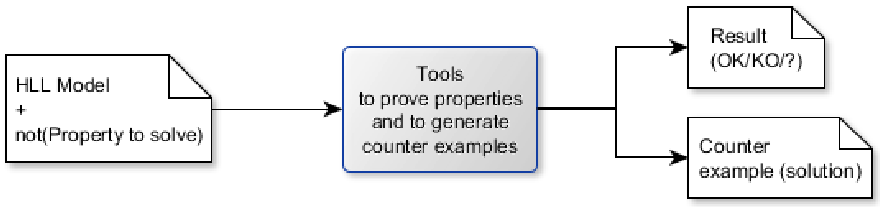

Figure 3: Solving Properties of an HLL Model

Proving the Equivalence of 2 HLL Models: The prover may be used to prove that 2 HLL models with the same interface (the same input streams and output streams) are equivalent. To do so, an equivalent model is produced with proof obligations stating that for all input streams values, each output of the first HLL model is equal to the output of the second model with the same name.

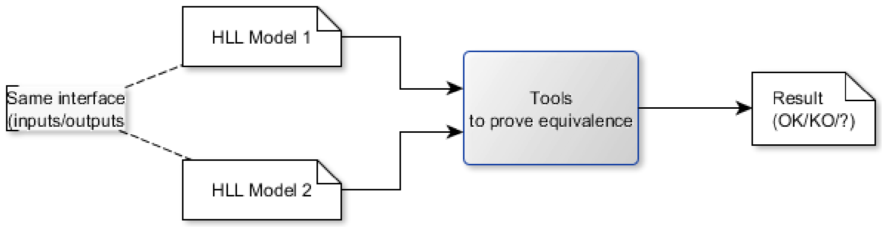

Figure 4: Proving the Equivalence of 2 HLL Models

3 Specified properties and results

Name	Type	Model	SRS coverage	section
isolate	simple proof	Modes	4.6.2 C1	
no_power	simple proof	Modes	4.6.2 C4	
level case	use case	Level	5.10	
shunting initiated by driver	validation	Modes	5.6	
start of mission	validation	Modes	5.4	

3.1 Application of S3: Static analysis

At first, the formal verification of the model was used to find bugs in the developed SCADE model. S3 adds some proof obligations to assess that the HLL model is correctly defined:

- Indexes of arrays belong to their ranges
- Latch definition range check
- No division by 0
- No overflow and no underflow on arithmetic expressions
- Output and constraint initialization check

Besides, the translators from a language to HLL can also generate proof obligations to be analyzed by S3, to check that the code does not have undefined behavior with respect to the source language.

For example the C-translator adds some proof obligations to ensure conformance with the C99 standard.

Results The three models have been verified on their topnode:

Scade model	Top Node	Results
Modes	ManageModes	PO 1-5: valid
Levels	ManageLevels	PO 1-5: valid
ModesAndLevels	ManageLevelAndMode	PO 1-5: valid

3.1.1 Conditions to Isolate mode

Files used for the proof The proof is defined in the file `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/05-Work/S3/Small_Proof/isolate.hll` and it is verified on the top node *ManageModes* of the SCADE model *Modes*.

What is proved ? The Condition 1 of SRS § 4.6 "The driver isolates the ERTMS/ETCS on-board equipment" is proved, ie; as soon as the input of SCADE model `Data_From_DMI.'ETCS_Isolated'` becomes true, the output `currentMode` becomes 'isolated' (`Level_And_Mode_Types_Pkg::IS`) and internal condition is activated.

Constraints used None.

Results The property is proved.

3.2 Verification of use case

levels

Files used for the proof The proof is defined in the file `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/05-Work/S3/Proof_SoM/shunting_initiated_by_driver.hll` and it is verified on the top node *xxx* of the SCADE model *Level Management*.

What is proved ?

Constraints used

Results

3.3 Validation of informal specification

3.3.1 Procedure Shunting initiated by Driver

Files used for the proof The proof is defined in the file `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/05-Work/S3/Proof_SoM/shunting_initiated_by_driver.hll` and it is verified on the node *Procedure_SH_Initiated_By_Driver* of the SCADE model *Modes Management*. The same proof is also defined in the file `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/05-Work/S3/Proof_SoM/shunting_initiated_by_driver_topnode.hll` to be verified on the top node *ManageModes* of the SCADE model *Modes Management*.

What is proved ? We want to prove that the procedure SH_Initiated_By_Driver is a correct implementation of the section 5.6 Shunting Initiated By Driver of SRS-26.

To prove this, a specification of the flowchart is proposed in the property file. However, the flowchart is not entirely specified: elements `D030`, `A030`, `A095`, `S100` and `A115` of the flow chart in SRS-26 are out of the scope of the mode management function.

Constraints used One hypothesis is used in this model to avoid a counter-example: when we are in `A100` the request of "End of Mission" procedure correspond to the value of the input `On-going Mission` as it is specified in the SCADE model.

This hypothesis is justified by the fact that, according to `A050`, `D040` and `A100`, if the input `On-going Mission` is `True` then the "End of Mission" request is `True`, so equal to `On-going Mission`. Also, as transition to SH mode is enable (`A050`) when "End of Mission" request is made, the system should go to SH mode (or another mode except SB mode).

Results Considering the constraint and our model, the SCADE model of SH_Initiated_By_Driver corresponds to the specification. Proof of this property allow to detect an error in SCADE model: operator *AND* was used instead operator *OR*. Besides the specification in SCADE of the computation of the output *End_Of_Mission* was corrected.

3.3.2 Procedure Start of Mission

Files used for the proof The proof is defined in the file `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/05-Work/S3/Proof_SoM/startofmission_topnode_proof.hll` and it is verified on the node *Procedure_StartOfMission* of the SCADE model *Modes*.

What is proved ? We want to prove that the procedure Procedure_StartOfMission is a correct implementation of the section 5.6 Procedure Start of mission of SRS-26.

Only the down part of the flowchart, from S10 and from S20, relative to the modes management, is specified in the SCADE model.

Constraints used

1. Level should not change : Level can be change at the beginning of Start of Mission procedure, but when we go in the step of mode selection, the level should not change.

2. Train Data should not change : Train Data are validated at the beginning of Start of Mission procedure, they shall be valid to start mode selection and we consider then that their validity should not change.

3. The train shall stay at standstill during all the procedure. Indeed in Stand-By mode, standstill shall be ensure by supervision function (see SRS-26 §4.4.7.1.5).

4. The "On Going Mission" variable, input of SH Initiated by Driver, is forced to False. This is justified by SRS 5, section "5.4.6 Entry to Mode Considered as a Mission".

Results Considering constraints and our HLL model, the SCADE model of Procedure Start of Mission corresponds to the specification. Some errors have been detected in the SCADE model: links between nodes were wrong.

3.3.3 Procedure On-Sight

On going work

Files used for the proof The proof is defined in the file `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/05-Work/S3/Proof_SoM/xxx` and it is verified on the top node *xxx* of the SCADE model *Modes Management*.

What is proved ?

Constraints used

Results

3.4 Comparison of Scade models

levels

Files used for the proof The proof is defined in the file `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/05-Work/S3/Proof_SoM/shunting_initiated_by_driver.hll` and it is verified on the top node *xxx* of the SCADE model *Level Management*.

What is proved ?

Constraints used

Results

4 Conclusion