The SafeCap trajectory: industry-driven improvement of an interlocking verification tool (Industrial paper)

Alexei Iliasov[1], Dominic Taylor[2], Linas Laibinis[3], and Alexander Romanovsky[1,4]

[1] The Formal Route, UK
[2] Systra Scott Lister, UK
[3] Institute of Computer Science, Vilnius University, Lithuania
[4] School of Computing, Newcastle University, UK

Abstract. This paper reports on the industrial use of our formal-method based interlocking verification tool, called SafeCap, and on what we needed to change in SafeCap as a result of our experience in applying it to a large number of commercial signalling projects. The substantial efforts dedicated to tool improvement are caused by the novelty of the technology and by a substantial gap to be bridged between the academic prototype, developed initially, and the industry-strength tool SafeCap has become now. It is our belief that when such innovative tools and technologies are developed for industrial use it is often impossible to fully understand and correctly elicit the complete set of requirements for their development. The paper describes the extensions added and the modifications made to the functionality of SafeCap after it was demonstrated to be successful in a number of real (but not live) signalling projects and, as a result of this, was formally approved for use in the UK railway. We believe this experience will be useful for the developers of formal verification methods, tools and technologies to be deployed in industry.

Keywords: Safety verification · Railway · Automated theorem proving · Scalability · Industrial deployment · Solid State Interlocking.

1 Introduction

Effective signalling is fundamental to the safe and efficient operation of railway networks. At the heart of any signalling system there are one or more interlockings. These devices constrain authorisation of train movements as well as movements of the infrastructure to prevent unsafe situations arising. One of the earliest forms of computer-based interlocking was the Solid State Interlocking (SSI) ([4], [1]), developed in the UK in the 1980s. Currently, SSI is the predominant interlocking technology used on UK mainline railways. It also has applications overseas, including in Australia, New Zealand, France, India and Belgium. Running on bespoke hardware, SSI software consists of a generic application

(common to all signalling schemes) and site specific geographic data. The latter configures a signalling area by defining site specific rules, concerning the control of the signalling equipment as well as internal latches and timers, that the interlocking must obey. Despite being referred to as data, a configuration resembles a program in a procedural programming language and is iteratively executed in a typical loop controlling the signalling equipment.

This paper discusses the SafeCap tool used for formal verification of geographical data of SSI and derived (Smartlock and Westlock) interlockings [8], [9], [6]. It provides scalable and fully-automated verification by mathematical proof. The tool is now actively used for safety verification of mainline interlockings by several UK signalling companies. SafeCap is deployed at different phases of SSI preparation to check the designs and to provide signalling engineers with the diagnostics to help them to enhance (and, where necessary, to correct) the design data. It has proven to be successful in improving the safety of the signalling systems and reducing the rework cycle. SafeCap is extremely efficient, it takes but few minutes to verify the safety of any UK interlocking on a fairly standard desktop computer.

The development of SafeCap started in 2013 in a series of public projects led by Newcastle University. It was originally conceived as an experimental extendable Eclipse-based tool for railway simulation and verification.

From 2017 we have focused our work exclusively on developing a fully-reworked version of SafeCap targeting SSI verification (described in [6][1]). Two main decisions leading to the industrial adoption of the tool were made during this period, namely,

- to focus on fully-automated scalable verification by mathematical proof;
- to ensure that the tool inputs the data developed by signalling engineers and outputs the diagnostics reports exclusively presented in terms of these data.

These decisions in effect allowed us to successfully address the majority of the factors limiting the acceptance of formal methods by industry (see, for example, Section 5.5 of paper [5]).

Since 2019 we have been conducting commercial projects and making substantial improvements to SafeCap. This has been a period of extensive use, during which we have verified over 60 interlockings in England, Wales and Scotland, including those controlling the most complex layouts in London and Birmingham. We have worked with a mixture of SSI-derived interlocking technologies and layout complexities, written at different times by different design offices. In particular, during this time it has been demonstrated that SafeCap is now fully equipped for conducting formal, scalable and automated verification of SSI/Smartlock/Westlock interlockings developed for ETCS (European Train Control System). This has been an important period during which we have been using the experience gained in the commercial projects to enhance the tool and to make it more useful and attractive for our customers.

[1] The paper was prepared in 2020 and submitted to the journal in early 2021.

This industrial paper focuses on the changes made to improve SafeCap during the latter phase of its development and on the industry-driven research that formed the foundations for these improvements. The experience from commercial SafeCap applications gained during this phase has been a key for meeting the needs of the signalling companies that use SafeCap.

The rest of the paper is structured as follows. The next section discussed the three main categories of SafeCap improvements, providing inside information about each improvement implemented. Section 3 reports on the comparative statistics from the live commercial projects delivered before the improvement phase and after the improvements were implemented. The last section discusses some conclusions and future directions of our work on SafeCap.

2 Tool improvements

SafeCap is a novel technology for which it was not possible to define a full set of requirements at the time when we started its development. During this development, there has been a continuous process of eliciting new requirements and modifying many of the initial ones. The main improvements have necessarily been based on learning from experience.

SafeCap is a unique tool, based on substantial advances in formal methods and supporting verification tools, which is actively used by industry. This means that all its improvements are driven by the industrial needs, and that all research advances supporting these improvements are driven by these needs.

The three main categories of tool improvements discussed in this paper are improvements of safety properties (i.e., how SafeCap understands safety), of the employed proof systems (i.e., how SafeCap verifies safety), and of the produced diagnostics and reporting (i.e., how SafeCap reports the verification results).

2.1 Improvements of safety properties

Safety properties assert the safe behaviour of the interlocking system that SafeCap verifies. When the system is demonstrated to satisfy all these properties, it is considered to be safe. The properties play a critical role in SafeCap formal verification, which is why we put substantial efforts ensure they are correct, unambiguous and relevant for signalling engineers.

During the improvement period we have finalised and started using a systematic process for property engineering. The process we are applying now guides us during property identification, modification, documentation, testing and correction, and allows us to document all these steps to make the property development accountable.

In the railway signalling domain, opportunities to specify a complete set of safety properties from scratch are rare and typically only occur on newly build, stand-alone metro lines. In other cases, interlockings are usually constrained by legacy considerations. As SafeCap targets the safety verification of UK mainline interlockings, safety properties had to be inferred from the railway standards.

The argument for the correctness and completeness of these safety properties thus stems from their traceability to the standards and the pedigree of those standards, which have evolved over many decades through design, risk assessment, operational experience and lessons learnt from accidents. Identification of the safety properties for SafeCap starts with a thorough review of the relevant standards (we now use 22 UK standards) to elicit specific safety properties applicable to the interlocking functionalities that fall within the scope of SafeCap verification.

When defining safety properties, consideration needs to be given not only to the safety requirements contained in railway standards, but how those are actually implemented in data. Failure to do this correctly results in many false positives. The SSI data is effectively a computer program that determines site-specific functionality of the interlocking system. It includes files defining interlocking identities (files of the following types: FLG, PTS, QST, ROU, SIG, TCS) which define the names of various variables corresponding to trackside equipment, and files which define functionality in terms of those identities (FOP, IPT, MAP, OPT, PFM, PRR). During the improvement phase, in addition to identifying the new safety properties from the textual standards, we added a substantial number of new properties by importing new types of files (including, IPT/OPT files) which allowed us to formulate new properties in terms on newly added types of identities from these files and their functionalities. This includes, for example, checking the aspect level conditions in the IPT/OPT files.

In the last three years we have increased the number of safety properties SafeCap verifies from 40 to 136, which substantially improved the coverage and the quality of SafeCap verification.

Dealing with false positives has became an important part of safety property development due to the complexity of safety requirements described in standards, which often allow for many exceptions to general rules and how the properties are implemented in data. The false positives encountered during the SafeCap verification process arise for various reasons: over-simplification of the property expressed in the textual standard; failure to recognise and exclude scenarios that cannot plausibly happen in the railway domain; the effect of safety over-approximation in the formal representation of a property due to limited scope of a mathematical model of railway; and, lastly, due to limitations inherent to theorem proving in application to inductive safety invariant verification. The first three types of false positives are dealt with by improving the properties (which sometimes can increase their complexity and affect the verification time) and extending railway model. Dealing with false positives of the last kind is explained in more detail in Section 2.2, which also provides more information about how the SafeCap prover works.

Another aspect of property engineering is the need to have robust change management procedures. To this end, SafeCap uses regression testing to check that previously found, and manually confirmed, violations are still present and that no new violations appear after a property is revised. To achieve this, SafeCap reproduces the effect of a source code error seeding by directly manipulating

proof conjectures, generated from interlocking code (without any seeded errors) and the safety property itself. More details about that are given in Section 2.2.

In addition, due to the critical importance of the safety properties for SafeCap verification, traceability to the referred standards of all new and revised properties are now manually checked by an independent checker from our team.

2.2 Improvements of the proof system

As its primary means of mathematical proof, SafeCap employs a custom rewriting-based prover. A symbolic transition system, build via symbolic execution of input signalling data, is used to generated, by instantiation of a schematic axiom of inductive safety invariant, conjectures called *proof obligations*. Each such proof obligation attempts to establish that a particular state transition maintains safety invariant. A conjecture is a logical sequent, consisting of a list of hypotheses (derived from state transition pre-conditions) and a goal (derived safety invariant).

Proofs are computations over the conjectures that aim to find a chain of predefined rewrite steps resulting in a copy of a current goal in hypotheses or demonstration of falsity of hypotheses. The prover is completely automatic – it has a procedure for selecting a next rewrite step as well as for ignoring currently irrelevant hypotheses. For a select few situation, the prover employs a rule translating conjecture into a SAT problem and attempting to disprove the result via a SAT solver.

In addition to a verdict on whether a given conjecture is correct (i.e., a theorem), the prover also constructs the complete proof script detailing all computations carried out during the proof. It is not designed as a small kernel prover – we can freely add new rewrites steps without having to prove their correctness. Instead we rely on a generated proof script and an independent proof checker to demonstrate proof soundness.

The first-order logic underpinning the SafeCap mathematical notation is undecidable, hence there is no guarantee that the prover can deliver definite result within any fixed time. It is very rare that the prover can positively demonstrate that a conjecture is not a theorem; in most cases we simply let the prover exhaust all possible venues of proof and, if the proof is still not completed, declare the conjecture to be unproven. It is also possible to keep rewriting for an impractically long time because of, e.g., rewrite cycles emerging during the proof or when the number of options presented to the prover is simply too large to be ever explored. One typical case is the presence of disjunctive clauses with many composite parts (in the region of hundreds). In the absence of any other promising directions, the prover might be resigned to exploring all possible cases of the disjunctive cases until it is forcibly stopped.

Proof complexity ranges from a few steps for very simple cases to millions of steps for a few extreme situations. The median number, measured on a random sample of five projects, is 802 proof steps with 37 sub goals. The mean, however, is 2802 proof steps and 386 sub goals; hence, there is a relatively small population of much harder-to-prove conjectures.

The vast majority of proof conjectures we deal with are shown to be theorems. For the same sample of five projects, 99.7% of all the generated conjectures are shown to theorems. Thus, from the efficiency viewpoint, we aim to discharge simpler cases of provable conjectures in fewer steps and then spend more time on harder cases. Luckily, proofs of the same safety predicates tend to be very similar across different projects and this enables us to construct macro-steps that narrowly target certain patterns reappearing in the generated conjectures. Here a macro-step is a proof script that computes a sequence of rewrite rules for a given set of hypotheses and a goal. For most situations, the script simply encodes pattern matching on conjecture elements to detect certain recurring situations. In a small number of cases, scripts contain conditional steps.

In the end, we are able to discharge thousands of proof obligations within seconds. As a comparison, it takes the Why3 framework[3] (with the Alt-Ergo backend) around twenty seconds to discharge some simpler proof obligations.

Attempting to prove a conjecture is only the first step in the process of finding safety violations. A failed conjecture on its own only tells us that a certain safety property does not hold for given interlocking data. This is not useful to railway engineers, who expect to see a link to a specific line in the interlocking data as well as identification of relevant signalling plan elements. There is a world of difference between saying that one or more points are not deadlocked correctly somewhere in the data and a statement naming a specific set of points and giving the precise location in the interlocking data where the violation occurs.

To present a failed conjecture in the manner that is useful to railway engineers, SafeCap uses two techniques: (i) undoing some proof steps (without altering the overall verdict) to reach a state that is easier to analyse and explain, and (ii) extracting the context information from a conjecture goal and its hypotheses. The first technique undoes the proof steps in which variables or identifiers linking the proof to the signalling plan are renamed or removed. The second technique uses source code information provided by the data parser to locate relevant data and extract a human-readable summary of a failed proof state from the conjecture hypotheses (e.g. occupied tracks, locked sub routes and etc.)

The extracted proof state and source code tracing information are combined to generate a human readable summary of a found safety violation. This is formatted using the predefined templates to describe the purported violation. The summary contains, among other things, the location, in interlocking data, of the original source code of an offending state transition, highlights of relevant elements on the signalling plan, and the proof context summarising, in human readable terms, a set of signalling system states in which the violation occurs.

One common approach to identifying and reporting a property violation is positing any one example for which the given property does not hold. Our experience clearly shows that this is not sufficient for industrial scale verification. The first reason is that the engineers need to have all the contexts, in which the property does not hold, reported and corrected. The second one is that in large and complex systems, like railways, there are often a few exceptional cases where

the engineers intentionally violate a general property for operational reasons; for example, allowing a train to shunt backwards and forwards at low speed on a non-passenger line without the signaller setting routes for each move. This renders possible situation where a benign case of a violation is reported and while serious cases that need corrective actions are not. At the core of our solution developed during the improvement period and presented earlier [7], lies an approach to name all possible violations in the terms of a unique combination of signalling assets plus atomic section of signalling code. This makes it impossible to wrongly combine several distinct violations.

2.3 Improvement of reporting

The SafeCap diagnostics report is the only information we provide to clients. For each safety property, it identifies whether the interlocking data complies with that property and, if not, where violations were found. The improvements in the properties and proof system, described above, have enabled a step change improvement in the precision with which violations are reported.

Previously it was only possible to group apparently similar violations based on where they occurred in the code and the pre-conditions (context) under which they occurred. This led to occasional misgrouping of distinctly different errors as well as repetitive reporting of the same error, both of which made reports difficult and time-consuming to analyse. With the new approach [7], what constitutes an individual property violation is now clearly defined and thus instances of the same violation identified by the proof system are grouped accordingly. Each individual violation is separately reported making reports longer than they were, but much easier to analyse. The approach also makes the reports consistent in the labelling of individual violations, facilitating comparisons between reports on different versions of the same interlocking data. This enables enumeration of all potential errors in a given interlocking data independently from proof obligation generation. This in turn enables us to detect missing (or extraneous) proof obligations.

Further improvements have been made in the diagnostics information provided for property violations. The SafeCap philosophy has always been to present findings in terms understood by signalling designers with no reference to the formal methods, intermediate formal notations, or generated prover outputs. Instead property violations are presented as textual descriptions accompanied by the section of code and graphical layout to which the violation applies and the failed proof states (contexts) that led to the violation. This has been further improved upon by additionally presenting all sub routines called by the applicable section of code and using coloured highlighting to identify the path through the code for which the violation was found. Illustrations of signal aspects have also been added to facilitate diagnosis of violations of new safety properties pertaining to signals.

3 Statistics from live projects

The following tables summarise the scale and verification effort for a random sample of industrial projects.

Table 1 shows the projects sampled from the period from 2020 to the first half of 2022. These projects were verified for 59 predicates defining safety properties. At that time we did not formulate any conditions for SSI logic covering output telegrams (procedures computing messages sent to various trackside equipment) and thus verification coverage as reflected in the first table below was only partial.

The *Routes*, *Signals*, *Points*, *LOC* columns characterise a project in terms of the total number of routes, signals, points and the total number of lines of codes (LOC) in signalling data (all types of files included).

The *POs, total* column gives the overall number of non-trivial proof conjectures generated for a project. Simple instantiation of safety property templates yields many more conjectures, many of which are trivial ones that have a copy of a goal in their hypotheses. Such trivial cases are detected and dropped during the generation phase and not included in this figure.

The *POs, unf* column is the number of conjectures for which proof was forcibly stopped due to an overrun in the number of generated proof branches or a predefined limit on proof time rather than simply running out of any applicable rewrite rules (what we regard as a proper termination). Logically, these are no different from failed, but properly terminated conjectures. In practice, their presence indicates a weakness in the prover or domain axiomatisation as the automated proof process is tuned to deadlock where successful proof is unlikely (empirically and drawing from the experience in this particular domain). They are also often indicative a computationally expensive runaway situation stemming from generation of proof sub branches, derivation of new hypotheses and possibly cyclical behaviour of rewrite rules. Finally *Proof time, s* gives the overall proof time in seconds. All time measurements are given for an AMD 5950X PC with 128GB memory using 16 cores; it excludes extra time needed for the construction of a state transition system from source SSI translation (typically under 5 seconds).

One conclusion to draw from Table 1 is that there is no simple correlation between the size of a railway interlocking and the required proof effort. However, we can notice that the projects P3 and P5 had complex swinging overlaps with complex control logic which, in its turn, leads to a higher number of state transitions. There is also a small number of cases where proof depends on detecting

Id	Routes	Signals	Points	LOC	POs, total	POs, unf	Time, s
P1	164	71	48	13244	12172	108	96
P2	92	45	36	9320	8098	0	60
P3	62	31	28	11002	11711	14	203
P4	47	28	23	8750	6003	0	22
P5	59	30	24	8573	17241	2	129

Table 1. SafeCap statistics from 2020 to mid 2022

Id	Routes	Signals	Points	LOC	POs, total	POs, unf	Time, s
P6	170	56	59	18917	48992	0	72
P7	46	67	25	16200	24002	0	8
P8	164	63	73	16027	172404	0	130
P9	42	48	22	16288	24566	0	2
P10	122	68	60	18745	344072	6	602

Table 2. SafeCap statistics from mid 2022 to 2023

a contradiction, e.g., in timing conditions expressed as simultaneous inequalities in hypotheses. Such proofs first exhaust all venues of rewriting and only then translate a subset of relevant hypotheses into an external SAT solver for a contradiction check. The higher number of proof obligations does normally lead to longer verification although this is not completely clear cut: much of the proof time is often spent discharging a small proportion of harder proof obligations.

Table 2 covers the period from the second half of 2022 to the first quarter of 2023, using the current version of SafeCap with all major improvements implemented. It verifies 136 safety properties with nearly all extra predicates addressing output telegrams.

One relevant trait of output telegrams associated with signals is that they are rarely small and simple – it is unusual to have signal output telegram with fewer than 600 state transitions and some have up to 9000. This means that the output telegram part of SSI completely dominates the rest in terms of number of state transitions. Hence, there was a marked increase in the number of generated proof conjectures when SSI output telegrams were included in the scope of verification.

This required efficiency improvements to the prover that were achieved mainly through the addition of macro rewriting steps combining several existing rewrite rules.

4 Discussion and Conclusion

The paper discusses our experience in improving the SafeCap tool since it was approved by Network Rail for the use by the UK railway industry and started to be used commercially. Before this, we have used real data of several (developed earlier) SSI interlockings for demonstrating the tool's usefulness to signalling companies in the UK and to prepare for its official approval. Since the approval, the real work on the live commercial projects with our industrial customers has led us to conducting substantial targeted improvements of the tool.

Our strong belief is that the improvement phase is now coming to an end as our substantial and diverse experience in verifying interlockings developed by different companies and design offices clearly shows that the SafeCap tool is sufficiently flexible and scalable.

In the course of this intensive, 3-year long period of interlocking project verification we have also drawn several lessons about how SSI data could be developed to make the verification process simpler and to reduce the risk of errors. It is interesting to note here that our recommendations for achieving

these two goals are very much overlapping, and that following them will make understanding and maintenance of the SSI code easier as well. We are now working on summarising this experience and these recommendations.

Our ongoing and future work on SafeCap focuses on a full tool re-development to streamline its architecture after the period of improvements and to make the tool more extendable to other verification applications and resilient to obsolescence. This future work additionally includes the following:

- the development of the safety assurance case for use of SafeCap as an alternative to established manual checking processes, including:
  - re-development of the software for T2 qualification in accordance with the CENELEC EN 50128 standard [2];
  - ensuring traceability and independent verification of safety properties;
  - bug reporting, tracking and fixing.
- the adaption of SafeCap to other technologies, including:
  - development of a front end to import data in the ladder logic or structured text formats;
  - prototyping using machine readable representations of GB relay signalling circuits, to which the existing safety properties could be applied;
  - future extensions to cover ladder logic interlockings and PLCs in railway signalling and other industries.

References

1. Solid State Interlocking. Code of practice for the testing and commissioning of SSI signalling schemes, SSI 8501, Issue 1. British Railways Board (1989)
2. EN 50128, "Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems". CENELEC (2020)
3. Bobot, F., Filliâtre, J.C., Marché, C., Paskevich, A.: Why3: Shepherd your herd of provers. In: Proccedings of Boogie 2011. pp. 53–64 (2011)
4. Cribbens, A.H.: Solid State Interlocking (SSI): an integrated electronic signalling system for mainline railways. Proc. IEE. **134**(3), 148–158 (1987)
5. Garavel, H., ter Beek, M.H., van de Pol, J.: The 2020 expert survey on formal methods. In: Formal Methods for Industrial Critical Systems - 25th International Conference, FMICS 2020, Vienna, Austria, September 2-3, 2020, Proceedings. LNCS, vol. 12327, pp. 3–69. Springer (2020)
6. Iliasov, A., Taylor, D., Laibinis, L., Romanovsky, A.: Practical Verification of Railway Signalling Programs. IEEE Transactions on Dependable and Secure Computing **20**(Jan-Feb), 695–707 (2023)
7. Iliasov, A., Laibinis, L., Taylor, D., Lopatkin, I., Romanovsky, A.: Safety Invariant Verification that Meets Engineers' Expectations. In: Dutilleul, S.C., Haxthausen, A.E., Lecomte, T. (eds.) Proceedings of Reliability, Safety, and Security of Railway Systems (RSSRail 2022). LNCS, vol. 13294, pp. 20–31. Springer (2022)
8. Iliasov, A., Taylor, D., Laibinis, L., Romanovsky, A.: Formal Verification of Signalling Programs with SafeCap. In: Proceedings of 37th International Conference, SAFECOMP 2018, Västerös, Sweden, September 19-21. LNCS 11093, Springer. pp. 91–106 (2018)
9. Iliasov, A., Taylor, D., Romanovsky, A.: Automated testing of SSI data. IRSE (Institution of Railway Signal Engineers) News 241 (February 2018)