

PROVER[®]
engineering a safer world[™]



The Prover iLock Process



The Prover iLock Process

Prover iLock is a tool suite that automates software development of computerized interlocking systems. New CENELEC SIL-4 certifiable interlocking systems can be produced in a few days, saving significant development resources. Prover iLock runs on an average computer.

What You Gain Using Prover iLock

Significant Savings

Using a traditional development process, every interlocking system requires significant engineering effort. The underlying signalling principles may be the same for a whole family of interlocking systems, but each system is nevertheless programmed individually to obtain a suitable adaption to the station it is supposed to control. Reuse of code from system to system is limited to copy-and-paste techniques with manual alteration, often introducing programming errors.

In the Prover iLock process, up-front effort is spent on specifying the underlying signalling principles themselves and formalizing them; thereafter the individual interlocking systems are automatically generated, simulated and verified by the push of a button. The initial investment pays off quickly: break-even occurs when a handful of interlocking systems have been developed.

	Traditional process	Prover iLock process
Formalizing generic signalling principles	N/A	18 man months
Creating interlocking system 1	3 man months	1 day
Creating interlocking system 2	2 man months	1 day
Creating interlocking system 3	4 man months	1 day
Creating interlocking system 4	10 man months	2 days
Creating interlocking system 5	3 man months	1 day
Creating interlocking system 6	2 man months	1 day
Creating interlocking system 7	6 man months	2 days
Creating interlocking system 8	3 man months	1 day

Figure 1. Resources required for the development of a number of interlocking systems applying identical signalling principles. All figures are examples, based on experience of average-complexity designs, including code generation, testing and safety verification.

Signalling principles can be divided into three categories: safety requirements, functional requirements and design principles. They can all be generically expressed, that is, in universal ways rather than about particular instances.

- ✓ **Safety requirements** Example: "A signal must only display a 'proceed' aspect if the route is clear."
- ✓ **Functional requirements** Example: "It shall be possible to run a train from one end of the station to the other."
- ✓ **Design principles** Example: "The red lamp of a main signal shall be lit precisely when the green lamp of the same signal is not lit."

Such generic principles can be given to Prover iLock, using a formal specification language. When Prover iLock is later provided with a particular station layout, it uses the formalized generic principles to generate a set of concrete principles adapted to the particular configuration. The generated design becomes detailed enough to be automatically translated to software for the system. The generated safety requirements are used to verify that the generated system is safe. The functional requirements are used to check, by simulation, that the system behaves as expected in particular situations.

- ✓ **Safety** Errors are eliminated using formal safety verification
- ✓ **Speed** Reliable systems are generated by the push of a button, and easier to certify, maintain and update
- ✓ **Savings** Development resources are minimized by extensive reuse through automation

Easily Changed Systems

With automatic generation it becomes easy to make changes in interlocking systems:

- When a station configuration is changed, a new interlocking system can quickly be generated, simulated and verified according to the new configuration. There is no need to rewrite any software. The whole process takes a few minutes.
- When signalling principles are changed, there is no need to go through all stations to see which ones are affected and need to be updated. Just make one modification of the generic design and regenerate all interlocking systems automatically.

Improved Documentation

A side effect is welcome: the precise generic signalling principles, earlier considered to be “tribal knowledge” among experienced engineers, now become explicitly expressed in a precise specification language. When senior engineers leave or retire, the generic signalling principles stay. When new engineers are employed, they can learn the details of the signalling principles by studying the formal specifications.

Support for Verification and Validation

CENELEC strongly recommends formal verification, that is, that mathematical methods are used to prove that the system fulfils the safety requirements. The reason is that formal verification gives full coverage, which a testing approach cannot give for reasonably sized systems due to the overwhelming number of possible input combinations that would have to be tested. Prover iLock supports formal verification. When a station configuration has been used to automatically generate an interlocking system, Prover iLock can automatically prove that all safety requirements are met, using mathematical logic methods.

Prover iLock also supports simulation of generated systems, either driven by generated test cases or by user interaction, much as if the user were operating the system. Distributed systems are also supported. The simulator can export traces, allowing test scenarios to be reproduced on real systems or by independent simulators.

How You Work Using Prover iLock

The process consists of two main steps. First, a Generic Application is produced by gathering all generic design principles and requirements and formalizing them. The main effort is spent on this step. The formalized specifications are written in a text editor of the user's choice while Prover iLock is assisting by pointing out errors and omissions in the specifications. Having completed the Generic Application, the next step is to produce a number of Specific Applications by graphical configuration and automatic generation. This task is completed entirely within Prover iLock, much as in a painting program: tracks are drawn with the mouse on a canvas and wayside objects are placed by drag-and-drop methods. The interlocking system is then automatically generated, simulated and verified by the push of a button.

Producing a Generic Application

The following tasks are performed to produce Generic Applications:

- Signalling principles for the area are gathered into a specification written in a natural language such as English. It is decided what assumptions should be made about limitations of the use of the Generic Application. These assumptions can later simplify the task of specifying the design: it will not be necessary to implement generic rules for cases that will not occur.
- An analysis is made of the collected principles, dividing them into three categories:
 1. Safety requirements that must never fail, not even in highly unexpected situations.
 2. Functional requirements on how the system shall behave in certain situations.

3. Design principles, that is, solutions to the problem of building a system that meets the requirements. An example may be: "the green lamp shall be lit when the route is locked and clear; the red lamp shall be lit whenever the green lamp is not lit." The assumptions about limitations can be used to determine how general the solutions need to be.

- The principles are formalized in the specification language PiSPEC as a number of text files produced in a text editor of the user's choice. PiSPEC has been designed to be the perfect language for interlocking systems specifications. It builds on wellknown concepts that are taught in engineering classes all over the world.
 1. Safety requirements are formalized in a Generic Safety Specification (GSS). It specifies requirements that shall be formally verified for every generated system.
 2. Functional requirements are formalized in a Generic Test Specification (GTS). It specifies requirements that shall be checked by simulation for every generated system.
 3. Design principles are formalized in a Generic Design Specification (GDS). It specifies how generated systems shall be designed. The formalization of the design includes formalization of the assumptions made about limitations of the use of the Generic Application, so that it can later be automatically checked that they are fulfilled every time the Generic Application is used.

```

1 class REQ_f
2
3 Variables: k;
4
5 Prove:
6 """ A route may only become locked if it is under control and there is a
7 locking command """
8 A := ALL rt (~PRE rt.is_locked & rt.is_locked ->
9           rt.route_control & rt.locking_command);
10
11 """ Occupied routes must not be considered being under control """
12 B := ALL rt ALL tc:rt.track_circuits (tc.is_occupied ->
13           ~rt.route_control);
14
15 """ A route may only give permission to proceed when it is under control """
16 C := ALL rt (rt.permission_to_proceed -> rt.route_control);
17
18 """ A route must not be considered being under control if it contains a
19 locally released switch """
20 D := ALL rt ALL sw:rt.switches (sw.is_locally_released -> ~rt.route_control);
21
22 """ An exit signal may only request green when all lines beginning at it
23 give permission to proceed """
24 E := ALL si ALL li:si.line_begins (si.is_requested_green ->

```

Figure 2. Examples of PISPEC safety requirements.

- The PiSPEC libraries are reviewed against the natural language specifications. PiSPEC has been designed to make reviewing convenient. By using traceability matrices it becomes easy to understand the connection between the formal expressions and their natural language counterparts.
- Basic rules for the use of the formal specifications are defined by simple Python scripts which communicate with Prover iLock via API's. Such scripts determine:
 1. How the graphical symbols placed on the canvas are interpreted as real rail yard objects. Example: every signal symbol in the track layout corresponds to a real signal with properties derived from the symbol's place on the canvas.
 2. How properties are set based on configuration files. Example: cab codes, speed limits, valid routes et cetera may be given in configuration files and read by Python scripts. Configuration files may be created by other tools used by application engineers.
- The Python scripts are manually verified by review. As Prover iLock is shipped with extensive Python libraries that are well tested, additional Python scripts in the Generic Application can be kept small and easy to review.

Producing Specific Applications

While the process described in the previous section is performed only once for a family of interlocking systems applying the same signalling principles, the process that will be described here is repeated for every Specific Application. It is performed entirely within Prover iLock.

- A new Specific Application is created: Prover iLock is launched and a new project based on the Generic Application is started.
- The built-in Layout Editor is used to draw a track layout for the Specific Application and to place wayside objects like signals. This is done with the mouse as in a painting program. An alternative is to import the station configuration from a file created by another program.

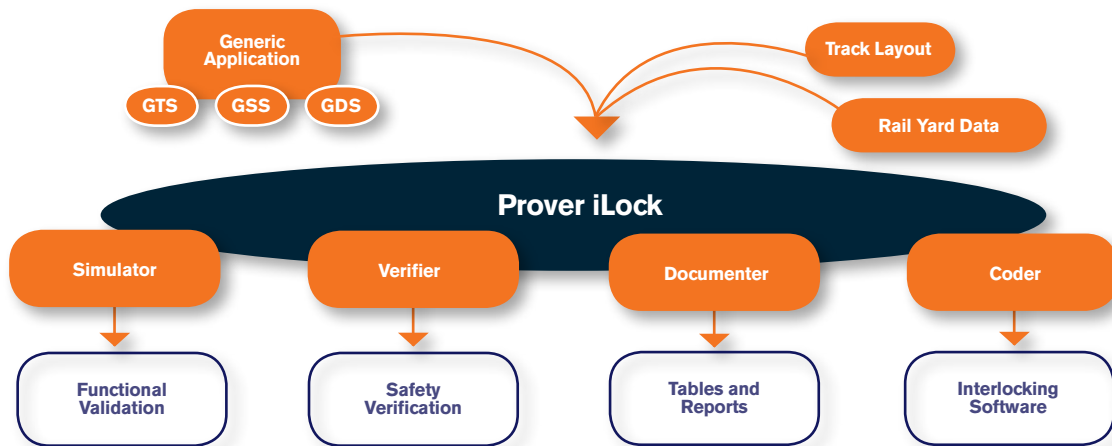


Figure 3. How a Specific Application is created from a Generic Application with track layout and other configuration data.

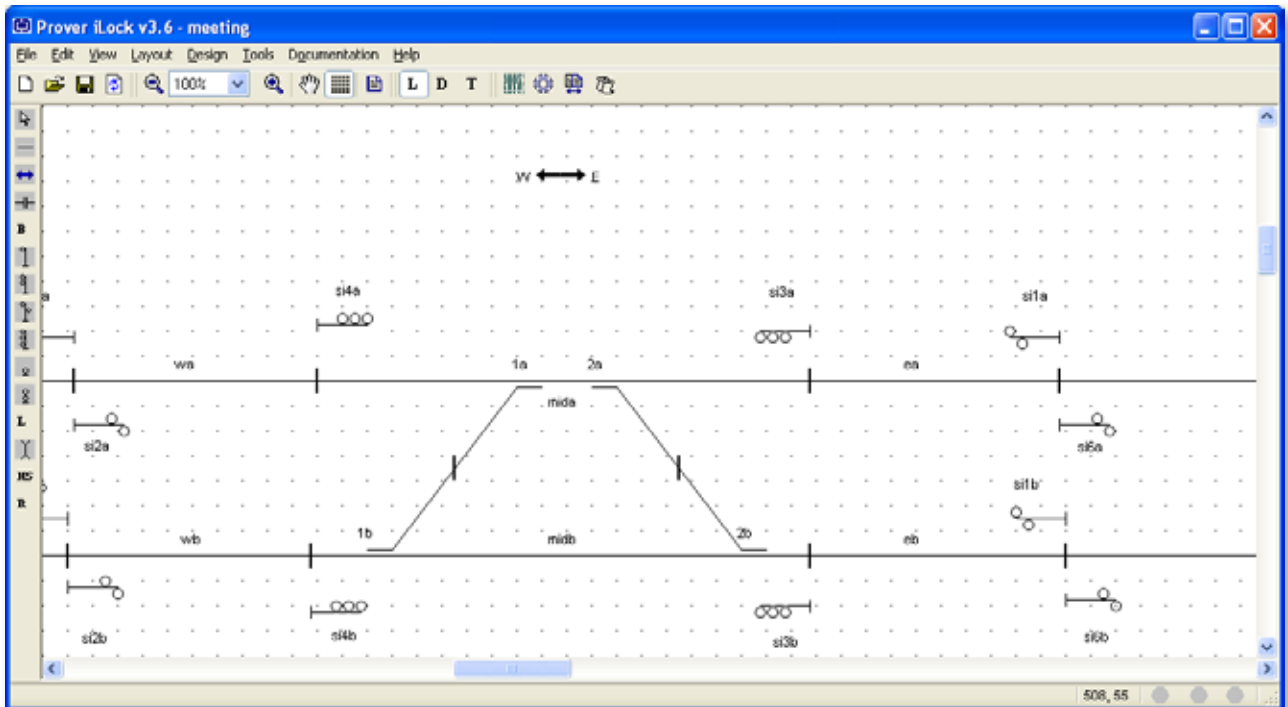


Figure 4. Tracks are drawn, wayside objects are placed on the canvas. All by drag-and-drop techniques.

- Objects are configured by checking boxes and by choosing from drop-down menus. Python scripts can be invoked for reading configuration data from file, or for dumping configuration data to file for review.
- An interlocking system based on the Generic Application is generated by the push of a button, and is kept in memory for investigation by various tools.

Prover iLock Modules

Prover iLock Simulator can simulate the interlocking system according to test scenarios that were generically specified in the Generic Test Specification. Fulfilled requirements

are displayed in green, while those that are violated are displayed in red. The failing ones can be investigated by a built-in debugger. The simulator also features interactive simulation.



Figure 5. Prover iLock Simulator runs through all test cases, marking passed tests green, and failing tests red.

- ✓ **Test cases** Simulation can be driven by test cases defined in a Generic Test Specification
- ✓ **Interactive simulation** The simulator allows the user to operate the system and investigate its behaviour
- ✓ **Traces** Exportable simulation traces, allowing test scenarios to be reproduced on real systems or by independent simulators

Prover iLock Verifier can perform formal verification of the safety requirements given in the Generic Safety Specification. Formal verification means that mathematical proof is used to ensure that the system fulfils the requirements. In this way, 100 % coverage is obtained in a few minutes. Again, requirements are displayed in green or red, depending on whether or not they are fulfilled. For red requirements, a viewer displays on the screen how the system can enter an unsafe state.

Prover iLock Documenter can produce various tables, including control tables, summaries of configuration parameters for review, or forms used for commissioning testing.

Prover iLock Coder can automatically produce software for the hardware platform. Platforms such as Westrace, VHLC, Microlok II and PLC are supported. It is also possible to generate source code in programming languages such as C.

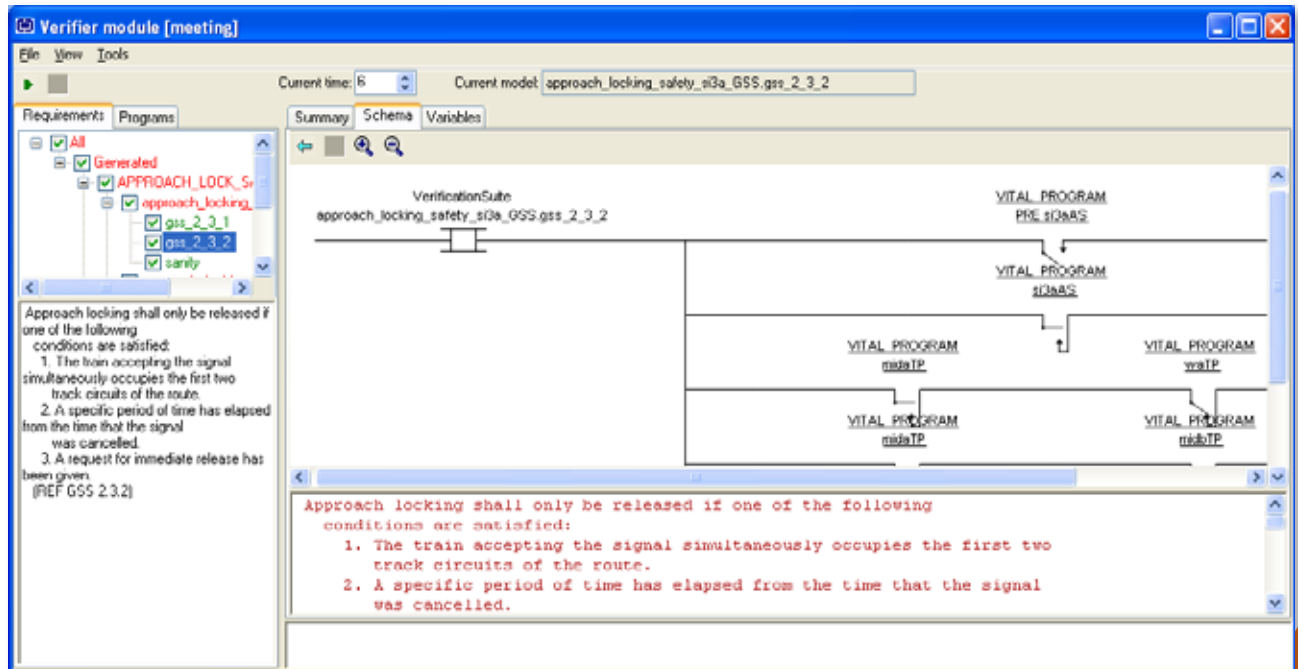


Figure 6. Prover iLock Verifier formally verifies the safety requirements. Failing safety requirements are marked in red. An example of a system state in which requirements fail is displayed using ladder logic in a style of the user's choice. By clicking on underlined variables, the user can navigate through the details of the example.

- ✓ **Development** Powerful development features in Prover iLock
- ✓ **Verification** Independent safety verification by Prover Certifier
- ✓ **Certification** CENELEC SIL-4 certifiable result

CENELEC SIL-4 Safety Cases

A typical design choice for CENELEC SIL-4 certified tools is to limit features to a minimum in order to save resources in the certification process. With Prover iLock, you can actually combine many useful features with certifiability. This has been made possible thanks to a complete separation of the certification process from the development process, and the observation that these processes require entirely different features. Debugging and simulation are needed during development only, while certified formal verification is needed during the certification process. Such formal verification is provided by Prover Certifier, an independent CENELEC SIL-4 certified product.



Prover Technology AB
Rosenlundsgatan 54
118 63 Stockholm
Sweden
Phone: +46 (0)8 617 68 00
Fax: +46 (0)8 653 69 00

Prover Technology S.A.S.
21 Rue Alsace Lorraine
31000 Toulouse
France
Phone: +33 (0)5 6227 5327
Fax: +33 (0)5 6227 5329

Prover Technology Inc.
2700 Chabot Drive
San Bruno, CA 94066
Phone: +1 (415) 963 4200
Fax: +1 (415) 963 4259