The interlocking formalization in RailML

This chapter develops a formalization that provides, together with the RailML v2.2 description in the previous chapter, the opportunity to exchange all data to engineer an interlocking system. Section x summarizes the literature that already elaborated on a possible way to formalize interlocking in RailML. Section x describes the process that led to the interlocking formalization. Section x concludes this chapter with the final XML interlocking formalization.

**Current status of the interlocking subschema in RailML**

[Lehmann and Albrecht (2008](#_ENREF_2)) proposed a RailML interlocking formalization as subschema of the infrastructure schema. They aim to capture the relations between those rail elements necessary to enable safe routes for trains. The study was initiated to measure the performance of various interlocking systems by means of simulation.

Lehmann and Albrecht approach the interlocking formalization from the viewpoint of the hardware. Therefore, they define the “mainSignalBox” as the attribute of interlocking which has, besides standard tags, a “type” (e.g. relay, mechanical and electronic) an “operationControlCenter”, an “ownControlRange” and a “subSignalBox”. The “operationControlCenter” allows for the possibility to link an interlocking to a dedicated control center. The “ownControlRange” provides the opportunity to define specific sections within an interlocking area that may operate as an interlocking area on its own. This function becomes obsolete for an electronic interlocking system. The “subSignalBox” contains mainly three sub elements relevant for capturing interlocking functionality: “routes”, “trackSections” and “overlaps”. Routes imply a list of possible options. TrackSections represent trackfree detection points. Overlaps include data on the location of danger points and overrunning distances of signals. Lehmann and Albrecht only designed the “route” sub element in more detail since the other two did not seem very relevant for simulation purposes.

A “route” contains the track characteristics for a certain section and the interdependencies with other rail elements. The authors, focusing on the German railway system, define a route as a block section; this counts for the Dutch railway system as well (in Dutch: “enkelvoudige rijwegen”) ([van der Meij, van der Werff, Janssen, Bartholomeus, & Dragt, 2013](#_ENREF_7)). A route typically refers to a rail element defining its start and its end. A route’s attributes include maximum speed, distance, interlocking setup time and interlocking release time. Special sub elements include references to rail elements, flank protection elements, overlap and relevant track sections. The next summarizes Lehmann and Albrecht’s interlocking schema structure.

<RailML>

<infrastructure>

<interlocking>

<subSignalBox id=”x” name=”y” type=”z”>

<routes id=”x” vmax=”y”>

<routeStart/>

<routeDestination/>

<elements/>

<flankProtectionElements/>

<overlaps/>

</routes>

<trackSections/>

<overlaps/>

</subSignalBox>

<ownControlRange/>

</interlocking>

<line/>

</infrastructure>

<timetable/>

<rollingStock/>

</RailML>

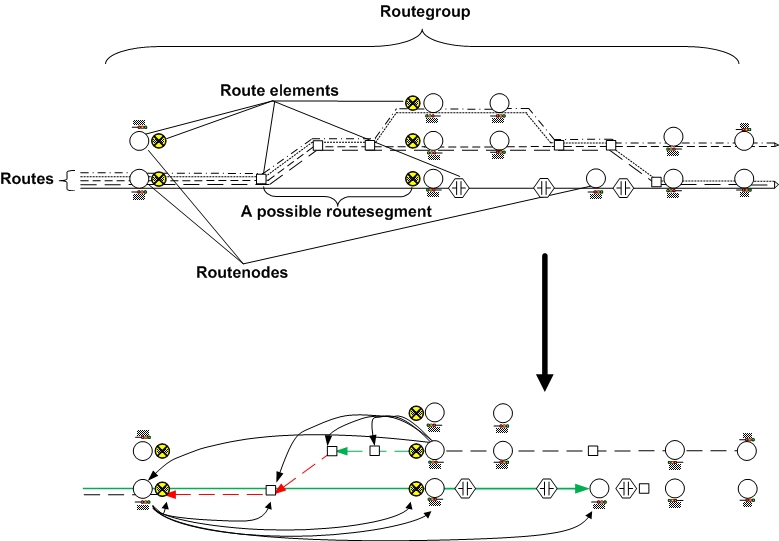
Lehmann and Albrecht explain that the German railway system tags every rail element in the same interlocking area with the same code. This rule allows easy identification of interlocking areas and open track areas. Currently, rail elements only have a connection with a track. The authors propose to study a connection between those rail elements and the interlocking by means of an attribute with reference tag.

[Fries (2003](#_ENREF_1)) proposes the use of an interlocking database with relations to a RailML file instead of an interlocking formalization in RailML. Fries proposes a database like MS ACCESS because of probable redundant data issues. In order to minimize the chance and risk of inconsistencies and the use of old data files, Fries believes that an IT tool needs to manage and track changes. RailML cannot do that. Schut and Palacios, both working on (rail) IT tools, acknowledge this issue ([Palacios, 2013](#_ENREF_3); [Schut & Dragt, 2013](#_ENREF_5)).

As a consequence, a conversion tool needs to transform the RailML infrastructure data into the interlocking database. The conversion tool can find the elements along a defined track and determine the relations between them. Fries mentions five challenges with this approach. First, the process could contain errors and the database has no means to find those errors itself. Second, RailML positions itself at meta level of detail to connect with a wide variety of databases, applications and output files. As a result, it may occur that RailML will sometimes not contain data at a micro level of detail necessary for a certain application as interlocking. Third, the conversion time may become substantial for large interlocking areas as the conversion tool needs to find every route with associated rail elements. Fourth, a database has more difficulties to capture some track descriptions compared to RailML. Fries provides some examples: the limited amount of variable conditions and the distinction between tracks with double track or more. Fifth, the database format does not make real connections between elements like RailML does with tracks, switches and so on.

For a few years, a group of representatives of Siemens, Alstom, Signon, Thales, Infrabel, OBB, SBB and the ON-time project group work on the official formalization of an interlocking sub elements in RailML ([RailML.org, 2013](#_ENREF_4)). At their last meeting in February 2013, they agreed on an UML object relation structure that might form the basis of the interlocking subschema (Appendix X). The RailML group thus follows the approach of Lehman and Albrecht to include interlocking in RailML instead of using the separate database approach of Fries. In contrast to Lehman and Albrecht however, the RailML group formalizes the interlocking structure from the viewpoint of requested train routes instead of the interlocking hardware. Figure X makes this very clear by visualizing the UML object structure.

The UML agreement implicates that, in terms of the RailML structure in XML, the interlocking scheme captures all route locking process. A route locking process starts with a train’s request for a route. Traffic control sets the sequence of routes so that the interlocking does not take care of route selection but only just locking. The interlocking may grant a route when (1) the route is clear, (2) all movable elements are in the correct position and (3) flank /conflict protection is ensured (chapter 2). This implies that an interlocking must check the rail elements with a variable condition and put them in the correct position, both for the requested route as the flank. A complicating factor arises when interlocking systems control signal aspects and their corresponding speed limits like the Dutch system of ATB.



The visual meaning of the object relations (from the UML in Appendix X) for an interlocking that the RailML interlocking group agreed on. A routegroup contains two or more different routes that a train can request to drive on. A route is defined as a realizable link between two routenodes. The Dutch Railway network defines a route as a block section due to which a node is always a signal ([van der Meij et al., 2013](#_ENREF_7)). Every route then contains two or more rail elements. Rail elements in the visualization include signals (circles), level crossings (yellow circles), switches (squares) and detection (hexacons). The UML defines a link between two rail elements as a routesegment.

The RailML interlocking group also defined how figure x translates to a (temporary) XML structure:

<interlocking>

<interfaces>

</interface>

</interfaces>

<routeGroups>

</routeGroup>

<routes>

<route>

<start>

</signalRef>

</trackTerminationRef>

</start>

<target>

</signalRef>

</trackTerminationRef>

</target>

</segments>

<segment>

<element>

</signalRef>

</trackSection>

</switchRef>

</crossingRef>

</derailerRef>

</trainDetectorRef>

</levelCrossingRef>

<flankProtection>

</signalRef>

</trackSection>

</switchRef>

</crossingRef>

</derailerRef>

</trainDetectorRef>

</levelCrossingRef>

</segment>

</segments>

<signals>

</signal>

</signals>

<signalTypes>

</signalType>

</signalTypes>

<signalAspects>

</signalAspect>

</signalAspects>

<signalAspectGroups>

</signalAspectGroup>

</signalAspectGroups>

<signalAspectDepencies>

<signalAspectDependency startSignalTypeRef=”” targetSignalTypeRef=’” >

<dependency>

<routeDescriptionAfterTargetSignal>

</routeKind>

</sequence>

</routeDescriptionAfterTargetSignal>

</dependency>

</signalAspectDependency>

</signalAspectDependencies>

</interlocking>

**The Development Process of the RailML Interlocking Formalization**

A four step approach leads to a rational, top-down way of formulating an interlocking XML structure for RailML:

1. determine the route interlocking process on a general level;
2. apply this to an actual interlocking area;
3. discover and formalize the data that misses in the RailML v2.2 (chapter x);
4. reduce the formalization to the core necessities and align with RailML group’s definitions.

Determine the route interlocking process on a general level

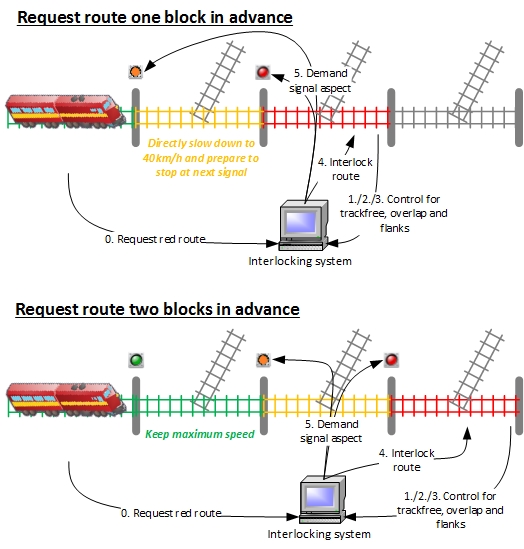
The interlocking system starts interlocking a route when a train demands a route. In the Netherlands, a route equals a block section; a block section equals a path from signal to the subsequent signal. Ideally, the interlocking system interlocks a route two block sections in advance to ensure an optimal speed profile and track capacity ([Wiggenraad, 2012](#_ENREF_8)). Figure x illustrates that a route request at two blocks distance allows a train to pass at the maximum allowed speed, assuming a free requested route in which a train can always come to a complete stop.

Upon receiving a request to interlock a route, an interlocking system will first check whether another train already claimed the route. Furthermore, the interlocking system will read the measurements of the detection system in place to exclude the possibility of rolling stock on the requested route. The interlocking also needs to confirm the absence of derailers, open bridges and water surge barriers on route. The interlocking will then check the route for the movable rail elements in place. In the Netherlands, changeable rail elements could include switches and level crossings. The interlocking needs to put switches in the correct position and demand an activated level crossing.

Before the interlocking may change the elements on route, it also needs to ensure flank protection. Chapter X lists the rules for flank protection. In the case of no danger from the flanks, the interlocking may alter the switches. The level crossing gets activated later by a train itself in order to minimize waiting times.

In the case that the rail elements are in the correct position and there is no danger from other rolling stock, the interlocking may communicate this status to the signaling system. The communication occurs by means of signal aspects. In the simplest form, a red aspect changes into yellow and a yellow aspect into green. The result is an active route.

Eventually, when a train successfully leaves the requested route, the route becomes passive and another train may request the route again. Then the process repeats itself. The interlocking does not need to change the start signal aspect as the Dutch signals do this automatically on the basis of neighboring signals and switches.



Apply the interlocking process to an actual interlocking area

A route request at the Stpn interlocking area shows the implication of the interlocking procedure in the previous paragraph. Assume the hypothetical situation that a train requests a route from signal 1404 to signal 1422 since it wants to pick up passengers at platform 3 (please refer figure x and Appendix X for the track topology and rail elements). Then, a train would request the route two signals earlier: signal 501 in the Haarlem area.

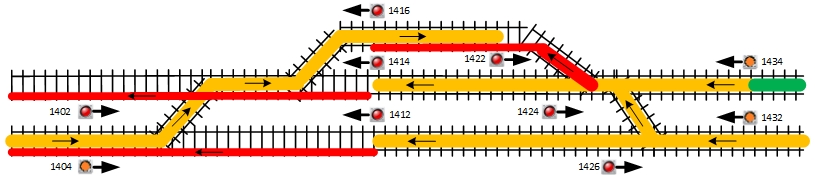
First, the interlocking system needs to check whether this route already overlaps with another active route. A possibility could be a route from signal 1402 to signal 1424.

Second, the interlocking system needs to ensure that no rolling stock is present on this route by reading the status of the track circuit sections. No further elements could constrain the route in the absence of water surge barriers, movable bridges and derailers.

Third, the flanks of the train need protection from other trains. Flank collisions could potentially occur from train movements starting at signal 1402 towards signal 1424/1422, train movements starting at signal 1432/1434 heading towards signal 1416, train movements starting at signal 1432 heading towards signal 1412 and train movements starting at signal 1414 heading towards signal 132/506. Potentially, because separate flank protection measures are not required in this case. A route from signal 1404 to 1422 directly prevents the possible flank movements by either causing overlap (e.g. for a route from signal 1434 to 1416) and fixing the switches in an impossible position to allow other routes (e.g. for a route from signal 1402 to 1424). Figure x visualizes how this works out.

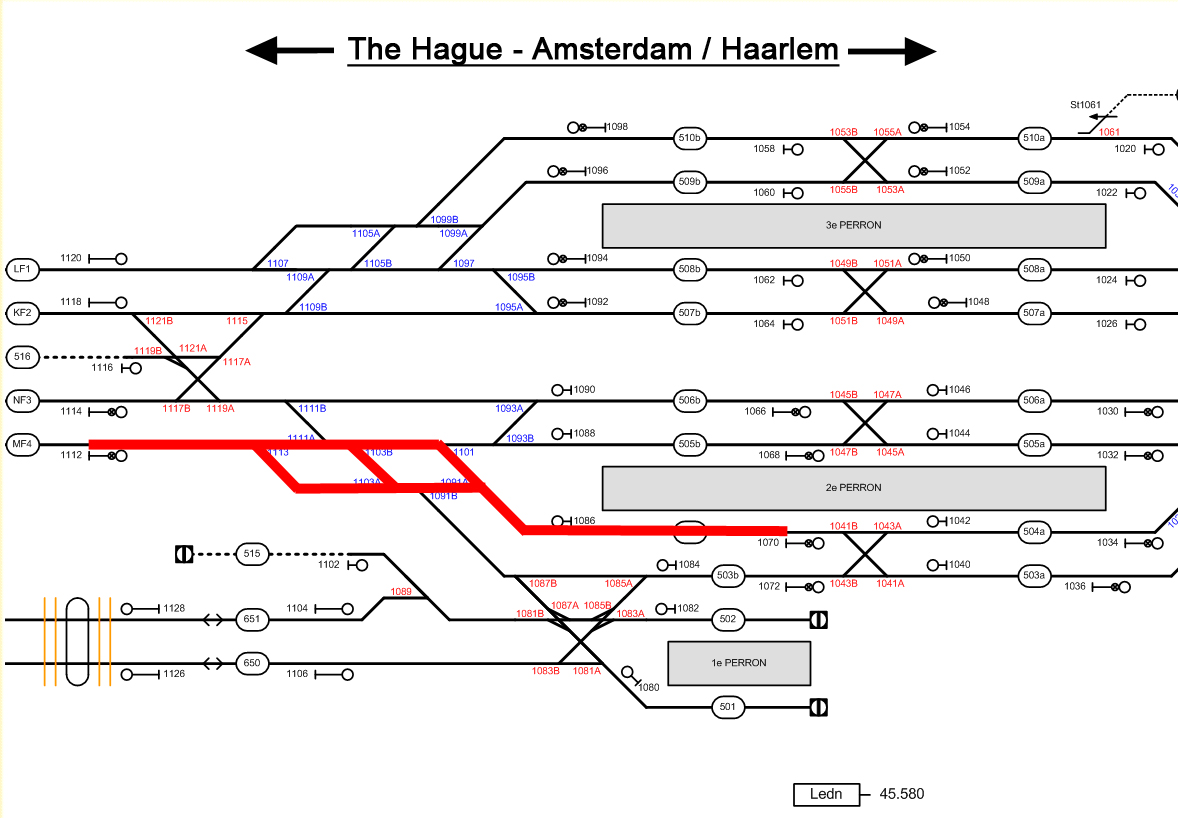
Fourth, the interlocking needs to manage five changeable rail elements: two level crossings and three switches. The level crossings need to be activated by the train; this occurs over a block section in advance. The interlocking also puts the switches into the correct position. This means that switch 1405, 1407 and 1409 should all bent.

Fifth, with the elements in the correct position, the interlocking should communicate a signal aspect to the starting signal 1404. Each area has their own set of signal aspect dependencies. The relations mostly depend on local speed restrictions that result in a different signal aspect. Following the example, the OS map (Appendix x) indicates two possible signal aspects: yellow and green flashing. The interlocking needs to communicate the yellow aspect when the target signal has a red or yellow flashing signal. The interlocking needs to communicate the green flashing aspect when the target signal has a yellow, yellow 4, green or flashing green aspect.



This figure shows that separate flank protection measures are not necessary when interlocking route 1404 to 1422. Train movement from signal 1412 or 1414 in contradicting direction, cannot be interlocked due to incompatible switch positions. Furthermore, a route from signal 1432 or 1434 to signal 1416 causes overlap with the interlocked route 1404 to 1422. The consequence is that these routes cannot be interlocked either; protecting all possible flank movements of route 1404 to 1422. The lines’ colors correspond to signal aspects.

The Stpn does not show one additional task for the interlocking system: route decision making. Figure x shows that some interlocking areas allow multiple routes between two signals. Therefore, the interlocking should know or determine which route to interlock. Currently, this occurs by means of a route hierarchy, i.e. experts determine that route 1 is preferred over route 2 [source]. The interlocking system first needs to check the above process for the preferred route. When the route cannot be interlocked, e.g. because of overlap, the interlocking needs to check the second preferred route, and so on.



The figure shows the interlocking area of Leiden. The red line indicates that a route from signal 1112 to signal 1070 could follow three different routes. Experts currently determine, probably on the basis of timetables, speed restrictions and so on, what the hierarchy of preferred routes is. The interlocking needs this data to interlock the desired route ([SporenplanOnline, 2010](#_ENREF_6)).

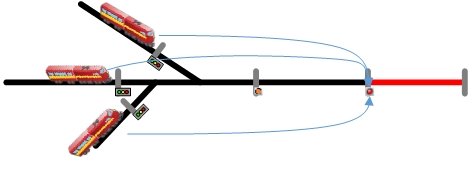
Discover and formalize the data that misses in the RailML v2.2

On the basis of the previous two sections, the following data/information misses in the RailML v2.2 presented in chapter x:

* RailML does not indicate the detector or track circuit border that announces a route request.
* RailML does not distinguish active or passive routes.
* RailML does cannot provide the current status of possible track obstruction objects like rolling stock and water surge barriers.
* RailML does not indicate tracks’ relations to other tracks and corresponding rail elements for the purposes of flank protection.
* RailML does not indicate the status of changeable rail elements, e.g. a switch’s position and/or level crossing activation.
* RailML does not provide a mean to choose between various route options.
* RailML does not contain signal aspect dependencies for each possible route.

The six data gaps represent the interlocking and a large part of the signaling system. Therefore, the formalization is developed from the viewpoint of both systems. The interlocking related data gaps follow a sequence from route request until route freeing. A route based formalization approach as proposed by the RailML group thus seems sound. A twist to the route based approach is given from the train’s perspective to request a route.

The Stpn interlocking example illustrates that a route starts at a dedicated signal and ends at a dedicated signal. One begin signal could have multiple end signals as figure x shows. A train may request a route, i.e. a clear begin signal, from multiple tracks as long as it is at at least two full blocks distance (figure x).

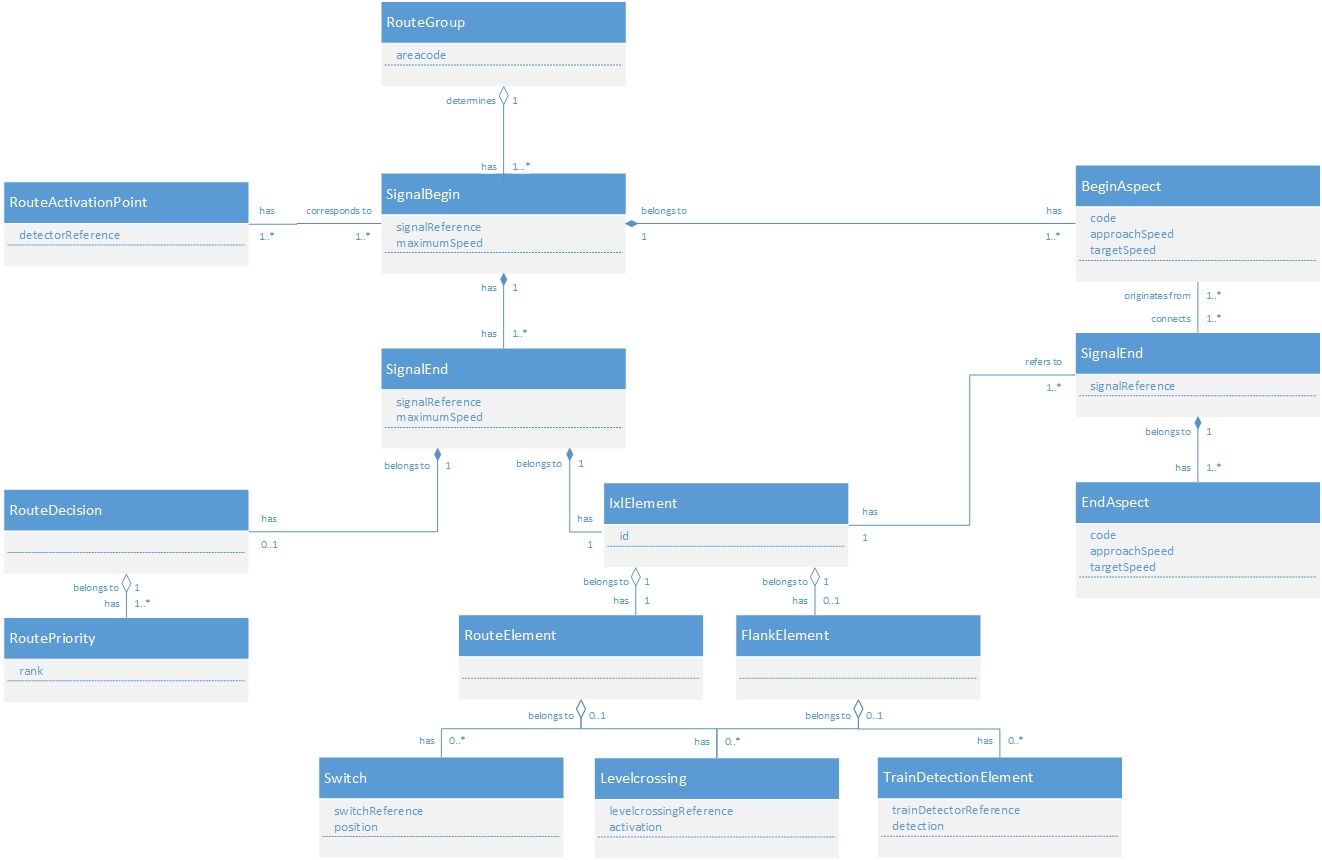


A track topology in which the red route can be activated from three different points.

The interlocking needs to refer to route specific, i.e. a signal begin and signal end pair, rail elements. Furthermore, the interlocking needs to refer to route specific flank rail elements, if any. In addition, each route may have route priorities.

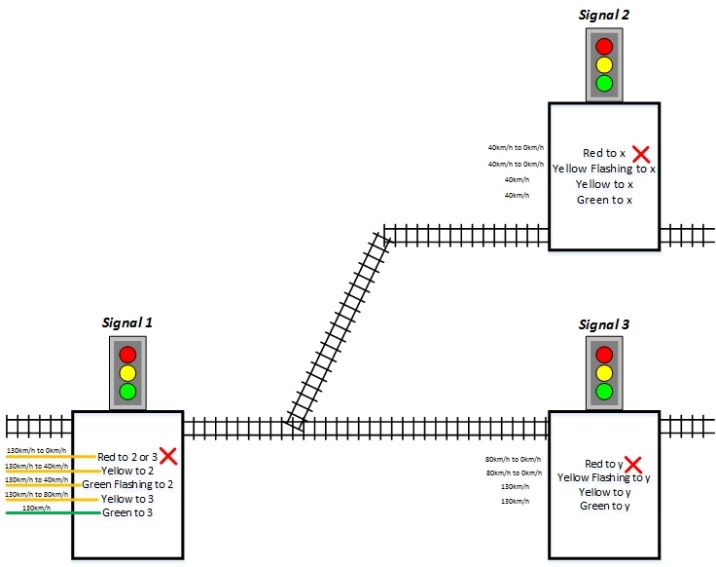
Besides interlocking of elements, the interlocking formalization also requires a definition of signal aspect dependencies in order to determine the aspect of the begin signal. Next to an end signal, each signal begin also has a list of aspects it can provide given the track topology and signal type. Each aspect enforces a speed regime on the requested route; modelled as an attribute. Furthermore, each aspect depends on the aspect generated by the end signal. Therefore, each aspect needs another list of aspects per end signal to unambiguously define the start signal; given that the route is free and the rail elements interlocked.

Figure x shows the result of this route based object relations in the form of an UML. Appendix x provides the XML structure for a Stpn interlocking area route from signal 1404 to 1422.

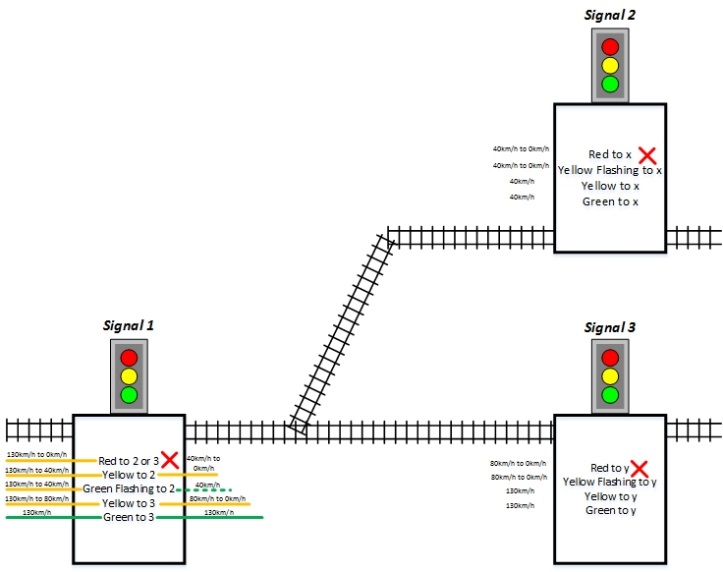


RailML interlocking formalization’s object relations on the basis of a route approach from the perspective of a train. The classes correspond to a Sptn route from signal 1404 to 1422.

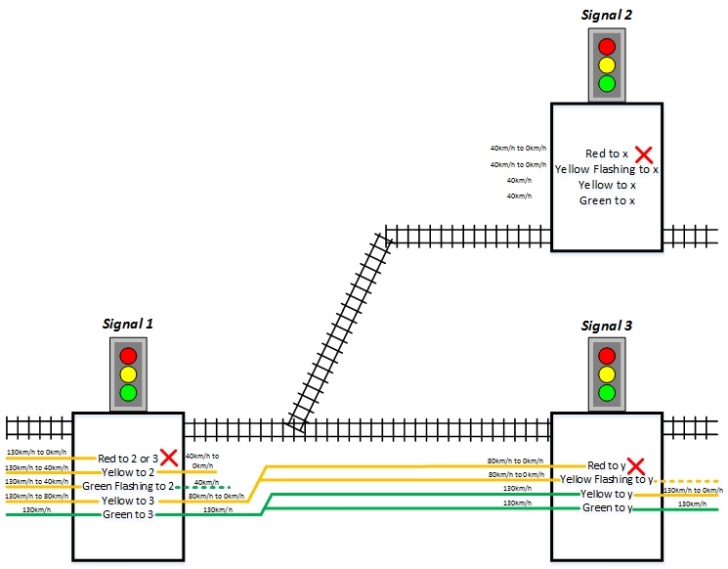
Another way to develop the interlocking formalization is from the perspective of the signals. Assuming that the interlocking system is smart enough to find the relevant rail elements on a route from the infrastructure description, it only needs to know the relations between the aspects of two subsequent signals. In the Netherlands, these relations always depend on the situation and cannot be derived from rules or algebra ([van der Meij et al., 2013](#_ENREF_7)). What an interlocking can determine based on rules, is the effect of a signal aspect. Table x shows those effects. Then, on the basis of a signal’s aspect code, approach speed profile and train’s target, all signal aspect dependencies follow as shown in figure x. The result corresponds to the Dutch OS maps.



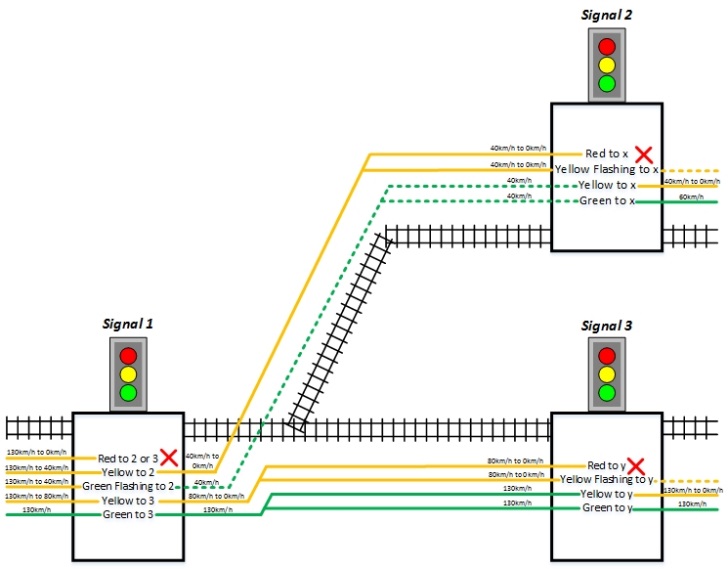
Part 1 of a fictive track topology to illustrate the formalizing approach for the signal aspect dependencies. Given are the signal aspects with corresponding targets and approach speed profiles per signal. Furthermore, all red signals do not have a signal aspect dependency.



Part 2 of a fictive track topology to illustrate the formalizing approach for the signal aspect dependencies. The effect of the entrance signal’s aspects on the approach speed profiles can be determined on the basis of the signal aspects.

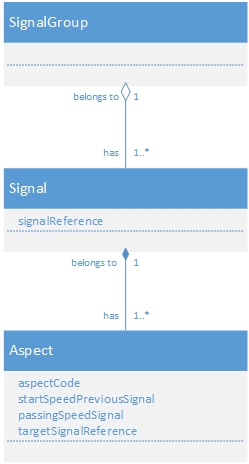


Part 3 of a fictive track topology to illustrate the formalizing approach for the signal aspect dependencies. The output speed profiles with target signal 3 of signal 1 can now be linked to the similar approach speed profiles of signal 3. Then, the output speed profiles of signal 3 can be determined as well.



Part 4 of a fictive track topology to illustrate the formalizing approach for the signal aspect dependencies. The output speed profiles with target signal 2 of signal 1 can also be linked to the similar approach speed profiles of signal 2. Then, the output speed profiles of signal 2 can be determined as well. And so forth with all other signals.

The RailML formalization of this signal dependency approach only describes signal aspects per signal in the interlocking area as shown in figure x. Each aspect at least needs to include the aspect code, speed profile and target signal for a complete picture. Appendix X contains an XML version of this approach for the route from signal 1404 to 1422 at Stpn.



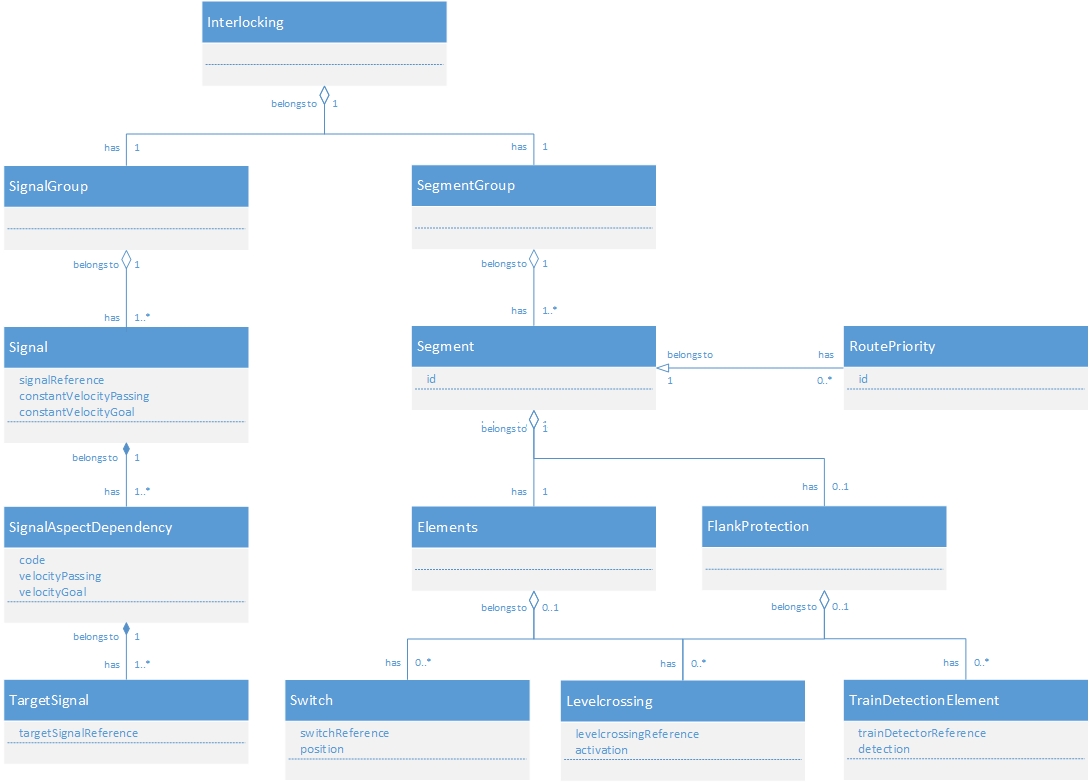
The UML for a RailML interlocking formalization on the basis of just signal aspect dependencies.

Reduce the formalization to the core necessities and align with RailML group’s definitions

The route based approach formalizes the interlocking system in a comprehensive way when compared to the signal approach. Furthermore, the route based approach includes redundant data. The signal approach namely showed that the speed profile, target and possible aspects per signal are sufficient to depict all signal aspect dependencies. In contrast, the route based approach also includes static maximum speeds and speed profiles with corresponding signal aspects of the route. In addition, the reference for each signal aspect dependency to the rail elements does not add additional information either. That is to say, the route formalization already groups the elements per signal end and thus route.

The signal aspect approach formalizes the signal aspect dependencies in a very concise way. The conciseness also limits transparency. The targetSignalReference as sub element instead of an attribute would directly distinguish the amount of different aspect codes per unique approach speed profile. This categorization is convenient for face validation as it represents the possible input signal aspect. When these do not correspond with the output of the preceding signal(s), a modeler can quickly take corrective addition. In addition, the signal input ‘lines’ on an OS map represent the same structure. Besides some lack in structure transparency, the signal based formalization misses a description of rail elements and possible route priorities. Those elements require a formalization so that the interlocking knows their required status, e.g. a switch should bent or stay in straight/normal position.

In order to mitigate the disadvantages of both approaches, the signal based formalization, including a slight restructuring with the targetSignalReference as sub element, forms the new aspect dependency scheme within the interlocking formalization. The ixlElement class complements the aspect dependency scheme to include the route and flank elements and route priorities. Figure x shows the UML of the final structure; the next also shows the final structure of the interlocking formalization in XML. The elements use the names of the RailML group as much as possible.



The final UML for the interlocking formalization in RailML. Notice that the UML corresponds to a mix of signal aspect classes of figure x and mostly ixl element sub classes of figure x.

The interlocking XML structure looks as follows:

<interlocking>

<signals>

<signal>

<AspectSpeedDependencies>

</targetRef>

</ AspectSpeedDependencies >

</signal>

</signals>

<routes>

<route>

<elements>

<signalRef/>

<trackSection/>

<switchRef/>

<crossingRef/>

<derailerRef/>

<trainDetectorRef/>

<levelCrossingRef/>

</elements>

<flankElements>

<signalRef/>

<trackSection/>

<switchRef/>

<crossingRef/>

<derailerRef/>

<trainDetectorRef/>

<levelCrossingRef/>

</flankElements>

</routePriorities>

</route>

</routes>

</interlocking>

Fries, N. (2003). Modellierung einer Eisenbahn-Infrastruktur in RailML. Dresden: TU Dresden.

Lehmann, M., & Albrecht, T. (2008). Erarbeitung eines XML-basierten Schemas zur Darstellung der sicherungstechnischen Streckenausrüstung in einem Fahrsimulator. In J. Krimmling (Ed.). Dresden: TU Dresden.

Palacios, E. G. (2013). [Interview: Siemens IT Development].

RailML.org. (2013, February 4 2013). RailML: Die XML-Schnittstelle für Eisenbahnanwendungen 2013, from <http://www.railml.org//index.php/index.html>

Schut, G., & Dragt, S. (2013). [Interview: Loxia].

SporenplanOnline (Cartographer). (2010). Tekeningen met spoornummers en seinen Nederland[Railway track maps].

van der Meij, D., van der Werff, M., Janssen, B., Bartholomeus, M., & Dragt, S. (2013, May 21). [Intermediary Prorail RailML Project Meeting].

Wiggenraad, P. (2012). CT4811 Design and Control of Public Transport Systems lecture 3: Timetable Design, Blocking Times. Retrieved from BlackBoard TU Delft website: