Forschungsvorhaben

"Untersuchung des Einsatzes formaler Methoden zur Spezifikation elektronischer Stellwerke am Beispiel des Lastenheftes ESTW-R (Regionalstrecken) der Deutschen Bahn AG"

1. Zwischenbericht

Formal Specification of Interlocking System Requirements
- Lastenheft ESTW-R (LH-ESTW-R) -

Yuen Man Hon

October 24, 2007

Technical University of Braunschweig
Institute of Railway System Engineering and Traffic Safety

Prof. Dr. -Ing. Jörn Pachl

This research project is supported by Karl-Vossloh-Stiftung
(Projectno. S047/10005/2006)
Abstract

Traditionally, system requirements of a computer-based interlocking system are written in natural language. It is difficult for the system development team to understand this document unambiguously and easily without domain specific language. Furthermore, the consistency of these requirements cannot be checked easily. Applying a formal framework for specifying the system requirements is proposed as a solution to handle these problems. A formal framework provides well-defined mathematical concepts to formulate system requirements. These well-formalized system requirements can be analyzed and understood easier and their consistency can be checked based on the mathematical concepts. In this work, a formal framework called Object Oriented LastenHeft (OOLH) is proposed to specify the system requirement of German regional computerized RIS Lastenheft ESTW-R (LH-ESTW-R). In OOLH, railway infrastructure elements are considered as objects and each object consists of attributes. LH-ESTW-R defines conditions that objects need to fulfil, such that establishment of safe routes can be guaranteed. The mathematical concepts used to specify these conditions are propositional logic and temporal logic in OOLH. These logical conditions can be transformed automatically into truth tables or decision tables, such that professionals can analyze the meaning of OOLH in different abstract levels. To check the consistency and correctness of OOLH, users can express an expected behavior of an RIS as a decision table or propositional formula, OOLH can be checked against them. The well-formalized and correct OOLH can be used to evaluate an establishment of a safe route for a given situation. Finally, the conformity of a design of an RIS to OOLH can be verified. A tool will be implemented to support analyzing, verifying and checking consistency of formulas in OOLH. In this tool, users can access the attributes and conditions of each object that need to be fulfilled during the establishment of a safe route. These conditions are expressed in propositional logic. They can be transformed into decision tables and truth tables. The expected behavior or a design of an RIS can be specified in decision tables in this tool, such that the correctness of OOLH or the conformity of the design against OOLH can be verified. A possible contradiction of a decision table is checked automatically by this tool.
Contents

List of Figures
List of Abbreviations
List of Symbols

1 Introduction

2 Formal methods
  2.1 Introduction
  2.2 Advantages of using formal methods

3 Railway interlocking systems and LH-ESTW-R
  3.1 Safe Routes
  3.2 LH-ESTW-R

4 Object Oriented Lastenheft (OOLH)
  4.1 Goals
  4.2 OOLH

5 Propositional logic and Temporal logic
  5.1 Propositional Logic
  5.2 Temporal Logic

6 Case Study
  6.1 Modeling of ZPZ
  6.2 Modeling of basic conditions
  6.3 Modeling of overlapping overlaps
  6.4 Modeling of object conditions
  6.5 Modeling of ZPZ condition for OOLH objects
  6.6 Discussion

7 Conclusion

A Railway Terms
List of Figures

2.1 System development process, V-model .......................... 6
3.1 Safe route ......................................................... 7
3.2 Point .............................................................. 8
4.1 Non-stop route (A,11) ............................................. 13
4.2 Tool for OOLH .................................................... 14
5.1 System satisfies $\text{AG}(\text{deadlock} \rightarrow \text{AG}(\text{deadlock}))$ .................................. 19
5.2 CTL formulas with operator AG and EG .......................... 21
5.3 LTL formulas with operator G and F .............................. 21
6.1 Point without movable frog and overlapping overlap ............ 27
6.2 Point without movable frog used as FLEL and DWEL ........... 33
List of Abbreviations

- CENELEC, Comité Européen de Normalisation Electrotechnique
- CTL, Computation Tree Logic
- DNF, Disjunctive Normal Form
- F, False
- FSM, Finite State Machine
- FüMBli, Festlegeüberwachungsmelder Blinklicht
- LH-ESTW-R, Lastenheft für das Elektronische Stellwerk Regional
- LTL, Linear Time Logic
- OBDD, Ordered Binary Decision Diagram
- OOLH, Object Oriented Lastenheft
- RIS, Railway Interlocking System
- SAT, Satisfiability solver
- T, true
- UML, Unified Modeling Language
- wwf, well formed formulas
- w.r.t, with respect to
- ZPZ, Zulassungsprüfung
List of Symbols

Propositional logic

- $\neg \phi$, not $\phi$
- $\phi \land \psi$, $\phi$ and $\psi$
- $\phi \lor \psi$, $\phi$ or $\psi$
- $\phi \rightarrow \psi$, if $\phi$ is true, then $\psi$ must be true
- $\phi \leftrightarrow \psi$, $\phi \rightarrow \psi$ and $\psi \rightarrow \phi$
- $\phi \equiv \psi$, $\phi$ and $\psi$ is logically equivalent
- $\phi \models \psi$, $\psi$ is the logical consequence of $\phi$ semantically
- $\phi \vdash \psi$, $\psi$ is the logical consequence of $\phi$ syntactically

Temporal logic

- $A$, all paths
- $E$, at least one path
- $G$, globally
- $F$, some future states
- $X$, next state
- $U$, until
- $W$(eak Until)
- $R$(elease)
- $Y$(previous state)
- $O$(nce)
- $S$(ince)
Chapter 1

Introduction

A railway interlocking system (RIS) is responsible for establishing safe routes for trains that are scheduled to pass through or stop at the railway station. Safe routes ensure that movements of trains along the station are safe, such that no train collisions or derailments can happen. Nowadays, in most railway systems, the functions of interlocking systems are realized by software and interlocking systems are called computer-based interlocking systems [Pac04]. Since interlocking systems are safety critical systems, it is important to ensure the correct behavior of computer-based interlocking systems. In order to develop such systems, the correctness and consistence of the input and the output artifacts of each phase of the software development, for example, system specifications, need to be checked and verified. This can be achieved when the system is developed by applying formal methods.

In [CEN06], CENELEC suggests the use of formal methods in specifying RIS. The focus of this work is to develop a formal framework for specifying the system requirements of computer-based interlocking systems. In this framework, the system requirements in natural languages are specified with mathematical concepts, such that their ambiguity can be reduced, their understandability can be improved and their consistency can be checked. As a result, the efficiency in developing a computer-based interlocking system can be increased. Furthermore, a computer tool can be developed based on these mathematical concepts to support analyzing these well-formalized specifications. It can also support the verification of a design against specifications during the design phase of the software development.

In this work, a formal framework for specifying the system requirements of German regional computerized RIS, LH-ESTW-R, is proposed. This framework is called OOLH. In OOLH, railway infrastructure elements are considered as objects and each object has attributes. LH-ESTW-R defines conditions that objects need to fulfil, such that establishment of safe routes can be guaranteed. There are two types of conditions. Conditions that objects need to fulfil in each phase of developing a safe route are called static conditions. Conditions that specify the relationship of phases of safe route development are called dynamic conditions. Static and dynamic conditions are defined based on attributes of objects, propositional logic and temporal logic [HR04] in OOLH.

During the translation of conditions from natural language to OOLH, the
consistency among conditions can be checked. In order to provide a chance for different professions to understand and analyze the meaning of OOLH, OOLH can be transformed into truth tables or decision tables. An expected behavior of an RIS can be formalized as a decision table or propositional formula. OOLH can be checked against them, such that the correctness of these translated conditions can be verified. The well-formalized and correct OOLH can be used to evaluate an establishment of a safe route with a given situation. Furthermore, the conformity of a design of an RIS to OOLH can be verified.

A tool is proposed to be implemented to support analyzing, verifying and checking consistency of formulas in OOLH. In this tool, users can access the attributes and conditions of each object that need to be fulfilled during the establishment of a safe route. These conditions are expressed in propositional logic. They can be transformed into decision tables and truth tables. The expected behavior or a design of an RIS can be specified in decision tables in this tool, such that the correctness of OOLH or the conformity of the design against OOLH can be verified. A possible contradiction of a decision table is checked automatically by this tool.

In this report, a general introduction to formal methods, RIS and LH-ESTW-R is given in chapter 2 and chapter 3. Chapter 4 describes the goals and the first approach of this work and the ideas of OOLH. Propositional and temporal logic are introduced in chapter 5. In chapter 6, a simple case study is given to demonstrate the idea of developing OOLH and applying propositional logic in specifying LH-ESTW-R.
Chapter 2

Formal methods

Formal methods provide a mathematical framework to users in specifying and analyzing system requirements [Win90, Mon03, TE04, fmv06]. Tools are implemented based on the mathematical frameworks to verify a system design against these system requirements. As a result, applying formal methods in developing systems becomes an important research topic [HB95, HP00]. In this chapter, a general introduction to formal methods is first given. The advantages of applying them in the system development process are then discussed.

2.1 Introduction

System requirements are said to be formally specified if they are expressed by applying mathematical concepts and mathematical logic. Formal methods provide a framework to specify system requirements formally. The first component of a formal method is a formal specification language [AP98]. The language comprises syntax (notations) and semantics (meaning of the notations). The semantic meaning of every notation is mathematically and clearly defined. As a result, one can formalize the same concept with equivalent notations and if the formalizations are correct, there will be only one meaning of those formalizations. Propositional logic is one of these mathematical specification languages. For example, the sentence $s_1$ "A point is not allowed to lock without reservation" can be formalized in propositional logic as follows:

$$s_1 \equiv \neg(\text{locked} \land \neg\text{reserved})$$

locked : point is locked
reserved : point is reserved

One can also formalize the sentence equivalently as $(\text{locked} \rightarrow \text{reserved})$. This sentence can be read as "if the point is locked, then it is also reserved". Although the concept is represented by different notations, the semantic meaning has not changed and there is only one way to interpret the concept. This
equivalent relationship can be found by the syntactical part of the language. For example,

\[ s_1 \equiv \neg(\text{locked} \land \neg \text{reserved}) \]
\[ \equiv (\neg \text{locked}) \lor \text{reserved} \]
\[ \equiv \text{locked} \rightarrow \text{reserved} \]

The second component of formal methods is verification of well-formalized specifications. If the specification language is well defined syntactically and semantically, computer programs can be written to analyze the specifications. This means a well-formalized specification or design can be verified against the expected behavior of the system. For example, a model checker implements the concepts of the formal method called model checking [CGP99], one can formalize a design of an interlocking system as a Finite State Machine (FSM) and the expected behavior of the system as checking conditions, for example, safety rules [HK06]. The model checker verifies the FSM against checking conditions. It produces a counterexample if the FSM does not conform with a condition.

2.2 Advantages of using formal methods

As mentioned above, formal methods consist of formal specification languages and frameworks to verify system specifications against the expected behavior of the system. These two components can bring advantages to users as follows:

- Reduction of ambiguity of specifications: most of the system requirements are described in natural language in the first place. Some sentences that are written in natural language are ambiguous and unclear. Terms are often not precisely defined. Ambiguous statements can be interpreted in different ways by different professions. For example, the sentence \( s_2 \) ”It can be indicated: Ks1 or Ks2 and additional Zs3-Indicator” (German: es kann angezeigt werden: Ks1 oder Ks2 und zusätzlich Zs3-Anzeige). This sentence can be interpreted in two ways as follows:

"It can be indicated: Ks1 or Ks2 and in both cases an additional Zs3-Indicator” (German: es kann angezeigt werden: Ks1 oder Ks2 und in beiden Fälle zusätzlich mit Zs3-Anzeige)

or

"It can be indicated: Ks1 or it can be indicated: Ks2 and additional Zs3-Indicator” (German: es kann angezeigt werden: Ks1 oder es kann angezeigt werden: Ks2 und zusätzlich Zs3-Anzeige)

By applying formal methods in the development process, system requirements need to be specified based on the formal specification language of
the corresponding formal methods, each requirement becomes a precise mathematical statement. A single interpretation can be deduced from the statement and misleading interpretations can be avoided. When the specification is unambiguous, it is easier for professions from different areas to discuss the specification or understand the system because terms are well defined and equivalent concepts can be deduced by the mathematical rules. In the case of computerized interlocking systems, if the specification is expressed by mathematical concepts or logics, computer programmers who do not have sufficient knowledge in railway systems, can relatively easy understand the specification. This can increase the efficiency of the software development process and the correctness of software systems.

- Increasing the understandability of specifications: It is sometimes difficult to understand and analyze the meaning of a statement that is written in natural language. With the help of mathematical concepts, one can capture the meaning and concept of a statement efficiently. For example, during the evaluation of ZPZ (German: Zulassungsprüfung), if a point is located within the requested route and is needed to be used as a route element, then one of the conditions that a RIS needs to check is $s_3$ "The point must not be locked in an improper position" (German: Die Weiche darf nicht in Nicht-Solllage verschlossen sein). The meaning of the sentence can be easily understood if it is written in propositional logic as follows:

$$s_3 \equiv \neg (\neg \text{solllage} \land \text{verschlossen}) \equiv (\text{solllage} \lor \neg \text{verschlossen})$$

- Checking the consistency and completeness of specifications: Inconsistency can be found if the specification is written using mathematical concepts. For example, figure 2.1 shows a system development process called V-model. In the stage of system requirements, these requirements are specified with a set of logical statements and within this set of logical statements. When it is noticed that there exists a statement which should be true and false at the same time, then inconsistency is said to be found. As mentioned above, formal methods support proving the conformity of the specification to the expected behavior of the system. If the specification
does not fulfil the expected behavior, it means that the system has not been specified completely or correctly. For example, the result produced during the phase of system requirements can be verified against the output from the phase of user requirements, it can check whether the developed system requirements satisfy the requirements from the users. The costs and resources can be reduced if the incorrectness of the system requirements specification can be found in an early stage of the system development process.

Figure 2.1: System development process, V-model
Chapter 3

Railway interlocking systems and LH-ESTW-R

LH-ESTW-R is the system requirements specification of regional computerized interlocking systems for Germany. RIS are responsible for setting up safe routes (German: Fahrstraße) based on safe route requests. Safe routes ensure no train collision would happen and the trains can be driven from the planned beginnings to the planned destinations. LH-ESTW-R specifies requirements that need to be implemented in a German regional computer-based interlocking system, such that development of safe routes can be guaranteed. In this chapter, the concept of safe routes is first given. The characteristics of LH-ESTW-R is then discussed.

3.1 Safe Routes

A safe route is composed of a route (German: Fahrweg), an overlap (German: Durchrutschweg/DWeg) and the flank protections for the route and overlap (see figure 3.1). The infrastructure elements that are located within the safe route must be set properly. They are locked as proper elements for this safe route based on their function provided for the safe route. For example, a point of a route must be set in a proper position such that the train runs into the correct track. In figure 3.2, in order to develop a route from A to B or B to A, the point
is set to the direction of the straight track (German: Stammgleis) and from A to C or C to A, the point is set to the direction of the diverging track (German: Zweiggleis). In this work, for the first case, the point is said to be set in a straight position and for the second case in a divergent position.

![Diagram of point with A, B, C, and arrows indicating directions]

Figure 3.2: Point

The route is used by the train to move from the start to the planned goal. Elements that are located between the start and the goal of the route are reserved and locked as route elements (German: Fahrwegelement). Their proper setting for safe routes is also marked during the reservation. If the current usage of the element is enquired later, this corresponding reserved setting is also provided at the same time. For example, the point W21 is put to the direction of the diverging track and is not reserved or locked for other safe routes. During the development of $R_{route1}$, W21 is reserved and is marked as the direction of the straight track because this is the proper setting of W21 for $R_{route1}$. Basically, if route elements are locked or reserved for a safe route, they cannot be used by other safe routes as route elements. Otherwise, collision might happen when a route element is locked or reserved for more than one route as route element and be used at the same time. In figure 3.1, the route $R_{route1}$ is (A.N2). This means, the route begins at the signal A and ends at the signal N2. The route elements of $R_{route1}$ are the point W21, the signal A, signal N2 and the track section between the signals.

The overlap is a track section that is in front of the route. This track section must be ensured to be clear, such that if the driver does not stop the train in time, the train will not collide with a train ahead. Elements along the overlap are locked as overlap elements, for example, W22 is locked as an overlap element for DWeg$_{route1}$ in figure 3.1.

Flank protections ensure that no trains can be driven into the safe route through turnouts or crossings. Turnouts or crossings are composed by points. This means, RIS needs to search flank protections for points that belong to the route and overlap. An element that provides flank protection to a route element or overlap element is called a flank protection element (German: Flankenschutzelement). Signals and points can be flank protection elements. The points that provide flank protections need to be set properly, such that trains cannot run into track sections of the safe route. For example, in figure 3.1, W11 and W12 provide flank protections to W21 and W22 respectively. W11 and W12 need to be set to the straight position, such that trains on track 1 cannot run into track 2. RIS are responsible for checking the possibility to reserve or lock each of the
infrastructure elements along the safe route in a proper role and position, and for monitoring these settings continuously until the safe route is properly used and released.

3.2 LH-ESTW-R

LH-ESTW-R contains the system requirements of regional computerized RIS and is written in German natural language. The interlocking logic of RIS that the computer-based RIS needs to be conform with for establishing, monitoring and releasing safe routes is specified in LH-ESTW-R. The implemented RIS needs to fulfill those requirements that have been written in this document. LH-ESTW-R describes the life cycle of safe routes and the conditions or constraints that the route, overlap and flank protection elements of a safe route need to fulfill during its life cycle. RIS needs to monitor each phase of the life cycle of safe routes. It ensures that the conditions of the elements during each phase of the safe route are satisfied. These constraints and conditions are expressed in statements or exceptionally allowed situations or cases. The statements can be relevant to safe routes, route, overlap and flank protection elements of safe routes. Exceptional situations, for example, overlapping overlaps, are described as scenarios. Rules that can be applied to check whether the setting of the elements is described as one of these exceptional situations, are not specified directly.
Chapter 4

Object Oriented Lastenheft (OOLH)

The advantages of specifying system requirements with formal methods has been discussed in chapter 2. The focus of this work is to develop such a formal framework to express LH-ESTW-R, such that railway engineers and computer scientists can benefit from applying formal methods in a system development process. Based on the characteristics of LH-ESTW-R, OOLH is proposed as the framework to achieve this goal. In this chapter, the goals of this work are further discussed in section 4.1 and OOLH is introduced in section 4.2

4.1 Goals

As mentioned above, computerized RIS must satisfy all the system requirements of LH-ESTW-R, such that safe routes can be established and monitored by the RIS. This implies the system development team of RIS needs to understand this document in order to specify, design, implement and test the system. However, it is difficult to understand this document unambiguously and easily because it is written in natural language. For example, without domain specific knowledge, it is difficult to understand the exceptional situations and technical terms. This could lead to misunderstandings. Furthermore, the consistency and completeness of requirements is difficult to be verified. It is not easy for computer scientists and railway engineers that participate in the system development process to understand these system requirements completely. As a result, there is a need to investigate a new way in specifying LH-ESTW-R systematically in order to increase the understandability of LH-ESTW-R and efficiency in developing interlocking systems.

The main goal of this work is to investigate the feasibility in specifying LH-ESTW-R by applying formal methods, in other words, searching the suitable mathematical concepts and logic to express the requirements, such that the requirements can be understood easier, the consistence and completeness of LH-ESTW-R can be analyzed and verification of a design of RIS against the requirements can be supported via this specification method.

The result of this work will bring railway engineers and computer scientists
advantages that have been mentioned in section 2.2: avoid ambiguity and support understanding and support verification. Checking the consistency and completeness means searching for the existence of contradictions in LH-ESTW-R and verifying whether the expected behavior of a RIS are guaranteed if the RIS conforms with LH-ESTW-R. The latter question means "if a system satisfies LH-ESTW-R, can safe routes be built?", one needs to prove whether establishment of safe routes is a consequence of the system requirements specification. Supporting verification means, the well formalized specification can be used later as checking conditions in verifications and answer the question "does a design model satisfy each requirement of LH-ESTW-R?".

4.2 OOLH

In order to express and verify the consistency of the requirements systematically, an object oriented approach in specifying LH-ESTW-R is proposed and will be investigated in this work. LH-ESTW-R that is specified based on this approach is called OOLH. Safe routes and infrastructure elements of safe routes are then considered as infrastructure objects. These objects has attributes. The conditions and exceptional situations of each phase during the life cycles of safe routes are modeled and expressed in logical formulas based on these attributes. They are then combined and transformed into conditions that each of the infrastructure objects need to satisfy during these phases, such that the exceptional situations can be captured by applying these formulas to check the attributes of each object. The phases of safe routes are modeled as states of objects. In order to reach a state, conditions or constraints of each infrastructure object have to be fulfilled. During the transformation of statements into formulas, the consistency of the requirements can be verified. The formalized requirements can be understood easier by analyzing the conditions that each object needs to fulfill in each phase of the safe route. The expected behavior of objects can be formalized based on their attributes as logical formulas. The formulas of LH-ESTW-R and the expected behavior can then be compared or in other words, the logical consequence of the formulas need to be checked. There is also a tendency in specifying the designs of interlocking systems by object-oriented modeling. For example, interlocking systems are composed of infrastructure objects and the design of these objects is specified by Unified Modeling Language (UML) state machines [BF04, HK06]. Formulas of objects can be used as checking conditions to verify the design. For example, the existence of states during the life cycle of the objects in the design can be verified. The object conforms with the formulas at each specified state.

In this first approach, logical formulas are specified by propositional or temporal logic [HR04]. The phases of safe routes are modeled as states of objects. In order to reach a state, conditions or constraints of each infrastructure component have to be fulfilled. This type of conditions is called static. Static conditions can be specified by using propositional logic. In some phases of safe routes, conditions need to be fulfilled continuously, such that the status of safe routes is guaranteed and they are related in sequence and order, for example, ZPZ (German: Zulassungsprüfung) and adjustments of points (German: Umstellung von
Weichen) are two phases of the safe route, points that belong to a safe route cannot be adjusted if the ZPZ of the safe route is not evaluated to be positive. In other words, points cannot be adjusted if the constraints or conditions of ZPZ for elements of the safe route are not satisfied. One of the possible ways in specifying time-oriented conditions is using temporal logic.

- Propositional logic: It is used to specify the conditions and constraints of each infrastructure element in each phase. For example, one of the conditions $\phi_1$ that a safe route $SR_1$ is evaluated as ZPZ positive is that the infrastructure elements which are located between the start and goal of the route $R_1$, must not be reserved or locked for other routes as route elements unless the reservation or locking is made for $SR_1$ and $SR_1$ is a non-stop safe route (German: Durchfahrt) (see figure 4.1). This is modeled as one of the conditions of the phase ZPZ positive. This specification can be modeled as a formula in propositional logic as follows:

$$ZPZ_{\text{positive}} \equiv \phi_1 \land \phi_2 \land ... \land \phi_n$$

$$\phi_1 \equiv FWEL(r) \lor FWEL(v) \rightarrow \text{durchfahrt} \land \text{solllage}$$

$ZPZ_{\text{positive}}$: ZPZ of the route is positive

$FWEL(v)$: the object is locked as a route element

$FWEL(r)$: the object is reserved as a route element

$\text{durchfahrt}$: the reservation or locking belongs to the part of the currently requested safe route

$\text{solllage}$: the object is set in a proper position

The formula $\phi_1$ can be read as if the object is locked or reserved as a route element ($(FWEL(r) \lor FWEL(v))$ is true). It is important to check whether the reservation or locking is made for $SR_1$ and the object is set in a proper position for the current evaluation. If it is neither the case (durchfahrt is false or solllage is false), then the formula is evaluated to false ($\phi_1$ is false) and ZPZ is negative ($ZPZ_{\text{positive}}$ is negative). If the object is not locked or reserved as a route object ($(FWEL(r) \lor FWEL(v))$ is false), $\phi_1$ is evaluated to true and the evaluation of $ZPZ_{\text{positive}}$ depends on the truth value of $\phi_2 \land ... \land \phi_n$.

- Temporal logic: It is used to specify the dynamic, ordering or sequence oriented conditions in LH-ESTW-R. For example, in order to let objects reach the state $ZPZ_{\text{positive}}$, the conditions of reaching this state $ZPZ_{\text{positive}}$ needs to be fulfilled. This can be specified in one of the temporal logics, for example in **Linear Time Logic (LTL)** as $\mathbf{G}$lobally($\text{state}(ZPZ_{\text{positive}})$ $\leftrightarrow ZPZ_{\text{positive}}$) and it has to be fulfilled for the system. If the system is composed of objects then this formula should be true for each of these objects.
To express the meaning or semantics of each propositional formula of objects, truth tables (see tables 4.1) and decision tables (see tables 4.2) are used. Formulas can be transformed into truth tables or decision tables, such that the formulas can be understood easier by railway engineers and computer scientists. This provides also a chance for the users to check the correctness of the transformed requirements. Decision tables and the approach in transforming a formula to a decision table need to be defined formally.

<table>
<thead>
<tr>
<th>solllage</th>
<th>verschlossen</th>
<th>solllage ∨ ¬verschlossen</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

T: true, F: false

Table 4.1: Truth table of \( s_3 \)

<table>
<thead>
<tr>
<th>C1</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set in proper position (solllage)</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>C2 locked (verschlossen)</td>
<td>-</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>A1 ( s_3 ) is evaluated to be True</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

Y: yes, N: no

Table 4.2: Decision table of \( s_3 \)

To support analyzing, verifying and checking consistency of formulas in OOLH, a tool is proposed to be developed (see figure 4.2). In this tool, propositional formulas of objects that are deduced from LH-ESTW-R can be accessed. These formulas can be transformed into decision tables or truth tables based on the needs of users. The expected behavior of the requirements can also be formalized as either logical formulas or decision tables. The logical consistence between the formulas from OOLH and the expected behavior are verified, such that the consistency and completeness of LH-ESTW-R are analyzed. By extending this idea, if the specification of an RIS from the development team is formalized as decision tables or logical formulas, the consistency of the design and OOLH can also be verified. The underlying supportive technique of verification are Ordered Binary Decision Diagrams (OBDDs) [Bry86] and SATisfiability.
solver[MMZZ01] (SAT). Checking whether the design conforms the time-oriented requirements of LH-ESTW-R can be verified by applying model checking. In this case, the design is specified as state machines and the requirements in temporal logic.

Decision tables are an important form of expressing the meaning of OOLH in this work, as a result, a formal definition of decision tables is given in this work. One of the advantages of using decision tables in expressing requirements is that the requirements can be expressed in a compact form by combining rules. The underlying technology that is proposed to generate such decision tables from a propositional formula in this work is OBDDs. The size or compactness of the decision tables depends on the size of OBDD. The compactness of OBDD depends on the ordering of the propositions. Algorithms have been developed from researchers to produce optimized orderings for building OBDD. Based on these algorithms, a compact decision table can be generated and represent the logic of the propositional formula. Further investigation is needed to search for a suitable algorithm and integrate this to the tool of OOLH.

![Decision Table Diagram](image)

**Figure 4.2: Tool for OOLH**
Chapter 5

Propositional logic and Temporal logic

In OOLH, propositional logic and temporal logic is the formal specification language to express the conditions that infrastructure objects need to satisfy during the development of a safe route. In this chapter, propositional logic and temporal logic is introduced in section 5.1 and 5.2.

5.1 Propositional Logic

Propositional logic is one of the classical logics to model declarative statements (or also called formulas) and concepts and support to reasoning. It comprises two components: syntax and semantics. The syntax includes the definition of notations that can be used in the logic and deduction rules to manipulate the notations. The semantics defines the meaning of notations. As a result, it is considered as formal specification language (see section 2.1). One of the important application of propositional logic is reasoning, it means deducing whether a formula is a logical consequence from a set of formulas and whether two formulas are equivalent. The first concept can be expressed as follows:

\[ \phi_1, \phi_2, ..., \phi_n \models \psi \]

\( \{\phi_1, \phi_2, ..., \phi_n\} \) is a set of formulas and the formulas are called premises. \( \psi \) is another formula and it is called the conclusion. \( \models \) is called semantic entailment. This expression is called a sequent. In the syntactical aspect, a formula has no meaning, it is considered only as a sequence of symbols. The meaning of a formula is defined by the semantics of the logic, a formula can be evaluated to either true or false. The sequent can be read as following, if \( \phi_1, \phi_2, ..., \phi_n \) are all evaluated to true, then \( \psi \) is also evaluated to true. The sequent is said to be valid and \( \psi \) is the logical consequence of \( \phi_1, \phi_2, ..., \phi_n \). The logical consequence of two formulas can also be checked by applying the syntactical rules of propositional logic. In this case, the semantic entailment symbol is replaced by the syntactic entailment \( \vdash \).
be expressed in the form of sequent as follows:

\[ M \models \phi \]

- \( M \) is the design model of the interlocking system
- \( \phi \) is a requirement of LH-ESTW-R

This formulation expresses the question "Does a design model satisfy each requirements of LH-ESTW-R?" and it means the requirement \( \phi \) is the logical consequence of the design model \( M \), in other words, \( M \) conforms to \( \phi \)

\[ \Gamma \models \psi \]

- \( \Gamma \) is the set of requirements of LH-ESTW-R
- \( \psi \) is the expected behavior of the interlocking system, building safe routes

This formulation expresses the question "If a system satisfies LH-ESTW-R, can safe routes be built?" and it means the expected behavior \( \psi \) is the logical consequence of the requirements \( \Gamma \). This means, the set of requirements is satisfied, a safe route can be built.

In propositional logic, the fundamental element is a proposition, called a formula. It can be considered as a declarative statement in natural language. In section 2.1, \( s_1 \) has been specified as a complex formula in propositional logic, it is called a complex formula because it is composed of atomic formulas and a logical connective. Complex formulas are simply called formulas in this report. For example, in section 2.1, \( s_1 \) is a complex formula. It is composed of two atomic formulas \textit{locked, reserved} and the connectives \( \neg, \land \). They cannot be further decomposed and do not contain any connectives. \( \neg \) and \( \land \) are called logical connectives. The syntactical part of propositional logic defines the possible connectives that are allowed to be used in propositional logic and the rules in forming formulas. These rules are expressed in Backus Naur Form (BNF) as follows:

\[ \phi ::= q | (\neg q) | (\phi \land \phi) | (\phi \lor \phi) | (\phi \rightarrow \phi) \]

- \( q \) is an atomic formula and \( \psi \) is a formula. The formulas that are formed based on these rules are called well formed formulas (wff). Some constructions of wff are standardized and considered as normal forms. One of these normal forms is Disjunctive Normal Form (DNF). A DNF is a disjunction of conjunctive clauses \( c \). Each clauses consists of atomic formulas which are connected by \( \land \) and these clauses are connected by \( \lor \). A wff \( \phi \) is considered in DNF if it is built based on the rules as follows:

\[ c ::= q | (\neg q) | (c \land c) \]

\[ \phi ::= c | (c \lor \phi) \]
The semantic part of propositional logic defines the meaning of the connectives and meaning of atomic formulas. Truth values are assigned to a formula in order to give a meaning to a sentence. The truth values in propositional logic are true \( T \) and false \( F \). The assignment of a truth value to a formula is called a valuation or model of a formula. This means, in propositional logic, each formula can be valued to be true or false. The meaning of connectives are also defined, the connectives \( \neg \), \( \land \) and \( \rightarrow \) are defined and presented as truth tables as shown in table 5.1.

Table 5.1 shows the truth tables of the three logical connectives, a truth table contains all possible valuations of the formula. In the truth table of the logical connective \( \land \), the first two columns express the possible valuations of the atomic formula \( \phi \) and \( \psi \), the third column expresses the results of the valuations. The logical connective can be interpreted as and in natural languages. For example, a statement \( s_4 \) "The point is in the left position and the signal is in clear aspect", if both conditions are satisfied (the valuations to \( T \)), then one considers \( s_4 \) to be correct (the valuation of \( T \)). If one of the conditions is not satisfied (the valuation to \( F \)), then \( s_4 \) is considered as incorrect (the valuation to \( F \)). This interpretation is expressed in the truth table. Truth tables allow one to consider the possible valuations or meanings of a formula. The truth tables of two formalizations of \( s_4 \) are shown in table 5.2 and 5.3.

<table>
<thead>
<tr>
<th>locked</th>
<th>reserved</th>
<th>( \neg )reserved</th>
<th>locked ( \land ) ( \neg )reserved</th>
<th>( \neg )(locked ( \land ) ( \neg )reserved)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Table 5.2: Valuations of \( \neg \)(locked \( \land \) \( \neg \)reserved)

As it has mentioned above, one can reason formulas by checking whether they are logical consequence and logical equivalences. In section 2.1, the concept of
equivalent sentences or formulas has been introduced, two formulas $\phi$ and $\psi$ are said to be equivalent if and only if they have the same valuations. This means, when $\phi$ is valuated to be $T$, then $\psi$ is valuated to be $T$. If $\phi$ is valuated to be $F$, then the truth value of $\psi$ is also $F$. This can be shown by comparing the results of the truth tables of the formulas. The result of truth tables 5.2 and 5.3 show that this two formulas are logically equivalent.

Another two concepts of propositional logic that need to be introduced are satisfiability and validity. A formula is said to be satisfiable, if the formula is assigned the value $T$ at least once among its evaluations. For example, $\neg(\text{locked} \land \neg\text{reserved})$ is satisfiable. This means, it is possible to satisfy the formula. A contradictory formula can never be satisfied. If the conditions and constraints of LH-ESTW-R are specified as a set of formulas $\phi_1 \land \phi_2 \land \ldots \land \phi_n$, one can check whether a contradiction exists in the set of formulas by analyzing its satisfiability.

A formula is said to be valid, if it is always valuated to be true. This means, the formula is correct or holds under any situation or possible combination of the atomic formula. For example, $(\text{locked} \lor \neg\text{locked})$ is a valid formula (see Table 5.4). The results of all valuations are $T$. Checking the logical consequence can be reformulated to checking whether $\phi_1 \rightarrow (\phi_2 \rightarrow \ldots (\phi_n \rightarrow \psi))$ is valid. For example, consider the sequent $\phi, \phi \rightarrow \psi \models \psi$. This sequent is considered as valid if and only if the premises $\phi$ and $\phi \rightarrow \psi$ are both true and the conclusion needs to be true. This can be checked by considering the truth table of the connective $\rightarrow$ in table 5.1. There is only a single valuation, such that both premises $\phi$ and $\phi \rightarrow \psi$ are $T$, this is the first line. In this line, $\psi$ is also valuated as $T$. As a result, this sequent is considered to be valid. As mentioned above, one can also check the validity of this sequent by checking the validity of $\phi \rightarrow ((\phi \rightarrow \psi) \rightarrow \psi)$. In table 5.5, the formula $\phi \rightarrow ((\phi \rightarrow \psi) \rightarrow \psi)$ is valuated to be $T$ in all the valuations. As a result, the formula is considered to be valid. This means, $\psi$ is the logical consequence of $\phi$ and $\phi \rightarrow \psi$.

<table>
<thead>
<tr>
<th>locked</th>
<th>$\neg$locked</th>
<th>locked $\lor$ $\neg$locked</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Table 5.4: Valuations of $\text{locked} \lor \neg\text{locked}$

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\psi$</th>
<th>$\phi \rightarrow \psi$</th>
<th>$\phi \rightarrow ((\phi \rightarrow \psi) \rightarrow \psi)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Table 5.5: Valuations of $\phi \rightarrow ((\phi \rightarrow \psi) \rightarrow \psi)$
5.2 Temporal Logic

Propositional logic is said to be static. A proposition `point = left ∨ right`, it is valued to be true in a model when there exits an element that can be turned to the right or the left in the system. The proposition will always be true during the life time of the system. However, there exist some conditions which change their truth values over time. For example, `locked` can be true if an element is locked and this element can be unlocked at another time point. Propositional logic do not support specification of dynamic or order oriented requirements explicitly. Although time can be defined as another variable in the language, the complexity of language will also be increased. RIS are reactive systems, their reactions and behavior are based on its interaction with the environment. As a result, there exists dynamic aspects in LH-ESTW-R. Temporal Logic can be used to model the dynamic behavior of systems and decrease the complexity of introducing time in the domain specific language.

Temporal logic is used to describe the properties of the system over time. Temporal logic models time as a sequence of states. As a result, one can use temporal logic to describe conditions that have to hold as the system evolves. For example, a condition – called proposition `p` – always has to be true in the life time of the system, this can be represented in temporal logic by `G(lobally)p`.

![Figure 5.1: System satisfies AG(deadlock → AG(deadlock))](image)

Temporal logic provides temporal operators, for example, `G`, for specifying the properties w.r.t. time. Computation Tree Logic (CTL) is one type of temporal logic that is commonly used to specify checking conditions in model checking. In CTL, time is represented as a tree-like structure called computational tree whose nodes are the states of the system. The future of the system can be represented by different branches that emanate from the current state. As a result, one can specify a condition that has to hold in some path. Or a condition has to hold in all paths, it means that condition must hold true no matter how the system evolves in the future. The corresponding notations are as follows:

- Path quantifiers (CTL) `E(for some paths)`
A (for all paths)

- Temporal operators (CTL)
  X (next state)
  G (globally, all the states)
  F (in some future states)
  U (Until)

Path quantifiers are used to describe the properties of paths. One can specify the condition that has to hold in a specific path by using temporal operators. With the path quantifiers and temporal operators, one can express, for example, "whenever the system reaches a deadlock, then it will remain in the deadlock forever" in CTL as AG(deadlock → AG(deadlock)) (see figure 5.1). Figure 5.2 shows a system M whose initial state satisfies CTL formulas. Linear Time Logic (LTL) is another type of temporal logic. In LTL, time evolves linearly and the future is composed of a set of linear time paths. As a result, there are no path quantifiers in LTL. An LTL formula has to hold for all paths (see figure 5.3). LTL provides all temporal operators that can be used in CTL to specify future events. Temporal operators Y, O and S are used to specify events that happened in the past.

- Temporal operators (LTL)
  X (next state)
  G (globally, all the states)
  F (in some future states)
  U (Until)
  W (weak Until)
  R (release)
  Y (previous state)
  O (once)
  S (since)
when all route elements are adjusted to the proper position and locked, this can
and changes to the state FuMBli (German:Fahrstraßenüberwachung Blinklicht)
R, for example, after the evaluation of ZPZ, RIS continues to monitor the route
and changes to the state ZPZ
logic for specification needs to be further investigated.

As mentioned above, ordering of phases and some events exists in LH-ESTW-R, for example, after the evaluation of ZPZ, RIS continues to monitor the route and changes to the state FuMBli (German:Fahrstraßenüberwachung Blinklicht) when all route elements are adjusted to the proper position and locked, this can be expressed in LTL as follows:

\[ G \left( (\text{state}(ZPZ_{\text{positive}})) \rightarrow F(\text{state}(FuMBli)) \right) \]

This expression states that during the life time of the system G, if it reaches the state \( ZPZ_{\text{positive}} \) state(\(ZPZ_{\text{positive}}\)), then the state FuMBli state(FuMBli) will be reached in the future F. This condition can be used later on during the design and implementation phase, in order to verify whether this correlation exists in the design or implementation.

At this stage of the work, the main focus is put on the possibility of specifying the static conditions of LH-ESTW-R with propositional logic. The time-oriented aspect of LH-ESTW-R has not been considered. The application of temporal logic for specification needs to be further investigated.
Chapter 6

Case Study

In this chapter, an example of applying propositional logic to specify requirements of LH-ESTW-R is given. This example is concerned with the evaluation of infrastructure elements within the overlap DWeg_{current} of the currently requested safe route SR_{current} during ZPZ. In LH-ESTW-R, there are two different sets of requirements that are applied to check the elements within DWeg_{current}. The first set of requirements describes the basic evaluation of DWeg_{current}. The second set of requirements expresses the criteria that elements within DWeg_{current} need to fulfill. The idea of OOLH is to combine these two sets of requirements into a consistent formula. This consistent formula can be used to verify the elements within DWeg_{current}. Instead of reading the system requirements to understand the ZPZ evaluation of DWeg, the consistent formula can provide the objects' attributes that an RIS needs to check, in order to decide whether the objects can be used as overlap elements. Section 6.1 describes the concept of the evaluation of ZPZ. The characteristics of the ZPZ requirements in LH-ESTW-R are also discussed in this section. The modeling of the basic conditions is given in section 6.2. Section 6.3 focuses on the modeling of overlapping overlaps. The object conditions that are relevant to the objects within DWeg_{current} are modeled in section 6.4. The consistent formula that is specified based on the basic conditions and the object conditions is given in section 6.5. A conclusion is given in the end of this chapter.

6.1 Modeling of ZPZ

ZPZ is one phase of the life cycle of a safe route. When a safe route is requested, RIS checks the condition of elements between the start and the goal of the route and elements within overlap and evaluates whether they can be reserved for this requested safe route. This is done during the phase ZPZ. If all elements which belong to the route can be reserved for this safe route, then ZPZ_{Route} is said to be positive. Similarly, all elements within DWeg_{current} are evaluated and ZPZ_{Dweg} is evaluated to be positive if they can be used as overlap elements for DWeg_{current}. The ZPZ evaluation of SR_{current} (ZPZ_{positive}) is said to be positive if ZPZ_{Route} and ZPZ_{Dweg} are both estimated to be positive. If there exist elements which cannot be reserved as route elements or overlap elements...
for SR\textsubscript{current}, then the ZPZ evaluation for SR\textsubscript{current} is said to be negative. This can be modeled as follows:

\[ ZPZ\text{positive} \equiv ZPZ\text{Route} \land ZPZ\text{Dweg} \]

This propositional formula expresses that the ZPZ evaluation of SR\textsubscript{current} is said to be positive (ZPZ\text{positive} is true), if both ZPZ evaluations of the route and overlap are positive (ZPZ\text{Route} is true and ZPZ\text{Dweg} is true).

As mentioned above, in LH-ESTW-R, there are two main parts which express the conditions that elements within DWeg\textsubscript{current} need to fulfill. The first part are the basic conditions for the evaluation of DWeg\textsubscript{current} (German: Grundsatz) (ZPZ\text{Dwegcond}1). The second part are the object conditions (German: ZPZ-Bedingungen je Fahrwegelement im Durchrutschweg) (ZPZ\text{Dwegcond}2). It contains conditions that are relevant to the objects within DWeg\textsubscript{current}. These two parts of requirements are considered as static conditions. It is assumed attributes of object will not changed during the evaluation of ZPZ.

As mentioned in section 4.1, these conditions are written in natural language. It is difficult to obtain an overview of the evaluation of ZPZ\text{Dweg} without an extensive reading and analysis of the text. Therefore, contradictions of conditions cannot be found easily. In OOLH, these problems are handled by applying propositional logic to specify these conditions in this example as follows:

1. The attributes of the objects are first defined and deduced based on these two parts of conditions.

2. The basic conditions and object conditions are translated to propositional logic based on these attributes and become ZPZ\text{Dwegcond}1 and ZPZ\text{Dwegcond}2 (see section 6.2 and 6.4), respectively.

3. The Goal is to combine these two parts together in order to formulate a condition or formula, ZPZ\text{Dweg}, for checking the evaluation of DWeg\textsubscript{current} of an infrastructure object. The concept of OOLH is to specify conditions that can be applied directly to infrastructure objects. As a result, ZPZ\text{Dweg} is defined based on the modeling of the object conditions ZPZ\text{Dwegcond}2.

4. To check the existence of contradictions in ZPZ\text{Dwegcond}2 and ZPZ\text{Dwegcond}1 and complete the specification of ZPZ\text{Dweg}, ZPZ\text{Dwegcond}2 and ZPZ\text{Dwegcond}1 are compared and analyzed (see section 6.5). Those interpretations of ZPZ\text{Dwegcond}1 that are evaluated to be true, must also be found true in ZPZ\text{Dwegcond}2. As a result, those accepted situations of ZPZ\text{Dwegcond}1 are included in ZPZ\text{Dweg}.

6.2 Modeling of basic conditions

There are three basic conditions that need to be considered for the evaluation of DWeg\textsubscript{current}. ZPZ\text{Dwegcond}1 is composed of these three conditions as follows:
ZPZ_{Dwegcond}\equiv condition1 \land condition2 \land condition3

The first condition condition1 consists of the concept of overlapping overlaps and it is described only generally. The complete modeling of overlapping overlaps is given in section 6.3.

- Condition1: "Grundsatz für ZPZ: Im Breich ... Ziel-D-Weg-Ziel ..., es darf also kein Fahrwegelement (FW-EL) noch für eine andere Fahrstraße verschlossen sein. ... Es gelten aber folgende Ausnahmen: a) Durch- fahrt ... b) Überlappende D-Wege: Im Bereich Ziel-D-Weg-Ziel dürfen ein oder mehrere D-Wege vorhanden sein, wenn die verschlossenen Weichen dieser D-Wege durch ihre Lage den neu hinzukommenden D-Weg nicht ausschließen.” [LH-ESTW-R F1, 2.2.1, P.13]

Condition1 states that ZPZ_{Dwegcond} of D Weg current is partly evaluated to positive if elements between the goal of the route (German:Ziel) and the goal of the overlap (German:D-Weg-Ziel) are not locked for another route except for two situations. These two situations are as follows:

1. SR current is a non-stop safe route (see figure 4.1) and these elements are locked as a route element of SR current

2. D Weg current is an overlapping overlap.

The Condition1 is specified in propositional logic as follows:

\[
condition1 \equiv \phi_1 \land (\neg FWEL(b) \rightarrow (\phi_2 \land (\neg DWEL(b) \rightarrow \phi_3)))
\]

\[
\phi_1 \equiv (FWEL(b) \rightarrow durchfahrt \land solllage)
\]

\[
\phi_2 \equiv (DWEL(b) \rightarrow überlappen)
\]

\[
\phi_3 \equiv (FLEL(b) \rightarrow solllage)
\]

\[
DWEL(b) \equiv (DWEL(v) \lor DWEL(r))
\]

\[
FWEL(b) \equiv (FWEL(v) \lor FWEL(r))
\]

\[
FLEL(b) \equiv (FLEL(v) \lor FLEL(r))
\]

\[
durchfahrt: \quad \text{the reservation or locking belongs to } SR_{current} \text{ and } SR_{current} \text{ is a non-stop route}
\]

\[
überlappen: \quad \text{D Weg}_{current} \text{ is an overlapping overlap}
\]

\[
solllage: \quad \text{the object is in the correct position for the currently requested safe route } SR_{current}
\]

\[
DWEL(v): \quad \text{the object is locked as an overlap element}
\]

\[
DWEL(r): \quad \text{the object is reserved as an overlap element}
\]

\[
FWEL(v): \quad \text{the object is locked as a route element}
\]

\[
FWEL(r): \quad \text{the object is reserved as a route element}
\]

\[
FLEL(v): \quad \text{the object is locked as a flank protection element}
\]

\[
FLEL(r): \quad \text{the object is reserved as a flank protection element}
\]
The ZPZ evaluation of $DWeg_{Dwegcond1}$ is said to be partly positive, if the formulas $\phi_1$ is evaluated to true. $\phi_1$ states that if an element within the overlap has already locked or reserved as a route element ($FWEL(b)$ is true), then it must be reserved or locked for $SR_{current}$ in a proper position ($solllage$ is true) and $SR_{current}$ is a non-stop route ($durchfahrt$ is true). Otherwise, if it is locked for another routes ($FWEL(b)$ is true, $durchfahrt$ is false and $solllage$ is false), then $\phi_1$ is evaluated to false, $condition1$ and $DWeg_{Dwegcond1}$ are also evaluated to false. If elements are not locked as route elements ($FWEL(b)$ is false), $\phi_1$ is true and elements can be locked or reserved as an overlap element ($DWEL(b)$ is true) which is not allowed except $DWeg_{current}$ is an overlapping overlap. This is modeled as $\phi_2$. It means if the object is locked as an overlap element for other routes, then one needs to check whether the usage of the element forms an overlapping overlap. If $DWeg_{current}$ is not an overlapping overlap ($\overlappen$ is false), then $\phi_2$ is false and $ZPZ_{Dwegcond1}$ is evaluated to negative. $\phi_2$ is also evaluated to true if the element is not locked as an overlap element ($DWEL(b)$ is false).

When $\phi_1$ and $\phi_2$ are evaluated to true, the last condition that the element needs to fulfil is $\phi_3$. Based on the description of Condition1, an element within $DWeg_{current}$ is not allowed to be reserved or locked for other routes except the mentioned situations. Flank protection has not been mentioned in these two situations. When one translates Condition1 into propositional logic directly, then an overlap element is not allowed to be locked or reserved as a flank protection element of other routes and it is specified as $\neg FLEL(b)$.

- **Condition2:** "Im Bereich ... Ziel-DWeg-Ziel darf keine Befahrbarkeitssperre gesetzt sein... Dies gilt auch für den 1. Streckenabschnitt, d.h. auch im Bahnhofsbereich wird ZP negativ, wenn im 1. Streckenabschnitt noch eine Bsp gesetzt ist." [LH-ESTW-R F1, 2.2.2, P.13]

$$condition2 \equiv \neg Bsp$$

$Bsp$: The object is blocked for driving purpose
(German: Befahrbarkeitssperre)

In this condition, it states that if an element is blocked for driving purpose ($Bsp$ is true), the ZPZ evaluation of $DWeg_{current}$ is evaluated to false. This need to be included in $ZPZ_{Dwegcond1}$.

- **Condition3:** "ZPZ-Bedingung für Gegenfahrstraße im Bereich Ziel Ziel-DWeg-Ziel Im Bereich Ziel-DWeg-Ziel darf keine Gegenfahrstraße eingestellt sein." [LH-ESTW-R F1, 2.2.4, P.13]

$$condition3 \equiv \neg (FWEL(b) \land \neg durchfahrt)$$

Condition3 states that if an element is used as a route element for another safe route ($FWEL(b)$ is true) and the traffic direction of this route is opposite
to the direction of SR\textsubscript{current} (durchfahrt is false), then the ZPZ evaluation of D Weg\textsubscript{current} are also evaluated to false.

Based on condition 1, 2 and 3, ZPZ\textsubscript{Dwegcond1} is specified in propositional logic as follows:

\[
ZPZ_{Dwegcond1} \equiv \neg \text{FWEL}(b) \rightarrow (\neg \text{D WEL}(b) \rightarrow \phi_3)) \land \neg \text{Bsp}
\]

6.3 Modeling of overlapping overlaps

In order to complete the modeling of ZPZ\textsubscript{Dwegcond1}, one needs to model the meaning of overlapping overlaps. In this example, overlapping overlaps are modeled based on their description in LH-ESTW-R as follows:

"D-Weg der Zugstraße...-D-Wege können sich gleich- oder gegenläufig übereinander, fall sie sich nicht über spitz zur Durchrutschrichtung liegende, verschlossene Weichen oder Weichen mit beweglichen Herzstückspitzen ausschließen."

[LH-ESTW-R F1, 1.7, P.10]

\[
\text{"\text{überlappen} \equiv DWEL(b) \land (solllage \lor (\neg bHSS \land \neg \text{spitz}))\}}
\]

\[
bHSS: \text{the object is equipped with a movable frog (German: mit beweglichen Herzstückspitzen)}
\]

\[
\text{spitz: SR\textsubscript{current} is a facing point movement}
\]

According to the textual description, a situation can be described as an overlapping overlap when the object within D Weg\textsubscript{current} has already been locked as an overlap element for other routes and neither of the following two cases happen:

1. the overlap element is reserved or locked in an improper position for D Weg\textsubscript{current} and D Weg\textsubscript{current} is a facing point movement. Or,
2. the locked or reserved overlap element is equipped with a movable frog and is in an improper position for the overlap of SR\textsubscript{current}.

This can be modeled as follows:

\[
\text{"\text{überlappen} \equiv DWEL(b) \land (\neg ((\text{spitz} \land \neg \text{solllage}) \lor (bHSS \land \neg \text{solllage})))}}
\]

\[
\equiv DWEL(b) \land (\neg (\text{spitz} \lor bHSS) \land \neg \text{solllage})
\]

\[
\equiv DWEL(b) \land (\text{solllage} \lor (\neg bHSS \land \neg \text{spitz}))
\]

The statement \text{"\text{überlappen} can be read as if the point within D Weg\textsubscript{current} has already been locked as an overlap element for other routes (DWEL(b) is true), then the point must be either in the correct position for D Weg\textsubscript{current} (solllage is true), or if it is not in the correct position (solllage is false), then one of two situations are not allowed: 1) the object is equipped with a movable frog or 2) the requested route is a facing point movement. (see figure 6.1).}
6.4 Modeling of object conditions

In LH-ESTW-R, there are other two conditions that are related to the evaluation of $ZPZ_{DWegcon2}$. These conditions and their formulas are as follows:

- **Condition 4**: "ZPZ - Bedingungen für Weiche ohne bewegliche Herzstückspitzen:
  (a) Die Weiche darf nicht in der Nicht-Solllage gegen Umstellen $Usp$ sein.
  (b) Die Weiche darf nicht in der Nicht-Solllage beansprucht sein. [LH-ESTW-R F1, 2.4.1, P.16]

- **Condition 5**: ZPZ - Bedingungen für Weiche mit beweglicher Herzstückspitze:
  (a) Die Weiche darf nicht in der Nicht-Solllage gegen Umstellen $Usp$ sein. (b) Die Weiche darf nicht in der Nicht-Solllage beansprucht sein. [LH-ESTW-R F1, 2.4.2, P.16]

\[
\begin{align*}
\text{condition 4} & \equiv \neg bHSS \rightarrow \neg (\neg \text{solllage} \land Usp) \land \neg (\neg \text{solllage} \land \text{EL}(b)) \\
& \equiv \neg bHSS \rightarrow (\neg \text{solllage} \lor \neg Usp) \land (\text{solllage} \lor \neg \text{EL}(b)) \\
& \equiv \neg bHSS \rightarrow (\text{solllage} \lor (\neg Usp \land \neg \text{EL}(b))) \\
\text{condition 5} & \equiv bHSS \rightarrow \neg (\neg \text{solllage} \land Usp) \land \neg (\neg \text{solllage} \land \text{EL}(b)) \\
& \equiv bHSS \rightarrow (\text{solllage} \lor \neg Usp) \land (\text{solllage} \lor \neg \text{EL}(b)) \\
& \equiv bHSS \rightarrow (\text{solllage} \lor (\neg Usp \land \neg \text{EL}(b))) \\
\text{EL}(b) & \equiv DWEL(b) \lor FWEL(b) \lor FLEL(b) \\
\end{align*}
\]

$EL(b)$: the point is subject to a route (German: beansprucht)

$Usp$: the object is blocked for adjustment (German: umstellen gesperrt)

Condition 4 can be read as follows: if the point within $DWeg_{current}$ is equipped with a movable frog ($\neg bHSS$ is false), then condition 4 will be evaluated to true because this constraint is only relevant to points without a movable frog. Otherwise, if the point is equipped without a movable frog ($bHSS$ is false), then $(\neg \text{solllage} \lor (\neg Usp \land \neg \text{EL}(b)))$ must be evaluated to true. In this case, if the point is set in a correct position for the requested route (solllage is true), then
the ZPZ evaluation for this element within DWeg\textsubscript{current} (condition \textnumero 4 is true) is positive. If the point is not set in a correct position (solllage is false), the point must not blocked (¬Usp is true) and locked for other routes as an overlap, a route or flank protection element (¬EL(b) is true). If condition \textnumero 4 needs to implement the concept of overlapping overlaps. condition \textnumero 4 should be a logical consequence of the definition of overlapping overlaps. If DWeg\textsubscript{current} is an overlapping overlap (überlappen is true), then condition \textnumero 4 should also be evaluated to true. As mentioned in 5.1, this sequent can be written as follows:

\[ \text{überlappen} \models ¬bHSS \rightarrow (\text{solllage} \lor (¬Usp \land ¬EL(b))) \]

The premises are the definition of overlapping overlaps. The consequence is the condition of points without movable frogs that are located within DWeg\textsubscript{current}. Whenever the premises are true, then the consequence must also be true. This can be checked by comparing the truth tables of the premise and consequence.

Table 6.1 and 6.2 are the truth tables of the premise and consequence. If there is an entry in table 6.1 is true \textit{T}, then the corresponding combination of atomic propositions must also be evaluated to true in table 6.2. There is one entry of the tables where the evaluations are not consistent. They are line 12 (Table 6.1) and line 6,7 (Table 6.2). In line 12 of truth table 6.1, SR\textsubscript{current} is a trailing point movement (spitz is false) to a point without a movable frog (bHSS is false). This point is set in an improper position (solllage is false) and locked as an overlap element for other routes (DWEL(b) is true), DWeg\textsubscript{current} is an overlapping overlap (überlappen is true).

<table>
<thead>
<tr>
<th>\textit{bHSS}</th>
<th>\textit{DWEL(b)}</th>
<th>\textit{solllage}</th>
<th>\textit{spitz}</th>
<th>(\textit{solllage} \lor (¬bHSS \land ¬spitz)) \land \textit{DWEL(b)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>6</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>8</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>9</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>10</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>11</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>12</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>13</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>14</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>16</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Table 6.1: Valuations of überlappen D-Weg
If overlapping overlaps are allowed in LH-ESTW-R, then the ZPZ evaluation of DWeg\textsubscript{current} is positive (given that φ₁ and φ₃ are true). However, in line 6 and 7 of truth table 6.2, this situation is not allowed and the ZPZ evaluation of the point within DWeg\textsubscript{current} is negative in both cases. According to knowledge of domain expert, the ZPZ evaluation of the situation that is described by line 6 must remain negative because the element is blocked in an improper position for DWeg\textsubscript{current} (Usp is false and solllage is false). However, the ZPZ evaluation of the situation that is described by line 7 can be true if overlapping overlaps is allowed. It shows that the specifications are inconsistent in LH-ESTW-R, the variable spitz should be introduced in condition4. The correct specification of the Condition4 is condition6 and condition6 is as follows:

\[
\text{condition6} \equiv \neg bHSS \rightarrow \neg (\neg \text{solllage} \wedge Usp) \wedge (\text{spitz} \rightarrow \neg (\neg \text{solllage} \wedge \neg \text{EL}(b))) \\
\equiv \neg bHSS \rightarrow (\text{solllage} \vee \neg Usp) \wedge (\neg \text{spitz} \vee (\text{solllage} \vee \neg \text{EL}(b))) \\
\equiv \neg bHSS \rightarrow (\text{solllage} \vee (\neg Usp \wedge (\neg \text{spitz} \vee \neg \text{EL}(b))))
\]

This formula condition6 can be read as if a point is equipped without a movable frog (\neg bHSS is true) and ZPZ\textsubscript{Dweg} of this element is evaluated to positive in two situations. These situations are as follows:

1. when the point is set in a proper position for DWeg\textsubscript{current} (solllage is true).
2. when the point is set in an improper position (solllage is false), it cannot be blocked (\neg Usp is false) and, SR\textsubscript{current} is a trailing point movement.
(spitz is false) or the point is not reserved or locked for another route as an overlap, a route or flank protection element (EL(b) is false).

Truth tables 6.1 and 6.3 show that \textit{condition 6} is a logical consequence of \textit{überlappen}. When the entry of the truth table 6.1 is evaluated to true, then the corresponding combination of atomic proposition is also evaluated to true in truth table 6.3.

<table>
<thead>
<tr>
<th></th>
<th>bHSS</th>
<th>DW(b)</th>
<th>Usp</th>
<th>solllage</th>
<th>spitz</th>
<th>¬bHSS → (solllage ∨ (¬Usp ∧ (¬spitz ∨ ¬EL(b))))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>7</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>8</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>9</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>10</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>11</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>12</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>13</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>14</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>16</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>17</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>18</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>19</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>20</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>21</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>22</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>23</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>24</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>25</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>26</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>27</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>28</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>29</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>30</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>31</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>32</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Table 6.3: Valuations of \(¬bHSS → (solllage ∨ (¬Usp ∧ (¬spitz ∨ ¬EL(b))))\)
<table>
<thead>
<tr>
<th>C1</th>
<th>Set in proper position (solllage)</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>C2</td>
<td>Blocked (Usp)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>C3</td>
<td>Reserved/locked as FL/DW/FWEL (EL(b))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>Equipped with movable frog (bHSS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>Dweg&lt;sub&gt;current&lt;/sub&gt; is facing point movement (spitz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>ZPZ&lt;sub&gt;Dweg&lt;/sub&gt; is positive(ZPZ&lt;sub&gt;Dweg&lt;/sub&gt;)</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Table 6.4: Decision table of solllage∨(¬Usp∧¬EL(b)∨¬bHSS∧¬spitz))

6.5 Modeling of ZPZ condition for OOLH objects

Based on condition 5 and condition 6, the ZPZ constraint that is used to evaluate ZPZ<sub>Dweg</sub> for objects within Dweg<sub>current</sub> can be modeled based on the formula ZPZ<sub>Dwegcond2</sub> as follows:

\[
ZPZ_{Dweg} \equiv ZPZ_{Dwegcond2} \\
\equiv \text{condition 5} \land \text{condition 6} \\
\equiv (¬bHSS \rightarrow (solllage \lor (¬Usp \land (¬spitz \land ¬EL(b)))) \\
\land (bHSS \rightarrow (solllage \lor (¬Usp \land ¬EL(b)))) \\
\equiv (bHSS \lor solllage \lor (¬Usp \land (¬spitz \lor ¬EL(b)))) \\
\land (¬bHSS \lor solllage \lor (¬Usp \land ¬EL(b)))) \\
\equiv solllage \lor ((bHSS \lor (¬Usp \land (¬spitz \lor ¬EL(b)))) \\
\land (¬bHSS \lor (¬Usp \land ¬EL(b)))) \\
\equiv solllage \lor (¬Usp \land ¬EL(b)) \lor ((bHSS \lor (¬Usp \land ¬spitz)) \\
\land (¬bHSS) \\
\equiv solllage \lor (¬Usp \land ¬EL(b)) \lor (bHSS \land ¬bHSS) \\
\lor (¬bHSS \land ¬Usp \land ¬spitz) \\
\equiv solllage \lor (¬Usp \land (¬EL(b) \lor (¬bHSS \land ¬spitz)))
\]

ZPZ<sub>Dweg</sub> can be represented as a decision table (see table 6.4) [dt06]. The first column of the decision table is composed of two sections: the possible conditions of the system and the actions that would be triggered based on the conditions. In table 6.4, row 1 - 5 of the first column are the conditions of a point and row 6 of the first column is the action. In this table, the action is the evaluation of ZPZ<sub>Dweg</sub> of the point. The other columns show the possible alternatives of the conditions and the corresponding action. Y and N indicate Yes and No, respectively. In action entries, Y indicates that the action would be triggered based on the combination of conditions. For example, in the second column, if the point is set in a proper position, then ZPZ<sub>Dweg</sub> of the point is positive. The third column indicates that the point is set in an improper position for the Dweg<sub>current</sub> and it is blocked, then ZPZ<sub>Dweg</sub> of the point is negative.

There are three problems with this direct translation from conditions written in a natural language to a formula ZPZ<sub>Dweg</sub>. The problems are as follows:
1. \( ZPZ_{DWeg} \) states if the point is set in a proper position for the
\( DWeg_{\text{current}} \) (\text{solllage} is true), then \( ZPZ_{DWeg} \) of the point is positive
independent of the reservation or locking of the point, even the point is
locked or reserved as a route element of another route (\( FWEL(b) \) is true).

2. when the non-blocked point is not in a proper position (\( \text{solllage} \) is false and
\( Usp \) is false), it is not equipped with a movable frog (\( bHSS \) is false)
and the \( SR_{\text{current}} \) is coming from the trailing point (\( \text{spitz} \) is false), then
\( ZPZ_{DWeg} \) of the point is also positive independent of the reservation or
locking of the point, even it is locked or reserved as a route element.
However, based on \( ZPZ_{DWeg_{\text{cond1}}} \), if an element within \( DWeg_{\text{current}} \) is
locked or reserved as a route element and this reservation is not made for
the non-stop \( SR_{\text{current}} \) and is not set in a proper position for \( DWeg_{\text{current}} \)
(\( \text{durchfahrt} \) is false or \( \text{solllage} \) is false), then \( ZPZ_{DWeg_{\text{cond1}}} \) is false and
the ZPZ evaluation of \( DWeg_{\text{current}} \) is negative. This is the basic constraint
that is applied to all the elements within \( DWeg_{\text{current}} \).

3. The third problem is when the element is blocked for driving purpose, the
ZPZ evaluation of \( DWeg_{\text{current}} \) must be negative. This condition has not
been considered in \( ZPZ_{DWeg} \).

In OOLH, constraints for evaluation of \( ZPZ_{DWeg} \) that the infrastructure
elements within \( DWeg_{\text{current}} \) need to satisfy are captured in the objects. These
constraints from \( ZPZ_{DWeg_{\text{cond1}}} \) must be included in \( ZPZ_{DWeg} \) as follows:

\[
ZPZ_{DWeg} \equiv (\text{solllage} \land (FWEL(b) \rightarrow \text{durchfahrt})) \lor (\neg FWEL(b) \\
\land \neg Usp \land (\neg(FLEL(b) \lor DWEL(b)) \lor (\neg bHSS \land \neg \text{spitz})))
\land \neg Bsp
\equiv (FWEL(b) \rightarrow (\text{durchfahrt} \land \text{solllage})) \land (Usp \rightarrow \text{solllage})
\land (\neg(FLEL(b) \lor DWEL(b)) \lor \text{solllage} \lor (\neg bHSS \land \neg \text{spitz}))
\land \neg Bsp
\]

According to this \( ZPZ_{DWeg_{\text{cond2}}} \), \( ZPZ_{DWeg} \) is positive if the following three
conditions are evaluated to true:

1. first condition is true if the element is locked or reserved as a route element,
then this reservation must be made for the non-stop \( SR_{\text{current}} \) and it is
reserved or locked in a proper position for \( DWeg_{\text{current}} \) (\( FWEL(b) \) is true,
\( \text{durchfahrt} \) is true and \( \text{solllage} \) is true). Or the element is not reserved or
locked for other routes (\( FWEL(b) \) is false).

2. second condition is true if the element is not blocked (\( Usp \) is false) or,
if it is blocked (\( Usp \) is true), then it is blocked in a proper position for
\( DWeg_{\text{current}} \) (\( \text{solllage} \) is true).

3. third condition is evaluated to true if the object is not reserved or locked as
an overlap or flank protection element (\( FLEL(b) \) is false and \( DWEL(b) \)
is false) or if this element is either reserved or locked as an overlap or
flank protection element \( (FLEL_\text{true} \text{ or } DWEL_\text{true}) \), then one must check whether this point is equipped with a movable frog and the traffic movement of \( SR_{\text{current}} \) \( (spitz) \). If this point is equipped without a movable frog \( (\neg bHSS \text{ is true}) \) and the traffic movement is a trailing point movement \( (spitz \text{ is false}) \), the third condition is evaluated to true (see figure 6.2). Table 6.5 is the decision table of \( ZPZ_{\text{Dwegcond2}} \).

![Figure 6.2: Point without movable frog used as FLEL and DWEL](image-url)

Table 6.5: Decision table of \( ZPZ_{\text{Dweg}} \)

<table>
<thead>
<tr>
<th>c1</th>
<th>c2</th>
<th>c3</th>
<th>c4</th>
<th>c5</th>
<th>c6</th>
<th>c7</th>
<th>c8</th>
<th>c9</th>
<th>c10</th>
<th>c11</th>
<th>c12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocked for driving purpose ( B_{sp} )</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Reserved/locked as FWEL ( (FWEL_\text{b}) )</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Lock for non-stop route ( (durchfahr) )</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Blocked ( (\neg spitz) )</td>
<td>-</td>
<td>-</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Reserved/locked as DW-EL ( (DWEL_\text{b}) )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Equipped with movable frog ( (bHSS) )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Is facing point movement ( (spitz) )</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### 6.6 Discussion

In this example, the advantages of specifying requirements in proposition logic and the idea of transforming LH-ESTW-R to OOLH have been demonstrate. As mentioned before, there are two type of requirements in LH-ESTW-R, the first type is basic statements which are related to safe routes and consist of exceptional situations like overlapping overlaps. The second type is rules that are applicable to infrastructure elements. These requirements should be consistency. When they are translated into propositional formula, their consistence and completeness can be verified by comparing the truth tables of the formulas consecutively as it has been shown in this example. In OOLH, these two conditions are checked and combined into a consistent condition of the object \( ZPZ_{\text{Dweg}} \). \( ZPZ_{\text{Dweg}} \) can be evaluated by applying this formula to all the objects within the overlap. A consistent formula can be translated into a truth table or decision table, the requirement can be analyzed in a different abstract level based on the preferences of railway engineers and computer scientists. Furthermore, during the design phase of a interlocking system development, one can verify the correctness of
the ZPZ design based on this formula. This means, if the object reaches the state ZPZ positive, the formula must be satisfied. Similarly, if the decision table has been used to describe the interlocking logic of a design, the conformity of the decision table to the OOLH can be verified by translating it to propositional formulas. The logical equivalence can be checked among the formulas. As mentioned above, a OOLH tool is proposed to be developed to achieve all these tasks automatically.
Chapter 7

Conclusion

In this report, the advantages of applying formal methods to model system requirements has been discussed. With the framework of formal methods, mathematical concepts and logics are applied to specify system requirements. Chapter 5 has shown that the syntax and semantics of mathematical specification languages are formally defined. The meaning of each logical statement or formula is clear and unambiguity does not exist. Equivalent concepts and formulas can be deduced based on the well-defined syntax and semantics of the language.

System specifications that are written in natural languages have the opposite characteristics. Most of the concepts and terms in natural languages are not well-defined. Statements and sentences become unclear and ambiguous. This leads to misunderstandings easily. It is very difficult for the system development team to fully understand and transform the corresponding specification into computer programs. Furthermore, the correctness of specifications is difficult to be verified. This means, if the system is developed based on this specification and conformity of the system against this specification is verified, the system might not have the expected behavior or deliver the intended functionalities. One of the solutions in handling these problems is applying the well-defined mathematical concepts and logics to model and specify system requirements as a set of logical formulas. As a result, ambiguity of specifications can be reduced and the consistency of specifications can be verified. The set of logical formulas can be used as checking conditions to verify design of systems during the software development process.

The focus of this work is the investigation of a mathematical based method in specifying LH-ESTW-R which is the system requirements specification of regional computerized RIS and is written in German natural language. Based on the characteristics of LH-ESTW-R, an object oriented approach in modeling this specification is proposed. Domain objects are defined and system requirements are captured in objects. The specified LH-ESTW-R is called OOLH. The mathematical concepts that are applied in this approach are propositional logic. At this stage of the work, the main focus is put on applying propositional logic to specify static conditions of LH-ESTW-R. A tool would be developed in order to support analyzing OOLH with truth tables and decision tables. Applying temporal logic to specify time-oriented conditions will be further investigated.

By applying an OO approach to specify LH-ESTW-R, its correctness and understandability can be analyzed and improved. The efficiency of developing
computerized RIS would also be increased. The first step in this work is to apply this approach to model and specify one of the phases of RIS, ZPZ. A simple example has been developed in chapter 6.
Bibliography


Appendix A

Railway Terms

This appendix lists railways terms that are used in this report in German (defined attributes of objects) and the corresponding possible English translation.

- Anzeige, Indicator
- Beansprucht \((b)\), subject to
- Befahrbarkeitssperre \((Bsp)\), blocking for driving purpose
- Beweglichen Herzstückspitzen \((bHSS)\), movable frog
- Durchrutschweg/DWegelement \((DWEL)\), overlap element
- D-Weg-Ziel, goal of the overlap
- Fahrstraße, safe route
- Fahrweg, route
- Flankenschutzelement \((FLEL)\), flank protection element
- Grundsatz, basic condition
- Gleis, track
- Infrastrukturelement, infrastructure element
- Reservieren \((r)\), reserved
- Stammgleis, straight track
- Stellwerk, railway interlocking system
- Spitz befahren \((spitz)\), facing point movement
- Transportschutzweichen, protection transportation points
- Umstellsperrre \((Usp)\), blocked for an adjustment
- Verschließen \((l)\), lock
• Weiche, point
• Ziel, goal
• Zweigleis, diverging track