Received 11 July 2024, accepted 19 July 2024, date of publication 29 July 2024, date of current version 12 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3435468

RESEARCH ARTICLE

Formal Verification Method of CTCS-2 Level Train Control Engineering Data Based on the Reduced Ordered Binary Decision Diagram

HAO ZHANG[1], **QING XU**[1,2], **AND KE YE**[2,3]
[1]School of Automation and Electrical Engineering, Lanzhou Jiaotong University, Lanzhou 730070, China
[2]Beijing National Railway Research and Design Institute of Signal and Communication Group Company Ltd., Beijing 100070, China
[3]School of Mechanical and Energy Engineering, Beijing University of Technology, Beijing 100124, China

Corresponding author: Hao Zhang (2335596542@qq.com)

This work was supported in part by Beijing National Railway Research and Design Institute of Signal and Communication Group Company Ltd.

ABSTRACT The precise generation of train control curves for the on-board Automatic Train Protection (ATP) of the Chinese Train Control System Level 2 (CTCS-2) relies significantly on accurate train control engineering data, which serves as a critical element in ensuring the safe operation of trains. However, the traditional verification methods of train control engineering data depend on manual validation, which lacks timeliness and completeness and makes it easy to overlook potential errors. On the other hand, traditional verification rules are derived from railway technical specifications and standards, expressed in textual language that is often ambiguous, which leads to insufficient completeness in data verification. To address these challenges, this paper proposes a formal verification method for train control engineering data based on a Reduced Ordered Binary Decision Diagram (ROBDD). First, the attribute constraints of the train control data and the implicit constraint relations between different data objects are mined using existing railway technical specifications and expert knowledge of railway signals. These constraints are then converted into Boolean function models. Second, we formulate the ROBDD generation algorithm and the evaluation decision algorithm of the Boolean function using the equivalent canonical data structure ROBDD of the Boolean function. Finally, based on the train control engineering data from actual passenger-dedicated lines, the automatic verification method is developed by constructing four types of "mutation" data, including mutate, swap, add, and remove. The test cases indicate that our proposed formal verification method is feasible and capable of achieving high efficiency and completeness in the verification of CTCS-2 level train control engineering data.

INDEX TERMS Formal verification, train control engineering data, rule extraction, ROBDD, mutation testing, rule rating, rule optimization.

I. INTRODUCTION

The CTCS-2 level train control system is a typical data-driven system consisting of three components: data, software, and hardware [1]. It effectively realizes the separation of generic control logic and train control configuration data [2]. Configuration data serves as foundational data for train movement authorization, balise telegram compilation, track circuit coding, and other functions. Consequently, even if the predefined control logic is accurate, errors in the train control configuration data can result in abnormal braking, overspeed, tailgating, and derailment and overturning of the train during operation [3]. A poignant illustration of the impact of configuration data errors is exemplified by the 4.28 Jiaoji Railway major accident, where the omission of temporary speed restriction data played a decisive role [4].

However, the traditional validation method of CTCS-2 level train control engineering data relies mainly on manual

The associate editor coordinating the review of this manuscript and approving it for publication was Ton Duc Do.

validation, which has the following problems. First, due to the diversity of data types and the large number of complex descriptive objects involved, manual validation is extremely inefficient. Moreover, this manual validation process often focuses on identifying fundamental errors, such as data format, unit of measurement, and data overrun. Second, the transformation of the station yard on existing lines of the universal speed railway, coupled with reconnaissance survey error correction and changes in line environment, necessitate modifications to train control engineering data. Frequent alterations in the entire line production cycle impose tight timelines and a substantial workload on the validation of train control engineering data. This leads to an inability to conduct comprehensive testing, making it prone to overlook potential errors in the train control engineering data [5]. Therefore, it is imperative to explore an efficient and complete automated verification method for train control engineering data.

The completeness of automated verification for train control engineering data primarily depends on the complete extraction of association rules among train control data and the construction of the general mathematical models of those rules. However, the conventional verification rules of train control data are typically derived from railway technical specifications and standards, employing textual language as a guiding principle. This language introduces ambiguity, and fails to explicitly delineate the association relationships among train control data. Consequently, verification tools developed based on these rules can only identify data type, format and other errors, lacking the capability to discern implicit relationships between train control data. For example, there is insufficient verification of mutation types, such as spurious data coincidences. To overcome this limitation, this study integrates the expertise of railway signaling professionals with existing railway technical specifications and the requirements of train control engineering data tables to delve deeper into the implicit association rules within train control engineering data. Subsequently, formal Boolean function models are adopted to represent the verification rules, thereby mitigating the ambiguity inherent in natural languages. On this basis, the equivalent canonical data structure of the Boolean function is utilized to construct the algorithm, and then the automated verification tool is developed to achieve the efficient and complete verification of train control engineering data.

The primary contributions of this study are threefold. Firstly, we are rooted in the integration of existing railway technical specifications, the requirements of train control engineering data tables, and the expertise of railway signaling professionals. Through this collaborative approach, a comprehensive exploration of the attribute constraint relationships within the train control data itself and the implicit association rules between different data objects was conducted. This exploration lays the rule foundation necess-ary for achieving efficient and complete verification of train control engineering data. Secondly, we transformed the extracted verification rules of train control data into Boolean function models, which not only enhance the scalability of the models but also eliminate the natural language ambiguity that usually takes railway technical specifications as the guiding principle of the verification rules. Furthermore, the Reduced Ordered Binary Decision Diagram (ROBDD), an equivalent canonical data structure of the Boolean function model, is used to verify the satisfiability of the train control data. Compared with the traditional B method, it drastically reduces the state explosion faced by verifying big data sets of train control. Finally, we carried out verification sensitivity analysis by constructing four types of "mutated" data samples: mutate, swap, add and remove. We then mapped the sensitivity into user-friendly rule ratings to identify missing verification rules, thus achieving the purpose of bottom-up closed-loop feedback to complement verification rules. Different from conventional formal verification, we achieved the combination of formal model checking and statistical verification of "mutation testing" and introduced a user-friendly rule rating system to assist in the construction of verification rules.

The remainder of this paper is organized as follows. Section II conducts a literature review related to the verification of train control data. Section III introduces the train control engineering data and then extracts the verification rules implicit in this data. Section IV proposes an ROBDD-based satisfiability verification method for train control enginee-ring data. The Boolean function models of verification rules are constructed and converted into the canonical data structure ROBDD. A case study is presented in Section V. Section VI provides the discussion. Section VII presents the conclusions of this study.

II. LITERATURE REVIEW

In recent years, both academic and industrial circles have increasingly recognized the paramount importance of ensuring accurate train control data in contemporary train control systems. Many efforts have been directed towards validating the correctness of data. Nash et al. proposed a comprehensive XML-based RailML railway data interaction format [6] to address the problem of data interaction between different applications. An illustrative study involves security validation using the RailML database for the interlocking system at Santpoort Nord station in the Netherlands [7]. Based on RailML, the International Union of Railways (UIC) proposed RailTopoModel [8], a tool for logically defining the data object model related to railway infrastruc-ture, which enables the construction of topological models of railway networks with different levels of detail [9], [10]. In literature [11], RailTopoModel was used to construct a high-speed railway turnout data model. Based on the RailTopoModel standard, Stefan Bischof et al. developed a Rail Topology Ontology model to enable the integration of disconnected data from different subsystems of the railway in a knowledge graph [12]. In addition, Menéndez et al. developed a Railway Network Analyzer (RNA) using Python, which automates the generation of rail yard signals containing interlocking tables based on the railway layout stored in RailML [13]. However, these studies focused on railway simulation and planning data, with insufficient validation of the correctness of the train-control data, not to mention the lack of exploring the attribute constraint relationships between different train control data objects.

Currently, the verification of train control data relies primarily on formal verification tools based on B-method. This involves converting the security requirements, initially described in natural language, into security attributes defined by a formal language. Subsequently, model checking or theorem proving is employed to verify whether the train control data satisfy the safety properties [14], [15]. For example, Badeau et al. used the B-method to create a formal model of metro CBTC system configuration data and their properties, and determined whether the data satisfy specific properties by theorem proving [16]. Hansen et al. verified the railway topological relationships using the B-method and constructed a ProB model checker based on the B-method to verify the data [17]. Lecomte et al. demonstrated how to validate large datasets against safety attributes in the railway domain using the ProB model checker, and verified close to 5000 pieces of data from the L12 line in Mexico [18]. Falampim et al. formally verified the Zone Controller (ZC) data of the CBTC system using the ProB tool, which is more efficient than Atelier B [19]. Hadad et al. built an AADL model for the movement authority (MA) of the Chinese Train Control System (CTCS) and mapped it to Event-B. The RODIN platform's ProB tools are then used to formalize the verification of the safety and liveness properties of safety-critical systems [20]. In addition, on the basis of the B-method, the OVADO validation tool has been formed. OVADO is a dual-channel data validation mechanism designed in the pre-development stage of the data, using the B-assertion mechanism to check the reasonableness of the data specification on the one hand, and carrying out detailed mathematical verification of the data on the other hand at the same time [21]. The literature [22] utilized the OVADO tool to transform the data validation of railway systems into the generation of configuration data. The theoretical foundation of the B-method is first-order logic and set theory, which transform strongly constrained relations between data into set-mapping relations of formalized predicates. However, it has weaker capabilities in data representation and data processing, which can easily lead to an explosion of the state space in the verification process for big data sets. Secondly, formal models based on the B-method are too resistant to changes necessary to accommodate missing requirements; and the logical notation of the B method is relatively poor in legibility, which is not convenient for communicating and convincing stakeholders (railway signal engineers and formal method experts) in the adequacy of the rule elicitation exercise.

Furthermore, in terms of verifying railway data using formal logic and set theory, other formal verification methods have been proposed in addition to the mainstream B-method. For example, Haxthausen et al. introduced the utilization of the Linear Temporal Logic (LTL) language to articulate graphical interlocking routing data and data verification rules. They employed the Bounded Model Checking (BMC) method to verify whether the data aligned with the specified rules [23] and developed it as a static data verification tool. Along this line, Peleska et al. used a subset of LTL to describe the data and configuration rules of distributed geographical interlocking systems (IXLs), and converted the LTL model checking into a Computation Tree Logic (CTL) model checking problem through an over-approximation method. Moreover, the DVL-Checker tool was developed to check whether violations of the configuration rules exist. CTL is more efficient than LTL verification due to the existence of an efficient algorithm for global model checking, which was successfully applied to the data validation of geographical IXLs by the Siemens Mobile Company [24]. Although Bounded Model Checking and Computation Tree Logic can effectively mitigate the state explosion faced by model checking, extracting complete verification rules and formally describing them often takes years for experts in the field of railway signaling. Moreover, there is a lack of a method to quantitatively prove that the set of verification rules may have omissions, in order to assist in the construction of more complete verification rules. Luteberget et al. utilized a variant of the data query language, Datalog, to construct a verification rule model for interlocking system data and used RailML as the data interaction format. On this basis, a tool that can be integrated into CAD software for verifying railway yard data was developed [25]. Iliasov et al. developed SafeCap, a modern toolkit for railway network modeling, simulation, and formal verification. They applied SafeCap to conduct formal analysis and automated scalable safety verification of Solid State Interlocking (SSI) programs [26], [27]. However, the aforementioned research primarily focuses on graphical interlocking route data, formally describing the interlocking constraint relationships among turnouts, signals and track sections of interlocking routes in railway yards. This study centers on the formal verification of data from tables such as the balise position table, main line signal data table, line gradient table, line velocity table, and other tables within the CTCS-2 train control engineering data.

III. EXTRACTION OF VERIFICATION RULES FOR TRAIN CONTROL ENGINEERING DATA

A. TRAIN CONTROL ENGINEERING DATA

The CTCS-2 level train control data are the core data source for generating on-board ATP target distance continuous speed control mode curve, which consists of three parts: train control basic data, train control engineering data tables and train control configuration data (including telegrams). Among them, train control basic data are the basis for compiling train control engineering data, which are divided into the basic data of the track maintenance category, the basic data of the signal and communication categories, and the basic data of the traction power supply category. The

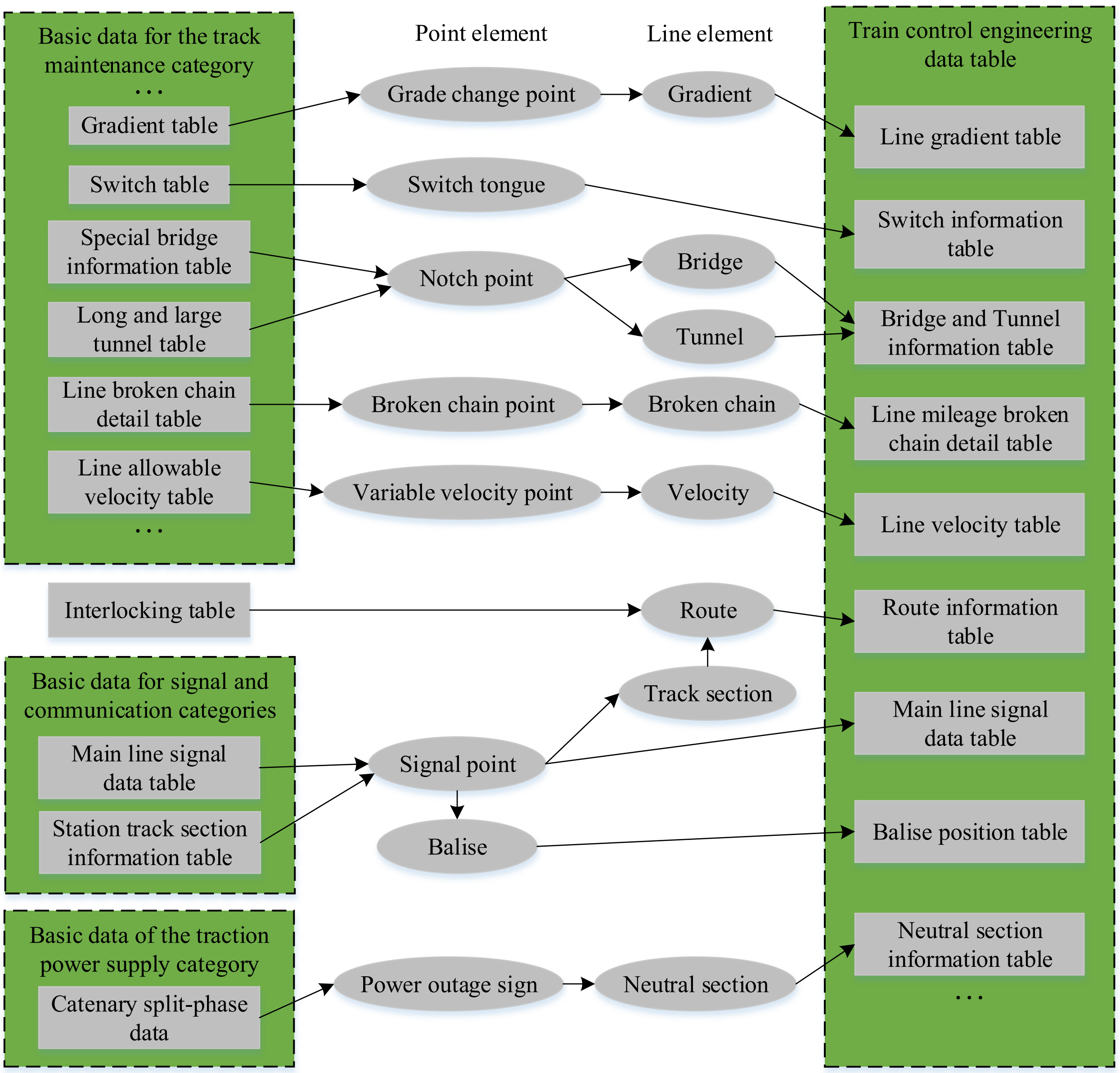

Basic data for the track maintenance category
Gradient table
Switch table
Special bridge information table
Long and large tunnel table
Line broken chain detail table
Line allowable velocity table
Interlocking table
Basic data for signal and communication categories
Main line signal data table
Station track section information table
Basic data of the traction power supply category
Catenary split-phase data
Point element
Line element
Grade change point
Gradient
Switch tongue
Notch point
Bridge
Tunnel
Broken chain point
Broken chain
Variable velocity point
Velocity
Route
Track section
Signal point
Balise
Power outage sign
Neutral section
Train control engineering data table
Line gradient table
Switch information table
Bridge and Tunnel information table
Line mileage broken chain detail table
Line velocity table
Route information table
Main line signal data table
Balise position table
Neutral section information table

FIGURE 1. **The generation source and specific composition of CTCS-2 level train control engineering data.**

basic data of track maintenance include the name of the line, the beginning and ending mileage of the main line, the allowable velocity of the line, the gradient of the line, the information about special bridges and tunnels, the mileage of switches, etc. The basic data of signal and communication include the position and type of train signals, the length and carrier frequency of the track circuit, the kilometer post of the insulated joint and the divisional point of the track circuit, and the position of the balise. The basic data of traction power supply mainly consist of the kilometer post of the power outage signboards in the neutral section. The basic data of the three types of train control are compiled by the construction personnel of the Engineering Bureau in cooperation with the signal and communication, track maintenance, and power supply operation and maintenance units of the Railway Bureau according to the CAD drawings provided by the survey and design units, such as the railway line signal layout plans, through preliminary measurements, final measurements, and reviews. On this basis, Railway Engineering Design Institute compiles train control engineering data table according to the train control basic data and interim measures for the management of train control data. The train control system integrator compiles the train control configuration data according to the train control engineering data table, which is used to finally generate the on-board ATP accurate train control curve. Figure 1 shows the source and specific composition of CTCS-2 level train control engineering data. Figure 2 shows the signal layout plan of the railway down line, where v_i and $l_{v_i} (i = 1, \cdots, 4)$ represent the value of the line restrictive velocity and length

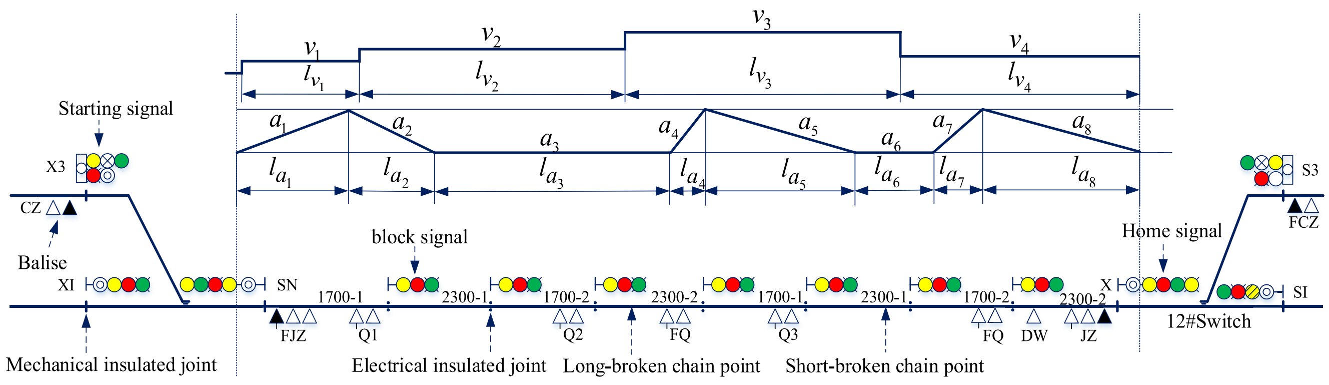


FIGURE 2. Signal layout plan of the railway down line.

of the restrictive velocity section, a_i and $l_{a_i}(i = 1, \cdots, 8)$ denote the gradient value and length of the gradient.

The CTCS-2 level train control engineering data are primarily categorized into trackside infrastructure data and line parameters, which include the line velocity table, line gradient table, line broken chain detail table, balise position table, main line signal data table, and so on. Each type of data contains corresponding data objects, and each type of data object has its own set of data attributes. Therefore, the CTCS-2 level train control engineering data tree structure can be constructed to provide a clear division of data categories for the verification of train control engineering data. The tree structure of CTCS-2 train control engineering data is shown in Figure 3.

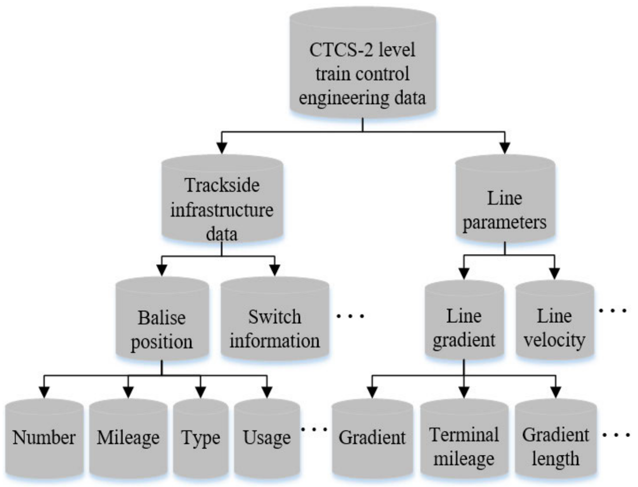


FIGURE 3. CTCS-2 level train control engineering data tree structure.

The train control engineering data are all stored in the form of Excel tables, which record the respective attribute values of different train control data objects. For example, the balise position table includes the balise name, number, type, mileage, and usage. Similarly, the line gradient table contains the gradient, gradient length, gradient end mileage, and broken chain point marker. As illustrated in Table 1, the balise position table is used as an example to display specific data.

TABLE 1. Example of balise position table.

Serial number	Balise name	Balise number	Mileage	Type	Usage
1	B7892	028-1-35-002	K789+400	Passive	Q
2	BS-1	028-1-35-004-1	K788+335	Passive	JZ
3	BS-2	028-1-35-004-2	K788+330	Active	JZ
4	BXN-1	028-1-35-006-1	K786+270	Active	FJZ
5	BXN-2	028-1-35-006-2	K786+265	Passive	FJZ
6	B7852	028-1-35-008	K785+300	Passive	Q

B. EXTRACTION OF TRAIN CONTROL ENGINEERING DATA VERIFICATION RULES

The verification rules of train control engineering data typically derive from the “The balise application principle for the CTCS-2 level train control system” and “Interim measures for the management of train control data.” Building on these foundations, the expertise of railway signal professionals is integrated to delve further into the attribute constraints of train control data and the associative relationships between distinct data objects. Through this collaborative effort, this study aims to extract implicit verification rules for train control engineering data.

Definition 1: In conjunction with the balise position table, the attributes of balise data can be represented by a 4-tuple, $B_a = (B\_{name}, B\_{number}, B\_{kilompost}, B\_{type})$, denoting the balise name, number, kilometer post, and type, respectively.

Rule 1: The categorization of balise types is defined as either active or passive, expressed as $B\_{type} = \{1, \ 2\}$.

Rule 2: According to the principle of balise setting, there exists a set of constants, denoted by S, that represent the installation distances between different types of balise groups and adjacent signals. If the distance between adjacent balises in a group is a constant $C = 5m$, the absolute value of the difference between balise group link distance B_{link} and intergroup distance $B\_{classpace}$ is 10m, which indicates that the balises installed on the line are continuous.

$$\left|B\_{kilompost_i} - S\_{kilompost_i}\right| \in S \tag{1}$$

$$\left|B\_{link_j} - B\_{classpace_j}\right| = 10$$
$$\Leftrightarrow \left|B\_{kilompost_{i+1}} - B\_{kilompost_i}\right| = C \tag{2}$$

Definition 2: Interruption in the continuity of line mileage stake marks resulting from local rerouting or segmental surveying is termed a broken chain. If mileage stake marks overlap, it is called long chain, whereas interrupted mileage stake marks are called short chain. Combined with the line broken chain detail table, the attributes of the line broken chain data can be expressed as a 3-tuple, $D_c = (D\_{type}, D\_{kilompost}, D\_{length})$.

Rule 1: $D\_{type} = \{1, 2\}$ denotes line long chain and short chain, respectively.

Rule 2: The length of a broken chain is equal to the difference between the end mileage and starting mileage of the line broken chain [28], which can be expressed as:

$$\left|D\_{kilompost_{i+1}} - D\_{kilompost_i}\right| = D\_{length_i} \tag{3}$$

Definition 3: In conjunction with the line gradient table, the attributes of the line gradient can be defined as a 3-tuple, $G_p = (G\_{value}, G\_{length}, G\_{termileage})$, representing the line gradient, gradient length, and gradient end mileage, respectively.

Rule 1: The line gradient should not exceed the maximum allowable line gradient value. For example, if a successive route is not provided at the end of the receiving route, the descending grade gradient within the braking distance out from the home signal must not exceed the 6‰.

$$\left\{G\_{value_1}, G\_{value_2}, \cdots, G\_{value_n}\right\} \leq G\_{value_{MAX}} \tag{4}$$

Rule 2: The difference between the kilometer posts of two adjacent gradient points minus the corresponding actual section length equals 0, indicating that the two adjacent gradient points are continuous. Conversely, if the difference between the kilometer posts of two adjacent gradient points is greater than the corresponding actual section length, there is a short chain; if it is less than the actual section length, a long chain exists.

$$\left|G\_{termileage_{i+1}} - G\_{termileage_i}\right| - G\_{length_i} = \pm D_{length_i} \tag{5}$$

Definition 4: In conjunction with the line velocity table, the attributes of line velocity data can be represented as a 3-tuple, denoted as $V_s = (V\_{value}, V\_{length}, V\_{termileage})$, representing the value of the line restrictive velocity, the length of the restrictive velocity section, and the terminal mileages of the line restrictive velocity, respectively.

Rule 1: The fixed velocity limit of a line cannot exceed the maximum velocity allowed on the line, which can be defined as follows:

$$\left\{V\_{value_1}, V\_{value_2}, \cdots, V\_{value_n}\right\} \leqslant V\_{value_{MAX}} \tag{6}$$

Rule 2: The difference between the kilometer posts of two adjacent velocity points minus the corresponding actual section length is equal to zero, indicating that the two adjacent velocity points are continuous. Conversely, if the difference between the kilometer posts of two adjacent velocity points is greater than the corresponding actual section length, there is a short chain; if it is less than the actual section length, a long chain exists [28].

$$\left|V\_{termileage_{i+1}} - V\_{termileage_i}\right| - V\_{length_i} = \pm D\_{length_i} \tag{7}$$

Definition 5: In conjunction with the main line signal data table, the attributes of line signal data can be defined as a 7-tuple. $S_d = (S\_{name}, S\_{type}, S\_{kilompost}, S\_{insuljoitype}, S\_{Trackname}, S\_{Frequency}, S\_{Tracklen})$ denotes the signal point name, type, kilometer post, insulated joint type, track section name, carrier frequency, and length, respectively.

Rule 1: The signal point type $S\_{type} = \left\{1, 2, 3, 4, 5, 6, 7\right\}$ is defined to denote the home signal, starting signal, block signal, route signal, shouting signal, station exit, and no signal, respectively.

Rule 2: Define the carrier frequency type $S\_{Frequency} = \{1, 2, 3, 4, 5\}$, indicating that the track section carrier frequencies are 1700 HZ,2000 HZ,2300 HZ,2600 HZ and no carrier frequency. Among them, the carrier frequencies of the adjacent sections of the up-line and down-line are configured alternately,$S\_{Frequency_i} \neq S\_{Frequency_{i+1}}$. The selection of carrier frequencies -1 and -2 for the front and rear sections adjacent to this section is also staggered. For example, the block section carrier frequencies of the down line are alternately configured according to $1700-1,\ 2300-1,\ 1700-2,\ 2300-2,\ 1700-1$, and so on. The block section carrier frequencies of the up line are configured alternately according to $2000-1,\ 2600-1,\ 2000-2,\ 2600-2,\ 2000-1$, and so forth.

$$\left\{S\_{Frequency_{i-1}} - 1\right\} \neq \left\{S\_{Frequency_{i+1}} - 2\right\} \tag{8}$$

Rule 3: Define the insulated joint-type $S\_{insuljoitype} = \{1, 2\}$, representing the electrical and mechanical insulated joints respectively. There must be an insulated joint near the location of the signal, that is, for $\forall\ S\_{name},\ \exists\ S\_{insuljoitype}$. However, the converse is not necessarily true. For example, if the signal point type is a station exit $S\_{type} = 6$, then it corresponds to a mechanical insulated joint $S\_{insuljoitype} = 2$. Secondly, there must be an electrical insulated joint near the block signal location, $S\_{insuljoitype} = 1 \Rightarrow S\_{type} \in \{3, 7\}$.

Rule 4: The difference between the mileages of two adjacent signal points minus the length of the corresponding actual track section is equal to zero, indicating that the two adjacent signal points are continuous. In contrast, if the difference between the mileages of two adjacent signal points is greater than the corresponding actual track section length, there is a short chain; if it is less than the actual track section length, there is a long chain.

$$\left|S\_{kilompost_{i+1}} - S\_{kilompost_i}\right| - S\_{Tracklen_i} = \pm D\_{length_i} \tag{9}$$

Definition 6: The switch data attribute can be defined as a 4-tuple,$P_t = (P_{id}, P\_{kilompost}, P\_{position}, P\_{offset})$, denoting the switch number, kilometer post, position, and offset, respectively.

Rule 1: The switch status is categorized into normal and reverse positions, which can be described as $P\_{position} = \{1, 2\}$.

Rule 2: If the link offsets of both the switch main line and side-track are not zero, then the link offsets of the switch junction line must be 0. Otherwise, if the link offsets of both the switch main line and sidetrack are 0, then the link offsets of the switch junction line must not be 0.

$$P\_offset\_m \neq 0 \cap P\_offset\_s \neq 0 \leftrightarrow P\_offset\_J = 0 \tag{10}$$

IV. SATISFIABILITY VERIFICATION OF TRAIN CONTROL ENGINEERING DATA BASED ON ROBDD

We formulated Boolean function models based on the extracted logical verification rules from the train control engineering data. Subsequently, by utilizing the equivalent canonical data structure of the Boolean function, namely ROBDD [29], [30], the satisfiability determination of the Boolean function under the given input mode is transformed into the depth-first search process within the ROBDD. To achieve this, the ROBDD construction algorithm and the Boolean function evaluation and determination algorithm are designed separately to realize the automated verification of train control engineering data.

A. CONSTRUCTING BOOLEAN FUNCTION MODELS OF VERIFICATION RULES

Model 1: Constructing a Boolean function model of intra-group balise position continuity for line settings

Let the Boolean variable $a_i(i = 1, 2, \cdots, k)$ denote that the difference between balise group link distance and balise group spacing is equal to $10m$, $\left|B\_link_j - B\_classpace_j\right| = 10m$. The Boolean variable $b_i(i = 1, 2, \cdots, k)$ denotes that the spacing between adjacent balise in the group is equal to the constants $5m$, $\left|B\_kilompost_{i+1} - B\_kilompost_i\right| = 5m$. Combined with Rule 2 of the balise position table, the logical relationship between a_i and b_i can be described as:

$$f_B = (a_1 \leftrightarrow b_1) \wedge (a_2 \leftrightarrow b_2) \wedge \cdots \wedge (a_k \leftrightarrow b_k) \Leftrightarrow (a_1 + b'_1)(a'_1 + b_1) \cdots (a_k + b'_k)(a'_k + b_k) \tag{11}$$

where $\leftrightarrow$ denotes if and only if. Let $x_j(j = 1, 3, 5, \cdots, 2k-1)$ and $x_j(j = 2, 4, 6, \cdots, 2k)$ denote a_i and b_i respectively, then the corresponding Boolean function model prototype of the above equation (11) is shown in equation (12), where $n = 2k$.

$$\begin{aligned} f(x_1, x_2, \cdots, x_j, \cdots, x_n) &= (x_1 + x'_2)(x'_1 + x_2) \cdots (x_{j-1} + x'_j) \\ &\quad (x'_{j-1} + x_j) \cdots (x_{n-1} + x'_n)(x'_{n-1} + x_n) \end{aligned} \tag{12}$$

Model 2: Constructing the Boolean functions models for the continuity judgment of adjacent velocity, signal, and gradient points.

Assume that the algebraic variable $a_i(i = 1, 2, \cdots, k)$ denotes the absolute value of the difference between the kilometer posts of the adjacent velocity points, signal points and gradient points, and the algebraic variable $b_i(i = 1, 2, \cdots, k)$ denotes the length of the corresponding actual section, so that the Boolean function f_i denotes $a_i = b_i$. The algebraic variable $c_i(i = 1, 2, \cdots, k)$ denotes D_length_i. Assume that the Boolean funct-ion g_i denotes $c_i = 0$, and the Boolean function h_i denotes $c_i \neq 0$. Then f_i, g_i, h_i under the same variable order π satisfy the $if f_i$ $then$ g_i $else$ h_i,$ITE(f_i, g_i, h_i) = (f_i \wedge g_i) \vee (\neg f_i \wedge h_i)$ paradigm [31], namely:

$$\begin{aligned} f_B &= ITE(f_1, g_1, h_1) \wedge ITE(f_2, g_2, h_2) \wedge \cdots \wedge ITE(f_k, g_k, \\ &\quad h_k) \Leftrightarrow (f_1 g_1 + f'_1 h_1)(f_2 g_2 + f'_2 h_2) \cdots (f_k g_k + f'_k h_k) \end{aligned} \tag{13}$$

Let $x_j(j = 1, 4, 7, \cdots, 3k-2)$, $x_j(j = 2, 5, 8, \cdots, 3k-1)$ and $x_j(i = 3, 6, 9, \cdots, 3k)$ denote f_i, g_i and h_i respectively, then the above equation (13) corresponds to the prototype of the Boolean function model as shown in equation (14), where $n = 3k$.

$$\begin{aligned} &f(x_1, x_2, \cdots, x_j, \cdots, x_n) \\ &\quad = (x_1 x_2 + x'_1 x_3)(x_4 x_5 + x'_4 x_6) \\ &\qquad \cdots (x_{j-2} x_{j-1} + x'_{j-2} x_j) \cdots (x_{n-2} x_{n-1} + x'_{n-2} x_n) \end{aligned} \tag{14}$$

If there are broken chains or spurious data coincidence at the points of the broken chain, the Boolean logical operator "same or" can be used to further determine whether the length of the calculated broken chain is consistent with the broken chain table. Assuming that the algebraic variable $c_i(i = 1, 2, \cdots, k)$ denotes the difference between the absolute value of the kilometer post difference of adjacent velocity points, signal points, gradient points and the corresponding length of the actual section, and the algebraic variable $d_i(i = 1, 2, \cdots, k)$ denotes the corresponding length of broken chain in the broken chain detail table $\pm D\_length_i$, then the logical relationship between c_i and d_i can be described as:

$$\begin{aligned} f_B &= (c_1 \odot d_1) \wedge (c_2 \odot d_2) \wedge \cdots \wedge (c_k \odot d_k) \\ &\Leftrightarrow (c_1 d_1 + c'_1 d'_1)(c_2 d_2 + c'_2 d'_2) \cdots (c_k d_k + c'_k d'_k) \end{aligned} \tag{15}$$

Let $x_j(j = 1, 3, \cdots, 2k-1)$ and $x_j(j = 2, 4, \cdots, 2k)$ denote c_i and d_i respectively, then the corresponding Boolean function model prototype of the above equation (15) is shown in equation (16), where $n = 2k$.

$$\begin{aligned} &f(x_1, x_2, \cdots, x_j, \cdots, x_n) \\ &\quad = (x_1 x_2 + x'_1 x'_2)(x_3 x_4 + x'_3 x'_4) \\ &\qquad \cdots (x_{j-1} x_j + x'_{j-1} x'_j) \cdots (x_{n-1} x_n + x'_{n-1} x'_n) \end{aligned} \tag{16}$$

Model 3: Boolean function model of signal point data attributes.

Suppose that the Boolean variable a_1 denotes that the signal point type satisfies $S\_type = \{1, 2, 3, 4, 5, 6, 7\}$; Boolean variable b_1 denotes that carrier frequencies of the adjacent sections of the up and down lines satisfy the configuration alternately $S\_Frequency_i \neq S\_Frequency_{i+1}$; Boolean variable c_1 denotes that the carrier frequency -1, -2 selection of the front and rear sections adjacent to this section is also staggered configuration, that is, $\left\{S\_Frequency_{i-1} - 1\right\} \neq \left\{S\_Frequency_{i+1} - 2\right\}$; Boolean variable d_1 denotes that the insulated joint type satisfies $S_{\_insuljoitype} = 1 \Rightarrow S_{\_type} \in \{3, 7\}$; Boolean variable $e_i(i = 1, \cdots, n)$ denotes that the adjacent signal points of the section are continuous, then a_1 and b_1, c_1, d_1 are the necessary condition for the continuity of the adjacent signal points of the section. Then the conditional logic relationship between a_1, b_1, c_1, d_1 and $e_i(i = 1, \cdots, n)$ can be described as:

$$f_B = (a_1b_1c_1d_1) \rightarrow (e_1e_2e_3 \cdots e_n) \Leftrightarrow (a_1b_1c_1d_1)' + (e_1e_2e_3 \cdots e_n) = a_1' + b_1' + c_1' + d_1' + (e_1e_2e_3 \cdots e_n) \quad (17)$$

The above equation corresponds to the Boolean function prototype:

$$f(x_1, x_2, \cdots, x_n) = x_1' + x_2' + x_3' + x_4' + x_5 \cdots x_n \quad (18)$$

Model 4: Boolean function model of switch data attributes

Suppose the algebraic variables $a_i(i = 1, 2, \cdots, k)$ and $b_i(i = 1, 2, \cdots, k)$ denote the offsets $P_{\_offset\_M}$ and $P_{\_offset\_S}$ of the main and side lines of the switch, respectively, and the algebraic variable $c_i(i = 1, 2, \cdots, k)$ represents the offset $P_{\_offset\_J}$ of the junction lines of the switch. Assuming that the Boolean function f_i denotes $P_{\_offset\_M} \neq 0 \cap P_{\_offset\_S} \neq 0$, the Boolean function g_i denotes $P_{\_offset\_J} = 0$, and the Boolean function h_i denotes $P_{\_offset\_J} \neq 0$. Then, f_i, g_i, h_i satisfy *if f_i then g_i else h_i* under the same order of variables π, $ITE(f_i, g_i, h_i) = (f_i \wedge g_i) \vee (\neg f_i \wedge h_i)$ paradigm [31].

$$f_B = ITE(f_1, g_1, h_1) \wedge ITE(f_2, g_2, h_2) \wedge \cdots \wedge ITE(f_k, g_k, h_k) \Leftrightarrow (f_1g_1 + f_1'h_1)(f_2g_2 + f_2'h_2) \cdots (f_kg_k + f_k'h_k) \quad (19)$$

Let $x_j(j = 1, 4, \cdots, 3k-2)$, $x_j(j = 2, 5, \cdots, 3k-1)$ and $x_j(j = 3, 6, \cdots, 3k)$ denote f_i, g_i and h_i respectively, then the above equation (19) corresponds to the prototype of the Boolean function model as shown in equation (20), where $n = 3k$.

$$\begin{aligned} &f(x_1, x_2, \cdots, x_j, \cdots, x_n) \\ &= (x_1x_2 + x'_1x_3)(x_4x_5 + x'_4x_6) \\ &\quad \cdots (x_{j-2}x_{j-1} + x'_{j-2}x_j) \cdots (x_{n-2}x_{n-1} + x'_{n-2}x_n) \quad (20) \end{aligned}$$

B. EQUIVALENT CANONICAL FORM OF BOOLEAN FUNCTIONS BASED ON ROBDD

Because the satisfiability problem of the Boolean function is NP-complete, the state combination explosion problem caused by it severely restricts the size of the problem to be solved. Therefore, it is necessary to find an equivalent paradigm for the representation and operation of the Boolean function. ROBDD is obtained by adding variable order and simplification rules on the basis of binary decision diagram (BDD) [32], which reduces the time complexity of the algorithm and makes it an equivalent canonical data structure describing the Boolean function family $\#_\pi f(x_1, x_2, \cdots, x_n)$. In the given variable order $\pi : x_1 < x_2 < \cdots < x_n$ and input pattern, a path starting at the root node v and ending at the final node is obtained via depth-first search. If the value of the function at the final node is 1, it indicates that the Boolean function can be satisfied under this input mode, thus realizing the verification of the train control engineering data; the time complexity is $O(n)$. Therefore, this paper transforms the evaluation determination of Boolean function into depth-first search of ROBDD.

For the Boolean function $f(x_1, x_2, \cdots, x_n)$ from $B^n = \{0, 1\}^n$ to $B = \{0, 1\}$, OBDD is obtained through recursive construction based on Shannon's expansion theorem [33] $f(x_1, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n) = x_i \cdot f(x_1, \cdots, x_{i-1}, 1, x_{i+1}, \cdots, x_n) + x_i' \cdot f(x_1, \cdots, x_{i-1}, 0, x_{i+1}, \cdots, x_n)$ under a given variable order π. Recursive expansion is expressed as follows:

$$\begin{aligned} &f(x_1, x_2, \cdots, x_n) \\ &= x_1' \cdot f(0, x_2, \cdots, x_n) + x_1 \cdot f(1, x_2, \cdots, x_n) \\ &= x_1' \cdot \left[x'_2 \cdot f(0, 0, x_3, \cdots, x_n) + x_2 \cdot f(0, 1, x_3, \cdots, x_n)\right] \\ &\quad + x_1 \cdot \left[x'_2 \cdot f(1, 0, x_3, \cdots, x_n) + x_2 \cdot f(1, 1, x_3, \cdots, x_n)\right] \\ &= \cdots = x'_1 \cdot x'_2 \\ &\quad \cdot x'_3 \cdots x'_n \cdot f(0, 0, \cdots, 0) + x'_1 \cdot x'_2 \cdots x'_{n-1} \cdot x_n \\ &\quad \cdot f(0, 0, \cdots, 0, 1) \\ &\quad + \cdots + x_1 \cdot x_2 \cdots x_n \cdot f(1, 1, \cdots, 1) \quad (21) \end{aligned}$$

A noteworthy point is that each node u on the OBDD represents a Boolean function f_u from $\{0, 1\}^{n-k} (k \leqslant n)$ to $\{0, 1\}$ satisfying:

1) If u is a final node, then $f_u = u.val \in \{0, 1\}$.
2) If u is not final node, then $f_u = u.var \cdot f^{u.high} + (u.var)' \cdot f^{u.low} = u.var \cdot f_{u.var=1} + (u.var)' \cdot f_{u.var=0}$, where $f^{u.high}$ and $f^{u.low}$ respectively denote the Boolean functions obtained by taking values 1 and 0 of the variables $u.var$ in the Boolean function f_u, that is, the Boolean functions corresponding to the children of node u, $u.high$ and $u.low$ [34]. On the basis of OBDD, by applying simplification rules to stipulate the OBDD and pruning the decision tree, the unique reduced ordered binary decision diagram (ROBDD) denoting the Boolean function family $\#_\pi f(x_1, x_2, \cdots, x_n)$ can be obtained. The main idea of the simplification rules is to remove redundant nodes from the OBDD [34], [35], as shown in Figure 4.

Rule 1: (S-deletion rule) for node u in OBDD, if $u.low = u.high$, node u is deleted, and the parent node of node u is directly connected to the corresponding node of $u.low$.

Rule 2: (Merge rule) For nodes u and v in OBDD, if $u.var = v.var$, $u.low = v.low$, $u.high = v.high$, then node u is deleted and the parent of node u is connected directly to node v.

Proof (mathematical Induction:) Generalize the number of variables n for the Boolean function $f(x_1, x_2, \cdots, x_n)$ and set the variable order π as $x_1 < x_2 < \cdots < x_n$.

When $n = 0$, there are only two Boolean functions with constants 0 and 1; that is, there are OBDDs representing final nodes 0 and 1. It can be observed that for this OBDD,

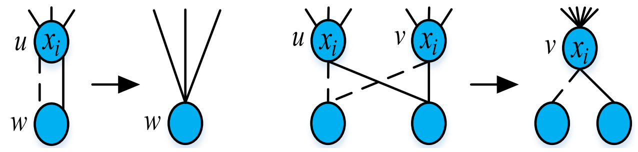


FIGURE 4. Deleting irrelevant nodes and merging child nodes.

simplification rules 1 and 2 are not satisfied. Any ROBDD containing at least one nonfinal node necessarily corresponds to a non-constant Boolean function. Therefore, the OBDDs representing the final nodes 0 and 1 must be the ROBDDs.

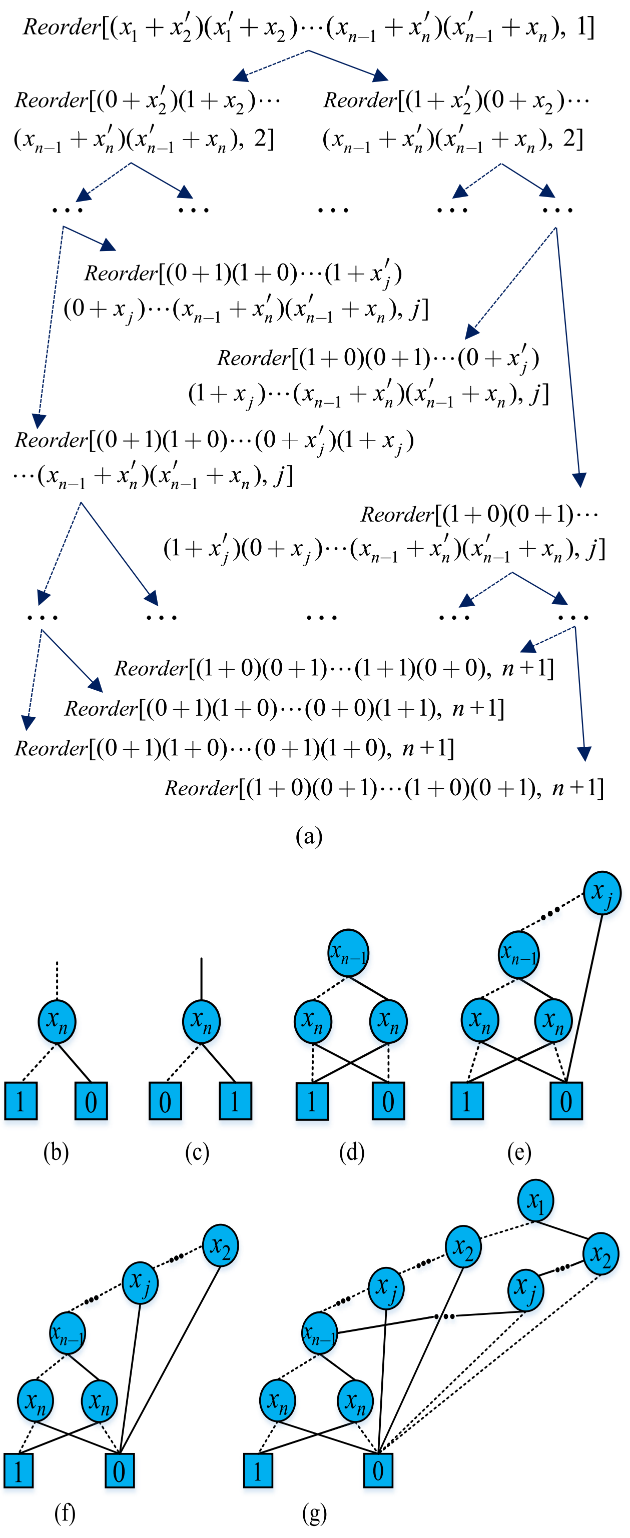
Assuming $n = k$, by continuously applying the simplification rules to the OBDD represented by the Boolean function $f(x_1, x_2, \cdots, x_k)$ of the k variables, the OBDD obtained is ROBDD when all nodes of the OBDD no longer satisfy the simplification rules.

We now prove that the simplification rules are applied continuously to the Boolean function $f(x_1, x_2, \cdots, x_{k+1})$ with $k+1$ variables when $n = k+1$. When all nodes of the OBDD do not satisfy the simplification rules, the OBDD obtained is ROBDD. For the variable x_1, the Boolean function $f(x_1, x_2, \cdots, x_{k+1})$ performs Shannon decomposition, obtaining $f(x_1, x_2, \cdots, x_{k+1}) = x_1 \cdot f(1, x_2, \cdots, x_{k+1}) + x_1' \cdot f(0, x_2, \cdots, x_{k+1}) = x_1 \cdot f_{x_1} + x_1' \cdot f_{x'_1}$. Due to the fact that both the f_{x_1} and $f_{x'_1}$ are Boolean functions of k variables, the ROBDDs representing f_{x_1} and $f_{x'_1}$ are obtained by continuously applying simplification rules to the OBDDs that they represent. Let f_{x_1} and $f_{x'_1}$ correspond to the nodes u_1 and u_0, respectively; the corresponding ROBDDs are denoted as G^{u_1} and G^{u_0}, and $f^{u_1} = f_{x_1}, f^{u_0} = f_{x'_1}$.

If $u_1 = u_0$ (namely $f^{u_1} = f^{u_0}$), and thus $f^{u_1} = f_{x_1} = f^{u_0} = f_{x'_1}$, it follows that $G^{u_1} = G^{u_0}$ is the unique ROBDD G of f. Because, according to the variable order π requirement, if the variable x_1 appears in the ROBDD G of f, then it must be the labeled variable of the root node of G. However, if $f^{u_1} = f_{x_1} = f^{u_0} = f_{x'_1}$, the root node of G has the same successor 0-branch child node as the 1-branch child node, thus satisfying simplification rule 1 and resulting in the deletion of the root node of G, x_1 does not appear in the ROBDD representation of the Boolean function f.

If $u_1 \neq u_0$, then $f^{u_1} \neq f^{u_0}$, such that the node u satisfies $f^u = x_1 \cdot f^{u_1} + x_1' \cdot f^{u_0}$, $u.var = x_1$, $u.low = u_0$, $u.high = u_1$, thus $f^u = f$. Assume that there exists another node v, which satisfies $f^v = f$. According to the requirements of the variable order π, there must be $v.var = x_1$. Furthermore, from $f^v = f$, it can be seen that $f^{v.high} = f_{x_1}$, $f^{v.low} = f_{x'_1}$. Hence, it necessarily follows that $u.var = v.var$, $u.low = v.low$, $u.high = v.high$. Subsequently, based on the merging rule, the node v can be deleted. The proof is complete.

It can be proven by the above mathematical induction method that, under the given variable order π, the S-delete rule and merge rule are continually applied to the OBDD of the Boolean function corresponding to the logic rule of any train control engineering data. When all nodes of the OBDD do not satisfy the aforementioned two simplification rules, the obtained OBDD must be a ROBDD.


$Reorder[(x_1+x_2')(x_1'+x_2)\cdots(x_{n-1}+x_n')(x_{n-1}'+x_n),\ 1]$
$Reorder[(0+x_2')(1+x_2)\cdots(x_{n-1}+x_n')(x_{n-1}'+x_n),\ 2]$
$Reorder[(1+x_2')(0+x_2)\cdots(x_{n-1}+x_n')(x_{n-1}'+x_n),\ 2]$
$Reorder[(0+1)(1+0)\cdots(1+x_j')(0+x_j)\cdots(x_{n-1}+x_n')(x_{n-1}'+x_n),\ j]$
$Reorder[(1+0)(0+1)\cdots(0+x_j')(1+x_j)\cdots(x_{n-1}+x_n')(x_{n-1}'+x_n),\ j]$
$Reorder[(0+1)(1+0)\cdots(0+x_j')(1+x_j)\cdots(x_{n-1}+x_n')(x_{n-1}'+x_n),\ j]$
$Reorder[(1+0)(0+1)\cdots(1+x_j')(0+x_j)\cdots(x_{n-1}+x_n')(x_{n-1}'+x_n),\ j]$
$Reorder[(1+0)(0+1)\cdots(1+1)(0+0),\ n+1]$
$Reorder[(0+1)(1+0)\cdots(0+0)(1+1),\ n+1]$
$Reorder[(0+1)(1+0)\cdots(0+1)(1+0),\ n+1]$
$Reorder[(1+0)(0+1)\cdots(1+0)(0+1),\ n+1]$
(a)
x_n 1 0
(b)
x_n 0 1
(c)
x_{n-1} x_n x_n 1 0
(d)
x_j x_{n-1} x_n x_n 1 0
(e)
x_2 x_j x_{n-1} x_n x_n 1 0
(f)
x_1 x_2 x_2 x_j x_j x_{n-1} x_n x_n 1 0
(g)
FIGURE 5. Recursive construction process of ROBDD for model 1.

Take the Boolean function model one $f(x_1, x_2, \cdots, x_n) = (x_1 + x_2')(x_1' + x_2)\cdots(x_{n-1} + x_n')(x_{n-1}' + x_n)$ as an example, we invoke the *Reorder* ( ) algorithm to construct its ROBDD, recursively constructing the ROBDD process as shown in Figure 5. Among them, Fig. (a) illustrates the recursive construction process of ROBDD based on Shannon's expansion theorem, Fig. (b) represents the ROBDD formed by invoking $Reorder(0, \cdots, 0, 0, x_n)$, Fig. (c) depicts the ROBDD formed by invoking $Reorder(0, \cdots, 0, 1, x_n)$, and Fig. (d) shows the ROBDD formed by invoking $Reorder(0, \cdots, 0, x_{n-1}, x_n)$. By analogy, we obtain Fig. (e) which is the ROBDD formed by invoking $Reorder(0, \cdots, 0, x_j, \cdots, x_n)$, and Fig. (f) represents the ROBDD formed through $Reorder(0, x_2, \cdots, x_j, \cdots, x_n)$. Ultimately, we arrive at Fig. (g), which is the final ROBDD representing the Boolean function $f(x_1, x_2, \cdots, x_j, \cdots, x_n)$.

C. ALGORITHM DESIGN

The time complexity of operations based on ROBDD depends on the size of the ROBDD. According to Shannon's expansion theorem, the size of ROBDD heavily depends on the variable order π, and the f_u obtained from the decomposition differs according to different variable orders. The choice of variable order determines whether the number of non-final nodes of ROBDD grows linearly or exponentially. Therefore, this study adopts the ROBDD reordering algorithm, whose time complexity is $O(N_{inp} \cdot N_{out})$, where N_{inp} and N_{out} are the numbers of input OBDD nodes and output ROBDD nodes, respectively.

1) CONSTRUCTING ROBDD FOR BOOLEAN FUNCTION

The algorithm generates OBDD nodes from top to bottom according to the variable order π in a breadth-first manner, placing nodes with the same labeled variables at the same level. The OBDD node v representing the Boolean function $f(x_1, x_2, \cdots, x_n)$, namely the root node of the OBDD, is first generated, where the marker variable of node v is $\pi[0]$. Then, OBDDs with labeled variables $\pi[j]$ are generated sequentially according to the variable order π. Two tables with the same structure, *current_table* and *working_table*, are introduced in the implementation.*current_table* is used to store the nodes of the current processed layer, storing table items of the form $< f, u >$, denoting the node u of the Boolean function f. the table *working_table* is used to store the next layer nodes generated from the previous layer nodes. Before generating $< g, v >$, the algorithm first uses g and v as keywords to search whether $< g, v >$ exists in *working_table*. If the table item exists, the result is returned directly; otherwise, a new node v is generated and inserted into *working_table*. When the algorithm is executed until the node in the *working_table* is the final node, an OBDD that satisfies the required variable order π has been generated. Finally, the *Reduce*() simplification rules algorithm is invoked to simplify OBDD into ROBDD.

2) EVALUATION DECISION OF BOOLEAN FUNCTION

The evaluation decision of the Boolean function is determined by performing a depth-first search for ROBDD denoting f under the input mode $x = (x_1, x_2, \cdots, x_j, \cdots, x_n), x_j \in$

Algorithm 1 Constructing the ROBDD of Boolean Function

Input: Boolean function $f(x_1, \ldots, x_n)$ and variable order π
Output: ROBDD of Boolean function $f(x_1, \ldots, x_n)$

```
1:  function Reorder(char *f, char π[n])
2:      vertex * u, *v; char *g
3:      Initialize working_table, current_table to { }
4:      if (!(v = (vertex*)malloc(sizeof(vertex)))) then
5:          exit(OVERFLOW);
6:      else
7:          v−> index = π[0]
8:          Insert table item < f, v > in current_table
9:          for (j = 1; j < n; j++;) do
10:             for For each node u in the current_table do
11:                 $g = f_{x_j=0}$;
12:                 if (exist table item < g, v > in
                        working_table) then
13:                     u−> type.kids.low = v
14:                 else
15:                     if (!(v = (vertex*)malloc(sizeof
                            (vertex)))) then
16:                         exit(OVERFLOW);
17:                     else
18:                         v−> index = π[j]
19:                         u−> type.kids.low = v
20:                         Insert table item < g, v > in
                                working_table
21:                     end if
22:                 end if
23:                 $g = f_{x_j=1}$;
24:                 if (exist table item < g, v > in
                        working_table) then
25:                     u−> type.kids.high = v
26:                 else
27:                     if (!(v = (vertex*)malloc(sizeof
                            (vertex)))) then
28:                         exit(OVERFLOW);
29:                     else
30:                         v−> index = π[j]
31:                         u−> type.kids.high = v
32:                         Insert table item < g, v > in
                                working_table
33:                     end if
34:                 end if
35:                 Delete node u from current_table
36:             end for
37:             current_table = working_table
38:             if all table items in working_table are
                    < 0, u > or < 1, v > then
39:                 break;
40:             end if
41:             Set working_table to {}
42:         end for
43:         Reduce(v)
44:     end if
45: end function
```

Algorithm 2 Evaluation Decision of Boolean Function

```
Input: The root node $v$ of the ROBDD representing the
    Boolean function $f$, input mode $x$
Output: Value of $f(x)$. If $f(x) = 1$, return true; otherwise,
    return false
1:  function Bool Evaluation($vertex * v$, bool $x[n]$)
2:      Construct a dictionary (hash table) with keys as ($x_1$,
          $x_2, \ldots, x_n$, leaf node) and values as $(0,1,2,\ldots,n)$
3:      int $j = 1$;
4:      $vertex * u$;
5:      $u = v$;
6:      while ($j <= n + 1$) do
7:          if ($u- > var == 0$) then
8:              return(false)
9:          else if ($u- > var == 1$) then
10:             return(true)
11:         else
12:             if ($x[j-1] == 0$) then
13:                 $u = u- > type.kids.low$
14:                 if (hashtable[$u- > var$] != $j$) then
15:                     $j$ = hashtable[$u- > var$] + 1
16:                 else
17:                     $j = j + 1$
18:                 end if
19:             else
20:                 $u = u- > type.kids.high$
21:                 if (hashtable[$u- > var$] != $j$) then
22:                     $j$ = hashtable[$u- > var$] + 1
23:                 else
24:                     $j = j + 1$
25:                 end if
26:             end if
27:         end if
28:     end while
29: end function
```

$\{0,\ 1\}$, and obtaining a path starting from the root node v to a certain leaf node, where the value of $f(x)$ is the value of the leaf node. If the output is true, this indicates that the Boolean function can be satisfied under the current input mode, thus achieving the automatic verification that the train control engineering data satisfy the logic rules, and the time complexity of the algorithm is $O(n)$.

V. CASE STUDY

Based on the Boolean function model and ROBDD algorithm design, an automatic verification tool for train control engineering data was developed and experimental analysis was carried out using the train control data from the actual Passenger Dedicated Lines. The line mileage reached 450 km, while the number of data records reached 4600. As shown in Table 2, taking the line gradient table as an example, and drawing on mutation test idea of interlocking route data in literature [36], the non-verification model designer constructs four types of "mutation" data: mutate, swap, add, and remove, then records the modification content. The type of "mutate" includes spurious data coincidence, such as the existence of a short chain at kilometer post K885+131, which will lead to the continuity of adjacent gradient points if the corresponding actual grade length is incorrectly modified to 400 m.

TABLE 2. Partial example of line gradient table.

Serial number	Gradient (‰)	Length of gradient(m)	Terminal mileage	Remark
1	3.7	982	K890+231	
2	1.6	1500	K888+731	
3	-2.5	1100	K887+631	
4	0	2100	K885+531	
5	2.4	200	K885+131	Broken chain
6	-1.2	1300	K883+831	

The sensitivity β for identifying erroneous train control engineering data and the time t required for data verification were used as evaluation indexes. Where $\beta = R/W$, the R is the number of identified "mutation" train control data, and W is the amount of constructed "mutation" data. where the limit value of sensitivity is 0, and close to 0 indicates a complete lack of sensitivity of validation rules to data of "mutation" types. Conversely, a ratio of 1 or close to 1 indicates that the sensitivity has reached the expected value of the verification rule. Usually, the statistically obtained sensitivity is neither 0 nor 1, and there is no fixed explanation for this. Therefore, based on the experience of experts in the field of railway signaling, we define a simple mapping from sensitivity to categorical values (rule ratings). The sensitivity is detected until it converges. Specifically, we compared the calculated mean and variance values of each set of random subsamples containing 50 % of the sample data to be validated with the overall sample and tested whether the two normal distributions were equal.

A. RULE RATING

Rule rating is a user-friendly interpretation of sensitivity. The purpose is to simply state the extent to which the verification rules constrain random variations in certain types of data in the train control engineering data tables. We use letter codes ranging from dash marks(-), which indicate 0 or very low sensitivity, to double-capital letters (e.g., XX), which indicate high or complete sensitivity. The following Table 3 shows the mapping between sensitivity and rule ratings. For different types of train control engineering data tables, there may be differences in the classification of rule rating categories for sensitivity.

In practice, a rule rating is given to a column or a certain concept. If a verification rule syntactically involves multiple columns, a separate rule rating is provided for each column.

The verification sensitivity of the mutation data of the line gradient table under a single rule is shown in Table 4, which shows the sensitivity of the error gradient point data

TABLE 3. The mapping between sensitivity and rule ratings.

Sensitivity	Rating
< 0.01	-
≥ 0.01 and < 0.3	m, s, t, f, x
≥ 0.3 and < 0.7	mm, ss, tt, ff, xx
≥ 0.7 and < 0.97	M, S, T, F, X
≥ 0.97	MM, SS, TT, FF, XX

of different "mutation" types. The median values of the sensitivities of the five groups of samples in Table 4 for different "mutation" types are used to generate a visual radar chart, as shown in Figure 6. The sensitivity for identifying the error data of "mutate" type was the highest, with a median of 97.9%, and its mapping rule rating was SS, which has reached the expectation of the verification rule, indicating that Boolean function model 2 can effectively identify the mutation data of the "mutate" type, especially the continuity of the adjacent gradient points caused by the coincidence of spurious data. The "swap" and "remove" type mutation data were misidentified as mutate-type mutation data, resulting in significantly reduced sensitivity, with rule ratings of xx and tt, respectively. The "add" mutation type data have the anomaly that the data of increasing gradient points still maintain the continuity of adjacent gradient points, thus its sensitivity is the lowest, with a median of 28.1%, corresponding to a rule rating of x. Obviously, in practical applications, a single rule cannot address all mutation cases, and we usually need to evaluate a collection of rules related by their intent. Taking the line gradient table as an example, a conjunct of all rules referencing a given column can deliver its strong sensitivity measure to obtain a more relevant rule rating for a given column. Second, for the same column, there are differences in sensitivity under different verification rules; therefore, it is impossible to judge whether the verification rules of a given column have strong constraints on the type of mutation. For example, if the sensitivity of column c under rule R1 is mm, whereas the sensitivity of column c under rule R2 is M, it is inferred that rule R1 may be contained in R2. Therefore, a sound way to infer the overall rule rating from the individual rule rating of a column is to use the max operator.

TABLE 4. Sensitivity of data for different "mutation" types.

Number of gradient points	Volume of erroneous data	Mutate	Swap	Add	Remove
520	200	0.986	0.675	0.292	0.484
840	480	0.983	0.671	0.285	0.479
1280	720	0.979	0.669	0.281	0.476
1720	840	0.977	0.659	0.278	0.473
2200	1000	0.975	0.665	0.277	0.475

B. MISSING RULES

The verification rule construction is based on railway technical specifications, requirements of train control engineering data tables, and domain expert knowledge. However, the process is not completely systematic. Hence, it cannot guarantee that all the necessary verification rules will be provided. Therefore, it is crucial to establish whether any verification rules were missing before the train control engineering data verification method itself was SIL-qualified [37].

By providing per-column ratings for a collection of rules that constrain the column data, we can understand which rule types have been defined and which rules are still missing that fail to meet the constraints expected by the domain expert, and then evaluate the number and content of the missing rules to achieve a specific expected rating (e.g., MM, SS, etc.). Table 5 shows each column rating of a collection of rules for the constraint column data of the actual passenger dedicated line gradient table.

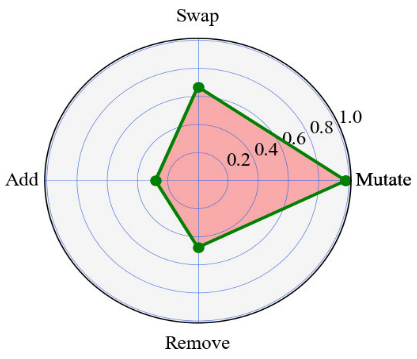


FIGURE 6. Visualization of the medians of data sensitivity for different "mutation" types.

TABLE 5. Rule ratings for line gradient table of actual passenger dedicated lines.

Gradient(‰)	Length of gradient(m)	Terminal mileage	Remark
x	S	T	MM
mm	xx	M	X
t	M	M	TT
ss	F	F	SS
f	T	T	MM

Obviously, most of the rule ratings shown in Table 5 achieve moderate ratings, except for the broken chain marker recognition column, which could be explained by missing or incorrect verification rules. For this reason, we statistically extracted all the undetected "mutation" data from the five groups of samples, then arrived at the following conclusions:

1) The rule ratings for both the gradient length and the terminal mileage columns of the line gradient table shown in Table 5 are up to F, combined with the data sensitivity rule ratings for the "mutate" and "swap" mutation types of the line grade table shown in Table 4 and Figure 6. Evaluate the "swap" mutation type to satisfy the expected rule rating MM (or SS) missing verification rule $\left|G\_termileage_{i+1} - G\_termileage_{i}\right| = G\_length_{i-1} \cap \left|G\_termileage_{i} - G\_termileage_{i-1}\right| = G\_length_{i}$, which can be described as an ITE paradigm solution. Let

the Boolean variables a_i and b_i respectively represent $\left|G\_termileage_{i+1} - G\_termileage_i\right| = G\_length_{i-1}$ as well as $\left|G\_termileage_i - G\_termileage_{i-1}\right| = G\_length_i$, and the Bo-olean function f_i indicates that $a_i \cap b_i$ is established. h_i denotes that the gradient lengths of adjacent gradient points in the line gradient table swap each other, while g_i indicates that they do not swap. Therefore, f_i, g_i, h_i satisfy *if* f_i *then* g_i *else* h_i under the same variable order π, namely $ITE(f_i,\ g_i,\ h_i) = (f_i \wedge g_i) \vee (\neg f_i \wedge h_i)$. At the same time, all the kilometer markers of the line gradient table either increase or decrease in sequence, that is, $(G\_termileage_i < G\_termileage_{i+1}) \vee (G\_termileage_i > G\_termileage_{i+1})$ $(i = 1, \cdots, n)$. Among them, the kilometer markers in the upward direction of the line decrease in sequence, while those in the downward direction increase in sequence, in order to verify the possibility of abnormal swapping of the kilometer markers in the line gradient table.

2) Combine the sensitivity rule ratings xx and tt of the "remove" and "add" mutation type data for the line gradient table presented in Table 4 and Figure 6. Compare the different versions of the line gradient tables and perform a consistency check to determine whether there are "remove" and "add" gradient point data.
3) In addition, by analyzing the error data of omission, it was found that some types of continuous data can only constrain the overrun data. For example, the descending grade gradient out of the home signal is mutated from to, or the gradient values within the same column are swapped. These still meet the upper limit where the maximum value does not exceed (Rule 1), but they cannot be identified.

In addition, through the verification of "mutation" data of the line velocity table, line broken chain detail table, balise position table, main line signal data table, etc., it was found that the missing data types, besides those mentioned in 3) above, also include the following two types. First, the train control engineering data such as balise usage and other textual information belong to natural language, which has only basic normative constraints and no clear quantitative constraint relationship. For example, FQ was modified to Q. Second, some of the isolated data were less constrained. Therefore, the essential reason that the above types of erroneous train control engineering data cannot be recognized is their own defects and weak data correlation. In other words, the verification method for the train control engineering data proposed in this study is feasible.

C. QUANTITY PREDICTION

Rule rating realizes the reversed complement and optimi-zation of bottom-up verification rules. Additionally, during the project development phase, it is critical to predict and plan the number of verification rules required to meet the expectations of the rule rating. Specifically, this planning manifests itself at three levels.

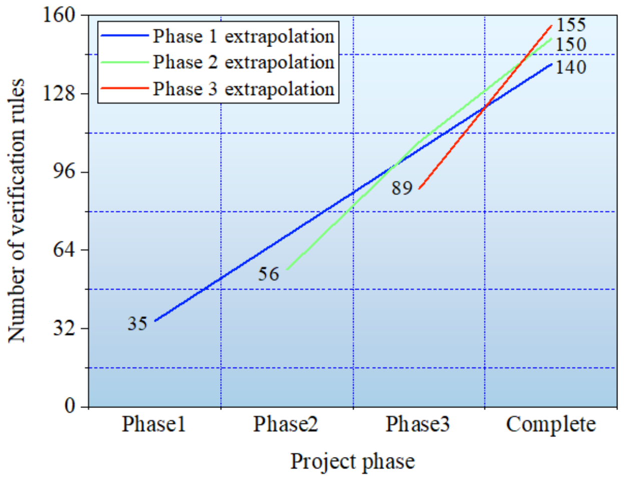


FIGURE 7. Extrapolation of the number of verification rules for different project phases.

1) Evaluate the number of verification rules required to reach the sensitivity threshold. For instance, at least 120 verification rules to constrain the train control engineer-ing data can approach or reach a sensitivity threshold of 0.8.
2) Evaluating the number of verification rules required for a column to achieve the desired rule rating.
3) At the early stage of project development, determining whether the train control data verification project is feasible in principle. For example, if 30 verification rules have been constructed but do not achieve the expected coverage, their feasibility should be judged at this time.

By subsampling a set of constructed verification rules, we can estimate the deviation and plot confidence intervals to further predict the number of rules required to achieve the desired rule rating. Figure 7 depicts the three extrapolations constructed in different project phases. Phase 1 is the extrapolation where the project has reached 35 verification rules; Phase 2 is the extrapolation where the project has reached 56 rules; and Phase 3 is the extrapolation where the project has reached 89 rules. Each successive phase contains all the verification rules from the previous phase, as well as other rules. Combined with Figure 7, it is predicted that 160 verification rules will be required to achieve a sensitivity of 90% (rule rating X).

Figure 8 shows a comparison, in terms of data validation time, between the verification method for train control engineering data proposed in this paper and those proposed in the literature [18], [19], [21]. When verifying 4000 pieces of actual passenger dedicated line train control engineering data, the method proposed in this paper only takes 32 seconds, which is better than other formalized verification methods based on the B language, among others.

VI. DISCUSSION

The construction of a user-friendly rule rating system aims to assist in the improvement of the verification rules for

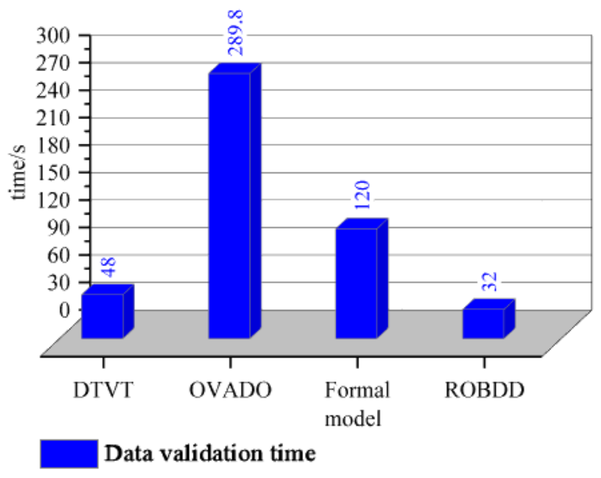


FIGURE 8. Runtime comparison of different verification methods.

train control data. In practical applications, by combining the column rating of the rule combination that constrains the column data of the train control engineering data table with the rule rating of the verification sensitivity for the four specific types of "mutated" data—mutate, swap, add, and remove—the missing rules are identified in reverse, and the number of verification rules required to reach the expected value of the rule rating is planned. Subsequently, the verification rules are complemented and optimized in a bottom-up manner. However, as sensitivity increases, it may become increasingly difficult to find the missing "smaller and smaller" verification rules. Nevertheless, the difficulty in enhancing sensitivity and its mapping rule rating may also indicate that the constructed set of verification rules possesses high quality in terms of validity and completeness. Additionally, the mapping interpretation between sensitivity values and rule ratings requires adjustments based on domain expert experience for different types of train control engineering data tables.

Compared to the traditional manual review of verification rules by domain experts, this study proposed a systematic method for bottom-up closed-loop feedback optimization of verification rules. This approach aims to identify potentially weak or missing verification rules by developing automated verification tools that ensure efficiency and completeness. While formal techniques, such as model testing and theorem proving, can prove the validity and completeness of rules, they are often time-consuming and labor-intensive. The method proposed in this paper serves as a compromise between the two extremes, combining model checking and statistical validation within the framework of "mutation testing."

VII. CONCLUSION

In this paper, a formal verification method of CTCS-2 level train control engineering data based on ROBDD was proposed. An automated verification tool for train control engineering data was developed by transforming the implicit logical association rules among the extracted train control engineering data into Boolean function models and their equivalent canonical ROBDD algorithms. This tool solved the problem of insufficient timeliness and completeness in traditional manual validation of train control engineering data, and it can effectively identify spurious data coincidences caused by mutations in the data. Furthermore, taking the verification sensitivity of "mutated" train control engineering data as a bridge, a user-friendly classification rule rating system was introduced to guide domain experts to make up for the missing verification rules. The case study showed that, due to the rigorous formal description of the logical relationships constraining train control data and the construction of a reasonable and detailed rule rating system, this method is feasible and complete in the enterprise's actual train control data verification process.

The four types of "mutated" data proposed in this paper were generated by non-model designers. In the future, genetic algorithms (GA) can potentially be used to build fitness functions based on the constraint relationships of train control engineering data, and mutation data generators satisfying the strong constraint relationships of train control engineering data can be synthesized to automatically generate a large amount of mutation data as verification samples.

In addition, we will attempt to use data mining techniques, such as the data association rule algorithms FP-Growth and Apriori, to mine the implicit association relationships among train control data, which is to expand the avenues for constructing verification rules. In the long run, developing a method to automatically synthesize missing rules is a new and highly challenging goal.

ACKNOWLEDGMENT

The CTCS-2 level train control engineering dataset for passenger dedicated lines used in this study originates from the Beijing National Railway Research and Design Institute of Signal and Communication Group Company Ltd. The extraction of verification rules was effectively assisted by domain experts from this company. Hereby, we express our heartfelt thanks.

REFERENCES

[1] N. Storey and A. Faulkner, "The characteristics of data in data-intensive safety-related systems," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.* Berlin, Germany: Springer, 2003, pp. 396–409.

[2] Y. Zhang, H. Wang, M. Chai, and R. Cheng, "Novel graph-based train control data verification method for Chinese train control system," *IEEE Intell. Transp. Syst. Mag.*, vol. 13, no. 3, pp. 45–57, Fall. 2021.

[3] U. Berger, P. James, A. Lawrence, M. Roggenbach, and M. Seisenber-Ger, "Verification of the European rail traffic management system in real-time Maude," *Sci. Comput. Program.*, vol. 154, pp. 61–88, Mar. 2018.

[4] M. Ouyang, L. Hong, M.-H. Yu, and Q. Fei, "STAMP-based analysis on the railway accident and accident spreading: Taking the China–Jiaoji railway accident for example," *Saf. Sci.*, vol. 48, no. 5, pp. 544–555, Jun. 2010.

[5] S. Ghosh, A. Das, N. Basak, P. Dasgupta, and A. Katiyar, "Formal methods for validation and test point prioritization in railway signaling logic," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 678–689, Mar. 2017.

[6] A. Nash, D. Huerlimann, J. Schuette, and V. P. Krauss, "RailML-A standard data interface for railroad applications," *WIT Trans. Built Environ.*, vol. 74, pp. 233–240, May 2004.

[7] M. Bosschaart, E. Quaglietta, B. Janssen, and R. M. P. Goverde, "Efficient formalization of railway interlocking data in RailML," *Inf. Syst.*, vol. 49, pp. 126–141, Apr. 2015.

[8] Ciszewski, W. Nowakowski, and M. Chrzan, "RailTopoModel and RailML-data exchange standards in railway sector," *Arch. Tran-sport Syst. Telematics*, vol. 10, no. 4, pp. 10–15, Nov. 2017.

[9] *Railtopomodel-Railway Infrastructure Topological Model*, document IRS30100, 2016.

[10] A. Hlubuček. (2023). *Integration of railway infrastructure topological description elements from the MicroL2 to the MacroN0, L0 Level of Detail*. Neural Network World. [Online]. Available: http://nnw.cz/doi023/NNW2

[11] A. Hlubuček, "Possibilities of high-speed railway turnout data description," *Acta Polytechnica CTU Proc.*, vol. 31, pp. 18–26, Jul. 2021.

[12] S. Bischof, and G. Schenner, "Rail topology ontology: A rail infrastructure base ontology," in *Proc. Int. Semantic Web Conf.*, vol. 12922, Cham, Switzerland: Springer, Oct. 2021, pp. 597–612.

[13] M. N. Menéndez, S. Germino, L. D. Díaz-Charris, and A. Lutenberg, "Automatic railway signaling generation for railways systems described on railway markup language (railML)," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 3, pp. 2331–2341, Mar. 2024.

[14] A. Ferrari and M. H. T. Beek, "Formal methods in railways: A systematic mapping study," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–37, Nov. 2022.

[15] A. Ferrari, F. Mazzanti, D. Basile, and M. H. ter Beek, "Systematic evaluation and usability analysis of formal methods tools for railway signaling system design," *IEEE Trans. Softw. Eng.*, vol. 48, no. 11, pp. 4675–4691, Nov. 2022.

[16] F. Badeau and M. Doche-Petit, "Formal data validation with event-B," 2012, *arXiv:1210.7039*.

[17] D. Hansen, D. Schneider, and M. Leuschel, "Using B and ProB for data validation projects," in *Proc. 5th Int. Conf. Abstract State Mach., Alloy, B, TLA, VDM, Z*, Linz, Austria: Springer, May 2016, pp. 167–182.

[18] T. Lecomte, L. Burdy, and M. Leuschel, "Formally checking large data sets in the railways," 2012, *arXiv:1210.6815*.

[19] J. Falampin, H. Le-Dang, M. Leuschel, M. Mokrani, and D. Plagge, "Improving railway data validation with ProB," in *Industrial DepLoyment of System Engineering Methods*. Berlin, Germany: Springer, 2013, pp. 27–43.

[20] A. S. A. Hadad, C. Ma, and A. A. O. Ahmed, "Formal verification of AADL models by event-B," *IEEE Access*, vol. 8, pp. 72814–72834, 2020.

[21] R. Abo and L. Voisin, "Formal implementation of data validation for railway safety-related systems with OVADO," in *Software Engineering and Formal Methods*, vol. 8368. Cham, Switzerland: Springer, Jan. 2014, pp. 221–236.

[22] F. Badeau, J. Chappelin, and J. Lamare, "Generating and verifying configuration data with OVADO," in *Proc. Int. Conf. Rel., Saf., Secur. Railway Syst.*, vol. 13294, Cham, Switzerland: Springer, May 2022, pp. 143–148.

[23] A. E. Haxthausen, J. Peleska, and R. Pinger, "Applied bounded model checking for interlocking system designs," in *Proc. Int. Conf. Softw. Eng. Formal Methods*. Madrid, Spain: Springer, Jan. 2014, pp. 205–220.

[24] J. Peleska, N. Krafczyk, A. E. Haxthausen, and R. Pinger, "Efficient data validation for geographical interlocking systems," *Formal Aspects Comput.*, vol. 33, no. 6, pp. 925–955, Dec. 2021.

[25] B. Luteberget, C. Johansen, and M. Steffen, "Rule-based consistency checking of railway infrastructure designs," in *Proc. 12th Int. Conf. Integr. Formal Methods*, Reykjavik, Iceland: Springer, Jun. 2016, pp. 491–507.

[26] A. Iliasov, D. Taylor, L. Laibinis, and A. Romanovsky, "Practical verification of railway signalling programs," *IEEE Trans. Depend. Secure Comput.*, vol. 20, no. 1, pp. 695–707, Jan. 2023.

[27] A. Iliasov, D. Taylor, L. Laibinis, and A. Roman-ovsky, "Formal verification of railway interlocking and its safety case," in *Proc. Saf.-Crit. Syst. Symp. (SSS)*, 2022, pp. 8–10.

[28] T. Wang, H. Zhao, and L. Zhu, "Satisfiability verification of engineering data safety rules of balise based on ROBDD," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 2386–2391.

[29] J. Clément and A. Genitrini, "Binary decision diagrams: From tree compaction to sampling," in *Proc. Latin Amer. Symp. Theor. Informat.*, Cham, Switzerland: Springer, Dec. 2020, pp. 571–583.

[30] K. Havelund and D. Peled, "BDDs for representing data in runtime verification," in *Proc. Int. Conf. Runtime Verification*, vol. 12399. Cham, Switzerland: Springer, Oct. 2020, pp. 107–128.

[31] M. Kubica, A. Opara, and D. Kania, "Binary decision diagrams," in *Technology Mapping for LUT-Based FPGA*. Cham, Switzerland: Springer, Nov. 2021, pp. 25–37.

[32] L. Xing, "Decision diagrams for complex system reliability analysis," in *Frontiers of Performability Engineering*. Singapore: Springer, 2024, pp. 51–67.

[33] R. Ma and L. Du, "Efficient pairing-free attribute-based blind signature scheme based on ordered binary decision diagram," *IEEE Access*, vol. 10, pp. 114393–114401, 2022.

[34] S. Li, Y. Song, and Y. Zhang, "Combinatorial test case generation based on ROBDD and improved particle swarm optimization algorithm," *Appl. Sci.*, vol. 14, no. 2, p. 753, Jan. 2024.

[35] J. Newton and D. Verna, "A theoretical and numerical analysis of the worst-case size of reduced ordered binary decision diagrams," *ACM Trans. Comput. Log.*, vol. 20, no. 1, pp. 1–36, Jan. 2019.

[36] L. Laibinis, A. Iliasov, and A. Romanovsky, "Mutation testing for rule-based verification of railway signaling data," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 676–691, Jun. 2021.

[37] *Railway Applications-Communication, Signalling and Processing Systems-Software for Railway Control and Protection Systems*, Standard European EN 50128, European Committee for Electrotechnical Standardization, Brussels, Belgium, Jun. 2011.

HAO ZHANG received the B.Eng. degree in rail transit signal and control from Lanzhou Jiaotong University, Lanzhou, China, in 2017, where he is currently pursuing the M.Eng. degree in traffic information engineering and control.

From 2023 to 2024, he participated in the research and development of the first "Lineside Electronic Unit (LEU) Detection Platform" for CTCS-2 train control system in China with Beijing National Railway Research and Design Institute of Signal and Communication. In addition, he developed an automated verification tool for train control engineering data. His research interests include formal modeling and validation of train control engineering data and on-board ATP multifunction vehicle bus fault diagnosis for CTCS-3.

QING XU received the B.Eng. degree in railway signal from Lanzhou Railway College, Lanzhou, China, in 1993.

She was a Research and Development Minister, the Director of the Technology Center, Beijing Railway Signal Company Ltd., a Chief Engineer, and a Chief Expert of CRSC (Beijing) Industry Group Company Ltd. Since October 2021, she has been a Deputy Chief Engineer of Beijing National Railway Research and Design Institute of Signal and Communication. Her research interests include formal modeling and validation of train control engineering data, measurement methods in Balise transmission systems, and on-board ATP multifunction vehicle bus fault diagnosis for CTCS-3. Her awards and honors include six science and technology awards from China Railway Society, the MAO Yisheng Railway Engineer Award, Beijing Outstanding Young Engineer Award, and Locomotive Medal.

KE YE was born in 1989. He received the B.S. degree in automation and the M.S. degree in electronics and communication engineering from Beijing University of Technology, China, in 2011 and 2017, respectively, where he is currently pursuing the Ph.D. degree in mechanical engineering. His research interests include measurement methods in Balise transmission systems, digital signal processing, and the application of non-destructive testing technology in transportation engineering.