A multi-representation method of building rules for automatic code compliance checking

Z. Zhang, N. Nisbet, L. Ma & T. Broyd
The Bartlett School of Sustainable Construction, UCL, London, United Kingdom

ABSTRACT: In the Architecture, Engineering and Construction (AEC) industry, design review is an important step that often leads to project delays, as the typical manual compliance checking process is error-prone and time-consuming. As an approach to accelerate this process and achieve a better quality of design, automatic compliance checking (ACC) has been researched for several decades. Rule interpretation and representation is a bottleneck of ACC. It focuses on the interpretation of regulations and the representation of them in a suitable computer-readable form. Despite extensive research efforts, a rule representation method that is suitable to represent all types of rules has yet to be proposed. To address this issue, this research proposed a multi-representation method that provides a "mix and match" for different representations and different types of rules, thereby representing all types of rules with suitable representations. This research is valuable to both academia and industry as it enables the representation of rules with less knowledge loss and more accuracy.

1 INTRODUCTION

In the Architectural, Engineering and Construction (AEC) industry, the design needs to be checked against regulations before the construction permit can be obtained. The compliance checking is traditionally conducted manually by rule experts, suffering from low efficiency, low accuracy and high cost. The errors may result in project delays or poor operational performance. To address these issues, automatic compliance checking (ACC) has been researched for more than 50 years. There have been two perspectives to approach ACC, one looks at the easy retrieval and query of building model data, and the other focusses on the interpretation and representation of regulations. The latter is claimed to be the bottleneck of the ACC process, which can take up to 30 % of the total time for implementing a rule (Solihin & Eastman, 2016).

Recognizing the important role of rule interpretation and representation in understanding and retaining construction design knowledge, various methods were developed to represent building codes. While these methods have been useful to handle various aspects of the knowledge domain, none of them has all the capabilities required to represent regulations (Macit İlal & Günaydın, 2017). There also lacks a thorough understanding of rules, representation methods and the relationships between them; and the methodological backdrop of the current representation development is weak (Zhang et al., 2022).

This paper thus aims to: 1) compare the methods of rule representation and identify their capabilities; 2) select suitable representation methods for different types of rules; 3) propose a multi-representation method that is capable of representing all types of rules. The relationships between rule types, capabilities, suitable representation for each rule class and a multi-representation method are shown in Figure 1.

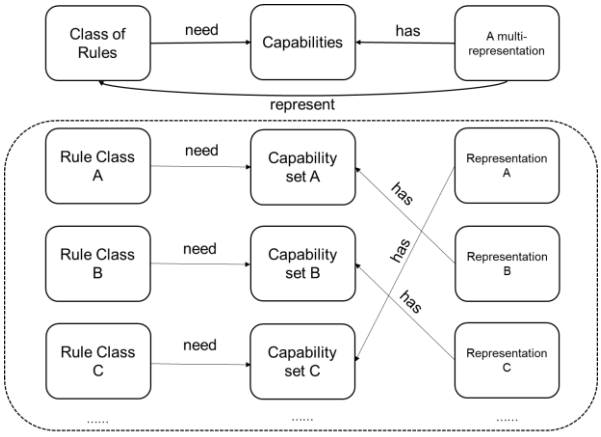

Figure 1. Relationships among class of rules, capabilities, and representation methods

The upper boxes and arrows show the overall relationships of class of rules, capabilities and the multi-representation method to be developed in this paper.

For each column, it has several subsets as shown in the lower boxes (i.e. rule classes, capability sets and representations). The multi-representation method can only be developed when different types of rules and capabilities required to represent them are understood. Once the suitable representation methods for different classes of rules are selected, a "mix and match" approach can be used to establish the multi-representation method.

2 METHOD

In this paper, design science research (DSR) is adopted. It is suitable for this research as it aims to provide practical solutions for industry problems (i.e. a representation method for building rules). DSR has five steps, including awareness of problem, suggestion, development, evaluation and conclusion (Vaishnavi, 2007). This paper presents the awareness of problem and suggestion as a solid foundation for developing a rule representation method. The suggestion of a new representation method is based on abductive reasoning. It involves the researchers selecting the "best" based on a set of pragmatic criteria. In this paper, the authors adopted the pragmatic criteria (Table 1) proposed in our previous work (Zhang et al., unpubl.). We seek to select the most capable representations based on those criteria. The evaluation will also help understand which representation is suitable for each type of rule.

3 EVALUATION OF EXISTING REPRESENTATION METHODS

3.1 *Evaluation Criteria*

Table 1. Required and desired capabilities for rule representation

Aspects	Required capabilities	Desired capabilities
Rule features	1) Requirement 2) Applicability 3) Selection 4) Exception 5) Definition 6) Outcome 7) Logical relationship	NA
Rule intensity	1) No calculation or simple calculation 2) Function and algorithm 3) Simulation	NA
Rule organization	1) Hierarchy 2) Cross-reference	NA
Implementation	1) Easy to use and understand 2) Dictionary 3) Independent of the rule engine 4) Independent of the data model	1) Conciseness 2) Translatability

Zhang et al. (unpubl.) proposed 16 required and two desired capabilities (Table 1) for representing building regulations. It is so far the most comprehensive set of criteria. Thus, we have chosen these criteria to evaluate the representations.

The evaluation of representation methods was twofold. Firstly, the current capabilities of representations were evaluated against the above-mentioned criteria. Secondly, they were also assessed regarding their potential of developing certain capabilities given their characteristics.

3.2 *Evaluation and Results*

Some early ACC developments use procedural code to represent and execute rules. An example of this is the CORENET e-PlanCheck project in Singapore (Solihin, 2004). It is a hard-coded approach, where the system is a "black box" with a built-in rule engine and procedural codes for executing rules, thus the representation is not independent of data model and rule engine. The system mainly focused on individual rule provisions and neglected cross-references. Because of the binary nature of the programming language, the checking outcomes only include "true" or "false", whilst other actions or side-effects cannot be represented. The system also requires extensive programming knowledge to develop, which domain experts typically do not have. However, its advantage lies in dealing with rules of high intensity using algorithms and functions and considering hierarchies of rules and logical relationships within a rule provision to get correct checking outcomes.

Production rule has been widely used for representing rules. It follows the pattern of "if <condition> then <action>". An example in the AEC industry is Tan et al. (2010). They proposed decision tables to present regulations in a concise and compact way, where each row is a production rule. Decision tables can represent applicability, selection and requirements in its cells. The actions of the checking include "pass", "fail" and "exceptional". Cross-references among rule provisions were also considered. However, the deficiency of this method is: 1) it cannot represent unknown and other side effects; 2) no explicit logical relationships are shown within each row; 3) it lacks consideration of the rule context (e.g. definitions of terms); 4) it failed to reveal the implicit knowledge and assumption of rules, thus making it difficult to check rules with higher intensity and/or performance-based rules.

Solibri (2022) is the most widely-adopted commercial ACC software. It used parametric tables to represent rules, which is easy to use after some simple training. Currently, there are 55 rule templates to check model and data quality and consistency, objects and properties, and geometries. The rule templates include object and space groups and some other pre-defined applicability such as "air well". APIs are also

available to allow experienced users to build their own rule templates. However, the rule templates only cover a limited number of objects, relationships and algorithms. Thus, Solibri has limited capability to deal with rules with higher intensity (e.g. rules requiring functions, algorithms and simulations). In addition, it must depend on the built-in rule engine and internal data model to finish the check. Like many other methods, it also only includes "pass" and "fail" as outcomes. The logical relationships within one rule provision are not clear. Any cross-references among rules and the hierarchy of regulations are not represented. Although Solibri allows the user to implement any definitions (e.g. fire compartment) using a mixture of rules and individual picks of entities, the broader context for applying a rule is limited to using a second rule as a "gate-keeper".

Predicate logic is a well-defined function in logic that can generate results of true/false (or undefined). It has been used to represent building rules by many researchers. It has quantifiers to distinguish "ALL" instances from the case that there "EXISTS" an instance, making it possible to represent applicability, selection and exception. It also uses logical connectives to denote conjunction, disjunction and negation. For example, Rasdorf & Lakmazaheri (1990) proposed a logic-based SASE (Standards Analysis, Synthesis and Evaluation) model that used predicate logic statements to represent rules. This model showed rule categories and relationships by linking rules using a tree-like organizational sub-model. The expressions of predicate logic are also independent of rule engine, with the potential of being independent of the building data model depending on the specific software. However, the drawbacks of predicate logic are: 1) as a two-value logic, it only has true or false as outcome; 2) it can become very lengthy and complex when the rules are complex; 3) it was not used to present definitions and other context; 4) it does not reflect the hierarchy of the regulations.

Having its root in predicate logic, conceptual graph (CG) was initially proposed by Sowa (1976) and later extended by Solihin & Eastman (2016) for representing building regulations, where different shapes were used to represent concept nodes, conceptual relations and represent functions intuitively. Further extensions include representing cross-references and exceptions, and denoting either a derived concept or a concept that requires functions/algorithms/simulations. CG also uses "or" and "not" to represent logical disjunction or negation, and the lines mean logical conjunction by default. Its graphical representation makes it easy to use and understand. Despite being a fairly comprehensive representation method, it has several drawbacks, including its current inability to represent applicability and selection and to show outcomes other than "pass" and "fail"; it cannot show hierarchies of rules; it does not represent definitions; and rules are represented using a model-dependent IFC format. It also does not have a dictionary to link the data model ontology to the rule ontology. However, an independent representation of rules might be possible in its future developments.

Visual Code Checking Language (VCCL) is a language-driven ACC solution proposed by Preidel & Borrmann (2016). It is a user-friendly visual programming language aiming to provide transparent "white boxes" (as opposed to black boxes) for rule checking. It has tailored methods for building rules, including logical, mathematical, geometric-topological, relational, building model related and utility methods. They are executed by the built-in rule engine. Each rule provision can be represented by linking multiple input ports, output ports and method nodes. Nevertheless, VCCL still failed to consider a broader context. It only has binary results including "true" and "false". It is also restricted by the IFC format and thus does not have a dictionary to map ontologies. In addition, it failed to consider the hierarchy of regulations and cross-references among rules, but the authors argue it might be possible for VCCL to incorporate these in a later version.

RASE (Hjelseth & Nisbet, 2011) is a semantic-based approach. RASE has been updated several times, here we evaluate the latest version of RASE (Beach et al., 2015) published in academic literature. It represents the semantic constructs of rules, including requirements, applicability, selection and exceptions. Its striking feature is the consideration of broader context by incorporating definitions and/or titles in the rule documents. It keeps the rule text as is and uses mark-ups to mark different semantic constructs. The mark-ups are easy to use once the user has some knowledge of semantic constructs. In addition, keeping the rules as is has several advantages: 1) the structure of regulations is retained, including cross-references, hierarchy, and logical relationships (although logical relationships may not be explicit); 2) the representation is independent of the rule engine and model data. To link the design and the regulation ontologies, a dictionary was developed. As for checking outcomes, RASE uses an open world assumption and its checking outcomes include "pass", "fail", "unknown" and other side-effects such as "add credits". A limitation is that it cannot explicitly present the knowledge needed to address the rule intensity.

More recently, some scholars developed representations using Semantic Web. Rules are represented using SPARQL that has similarities to SQL. The queries can then be applied to RDF/OWL representations of a model, often with extensions to simplify the model and underlying schema. The syntax in the semantic web requires specialized training. There has been little progress in exploiting geometry or computationally intensive queries, thus, it has a low capability of dealing with rule intensity. In Pauwels et al. (2011), they used N3Logic to represent building rules, which is essentially a form of "if <condition>

then <action>", with logical connectives to denote logical relationships. This production-rule-like representation limits expressiveness. Consequently, it does not have the capability of representing exceptions, outcomes other than "pass" or "fail". It also did not consider a broader context such as definitions. The semantic web representations have potential to be independent of the data model, although most of them currently use IFC. However, it will rely on the rule engine regardless of the representation form, and a dictionary can be used to link ontologies. Regarding rule organization, it can represent hierarchies of rules and cross-references.

considering the cross-references and hierarchies of rules. Despite its defects, this method recognized the importance that the representation needs to be independent of both the rule engine and the data model, and it used logical rules and facts to achieve that. In general, it requires some knowledge of logic to use but some simple training would suffice.

As a result of the analysis above, a summary of the shortlisted representation methods and their capabilities are shown in Table 2. The meaning of numbers is explained in the notes under Table 2. From Table 2, it is evident that no existing representation method has all of the capabilities required to represent building rules, although methods such as CG and RASE check most of the boxes. To minimize the knowledge loss during the rule interpretation and representation processes, a more well-rounded representation method is needed.

Table 2. Evaluation result of representation methods

Capability / Method	A	B	C	D	E	F	G	I	J	K	N	O	Q	References
Procedural Code	√	√	√	√	×	√	3	√	×	L	×	×	×	Solihin (2004)
Decision Table	√	√	√	×	P	√	2	×	√	H	×	√	√	Tan et al. (2010)
Solibri	√	√	×	√	×	×	2	×	×	H	×	×	×	Solibri (2022)
Predicate Logic	√	√	√	×	P	√	3	×	√	M	P	√	P	Rasdorf & Lakmazaheri (1990)
Conceptual Graph	√	P	√	×	P	√	3	×	√	H	×	√	P	Solihin & Eastman (2016)
VCCL	√	P	P	×	P	P	2	×	P	H	×	×	×	Preidel & Borrmann (2016)
RASE	√	√	√	√	√	×	P	√	√	H	√	√	√	Beach et al. (2015)
Semantic Web	√	√	×	×	×	√	1	√	√	L	√	×	×	Pauwels et al. (2011)
NLP	√	√	×	×	×	×	1	×	×	M	×	√	√	Zhang & El-Gohary Nora (2017)

Note: A=Requirements; B= Applicability/Selection; C=Exception; D= Definition; E= Outcome (Other actions/Side effects); F= Logical Relationships; G= Rule Intensity (level 1-3); I= Hierarchy; J=Cross-reference; K= Easy to Use and Understand; N= Dictionary; O= Independent of Rule Engine; Q= Independent of Data Model; H/M/L=High, Moderate, Low; P = Potential of developing the capability

Aiming at a fully-automated ACC system, semantic natural language processing (NLP) methods were proposed by researchers. Most research has narrowed the scope to the simplest subject-predicate sentences, ignoring subsidiary clauses, titles, lists and other matters. Zhang & El-Gohary (2017) have tried various approaches to build on the named-object identification, such as assuming all terms are requirements and matching sentences to pre-defined templates. They did not pay attention to the representation of exception, applicability and selection. The checking outcome includes "pass", "fail" and "unknown", but no other actions were considered. The method can only deal with rules with relatively low intensity. In addition, the focus is mainly on rule provisions, without

4 A CLASSIFICATION OF BUILDING RULES

In Section 3, it was recognized that there is no single representation method that is suitable to represent all types of rules based on the capabilities. As the suitable representation method needs to be independent of the rule engine and data model, it is envisaged that a multi-representation method could collaboratively address this issue by mixing and matching representations with different capabilities. To achieve this, it is important to first understand what are the different types of rules and what can be the criteria to distinguish them. Hence, in this section, the authors propose a new classification by analyzing example regulations: Health Building Notes (Department of Health and Social Care, 2017) and Approved Documents (Ministry of Housing, 2010).

4.1 *Manual or automatic checking*

A frequently mentioned topic in ACC is what type of rules can be checked automatically. Researchers held different opinions regarding this issue, which are ultimately reflected in their classifications. For example, Soliman-Junior et al. (2020) used quantitative/qualitative/ambiguous and ability to be translated into logical rules as criteria for classifying rules, where they claimed rules that cannot be translated into logical rules can only be checked manually.

However, as suggested by Zhang et al. (unpubl.), some rules are automatically checkable when they are more thoroughly analyzed. In this paper, the authors argue that only rules that include words or phrases that are subjective (i.e., based on or influenced by personal feelings, tastes, or opinions) cannot be checked automatically. It is because this type of rule is typically open to interpretation. It is thus difficult to have a specific object, or attribute value or value range to check, thereby making it difficult to be automated. For example, Rule 1 shows requirements regarding quality and aesthetics. Different people may have different ideas of "pleasant and welcoming".

"The main entrances and reception areas should be pleasant and welcoming."

Rule 1. A subjective requirement, HBN 00-01, Appendix 1, (Department of Health and Social Care, 2017)

Other rules are automatically checkable once sufficient information has been provided to define or clarify any insufficiently defined words or phrases (e.g., close to). Notably, although it has been a common belief among many scholars that only rules related to the design stage can be checked, the authors argue that operational rules can also be checked if they are appropriately interpreted. Rule *3* is an operational requirement for the shower seat. This requirement can inform the design and be interpreted as a requirement for adjustable shower seats.

"…The position of the shower seat should be adjusted between uses as required."

Rule 2. An operational rule, HBN 00-02, Rule 4.48 (Department of Health and Social Care, 2017)

4.2 *Rule classifications*

Based on the analysis of rules that have the potential to be automatically checked, a new classification method is proposed in this section. Incorporating the classification criteria proposed by scholars such as Solihin & Eastman (2015), Macit İlal & Günaydın (2017) and Hjelseth & Nisbet (2011), the new classification method recognized that a single criterion might neglect other criteria that distinguish rules within the class from each other. Hence, in this paper, we proposed a new classification using four criteria, namely semantic constructs, self-contained or linked explanatory, rule intensity and prescriptive or performance-based.

4.2.1 *Semantic constructs*

Semantic constructs concern the components of rule provisions with specific semantic meanings. Hjelseth & Nisbet (2011) proposed the RASE method, recognizing four semantic constructs of building rules: Requirement (R), Applicability (A), Selection (S) and Exception (E). Developed from their method, the authors included three more semantic constructs here: definitions (optional), outcomes (compulsory) and logical relationships (optional).

Definition is closely related to requirements, in the sense that it provides context for requirements and clarification for terms. Studies such as Zhang et al. (2022) stressed the importance of considering a broader context (e.g. definitions and titles) when analyzing rules.

Rule 3 shows the definition of the term TER. It is important for accurate calculation of TER to meet energy efficiency requirements.

"The Target CO2 Emission Rate (TER) is the minimum energy performance requirement for a new building based on the methodology approved by the Secretary of State in accordance with regulation 25. It is expressed in terms of the mass of CO2 emitted per year per square metre of the total useful floor area of the building."

Rule 3. Definition of TER, Approved Document L2A, Rule 2.2 (Ministry of Housing, 2010)

Outcomes are sometimes implicit in rule texts. They can take many forms. They can be as simple as pass or fail. Taking an open-world assumption (Hustadt, 1994), they can also be unknown when the information is not sufficient to make a judgement. They may include other actions and side-effects such as "adding 5 credits (points)" in BREEAM (Building Research Establishment, 2018), where the final credits and rating are based on the accumulation of credits awarded in different evaluation aspects.

Note that all rules have compulsory semantic constructs, but each rule can have different types of optional semantic constructs. As a result, one rule can have more or less semantic constructs than another rule. For example, the rule constructs of Rule 4 are marked below. The underlined part is the requirement. The part in bold is the selection, which is slightly different from "applicability" as a selection offers alternative subjects. It thus applies to both staff use and patient use.

*"Help call should be provided if the room is **for staff or patient use**."*

Rule 4. HBN 00-02,Rule 3.24, Semantic constructs in a rule (Department of Health and Social Care, 2017)

Logical relationships concern the logical connectives between words or phrases in rules. The same set of words and phrases, if linked using different logical connectives, can have very different meanings. Logical relationships mainly include "and", "or" and "not", meaning logical conjunction, disjunction and negation, respectively.

4.2.2 *Self-contained or linked explanatory*

Another criterion is self-contained and linked explanatory. Initially proposed by Macit İlal & Günaydın (2017), this criterion has a different meaning here. Self-contained means that a rule provision is self-explanatory (i.e. does not rely on other rule provisions). Linked explanatory rules refer to rules that do not have complete meanings themselves. They have to be looked at together with other rules. A typical example of linked explanatory rules is cross-references. Rule 5 shows a rule provision with cross-reference. It means that paragraph 5.27 needs to be consulted to make sure the rule is fulfilled.

"A landing should be provided at the top and bottom of each flight of stairs. The minimum clear landing depth is 1200 mm but must equal the clear stair width between handrails (see paragraph 5.27)"

Rule 5. A rule provision with cross-reference, HBN 00-04, Rule 5.4
(Department of Health and Social Care, 2017)

4.2.3 *Prescriptive or performance-based rules*

Prescriptive rules often present requirements explicitly, meaning that the path to compliance is clearly stated (e.g. the required properties, objects, and relationships). However, for performance-based rules, only the expected performance was asked, rather than how it can be achieved. The designers need to use their expertise to make sure the design meets the performance requirements. Performance-based rules are also not straightforward to check, as they either require domain experts to make the implicit knowledge explicit by rule interpretation or computationally intensive simulations. Rule 6 provides the required air change rate. It requires ventilation knowledge to interpret this rule into a machine-readable rule.

"The room will be considered fit for purpose if, with the ventilation system operating and all doors closed, the following parameters are achieved:
•the patient's room has an air change rate of at least 10 per hour;"

Rule 6. A performance-based rule, HBN 04-01 Supplement 1, Rule A2.12
(Department of Health and Social Care, 2017)

4.2.4 *Rule intensity*

Many studies have mentioned the difficulty of automatically checking building rules (Eastman et al., 2009), including: 1) rules written in human language can be ambiguous, subjective and thus hard to interpret and represent using machine-readable representations, and 2) some rules require intensive computations.

Since the term "rule complexity" has been adopted to describe rules with one or more of the above-mentioned characteristics without a consensus by academics and practitioners, the authors propose the term "rule intensity" in this paper. Compared with the multiple facets of "rule complexity", rule intensity only concerns the computational power required to check the rule automatically. Based on this, rules can be classified into three categories:

1) No calculations or simple calculations
2) Functions or algorithms
3) Simulations

No calculations or simple calculations means the requirement is straightforward and does not require complex calculation. For example, rules that only require checking the existence of objects or the property value of an object fall into this category (Rule 7). Rules that include only simple arithmetic calculations and no spatial or physical calculations also fall into this category.

"Bidets should be fitted with a sensor-operated over-rim supply."

Rule 7. A rule checking the existence of simple objects, HBN 00-02, Rule 2.3
(Department of Health and Social Care, 2017)

Rules with slightly higher intensity are included in the functions or algorithms category. This type of rule may involve combinatorial issues that deal with multiple objects and possibilities to compliance (Solihin & Eastman, 2015). For example, Rule 8 requires clear space around the toilet on the bath side to allow the accessibility of mobile hoist.

"The room layout utilises the minimum clear space requirement to the side of the toilet for mobile hoist transfer (that is, 1150 mm from the centreline of the toilet to the nearest obstruction), on the bath side of the toilet only."

Rule 8. A rule that requires function/algorithm to check, HBN 00-02, Rule 2.22
(Department of Health and Social Care, 2017)

Rules with the highest intensity are rules that require simulations. Many rules in this category are performance-based rules and they are typically found in fire codes, energy requirements, etc. They typically ask for a proof-of-solution or a feasible design to achieve performance requirements. Rule 9 asks for a design to satisfy the energy performance requirement as set by regulation 24. The approved software suggested will run simulations to help generate energy performance calculations and predictions.

"The TER must be calculated using one of the calculation

tools included in the methodology approved by the Secretary of State for calculating the energy performance of buildings pursuant to regulation 24.…"

Rule 9. A rule that requires simulation, Approved Document L2A, Rule 2.3 (Ministry of Housing, 2010)

5 PROPOSED RULE REPRESENTATION METHOD

5.1 *Representation selection for different types of rules*

Given the evaluation in Section 3.2 and the analysis of rules in Section 4, suitable representations can be selected by comparing the rule characteristics and the capabilities of representations (Figure 1). As the combination of different criteria for classifying rules can generate too many rule types, the authors only provide several examples here to demonstrate the process. For example, we can regard that a rule has requirements, definitions, and a checking outcome that includes other side-effects as Rule Class A. For Rule Class A, RASE would be the most suitable representation, as only RASE have both the capabilities of representing definitions and side-effects. There could also be a Rule Class B that has a requirement of level 3 rule intensity. In this case, conceptual graph could be the most suitable representation, as CG is easier to use when the rule intensity is high.

5.2 *A multi-representation method for all types of building rules*

A suitable rule representation method should be capable of representing all types of rules. To achieve this, three representation methods, namely RASE, predicate logic and conceptual graph, were selected to cover all capabilities, which forms a multi-representation method. Figure 2 shows when to select which representation.

RASE can deal with most of the required capabilities, except it does not have explicit logical relationships and does not make the implicit knowledge and assumptions explicit. It is ideal for larger scale documents because the time-saving advantage of mark-ups is especially evident when dealing with many documents.

Predicate logic deals with specific problems and individual rule provisions. It has a sound theoretical base, and it is very suitable for representing building rules. Conceptual graphs come into play when the rule intensity is high and the predicate logic statements get lengthy and hard to read.

6 DISCUSSION

It came with no surprise that from the evaluation of representations, no single representation meets all the 16 requirements. If our analysis is right, the proposed multi-representation method would address this issue. Nevertheless, the evaluation result shows several issues that need to be aware of when developing a new representation method.

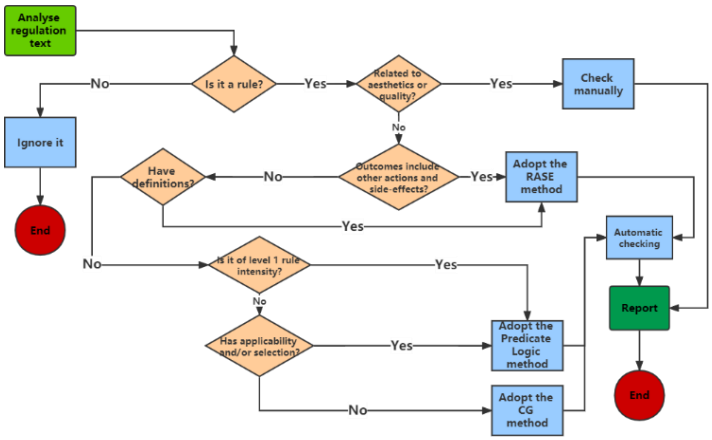

Figure 2. Selection diagram for the multi-representation method

1) The neglection of the broader context of rules

As shown in Table 2, only two methods can deal with definitions. Most methods lack the consideration of including the broader context such as regulation document titles, section descriptive sentences and definitions. However, these texts provide valuable context such as the applicability of rule provisions.

2) Only considering binary outcomes

Existing representations showed a general inability to represent outcomes other than "pass" and "fail" (except RASE). As the regulations become more complex, it is vital that representations are equipped with "unknown", other actions and side-effects to represent as many types of rules as possible.

3) Only dealing with rules with low intensity

In the current research and practices, scholars and practitioners seem to rush to develop an ACC system for proof-of-concept or implementation. For clearer demonstration and easier understanding, only rules with relatively low intensity were selected. This may result in the underestimation of rule intensity (Solihin & Eastman, 2015) or sacrifice the completeness of ACC system. Which rules require manual checking are often decided arbitrarily without sufficient evidence. The authors' experience shows that rules that are seemingly "uncheckable" by ACC systems can be checkable when being carefully interpreted.

4) The lack of attention to rule organization

Rule organization includes hierarchy and cross-references. While many methods have the capability to present cross-references, they rarely consider hierarchy of rules. Hierarchies may affect checking outcomes as they denote different constraint levels of

regulation documents and the superiority and inferiority of regulations.

5) The dependency on model data and rule engine

It seems that because BIM model is the most commonly used data model, most representations use objects, attributes and relationships in IFC directly to avoid the need to map among different ontologies. As a result, although ACC can be done, the representation is dependent on rule engine and building model data. This limits the expressiveness of representations and makes the representations difficult to be updated when regulations are revised.

7 CONCLUSION

This paper evaluated existing representation methods for building rules. It proposed a new classification method of rules and developed a multi-representation method to represent different types of rules. To the best of the authors' knowledge, this is the first research that mapped different types of rules to their suitable representation and proposed a representation method that meets all of the required capabilities. The method can help check more rules automatically, thereby improving the quality and efficiency of the design review process.

This paper inevitably has limitations. Firstly, more representation methods can be evaluated, and more suitable representations may be found. Secondly, the evaluation results might not be 100% accurate, as 1) representations were evaluated based on the descriptions in corresponding papers without implementing the method; and 2) many representations are still being developed. Thirdly, the method is proposed but not tested. The testing of this proposed representation will be included in our future research.

8 REFERENCES

Beach, T. H., Rezgui, Y., Li, H. & Kasim, T. 2015. A rule-based semantic approach for automated regulatory compliance in the construction sector. *Expert Systems with Applications* 42: 5219-5231.

Building Research Establishment 2018. BREEAM New Construction 2018 (UK).

Department of Health and Social Care 2017. DH health building notes. *In:* CARE, D. O. H. A. S. (ed.).

Eastman, C., Lee, J.-m., Jeong, Y.-s. & Lee, J.-k. 2009. Automatic rule-based checking of building designs. *Automation in Construction* 18: 1011-1033.

Hjelseth, E. & Nisbet, N. Capturing normative constraints by use of the semantic mark-up RASE methodology. 2011 2011. 1-10.

Hustadt, U. Do we need the closed world assumption in knowledge representation? Knowledge Representation Meets Databases, 1994 1994.

Macit İlal, S. & Günaydın, H. M. 2017. Computer representation of building codes for automated compliance checking. *Automation in Construction* 82: 43-58.

Ministry of Housing, C. L. G. 2010. Approved Documents.

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van de Walle, R. & Van Campenhout, J. 2011. A semantic rule checking environment for building performance checking. *Automation in Construction* 20: 506-518.

Preidel, C. & Borrmann, A. 2016. Towards code compliance checking on the basis of a visual programming language. *Journal of Information Technology in Construction* 21: 402-421.

Rasdorf, W. J. & Lakmazaheri, S. 1990. Logic-based approach for modeling organization of design standards. *Journal of computing in civil engineering* 4: 102-123.

Solibri. 2022. *Solibri Model Checker* [Online]. Available: https://www.solibri.com/solibri-office [Accessed 31/08/2021 2021].

Solihin, W. Lessons learned from experience of code-checking implementation in Singapore. BuildingSMART Conference, 2004 Singapore. BuildingSMART.

Solihin, W. & Eastman, C. 2015. Classification of rules for automated BIM rule checking development. *Automation in Construction* 53: 69-82.

Solihin, W. & Eastman, C. 2016. A knowledge representation approach in BIM rule requirement analysis using the conceptual graph. *Journal of Information Technology in Construction* 21: 370-402.

Soliman-Junior, J., Formoso, C. T. & Tzortzopoulos, P. 2020. A semantic-based framework for automated rule checking in healthcare construction projects. *Canadian Journal of Civil Engineering* 47: 202-214.

Sowa, J. F. 1976. Conceptual Graphs for a Data Base Interface. *IBM Journal of Research and Development* 20: 336-357.

Tan, X., Hammad, A. & Fazio, P. 2010. Automated Code Compliance Checking for Building Envelope Design. *Journal of Computing in Civil Engineering* 24: 203-211.

Vaishnavi, V. K. 2007. *Design science research methods and patterns: innovating information and communication technology.* Auerbach Publications.

Zhang, J. & El-Gohary Nora, M. 2017. Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking. *Journal of Computing in Civil Engineering* 31: 04016037.

Zhang, Z., Ma, L. & Broyd, T. 2022. Towards Fully-automated Code Compliance Checking Of Building Regulations: Challenges For Rule Interpretation And Representation. *2022 European Conference on Computing in Construction.* Ixia, Rhodes, Greece.

Zhang, Z., Nisbet, N., Ma, L. & Broyd, T. [No date]. Capabilities of rule representations for automatic compliance checking. Unpublished.