Suggested Citation: Yang, F., Zhang, J., Chen, Y., and Debs, L. (2022). "A new schema of logic representation and reasoning for automated building code compliance checking." Proc., Polytechnic Summit 2022, Darmstadt University of Applied Sciences, Darmstadt, Germany, 72-84.
For Final Published Version, please locate it at the 2022 GPEA Polytechnic Summit Proceedings here: https://arrow.tudublin.ie/eutpresscon/2/

A New Schema of Logic Representation and Reasoning for Automated Building Code Compliance Checking

Fan Yang[1[0000-0001-9842-719X]], Jiansong Zhang[1*[0000-0002-3638-1947]], Yunfeng Chen[1[0000-0002-0108-8484]], and Luciana Debs[1[0000-0002-9713-0957]]

[1] School of Construction Management Technology, Purdue University, West Lafayette IN 47907, USA
zhan3062@purdue.edu

Abstract: Manual building code compliance checking is a time-consuming, labor-intensive and error-prone process. Automated logic-based reasoning is an essential step in the automation of this process. There have been previous studies using logic programming languages for automated logic-based reasoning to support automated compliance checking (ACC) of building designs with building codes. As a high-performance implementation of the standard logic programming language, B-Prolog was widely used in these studies. However, due to the support of dynamic predicates and user-defined operators, the predicates' functions vary according to different user definitions; therefore, B-Prolog is sometimes not reliable for building code reasoning. As a more expressive, scalable, and reliable alterative to B-Prolog, Picat, a logic-based multi-paradigm programming language, provides a new and potentially more powerful platform for automated logic-based reasoning in ACC. To explore the potential value of Picat in ACC, in this study, the authors compared Picat and B-Prolog performance in automatically checking 20 requirement rules in the 2015 International Building Code. The experimental results showed that the automated checking for building codes in the B-Prolog version was faster than that in the Picat version, whereas the Picat version was more reliable than the B-Prolog version. This could be the result of B-Prolog using unification and Picat using pattern matching for indexing rules. More potential applications of Picat in ACC domain need further research. Furthermore, this schema could be used in the teaching of ACC to graduate construction students, illustrating the need to focus on the reliability, predictability and scalability of the process, in order to provide a practical solution to improving code compliance checking processes.

Keywords: Logic representation · Automated reasoning · Automated compliance checking · Building code · Artificial intelligence · Intelligent systems.

Suggested Citation: Yang, F., Zhang, J., Chen, Y., and Debs, L. (2022). "A new schema of logic representation and reasoning for automated building code compliance checking." Proc., Polytechnic Summit 2022, Darmstadt University of Applied Sciences, Darmstadt, Germany, 72-84.
For Final Published Version, please locate it at the 2022 GPEA Polytechnic Summit Proceedings here: https://arrow.tudublin.ie/eutpresscon/2/

1 Introduction

Building design and construction activities must meet applicable requirements in building codes to assure the safety and well-being of construction workers and end users (e.g., occupants). Building code compliance checking in the past has relied heavily on manual efforts and the experience, skills, and judgment of building professionals, which is a time-consuming, labor-intensive and error-prone process [1]. The ongoing development of computing technology, especially the emergence of Building Information Modeling (BIM), provides a great opportunity to the automation of building code compliance checking, which is expected to improve the efficiency and accuracy of such checking. As a digital representation of the entire life cycle of a building, BIM not only provides a platform for timely information sharing for all parties involved in the construction process (such as builders, designers, and owners), but also enables the transformation of physical characteristics of buildings into computer-interpretable digital representations. This provides a solid basis for automated building code compliance checking. By converting building models into an international standard format, i.e., Industry Foundation Classes (IFC), and expressing building code requirements as compliance checking rules that can be executed by a computer, the rules can be used to automatically check building design information in the IFC-based BIM, enabling automated compliance checking of building designs.

Logic representation and automated reasoning is an essential step in automated compliance checking of building designs with building codes. Because the binary nature (True/False) of the smallest reasoning unit of first-order logic (FOL) naturally fits the representation of expected result (compliance/noncompliance) of automated compliance checking, and FOL enables fully automated reasoning, it is therefore commonly used in the logic representation and automatic reasoning of building codes [2,3]. B-Prolog, a logic programming language developed based on FOL, was adopted by previous studies due to its high performance [1,4,5]. However, due to the support of dynamic predicates and user-defined operators, the predicates' functions in B-Prolog vary according to different user definitions, which makes the code requirement reasoning unreliable sometimes. Also, the cessation of active development of B-Prolog limits its future applications in the ACC field. At the same time, Picat, a multi-paradigm programming language developed based on and served as a successor of B-Prolog, is the logical replacement and upgrade for B-Prolog. Compared to B-Prolog, Picat is more expressive, scalable, and reliable because it incorporates many features of both declarative language and imperative language [6]. These advantages of Picat provide a solid basis for its potential application in the ACC, so there is a need to explore the application of Picat in the ACC domain.

Based on the previous ACC research findings leveraging B-Prolog, in this study, the authors propose a new logic-based Information Representation and Reasoning (InfoRR) schema tailored to Picat. The proposed schema was used to implement logic representation and automated reasoning for checking 20 building requirement rules in the 2015 International Building Code [7]. The running times of the B-Prolog and Picat versions of implementing the proposed schema were compared and analyzed. The research presented in this paper provides (1) a foundation for future research on the logic representation and automated reasoning of building codes in the ACC field using advanced logic programming platforms such as Picat, and (2) a schema that can be used to teach graduate level construction courses in automated code checking, considering the practical needs of municipalities in terms of reliability of the results, aligned with the scalability and predictability of ACC processes.

2 Background

2.1 Automated Compliance Checking

Automated compliance checking uses computing technology to check the compliance of building designs with applicable building codes in a way that is more accurate, efficient and economical compared to manual checking. Automated compliance checking is mainly built upon three parts: the digital representation of building designs, the computer-interpretable representation of building code requirements, and the automated compliance reasoning mechanism. Previous studies have investigated these three aspects extensively [4,8,9]. The following paragraphs mainly introduce the research work related to the computer-interpretable representation of building regulations.

Beach et al. and Rosenman & Gero proposed rule-based modeling methods to represent building codes [10,11]. Malsane et al. and İlal & Günaydın used object-oriented modeling to represent building codes [9,12]. In addition, some modeling methods based on the combination of ontology and natural language processing (NLP) have recently been proposed to achieve automated or semi-automated information extraction of building regulations [4,13].

Automated compliance checking systems were created and developed in different countries to enable efficient management of building construction projects, such as CORENET funded by Singapore's Ministry of National Development [14], Statsbygg Solibri system developed by Standards Norway and Norwegian BuildingSMART [15], SMARTcodes driven by the International Code Council (ICC) [16], and DesignCheck [15].

Although different systems and methods have been proposed in the past to pursue automated compliance checking, there are several limitations in the existing work: first, the existing logic-based representation methods are overly sensitive to the expressions

of different types of information in building regulations, such as element definitions and quantitative requirements for a building element. Second, with the revision and update of building regulations, the update of the electronic representation and corresponding checking logics imposed high requirements on the computing skills of users. Third, building codes vary greatly among different countries and regions, so high demands are placed on the scalability and comprehensiveness of the logic representation method of an ACC system that tries to be widely applicable. Fourth, the existing automated compliance checking systems need some manual effort and support, and have not achieved full automation. In this paper, the authors aim to help address the first limitation.

2.2 Logic Programming Language

Logic programming languages use logic facts to represent facts that already exist and define logic rules to conduct inferences based on the logic facts. As a declarative language, it can be used to represent knowledge and allows machines to automatically make inferences, so it is widely used in the field of artificial intelligence. Common logic programming languages include Prolog, Answer Set Programming (ASP) and Datalog. Among them, Prolog is the most commonly used logic programming language. B-Prolog, as a high-performance implementation of Prolog, provides an efficient and versatile logic programming system. It has been used to support different tasks in the building construction domain such as building code compliance checking [5] and modular construction analysis with robotics automation [17, 18]. There are three types of logic statements in B-Prolog: facts, rules, and directives. Logic facts are defined in the form of "*p(arg1,arg2,...,argn)*.", where *p* is the name of the predicate, and *arg1,arg2,...,argn* are the arguments of the predicate. After the logic facts are defined, they become the basis of machine reasoning in the logic programs. Logic rules are defined in the form of "*H:-B1,B2,...,Bn.*", where *H* is the head of the rule, *B1, B2,...,Bn* represent the body of the rule, and *:-* is the operator for implication (i.e., it separates the head from the body of a rule). The head of a rule is considered to be true when its body part is evaluated to true. Logic rules define the rules for machine reasoning, and can also be used as the constraint of reasoning. Therefore, B-Prolog is very suitable for solving constrained optimization problems. Directives are defined in the form of *":- B1,B2,...,Bn.*", which are mainly used to query the pragmatic information of the logic facts through the inference process [19]. Built upon B-Prolog, Picat was created and developed with logic programming concepts at the core. Picat builds a bridge between declarative language and imperative language because it incorporates many features of a declarative language, such as explicit unification, list comprehensions, constraints, as well as many features from an imperative language, such as assignments and loops [6].

Suggested Citation: Yang, F., Zhang, J., Chen, Y., and Debs, L. (2022). "A new schema of logic representation and reasoning for automated building code compliance checking." Proc., Polytechnic Summit 2022, Darmstadt University of Applied Sciences, Darmstadt, Germany, 72-84.
For Final Published Version, please locate it at the 2022 GPEA Polytechnic Summit Proceedings here: https://arrow.tudublin.ie/eutpresscon/2/

This makes Picat a simple and powerful language for various applications. Table 1 comparatively illustrates the syntaxes of B-Prolog and Picat used in this study.

Table 1. Syntaxes of B-Prolog and Picat

Meaning	B-Prolog	Picat
Conjunction	,	, or &&
Disjunction	;	; or ||
Negation	not or \+	not or \+
Implication	:-	=>
Predicate	predname(arg1,arg2,…,argn).	predname(arg1,arg2,…,argn).
Rule	predh(arg1,arg2,…,argn):- pred1(arg1,arg2,…,argn), pred2(arg1,arg2,…,argn),…, predm(arg1,arg2,…,argn).	Head,Cond=>Body. (Each takes the form predname(arg1,arg2,…,argn))
Function	fun(arg1,arg2,…, argn).	fun(arg1,arg2,…,argn) = return_value.
Foreach loop	foreach(E1 in D1, …, En in Dn, LocalVars, Goal).	foreach(E1 in D1, Cond1, ..., En in Dn, Condn) Goal end.

If-then	Cond -> Goal; Goalelse.	if Cond1 then
		Goal1
		elseif Cond2 then
		Goal2
		⋮
		elseif Condn then
		Goaln
		else
		Goalelse
		end.

3 Proposed Information Representation and Reasoning (InfoRR) Schema

Based on existing logic-based information representation and reasoning schemas proposed in previous research [4,5], in this study, the authors proposed a new information representation and reasoning (InfoRR) schema to leverage the most recent advancement in logic programming for automated building code compliance checking. Compared to

the previous schemas in literature, the proposed new schema divided the information representation into two parts: (1) the fundamental information elements, and (2) requirements, which were represented and stored in the form of logic facts and logic rules respectively. Fundamental information elements in this schema included instances that describe design information and building code concepts, and both were represented and stored in form of logic facts. The separate storage of logic facts and logic rules in the new schema facilitates information searching, reasoning, and compatibility with different logic programming implementations. Because both building code concepts and design information instances are represented in the form of logic facts, machines (i.e., logic reasoners) can automatically match and reason between the two under suitably defined logic rules, thereby outputting whether an instance of design information meets corresponding requirements of building regulations. In addition, the proposed new schema added secondary functions to obtain the running time of the codes, facilitating the comparison of different implementations. The structure of the proposed InfoRR schema is shown in Figure 1. Figure 2 shows the compliance checking process of building design information (e.g., Instance 1) to regulatory information (e.g., Logic Rule 1) under the proposed InfoRR schema. In this process, activation of logic rules is achieved through functional calls. Once a logic rule is activated, the applicability of the instance is checked by unification or pattern matching mechanism.

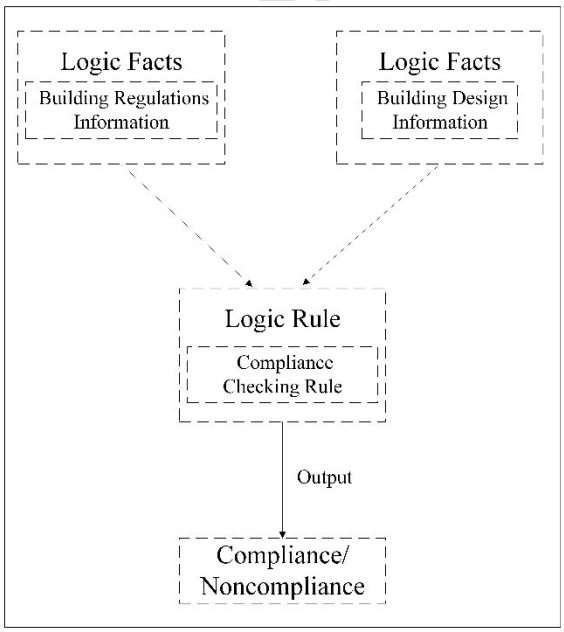

Figure 1. The proposed InfoRR schema

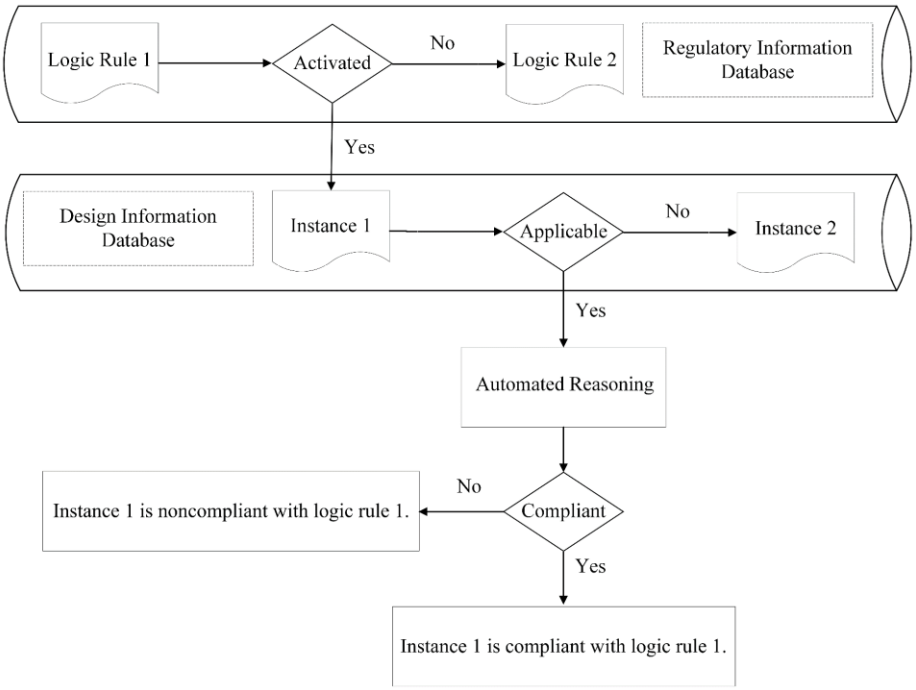

Figure 2. The compliance checking process of the proposed InfoRR schema

4 Experiment

The authors selected 20 building code requirements from Chapter 9 ("Fire Protection System") of the 2015 International Building Code [7] as the experimental basis. These requirements are mainly concerning the dimensions or quantities of fire protection equipment or related building elements. For each building requirement, the authors expressed the fundamental information elements with logic facts, wrote a corresponding compliance checking rule, and designed multiple building design instances that either comply with or violate the rule. Based on the logical analysis of the requirements, some rules may have more than one way of being violated, in which case multiple noncompliant instances would be created. The logic statements were implemented in both B-Prolog and Picat. It should be noted that logic-based reasoning can be implemented based on two assumptions: the open-world assumption and the closed-world

assumption. In the open-world assumption, true information is explicitly represented, and any information that is not known to be true is treated as unknown. In contrast, in the closed-world assumption, any information that is not known to be true is treated as false. According to previous research [5], logic-based reasoning based on the closed-world assumption is more suitable for automated compliance checking applications, because it can achieve higher recall in noncompliance detection compared to that based on the open-world assumption, which means that the probability of missing non-compliant instances will be lower. Therefore, the experiments in this study were based on the closed-world assumption.

Secondary functions were defined and used during the implementations to provide functional support. For example, the built-in time counting function was used to record the running time of the program, and other functions were defined to implement unit conversions and quantity comparisons, etc. In addition, a loop function was defined in the main program to run the program ten times and record the running time of each time, to reduce the influence of random errors on the time measurement results. The two versions of the program were written and run in the respective executable files of B-Prolog and Picat, respectively, and output the checking results of 47 instances. The distribution of 47 instances was summarized in Table 2. For example, the 2015 International Building Code requires that the minimum dimension of exterior wall openings should be no less than 30 inches (762cm) (Chapter 9 Provision [F] 903.2.11.1.1) [7]. In this study, an exterior wall opening with a dimension greater than 30 inches and an exterior wall opening with a dimension less than 30 inches were designed to comply with and violate this rule, respectively, to test the effectiveness of the corresponding compliance checking rule. Five of the selected requirements have two possible violation scenarios of the rule, so the authors designed 2 noncompliant instances for each of these requirements. Chapter 9 Provision [F] 904.12.1 of the 2015 International Building Code (i.e., "The manual actuation device shall be installed not more than 48 inches (1200 mm) or less than 42 inches (1067 mm) above the floor and shall clearly identify the hazard protected.") [7] is an example of such requirement. Similarly, there is one requirement that has 3 noncompliant instances as shown in Table 2 below.

Table 2. A summary of the distribution of 47 instances

Number of requirements	Number of compliant instances	Number of noncompliant instances	Total
14	1	1	28
5	1	2	15
1	1	3	4

5 Results and Discussions

The proposed InfoRR schema was successfully implemented in B-Prolog and Picat, and output compliance results for all instances correctly. In other words, both B-Prolog and Picat versions achieved 100% accuracy. Figures 3 through 6 showed the screenshots of part of code implementations and corresponding checking results of design instances corresponding to the same rule in B-Prolog and Picat. However, there was a significant difference in their running times. It is evident from Figure 7 that the running time of the B-Prolog version was about half of that of the Picat version. According to Zhou et al. [6], the Picat version is supposed to be at least as fast as the B-Prolog version because "the Picat compiler translates loops and list comprehensions into tail recursion, which is further converted into iteration by tail-recursion optimization." A tail recursion is a special form of recursion where the function calls itself at the end of the function. The Picat compiler can avoid allocating a new stack frame for a function through tail-recursion optimization, thus consuming less memory space. The results in this study seem contradictory to Zhou et al. [6]. However, the conclusion of Zhou et al. took the implementation of Picat and B-Prolog on matrix multiplication as an example, and did not involve the field of ACC [6]. At the same time, tail recursion was not used in the code implementations in this study, which could be one of the reasons why the Picat version was slower than the B-Prolog version. On the other hand, B-Prolog adopted unification to select the applicable logic rule for a call, whereas Picat used pattern-matching to execute logic rules. Pattern matching-based logic rules were fully indexed in Picat, whereas B-Prolog clauses usually indexed only one argument [6]. This difference could be another reason why the Picat version was slower than the B-Prolog version. However, the fully indexed feature gave Picat more scalability, which is reflected in the fact that the order of the logic clause elements that represent building code requirements did not affect the execution of the compliance checking in Picat. In B-Prolog, however, changing the order of (building code) regulatory information elements in a compliance checking rule could cause the rule to fail (to check). Figures 8 through 11 showed a comparison of the codes and outputs after changing the order of the regulatory information elements in the checking rule in B-Prolog. After the order of some logic clause elements was changed (e.g., "*greater_than(Covered_kiosks_displays_booths_concession_stands_or_equipment,quantity12)*" was changed to be the first element in this case), the program outputted an empty result, without explicitly indicating whether the instances comply with or violate this regulatory requirement (see Figure 11). In contrast, the change of order of the logic clause elements had no effect on the functionality of checking rules in Picat and the program could still output compliance results of instances after the changes, which showed that Picat version was more reliable and scalable.

Suggested Citation: Yang, F., Zhang, J., Chen, Y., and Debs, L. (2022). "A new schema of logic representation and reasoning for automated building code compliance checking." Proc., Polytechnic Summit 2022, Darmstadt University of Applied Sciences, Darmstadt, Germany, 72-84.
For Final Published Version, please locate it at the 2022 GPEA Polytechnic Summit Proceedings here: https://arrow.tudublin.ie/eutpresscon/2/

```
expect_yes8:-
findall((Openings),(openings(Openings),have
(Openings,A_minimum_dimension),greater_than_or_eq
ual(A_minimum_dimension,quantity8),quantity
(A_minimum_dimension)),Xs)...
```

Figure 3. Part of code from the B-Prolog implementation of InfoRR

```
| ?- expect_yes8
openings8_1, is, compliant, with, section, 903, 2, 11, 1,
1, in, ibc, year2015
```

Figure 4. Sample output from the B-Prolog implementation of InfoRR

```
expect_yes8=>
findall((Openings),(openings(Openings),have
(Openings,A_minimum_dimension),greater_than_or_eq
ual(A_minimum_dimension,quantity8),quantity
(A_minimum_dimension))) = Xs...
```

Figure 5. Part of code from the Picat implementation of InfoRR

```
Picat> expect_yes8
openings8_1, is, compliant, with, section, 903, 2, 11, 1,
1, in, ibc, year2015
```

Figure 6. Sample output from the Picat implementation of InfoRR

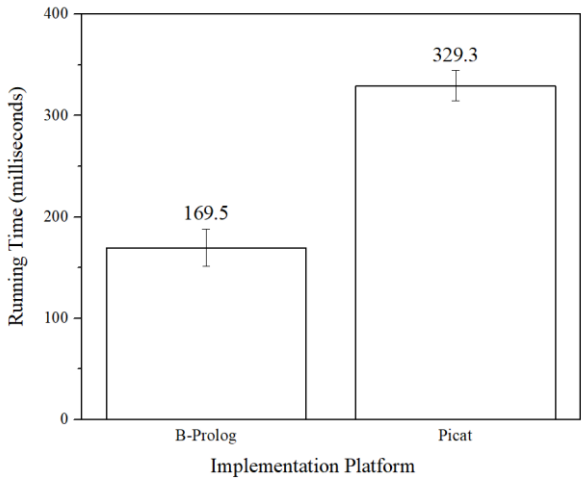

Figure 7. The runtime of the B-Prolog and Picat versions of the experiment

Suggested Citation: Yang, F., Zhang, J., Chen, Y., and Debs, L. (2022). "A new schema of logic representation and reasoning for automated building code compliance checking." Proc., Polytechnic Summit 2022, Darmstadt University of Applied Sciences, Darmstadt, Germany, 72-84.
For Final Published Version, please locate it at the 2022 GPEA Polytechnic Summit Proceedings here: https://arrow.tudublin.ie/eutpresscon/2/

```
expect_yes12:-
findall((Automatic_sprinklers),
(automatic_sprinklers
(Automatic_sprinklers),installed
(Automatic_sprinklers),in_or_under
(Automatic_sprinklers,Covered_kiosks_displays_boo
ths_concession_stands_or_equipment),greater_than
(Covered_kiosks_displays_booths_concession_stands
_or_equipment,quantity12),in_width
(in_width)),Xs)...
```

Figure 8. Part of the code in B-Prolog

```
| ?- expect_yes12
automatic_sprinklers12_1,is,compliant,with,sectio
n,903,3,3,in,ibc,year2015
```

Figure 9. Sample output from the B-Prolog code in normal order

```
expect_yes12:-
findall((Automatic_sprinklers),(greater_than
(Covered_kiosks_displays_booths_concession_stands
_or_equipment,quantity12),in_or_under
(Automatic_sprinklers,Covered_kiosks_displays_boo
ths_concession_stands_or_equipment),automatic_spr
inklers(Automatic_sprinklers),installed
(Automatic_sprinklers),in_width(in_width)),Xs)...
```

Figure 10. Part of the code in B-Prolog after changing the order of the regulatory information elements

```
| ?- expect_yes12

yes
```

Figure 11. Sample output from the B-Prolog code after changing the order

6 Contributions to the Body of Knowledge

This study contributes to the body of knowledge in three main ways. First, a new information representation and reasoning schema was proposed to harness the power of recent advancement in logic programming while facilitating the comparison of different implementation methods. Two logic programming languages (i.e., B-Prolog and Picat) were used for ACC implementation under this schema, which showed that the proposed schema was robust and scalable. Second, this study demonstrated that Picat was a more feasible implementation method for information representation and automated reasoning of ACC. Although Picat-based implementation ran slower than B-Prolog-based implementation currently, the former was less sensitive to the expression of different types

of regulatory information, and thus more reliable in the application of ACC. Last but not least, this research revealed the implications of the different mechanisms behind the different experimental results of the two implementations, and laid a solid foundation for future work on logic programming-based information representation and reasoning for ACC and for automation tasks in the architecture, engineering, and construction domain in general.

7 Implications for Teaching

Although much research has been published on ACC processes such as [16,17], with ongoing effort towards its improvement, little has been discussed about how to teach students about this process. In general, due to the complex nature and interdisciplinarity of this process, graduate students in advanced courses may be more exposed to this topic. To this end, our experiment and our findings can be used as a basis for graduate construction students researching ACC. Our motivation is to improve the ACC processes using a practice-based approach. Therefore, by considering issues of reliability of the results, along with the scalability and predictability of the process and comparing different programming languages available, our experiment can be used as a case study to discuss user-centered research and automation in construction.

8 Limitations and Future Work

The authors acknowledge the following limitations of this work. First, logic representation of regulatory information was implemented manually in this study. This process varies as individuals may understand and express the same building code requirement differently. Advanced artificial intelligence technologies, such as Natural Language Processing (NLP), could be helpful for the automation of this encoding process and reduction of the influence of individual differences and subjectivity. Second, the proposed InfoRR schema was only tested on dimensional and quantitative requirements in Chapter 9 of the 2015 International Building Code [7], and did not cover the other requirements of this building code. In order to represent and reason about other types of regulatory information, such as exceptions to a certain requirement and associations between different requirements, more modules need to be developed and embedded in the schema. Due to the complexity of logic relationship in building codes, the combination of multiple paradigm languages may need to be considered in the future. Third, the scalability and reliability of Picat were demonstrated through experiments in this paper, but the Picat-based implementation was not yet optimized with regard to its

running speed. Full applications of Picat with optimized performance in ACC need to be further investigated in future research.

9 Conclusions

The logic representation and automated reasoning of building codes and regulations are an important process in automated compliance checking. Based on previous work on B-Prolog, this study proposed an InfoRR schema applicable to Picat, and successfully applied the schema to the logic representation and automated reasoning of 20 building requirement rules in the 2015 International Building Code [7] and 47 building design instances. The proposed schema could accurately output compliance results for all design instances checked. Compared to the B-Prolog version, the Picat version took longer to run, but was less sensitive to the order of regulatory information elements in the compliance checking rules, meaning that Picat was more scalable and reliable in supporting ACC. One reason behind the test results could be that B-Prolog uses unification to select an applicable rule for a call, which usually only indexes one argument, whereas Picat uses pattern matching to call the logic rules, which fully indexed all rules. The result in this study provides preliminary conclusions for the potential application of Picat in ACC compared to B-Prolog. More applications and tests of Picat in ACC need to be explored in future research.

Acknowledgements

The authors would like to thank the National Science Foundation (NSF). This material is based on work supported by the NSF under Grant No. 1827733. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Zhang J, El-Gohary N M: Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. Automation in Construction, 73, 45–57 (2017).
2. Halpern J Y, Weissman V: Using First-Order Logic to Reason about Policies. ACM Transactions on Information and System Security, 11(4), 21:1-21:41 (2008).
3. Kerrigan S, Law K H: Logic-based regulation compliance-assistance. Proceedings of the 9th international conference on Artificial intelligence and law - ICAIL '03. Scotland, United Kingdom: ACM Press: 126, (2003).
4. Zhang J, El-Gohary N M: Automated Information Transformation for Automated Regulatory Compliance Checking in Construction. Journal of Computing in Civil Engineering, American

Society of Civil Engineers, 29(4), B4015001 (2015).
5. Zhang J, El-Gohary N M: Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking. Journal of Computing in Civil Engineering, 31(1), 04016037 (2017).
6. Zhou N-F, Kjellerstrand H, Fruhman J: Constraint Solving and Planning with Picat. Cham: Springer International Publishing (2015).
7. International Code Council. International Building Code 2015 IBC. Country Club Hills, Ill: International Code Council (2014).
8. Zhou P, El-Gohary N: Domain-specific hierarchical text classification for supporting automated environmental compliance checking. Journal of Computing in Civil Engineering, American Society of Civil Engineers, 30(4), 04015057 (2016).
9. Malsane S, Matthews J, Lockley S, et al: Development of an object model for automated compliance checking. Automation in Construction, 49, 51–58 (2015).
10. Beach T H, Rezgui Y, Li H, et al: A rule-based semantic approach for automated regulatory compliance in the construction sector. Expert Systems with Applications, 42(12), 5219–5231 (2015).
11. Rosenman M A, Gero J S: Design codes as expert systems. Computer-Aided Design, 17(9), 399–409 (1985).
12. Macit İlal S, Günaydın H M: Computer representation of building codes for automated compliance checking. Automation in Construction, 82, 43–58 (2017).
13. Xu X, Cai H: Semantic approach to compliance checking of underground utilities. Automation in Construction, 109, 103006 (2020).
14. Foo S T, Zhong Q: Construction and Real Estate NETwork (CORENET). Facilities, MCB UP Ltd, 19(11/12), 419–428 (2001).
15. Eastman C, Lee J, Jeong Y, et al: Automatic rule-based checking of building designs. Automation in Construction, 18(8), 1011–1033 (2009).
16. Nawari N O: SmartCodes and BIM. American Society of Civil Engineers, 928–937 (2013).
17. Wong Chong O, Zhang J: Logic representation and reasoning for automated BIM analysis to support automation in offsite construction. Automation in Construction, 129, 103756(2021).
18. Wong Chong O, Zhang J, Voyles RM, et al: A BIM-based approach to simulate construction robotics in the assembly process of wood frames to support offsite construction automation. Automation in Construction, 137, 104194 (2022).
197. Zhou N-F: B-Prolog user's manual. Version, 7, pp. 1994–2012 (1994).