A Prototype System for Fully Automated Code Checking

Jiansong Zhang[1] and Nora M. El-Gohary[5]

1) Ph.D., Assistant Professor, Department of Civil and Construction Engineering, Western Michigan University, Kalamazoo, MI, USA. Email: jiansong.zhang@wmich.edu
2) Ph.D., Assistant Professor, Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA. Email: gohary@illinois.edu

Abstract:

A number of important efforts have been made, both in academia and industry, to automate the code compliance checking process. While important progress has been achieved, existing automated compliance checking (ACC) systems are not entirely automated; manual effort is needed for extracting regulatory requirements from codes and encoding them in a computer processable rule format. To address this gap, this paper presents a new ACC system that leverages semantic natural language processing (NLP) techniques to automatically extract information from regulatory documents and represent them in a logic based rule format. The proposed systems is composed of three modules: (1) semantic NLP based regulatory information extraction and transformation algorithms to automatically extract regulatory information from regulatory documents into semantic tuples and automatically transform the extracted regulatory information in semantic tuples into logic rules; (2) semantic rule based design information extraction and transformation algorithms to automatically extract design information from building information models and transform the design information into logic facts; and (3) a logic-based reasoning algorithm to automatically reason about the logic rules and facts and generate compliance checking reports. To test the performance of the proposed system, as a whole, and how the different algorithms are performing together as one unit for detecting noncompliance instances, the proposed algorithms were implemented and integrated into a proof-of-concept prototype. The prototype was tested in checking the compliance of a design test case with Chapter 19 of the International Building Code 2009, and the performance was evaluated in terms of precision and recall.

Keywords: Automated compliance checking, natural language processing, building information modeling, logic-based reasoning.

1. INTRODUCTION

Building construction projects must comply with a multitude of regulations. The manual process of regulatory compliance checking is time consuming, costly, and error prone, similar to other manual processes (Boken & Callaghan, 2009). In comparison to the manual compliance checking, automated compliance checking (ACC) of building construction projects is expected to reduce the time, cost, and errors of the compliance checking task (Nguyen & Kim, 2011; Beach et al., 2013; Zhang & El-Gohary, 2013). Due to the anticipated benefits of ACC, many efforts were undertaken in the area of ACC in building construction. A good review of recent efforts in ACC is found in Eastman et al. (2009). These ACC efforts took various approaches, such as using different rules and programs to represent regulatory information, and using different models to represent design information.

Previous research and software development efforts paved the way for ACC in the architectural, engineering, and construction (AEC) industry. However, the state-of-the-art development in ACC still requires manual extraction of regulatory provisions/requirements, from textual regulatory documents, and encoding of these provisions/requirements into a computer-processable rule format. For example, Solibri Model Checker (Corke 2013) currently includes a set of 300 proforma-based rules that allow for some degree of user customization of rules. However, such customization does not allow for the creation of new rules. The development of new rules in Solibri Model Checker requires professional software engineering expertise and deep understanding of the software's environment and data structure (Corke 2013). The software tools developed by OptaSoft for ACC with ICC codes need major manual data entry and navigation (OptaSoft 2014). Such manually coded rules could be very effective in compliance reasoning with a specific set of requirement rules. However, they are difficult to scale up to the abundant amount of requirements from a multitude of regulations that are constantly changing. To enhance the scalability of rule development, some formalized representations of regulatory information were developed, with the intent to be generalized and flexible. Two of such representative efforts are the Requirement, Applies, Select, and Exception (RASE) method (Hjelseth & Nisbet, 2011) and the conformity-checking ontology (Yurchyshyna et al., 2008). These efforts improved the flexibility and generalization of regulatory information representation. However, their degree of automation in terms of regulatory information extraction and transformation are still limited. The extraction of regulatory information into the RASE representation and the conformity-checking ontology are still manually conducted.

To address these knowledge gaps, this study proposes a highly-automated ACC system that: (1) utilizes semantic natural language processing (NLP) and rule-based techniques to automatically extract and transform both regulatory information and design information for automated reasoning; and (2) utilizes a logic-based representation to represent both regulatory information and design information that could be automatically checked for compliance. This paper explains the details of the proposed ACC system, and the testing of the system on a test case.

2. BACKGROUND

2.1 Building Information Modeling for ACC

Building information modeling (BIM) is expected to help in various tasks during a building's life cycle, including ACC. Because of the popularity and promising outlook of BIM, many ACC efforts utilize BIM to represent design information, in two ways: industry foundation classes (IFC)-based BIM representation (Melzner et al., 2013; Tan et al., 2010) and proprietary BIM representation (Nguyen & Kim, 2011; Corke, 2013). The use of proprietary BIM representation may be necessary to protect intellectual property, but the "hidden" nature of proprietary BIM representation is not in favor of interoperability. In contrast, IFC is an open standard and facilitates interoperability. In addition, the use of IFC schema has the following three advantages: (1) the IFC schema is designed to cover all subdomains of the AEC industry and all phases of a construction project; (2) the IFC schema is defined using standard STEP description methods, which is the official "Standard for Exchange of Product model data" – ISO 10303; and (3) the IFC schema is registered as an official international standard ISO/IS 16739.

2.2 Natural Language Processing

Natural language processing (NLP) is a subfield of artificial intelligence (AI) that aims to enable computers to understand and process natural language text in a humanlike manner (Cherpas, 1992). NLP tasks take two main types of approaches: a rule-based approach or a machine-learning (ML)-based approach. A rule-based approach utilizes manually drafted rules to process text, whereas a ML-based approach utilizes ML algorithms (e.g., Support Vector Machines, Hidden Markov Models) to process text. In spite that the rule-based approach requires more manual efforts in generating rules in comparison to ML-based approach, the rule-based approach tends to achieve better text processing performance (Crowston et al., 2010). From another perspective, NLP approaches could be classified into shallow approaches and deep approaches, distinguished from each other by their different focus in text processing. If the focus of an NLP approach is on processing partial sentence or specific information, then the NLP approach is regarded as shallow. In contrast, if the focus of an NLP approach is on processing full sentence or entire meaning, then the NLP approach is regarded as deep (Zouaq, 2011). State-of-the-art NLP techniques achieved reasonable performances in shallow NLP tasks. However, deep NLP tasks are still challenging, because successful deep NLP requires elaborated knowledge representation and efficient reasoning about the domain, both of which remain to be challenges in AI (Tierney, 2012). Construction regulations are mostly represented in natural language text, which makes the use of NLP to process construction regulatory information (for ACC) a logical selection.

2.3 Automated Reasoning and Logic Programming

Automated reasoning utilizes the speed and precision of computers to conduct tedious tasks that have to be otherwise conducted by human experts (Konev et al., 2010). This domain has been well developed during the past decades and helped solve problems in mathematical proofs, software system and protocol verification, and knowledge-based reasoning (Konev et al., 2010). Automated theorem provers (i.e., reasoners) started to mature in the late 1960s and had achieved many successes, such as in proving the lattice complements conjecture and solving the Robbins problem in algebra (Harrison, 2007). A lot of effort in automated reasoning focused on first order logic (FOL) (Harrison, 2007), which is one type of predicate logic that has more than one correct and complete proof calculi (Hodges, 2001). Automated reasoning in FOL is most efficient using horn clause (HC), which is one of the most restricted forms of FOL. A HC is composed of a disjunction of literals of which at most one is positive. Built on HC, Prolog is a classic logic programming language to support logic-based automated reasoning (Portoraro, 2011). The advantage of formalizing a problem in logic programming is the automation of the solving process; the knowledge and facts could be well captured and represented by logic clauses. B-Prolog is an extended Prolog system for programming concurrency, constraints, and interactive graphics (Zhou, 2012). A predicate is the basic building block in B-Prolog logic programs. A predicate consists of a predicate symbol and one or more arguments in parenthesis following the predicate symbol. For example, "room(R)" is a predicate with the predicate symbol "room" and one predicate argument "R". In the syntax of B-Prolog, names starting with an upper-case letter represent variables, and names starting with a lower-case letter represent constants. The predicate "room(R)" thus declares the variable "R" as an instance of "room". There are three types of logic clauses in B-Prolog: rules, facts, and queries. A logic rule is composed of a rule head and a rule body, both of which are connected by the logical implication symbol ":-". A rule head is a single predicate representing the goal of the rule. A rule body consists of multiple predicates representing the conditions of the rule. For example, “bathroom(R) :- room(R), has_toilet(R).” is a logic rule describing that if “R” is a room and “R” has toilet, then “R” is a bathroom. A logic fact is a special kind of logic rule whose body is always true (thus no need to explicitly specify the body).

3. PROPOSED AUTOMATED COMPLIANCE CHECKING SYSTEM

The proposed system for automated compliance checking (ACC) consists of the following three modules (Figure 1): (1) regulatory information extraction and transformation algorithms for extracting regulatory information from construction regulatory documents (e.g., building codes) into semantic tuples and transforming the extracted regulatory information (in the form of semantic tuples) into logic rules; (2) design information extraction and transformation algorithms for extracting design information from IFC-based building information models (BIMs) into semantic tuples and transforming the extracted design information (in the form of semantic tuples) into logic facts; and (3) compliance reasoning algorithm for reasoning about both the extracted and transformed regulatory information (in the form of logic rules) and design information (in the form of logic facts) to check design compliance with regulations and generate compliance reports. The following subsections describe the details of the modules.

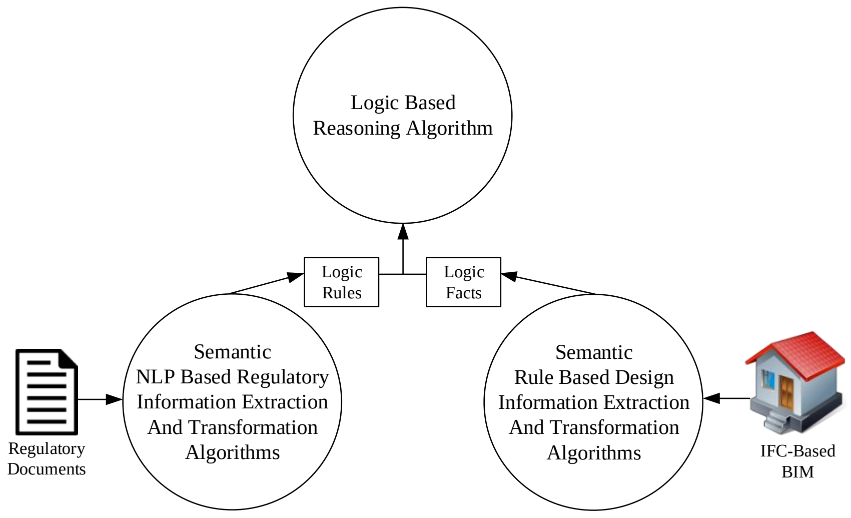

Logic Based
Reasoning Algorithm
Logic Rules
Logic Facts
Semantic NLP Based Regulatory Information Extraction And Transformation Algorithms
Semantic Rule Based Design Information Extraction And Transformation Algorithms
Regulatory Documents
IFC-Based BIM

Figure 1. Proposed Automated Compliance Checking System

3.1 Regulatory Information Extraction and Transformation Algorithms

The regulatory information extraction algorithm extracts regulatory information from regulatory documents (e.g., building codes) into semantic tuples, where each element in a tuple is a semantic information element instance. A semantic, rule-based NLP method is used for automated information extraction, where two types of rules were developed and used: information extraction rules (IE Rules) and conflict resolution rules (CR rules). The IE rules recognize target information for extraction, and the CR rules define the strategy for resolving conflicts in the extraction. Both types of rules are pattern-based. The patterns of IE rules consist of syntactic (syntax/grammar-related) and semantic (meaning/context-related, based on an ontology) features generated on the regulatory text. The patterns of CR rules are based on the relationship between the number of information instances that should be extracted for each semantic information element and the number of information instances that have been extracted by IE rules for that semantic information element.

The regulatory information extraction algorithm used a set of IE and CR rules, which were developed following the method proposed in Zhang & El-Gohary (2013). Chapters 12 and 23 of the International Building Code (IBC) 2006 were used as the development text when developing the IE and CR rules. IBC was selected because it is the most widely adopted building code in the United States. Figure 2 shows a sample set of IE rules and CR rules, along with the information that they extracted from the text. For example, one IE rule for extracting “quantity value” is based on the pattern “CD” “Unit,” where “CD” is the part-of-speech (POS) tag for “cardinal number” and “Unit” is a unit of measure. One IE rule for extracting “subject” is based on the ontology concept “building element,” which extracts all appearances of “building element” and its subconcept in any word form. One CR rule resolves the conflict of missing a “subject” information instance by using the same instance from the nearest sentence/clause (left or right) if those sentences/clauses describe the same content. These IE and CR rules led to the successful extraction of “wall” as a “subject” and “8” as a “quantity value” for the example regulatory requirement *RR1* from Chapter 19 of IBC 2009. The developed IE and CR rules were tested together with other information processing algorithms, as discussed in the experimental testing section.

- *RR1*: "In dwellings assigned to Seismic Design Category D or E, the height of the wall shall not exceed 8 feet, the thickness shall not be less than 71/2 inches, and the wall shall retain no more than 4 feet of unbalanced fill."

The regulatory information transformation algorithm transforms the extracted regulatory information (in the form of semantic tuples) into logic rules. The regulatory information transformation algorithm takes a semantic, rule-based NLP method to enable automation of the transformation. It utilizes pattern-matching-based rules where each pattern consists of a sequence of syntactic and semantic features. Two types of rules are used in the information transformation process: semantic mapping rules (SeM rules) and conflict resolution rules (CoR rules). The SeM rules define how to process the extracted information instances according to their semantic meaning. The semantic meaning of each information instance is defined by the information tag it is associated with and the context of the extracted information instance as reflected by the information tags of its surrounding information instances. The semantic meanings of information instances are utilized in patterns on the left-hand sides (LHS) of the SeM rules. For example, 'n' 'c' 'v' 'u' is used as a pattern for a SeM rule which identifies a sequence of "negation," "comparative relation," "quantity value," and "quantity unit." The CoR rules resolve conflicts between information tags. Two types of CoR rules are used: deletion CoR rules and conversion CoR rules. Deletion CoR rules delete certain information instances to resolve conflicts between information tags. Conversion CoR rules convert information tags of information instances into other types of information tags to resolve conflicts. Based on Chapter 12 and 23 of IBC 2006, a set of 297 SeM rules and 9 CoR rules were developed and used in the regulatory information transformation algorithm. The developed SeM and CoR rules were tested together with other information processing algorithms, as discussed in the experimental testing section.

Text with Features[1]

In dwellings assigned to Seismic Design Category D or E, the height (ontology concept "quantity") of the wall (ontology concept "building element") shall (POS tag "MD" for modal verb) not (gazetteer list "Negation") exceed (POS tag "VB" for base forma verb, gazetteer list "Comparative relation") 8 (POS tag "CD" for cardinal number) feet (gazetteer list "Unit"), ...

IE Rules and CR Rules

IE Rule 2: If "CD + Unit" is matched, extract the text matched with "CD" as an instance of "quantity value."
IE Rule 5: If ontology concept "building element" is matched, extract the matched text as an instance for "subject."
IE Rule 6: If ontology concept "quantity" is matched, extract the matched text as an instance for "compliance checking attribute."
IE Rule 7: If "MD + Negation + VB + Comparative Relation" is matched, extract the text matched with "Negation" and the text matched with "Comparative relation" together as an instance for "comparative relation."
CR Rule 1: If one instance exists for each semantic information element (except for subject restriction and quantity restriction), organize those instances into a tuple for the corresponding quantitative requirement.

Extracted Semantic Information Element Instances:

<Subject: Wall, Subject Restriction: N/A, Compliance Checking Attribute: Height, Deontic Operator Indicator: Obligation, Quantitative Relation: Exceed, Comparative Relation: Less than or equal, Quantity Value: 8, Quantity Unit: Feet, Quantity Restriction: N/A>

Text with Tags[1]

In dwellings assigned to Seismic Design Category D or E, the height ("a" for compliance checking attribute) of ("OF") the wall ("s" for subject) shall ("MD") not ("n" for negation) exceed ("c" for comparative relation) 8 ("v" for quantity value) feet ("u" for quantity unit), …

SeM Rules and CoR Rules

SeM Rule 1: "a" "OF" "s" "MD" "n" "c" "v" "u" -> a(A), s(S), has(S,A), not c(A, quantity(v,u)).

Logic Rules

compliance_Height_of_Wall(Wall) :- height(Height), wall(Wall), has(Wall, Height), not greater_than(Height, quantity(8,feet)).

Note: For simplicity only features related to the IE rules, CR rules, SeM rules and CoR rules are displayed

Figure 2. Sample IE, CR, SeM, CoR rules and their application

3.2 Design Information Extraction and Transformation Algorithms

The design information extraction and transformation algorithms extract design information from IFC-based BIMs into logic facts. Java Standard Data Access Interface (JSDAI) is used to access the entities and attributes in an IFC file of BIMs. A BIM IE algorithm based on JSDAI (Zhang & El-Gohary, 2015) was used to process each entity in an IFC file in a recursive and exhaustive manner. In the developed BIM IE algorithm, recursion is used in two ways: (1) when an entity is being extracted, not only the explicit attributes of the entity are extracted, but all explicit attributes that belong to the supertype of that entity and supertype of supertype (until no supertype can be found) of that entity are extracted too; (2) if an attribute is an aggregation data type (i.e., aggregation of multiple attributes), then the member attributes of the aggregation is recursively accessed for extracting values. The developed BIM IE algorithm exhaustively extracts the values for each attribute of each entity. Late binding information access method (in comparison to early binding information access method) in JSDAI was selected to support value extraction in the BIM IE algorithm. The late binding information access method accesses each attribute and entity using standard access methods in Java, and, thus, does not require the IFC schema to be available at the program compiling time. This feature of late binding information access method enables the BIM IE algorithm to extract information from BIMs based on any version of the IFC schema (e.g., IFC 2x3, IFC2x3-TC1, IFC4). With the extracted information from BIMs, an information transformation algorithm with two steps are used to transform the extracted design information into logic facts, including an initial transformation step and an alignment transformation step. The initial transformation step uses three main rules to transform the entities and attributes in an IFC-based BIM into logic facts. The three rules define the transformation methods for entity, attribute, and referenced attribute value, respectively. Referenced attribute value is a reference to another entity which is used as a value of an attribute. Reference attribute value is widely used in IFC-based BIMs to specify relationships between the entities. To match the generated logic facts with predicates in the logic rules that represent regulatory requirements, the alignment transformation step uses a set of semantic transformation (SeTr) rules to further transform the generated logic facts into more semantic representation consistent with the logic rule-represented regulatory requirements.

```
IFC Data

#39592=IFCHEIGHT();
#39594=IFCWALL($,$,$,$,$,$,$,$);
#39595=IFCACCBIRELATION($,$,$,$,'has',#39594,#39592,$);

Logic Facts Resulted From Initial Transformation

acc_bi_relation(acc_bi_relation39595).
has_type_name(acc_bi_relation39595,has).
has_relating_element(acc_bi_relation39595,wall39594).
has_related_element(acc_bi_relation39595,height39592).

Logic Facts Resulted From Alignment Transformation

has(wall39594,height39592).
```

Figure 3. Design information extraction and transformation example

3.3 Compliance Reasoning Algorithm

To enable automated reasoning, the compliance reasoning algorithm in this ACC system uses a set of functional built-in logic clauses. The functional built-in logic clauses implement basic computational functions (e.g., quantity level arithmetic operations) and define reasoning strategies (e.g., checking sequence). *LC1* shows an example functional built-in logic clause which initializes the checking of building elements sequentially. The corresponding checking sequence is defined in the list *L* which contains the building elements to check in order. The predicate "check(X)", which links to secondary logic clauses (e.g., the LC3 as described later) for arranging the checking against each rule, enables the activation of rules upon the matching of their activation conditions.

- *LC1*: checklist(L) :- foreach(X in L, check(X)).

A logic-based information representation schema using B-Prolog syntax is leveraged for enabling the compliance reasoning, which includes two parts: functional built-in logic clauses (which were described above) and information logic clauses (which are the logic rules and logic facts resulting from regulatory and design information extraction and transformation, as described above). Each regulatory requirement is represented by a primary logic rule and two secondary logic rules. The primary logic rule represents the conditions of the requirement in its rule body and the compliance status in its rule head. For example, *LC2* is a primary logic rule representing the requirement that "Courts shall not be less than 3 feet in width." Each primary logic rule is supported by two secondary logic rules, for representing the activation conditions and compliance checking consequences, respectively. The activation conditions for each requirement are directly generated from the premise conditions of the primary logic rule. Compliance checking consequences are the consequent actions for the compliance/noncompliance results. For example, *LC3* is a secondary logic rule representing the activation conditions for *LC2*, and *LC4* is a secondary logic rule representing the compliance checking consequences (output actions in this case) of *LC2*. Compliance checking consequences are automatically generated from the components of the logic body of the corresponding primary logic rule.

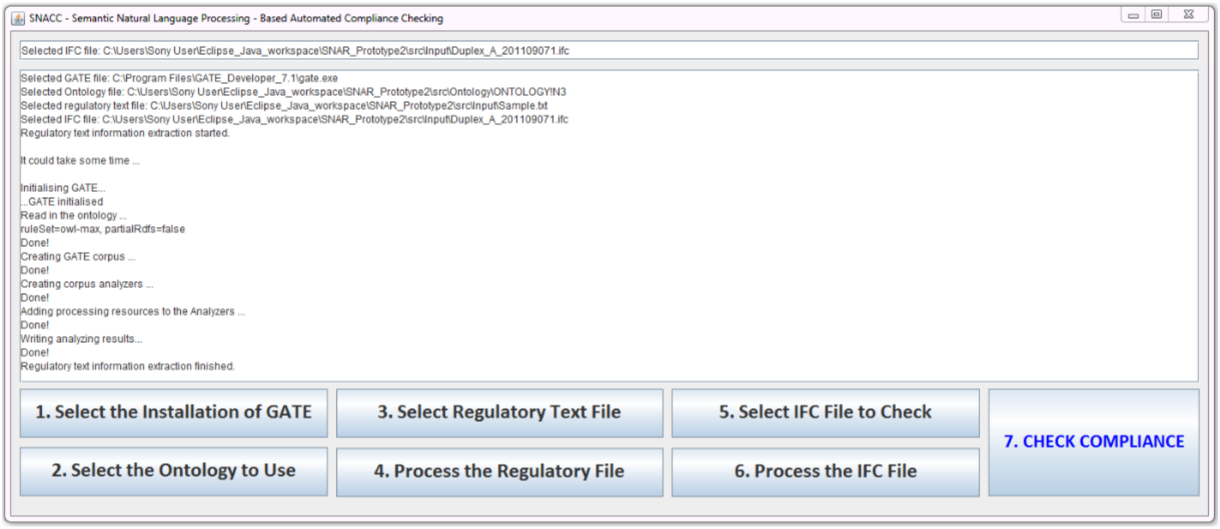
- *LC2*: compliance_width_of_court(Court) :- court(Court), width(Width), has(Court, Width), greater_than_or_equal(Width, quantity(3,feet)).
- *LC3*: check(X) :- … court(X),width(Y),has(X,Y) -> check_width_of_court(X);true, …
- *LC4*: check_width_of_court(X) :- compliance_width_of_court(X) -> writeln(X is width_compliant_based_on_Provision_1206(3)); writeln(X is width_noncompliant_based_on_Provision_1206(3)).

Under this information representation schema, a list of subjects (e.g., building elements) need to be specified for checking. The subjects in the specified list are then checked sequentially, implemented by a search strategy as follows: "for each selected subject instance, search over all activation conditions for matching, and check the instance against primary logic clauses whose activation conditions are met."

4. PROTOTYPE SYSTEM IMPLEMENTATION AND EXPERIMENTAL TESTING

4.1 Automated Compliance Checking System

The proposed ACC system was implemented in a proof-of-concept prototype system named Semantic Natural language processing-based Automated Compliance Checking (SNACC) system. The main platform of the SNACC system was built using Java programming language (Java Standard Edition Development Kit 6u45). The regulatory information extraction algorithm was implemented using resources from the General Architecture for Text Engineering (GATE) platform (Cunningham et al., 2012), using Java Annotation Patterns Engine (JAPE) rules to encode the IE rules. The interaction of the SNACC system with GATE was implemented using GATE Application Program Interface (API). The regulatory information transformation algorithm was implemented using python programming language (Python 2.7.3) because it was widely used for NLP research due to the abundance of modules that provide basic NLP functions. The "re" module (i.e., regular expression module) in Python was used for pattern matching to support the execution of SeM rules and CoR rules. The execution of the regulatory information transformation algorithms and their interaction with other parts of the SNACC systems were enabled using Jython (Jython 2.2.1). The design information extraction and information transformation algorithms were implemented in Java directly. The JSDAI runtime (JSDAI 4.3.0) was used to access information in IFC-based BIMs in the BIM information extraction and transformation algorithms. The automated reasoning capability was implemented using the B-Prolog Version 7.8 (Zhou, 2012) through its bi-directional interface with Java programming language. The graphical user interface of the SNACC system is shown in Figure 4. As shown in Figure 4, the SNACC system requires the download of the GATE tool and the availability of a building ontology as a premise to execute the regulatory information extraction and transformation algorithms. Then user could then select the regulatory document containing requirement text and the IFC file containing building design information to be checked. The information extraction and information transformation algorithms for regulatory information and design information could be executed independent of each other. After all information have been extracted and transformed, one pressing on the "check compliance" button activates the automated reasoning process using B-Prolog. The compliance checking results are then automatically shown in the text field upon completion of automated reasoning in B-Prolog.


SNACC - Semantic Natural Language Processing - Based Automated Compliance Checking
1. Select the Installation of GATE
3. Select Regulatory Text File
5. Select IFC File to Check
7. CHECK COMPLIANCE
2. Select the Ontology to Use
4. Process the Regulatory File
6. Process the IFC File

```
Selected IFC file: C:\Users\Sony User\Eclipse_Java_workspace\SNAR_Prototype2\src\Input\Duplex_A_201109071.ifc
```

```
Selected GATE file: C:\Program Files\GATE_Developer_7.1\gate.exe
Selected Ontology file: C:\Users\Sony User\Eclipse_Java_workspace\SNAR_Prototype2\src\Ontology\ONTOLOGY!N3
Selected regulatory text file: C:\Users\Sony User\Eclipse_Java_workspace\SNAR_Prototype2\src\Input\Sample.txt
Selected IFC file: C:\Users\Sony User\Eclipse_Java_workspace\SNAR_Prototype2\src\Input\Duplex_A_201109071.ifc
Regulatory text information extraction started.

It could take some time ...

Initialising GATE...
...GATE initialised
Read in the ontology ...
ruleSet=owl-max, partialRdfs=false
Done!
Creating GATE corpus ...
Done!
Creating corpus analyzers ...
Done!
Adding processing resources to the Analyzers ...
Done!
Writing analyzing results...
Done!
Regulatory text information extraction finished.
```

Figure 4. SNACC prototype system graphical user interface

4.2 Data for System Testing

The SNACC system was tested in checking the compliance of a BIM test case model (which was developed based on the Duplex Apartment Project from buildingSMARTalliance of the National Institute of Building Sciences) with the first 50 provisions from Chapter 19 of IBC 2009. A set of design facts were developed and added in the BIM of the Duplex Apartment Project to comprise the test case for design information. The test case included positive information set(s) and negative information set(s) for each quantitative requirement. A positive information set is a set of design information that complies with a regulatory provision. A negative information set is a set of design information that does not comply with a regulatory provision.

4.3 Testing Results and Analysis

The prototype ACC system was evaluated using precision and recall of noncompliance detection in comparison with a manually-developed gold standard. Precision is defined as the number of correctly detected noncompliance instances divided by the total number of noncompliance instances detected. Recall is defined as the number of correctly detected noncompliance instances divided by the total number of noncompliance instances that should be detected. The testing results are summarized in Table 1. As shown in Table 1, the recall of noncompliance detection is 98.0%, and the precision of noncompliance detection is 89.5%. These high performance results show that the proposed ACC system is promising. In addition, the proposed ACC system achieved higher recall (98.0%) than precision (89.5%), which shows its suitability for the ACC application; in noncompliance detection, recall is more important than precision. Recall errors are more critical because they might result in missing noncompliance instances, whereas precision errors could be easily double-checked and filtered out by the user. The ACC system was able to block errors in information extraction and information transformation steps from propagating into noncompliance detection results. Because all selected design subjects were checked, noncompliance instances were difficult to be missed in the proposed system. The only case where noncompliance instances could be missed is when errors exist in quantity type predicates [e.g., not greater_than(Height,quantity(8,feet)) in Figure 2]. In fact, errors in information extraction and information transformation steps affected 9 logic clauses in the experiment, yet none of them led to a missing noncompliance instance (i.e., false negatives). Most of these information extraction and information transformation errors still led to incorrectly detected noncompliance instances (i.e., false positives), which was why the precision was not as high as recall.

Table 1. Noncompliance detection testing results

Parameter/measure	Results
Number of noncompliance instances in gold standard	51
Recall of noncompliance detection	98.0%
Precision of noncompliance detection	89.5%

5. LIMITATIONS AND FUTURE WORK

The experimental results show that the proposed ACC system is promising in automatically extracting and transforming information from construction regulations and IFC-based BIMs and automatically conducting compliance checking. Despite the high performance achieved (98.0% and 89.5% for recall and precision, respectively) in noncompliance detection, two main limitations of the work are acknowledged, which the authors plan to address in their future/ongoing research. First, the ACC method and algorithms was were tested only on a part of a Chapter of IBC 2009 (ICC, 2009) and one test case of design information, primarily because the development of the gold standard for testing is highly time intensive. As part of future/ongoing research work, the authors will further test the method and algorithms on more building code chapters and more BIM model test cases. Similar high performance is expected because of the similarity in regulatory document expressions and consistency of information representation between different BIM models. However, the results may vary because of the possible variability in the syntactic and semantic text features across different chapters and/or codes and the different ways certain entities/attributes in IFC are used to represent design information. Second, the proposed algorithms were only tested in checking the compliance of BIM models with quantitative requirements. While the overall proposed ACC system is expected to be applicable to different types of requirements, the algorithms for each part of the system may need to be extended to cover other types of requirements (e.g., existential requirement). In future work, the authors will extend the system to cover existential regulatory requirements.

6. CONCLUSIONS

This paper presented a semantic NLP-based, logic-based, and JSDAI-based system for automated compliance checking of building designs with construction regulatory documents. The ACC system utilizes pattern matching-based rules consisting of syntactic and semantic features to automatically extract regulatory information from regulatory documents into semantic tuples, and automatically transform the extracted regulatory information into logic rules. The ACC system utilizes a JSDAI-based algorithm to extract design information from BIMs into semantic tuples, and utilizes semantic NLP-based transformation rules to transform the extracted design information into logic facts. The ACC method utilizes logic-based representation and compliance reasoning to automatically check the compliance of design information with regulatory information and generate compliance reports. The algorithms for all processing steps were implemented in different programming languages and integrated into one proof-of-concept prototype system. Experimental testing of the prototype system was conducted on a part of regulatory requirements from Chapter 19 of IBC 2009 and a test case building information model based on the Duplex Apartment Project from buildingSMARTalliance of the National Institute of Building Sciences. As a result, a recall of 98.0% and a precision of 89.5% were achieved for noncompliance detection. The testing results show that the proposed ACC system is promising. The ACC system was able to block certain information extraction and transformation errors from propagating into noncompliance detection results. The well-functioned prototype shows that the information extraction, information transformation, and logic-based reasoning algorithms could be integrated effectively, despite of their primary implementation in different types of programming languages. As part of future research, the proposed ACC method and algorithms will be tested on more building code chapters and more test cases, and extended to cover different types of regulatory requirements.

ACKNOWLEDGMENTS

The authors would like to thank the National Science Foundation (NSF). This material is based upon work supported by NSF under Grant No. 1201170. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

REFERENCES

Beach, T.H., Kasim, T., Li, H., Nisbet, N., and Rezgui, Y. (2013). Towards automated compliance checking in the construction industry, *H. Decker et al. (Eds.): DEXA 2013, Part I, LNCS 8055*, 366-380.

Boken, P., and Callaghan, G. (2009). Confronting the challenges of manual journal entries, Protiviti.

Corke, G. (2013). Solibri model checker V8, *AECMagazine: Building information modelling (BIM) for architecture, engineering and construction*, http://aecmag.com/software-mainmenu-32/527-solibri-model-checker-v8, accessed on February 01, 2015.

Cherpas, C. (1992). Natural language processing, pragmatics, and verbal behavior, *Anal. Ver. Behav.*, 10, 135-147.

Crowston, K., Liu, X., Allen, E., and Heckman, R. (2010). Machine learning and rule-based automated coding of qualitative data, *Proc., 73rd ASIS&T Annual Meeting: Navigating Streams in an Information Ecosystem, Association for Information Science and Technology*, Silver Spring, MD, pp.1-2.

Cunningham, H. et al. (2012). Developing language processing components with gate version 7 (a user guide), The University of Sheffield, Department of Computer Science, Sheffield, U.K.

Eastman, C., Lee, J., Jeong, Y., and Lee, J. (2009). Automatic rule-based checking of building designs, *Autom. Constr.*, 18 (8), 1011-1033.

Harrison, J. (2007). A short survey of automated reasoning, *Proc., AB 2007, the Second International Conference on Algebraic Biology*, Springer LNCS, 4545, pp. 334-349.

Hjelseth, E. and Nisbet, N. (2011). Capturing normative constraints by use of the semantic mark-up RASE methodology, *Proc., CIB W78 2011*, Conseil International du Bâtiment, Rotterdam, The Netherlands.

Hodges, W. (2001). Classical logic I - first-order logic, Goble, 2001, 9-32.

International Code Council (ICC). (2009). 2009 International Building Code, *2009 Int. Codes*, http://publicecodes.cyberregs.com/icod/ibc/2009/index.htm, accessed on February 05, 2011.

Konev, B., Schmidt, R.A., and Schulz, S. (2010). Special issue on practical aspects of automated reasoning, *AI Communications*, 23 (2010), 67-68.

Melzner, J., Zhang, S., Teizer, J., and Bargstadt, H. (2013). A case study on automated safety compliance checking to assist fall protection design and planning in building information models, *Construction Management and Economics*, 31 (6), 661-674.

Nguyen,T., and Kim, J. (2011). Building code compliance checking using BIM technology, *Proc., 2011 Winter Simulation Conference (WSC)*, IEEE, New York, NY, pp. 3395-3400.

OptaSoft, (2014). International Code Council, http://www.optasoft.com/ , accessed on January 06, 2014.

Portoraro, F. (2011). Automated reasoning, *The Stanford encyclopedia of philosophy (Summer 2011 Edition), Edward N. Zalta (ed.)*, http://plato.stanford.edu/archives/sum2011/entries/reasoning-automated/, accessed on December 26, 2014.

Tan, X., Hammad, A., and Fazio, P. (2010). Automated code compliance checking for building envelope design, *J. Comput. Civ. Eng.*, 10.1061/ 1195 (ASCE)0887-3801(2010)24:2(203), 203-211.

Tierney, P. J. (2012). A qualitative analysis framework using natural language processing and graph theory, *Int. Rev. Res. Open Distance Learn.*, 13 (5).

Yurchyshyna, A., Faron-Zucker, C., Thanh, N.L., and Zarli, A. (2008). Towards an ontology-enabled approach for modeling the process of conformity checking in construction, *Proc., CAiSE'08 Forum 20th Intl. Conf. Adv. Info. Sys. Eng.*, dblp team, Germany, 21-24.

Zhang, J., and El-Gohary, N. (2015). Automated extraction of information from building information models into a semantic logic-based representation, *Computing in Civil Engineering 2015*.

Zhang, J., and El-Gohary, N. (2013). Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking, *J. Comput. Civ. Eng.*, 10.1061/(ASCE)CP.1943-5487.0000346, 04015014.

Zhou, N. (2012). B-Prolog user's manual (version 7.8): Prolog, agent, and constraint programming, *Afany Software.*, http://www.probp.com/manual/manual.html, accessed on December 28, 2013.

Zouaq, A. (2011). An overview of shallow and deep natural language processing for ontology learning, *Ontology learning and knowledge discovery using the web: Challenges and recent advances*, IGI Global, Hershey, PA, 16–38.