Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking

Article *in* Automation in Construction · November 2016

DOI: 10.1016/j.autcon.2016.08.027

2 authors, including:

Jiansong Zhang
Western Michigan University

Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking

Jiansong Zhang[1]; and Nora M. El-Gohary[2]

Abstract

Existing automated compliance checking (ACC) systems are limited in their automation; they rely on the use of hard-coded, proprietary rules for representing regulatory requirements, which requires major manual effort in extracting regulatory information from textual regulatory documents and coding these information into a rule format. To address this limitation, this paper proposes a new unified ACC system that integrates: (1) semantic natural language processing techniques and EXPRESS data based techniques to automatically extract and transform both regulatory information (in regulatory documents) and design information [in building information models (BIMs)] for automated compliance reasoning, and (2) semantic logic-based information representation so that the reasoning could be fully automated. To test the proposed system, a BIM test case was checked for compliance with Chapter 19 of the International Building Code 2009. Comparing to a manually-developed gold standard, 98.7% recall and 87.6% precision in noncompliance detection were achieved.

Keywords: Automated code checking; Automated information extraction; Automated reasoning; Building information modeling (BIM); Natural language processing; Logic; Semantic systems; Automated construction management systems.

[1] Assistant Professor, Dept. of Civil and Construction Engineering, Western Michigan University, 1903 W Michigan Ave, Kalamazoo, MI 49008.

[2] Assistant Professor, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, 205 N. Mathews Ave., Urbana, IL 61801 (corresponding author). E-mail:gohary@illinois.edu; Tel: +1-217-333-6620; Fax: +1-217- 265-8039.

1 Introduction

The manual process of regulatory compliance checking is time-consuming, costly, and error-prone (Boken and Callaghan 2009). In the U.S., each building compliance review cycle usually takes several weeks (State of New Jersey 2014; City of Philadelphia 2015), and a construction project may be subject to multiple cycles of plan reviews due to design changes. At the city level, millions of dollars are spent on manual building compliance checking each year (Department of Buildings 2015). Failure to comply with building regulations could further result in fines, penalties, or even criminal court summons and prosecutions (Los Angeles Times 2015). Moreover, in an experiment conducted by Fiatech, more disparity than agreement was found when different plan review departments were asked to conduct manual code review of the same set of plans (Fiatech Regulatory Streamlining Committee 2012).

In comparison to manual compliance checking, automated compliance checking (ACC) of construction projects is expected to reduce the time, cost, and errors of the compliance checking process (Eastman et al. 2009, Tan et al. 2010; Nguyen and Kim 2011; Kasim et al. 2013; Zhang and El-Gohary 2013). However, the state-of-the-art ACC systems cannot achieve full automation because of relying on the use of hard-coded, proprietary rules for representing regulatory requirements, which requires major manual effort in extracting regulatory information from textual regulatory documents and coding these information into a rule format. For example, the COnstruction and Real Estate NETtwork (CORENET) project hard-coded rules in C++ programs, the Solibri model checker uses a proprietary proforma-based format to hard code rules, and several ACC efforts hard-coded rules for specific subdomains such as building evacuation (Choi et al. 2014), fall protection (Zhang et al. 2013), construction quality (Zhong et al. 2012), building safety design (Qi et al. 2011), building envelope performance (Tan et al. 2010), and accessibility (Lau and Law 2004). Such hard-coded rules could be very effective in reasoning about compliance with a specific set of requirements and specific regulatory sections in a certain period of time, but such rigid and static representation requires great effort in (1) adaptation to different regulatory codes/sections, and (2) maintenance/update across different time periods and in response to code revisions/updates. The use of hard-coded rules, thus, becomes effort-intensive and time-consuming because of the large number of codes and regulations and their frequent revisions/updates (Delis and Delis 1995; Dimyadi and Amor 2013).

In view of that, a number of researchers explored the development of generalized representations for the formalization of regulatory requirements, with the aim to facilitate soft coding of rules for supporting ACC. For example, Pauwels et al. (2011) proposed a semantic rule checking environment, in which Notation 3 (N3) Logic is used to represent requirement rules. Hjelseth and Nisbet (2011) proposed the Requirement, Applies, Select, and Exception (RASE) method to represent regulatory requirements. Yurchyshyna et al. (2010; 2008) developed a conformity-checking ontology that represents regulatory information, building-related knowledge, and expert knowledge on checking procedures, with a representation of regulatory requirements in the form of SPARQL Protocol and RDF Query Language (SPARQL) queries. Beach et al. (2013; 2015) extended the RASE method for a more powerful regulatory information representation at both “the block level (i.e., paragraph level) and inline (i.e., individual words or groups of words)”, which can be converted to Semantic Web Rule Language (SWRL) for reasoning. And, Dimyadi et al. (2014) represented regulatory requirements using the Drools Rule Language (DRL).

These efforts have undoubtedly contributed to the improvement of flexibility and reusability of regulatory representations for supporting ACC. However, they are still limited in terms of automated regulatory information extraction and transformation; the state of the art in ACC still requires major manual efforts in extracting regulatory information from textual regulatory documents and transforming/encoding these information into a computer-processable rule format. For example, in Pauwels et al. (2011), Hjelseth and Nisbet (2011), Yurchyshyna et al. (2010; 2008), Beach et al. (2013; 2015), and Dimyadi et al. (2014), the extraction of regulatory information and their encoding into N3Logic, the RASE representation, the SPARQL queries, the extended RASE representation, and the DRL rules, respectively, are still manually conducted. To facilitate the regulatory information extraction and conversion, the SMARTcodes project led by the International Code Council (ICC) developed tools to help ICC staff and building code officials mark-up the ICC codes with provided tags under a predefined SMARTcodes schema. The marked codes, then, can be automatically transformed into a "requirements model", which leverages the IfcConstraint entities within an Industry Foundation Classes (IFC) model and therefore is essentially an IFC constraint model (AEC3 2012). As the process suggests, the SMARTcodes project still requires manual rule extraction and encoding efforts in the form of marking-up tasks.

To address these gaps of knowledge, this paper proposes a new fully-automated ACC system [the authors call it semantic natural language processing (NLP)-based automated compliance checking (SNACC) system] that integrates three types of algorithms in one unified computational platform: (1) semantic NLP) algorithms to automatically extract the regulatory information from regulatory documents (e.g., building codes) and transforms the extracted regulatory information into logic rules, (2) semantic EXPRESS data processing algorithms to automatically extract the design information from building information models and transform the extracted design information into logic facts, and (3) semantic-based logic reasoning algorithms to automatically reason about the compliance of the logic facts with the logic rules. The automated analyses are facilitated by information representations that are semantic, logic-based, and generalized and flexible. This paper presents the integration of the proposed algorithms in a unified ACC system and discusses the experimental results of testing the proposed unified system using a test case.

2 Proposed approach to full automation in automated compliance checking

This paper proposes a fully-automated approach to ACC in construction. The approach relies on the use of a set of computational techniques in an integrated manner, in one unified system. The techniques include NLP, EXPRESS data processing, and logic reasoning, which are collectively used for automated information processing (both design information and regulatory information) and automated compliance reasoning. The automated processes are facilitated by semantic, logic-based representations that are generalized and flexible.

2.1 Information representation

The choice of information representation has strong implications on information processing and is of vital importance in facilitating automated processes. In ACC applications, specifically, there is a need for a "standard, generalized approach for formally representing building regulations in a digital format that would facilitate a variety of forms of reasoning about those codes in combination with digital building information models" (Garrett et al. 2014), including automated information extraction and information transformation to support complete automation of ACC. The proposed representation is semantic and logic-based, in a way which is generalized and flexible.

2.1.1 Semantic representation

The representation is semantic; it uses semantic information elements and a domain ontology. Semantic information elements represent the elements of a regulatory requirement, including "subject," "compliance checking attribute," "deontic operator indicator," "quantitative relation," "comparative relation," "quantity value," "quantity unit," "quantity reference," "restriction," or "exception." A building ontology is a semantic model for representing building domain knowledge in the form of concept hierarchies, relationships between concepts, and axioms. The semantic representation facilitates deep information processing (i.e., full-sentence analysis towards capturing the entire meaning of a sentence, as opposed to shallow processing that extracts partial information from a sentence). The semantic representation is also utilized to leverage domain knowledge in the reasoning process, in order to handle the complex relations involved in compliance reasoning and enable deep reasoning. This is important because the relations in regulatory provisions could be very complex. For example, Fig. 1 shows the many relations involved in one single regulatory provision in IBC 2006, leading to a very complex regulatory provision. The semantic representation also facilitates human understandability and interpretability of the formal representation, which is essential to facilitate usability and allow for human testing and verification of the information representation and the reasoning results.

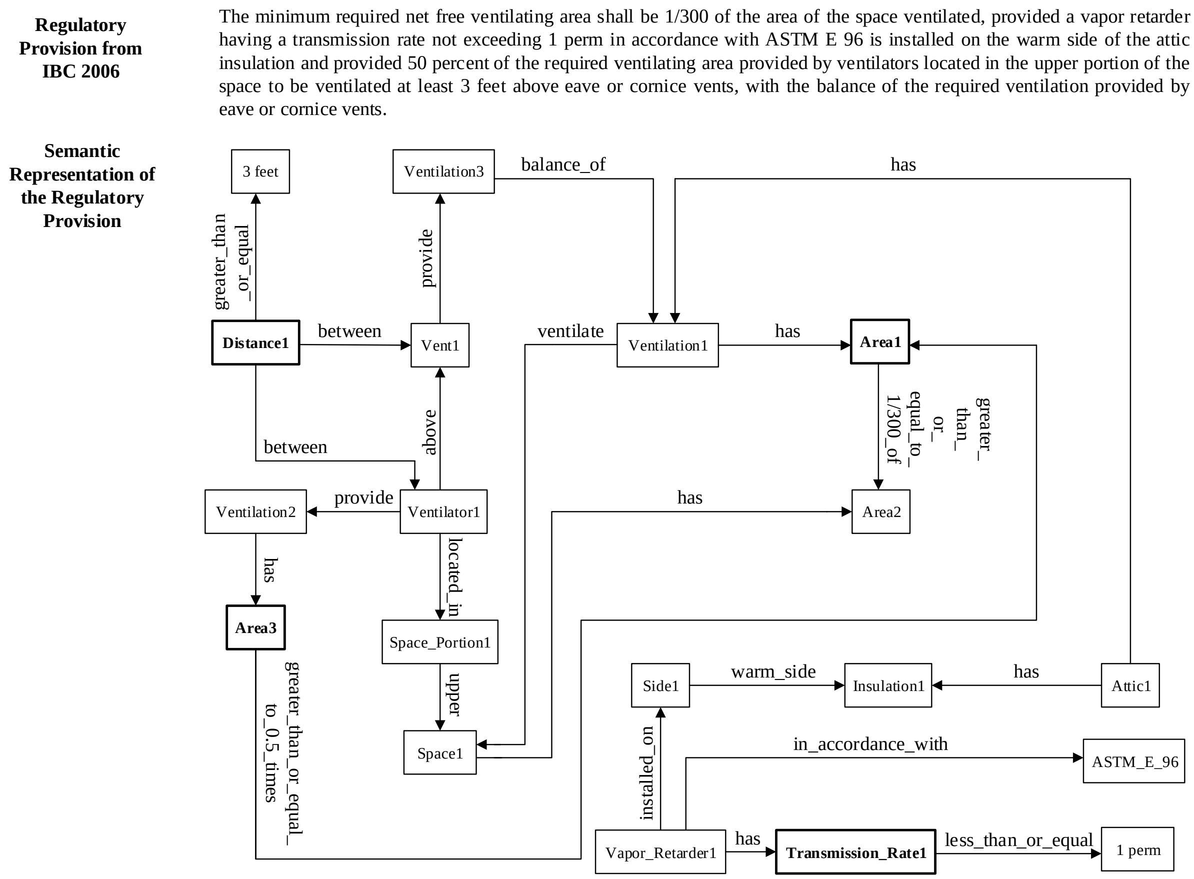

Fig. 1. An example to illustrate the complexity of relations in provisions.

2.1.2 Logic representation

The representation is logic-based: regulatory information are represented as logic rules, while design information are represented as logic facts. A logic-based representation was selected to take advantage of the well-matured logic-based reasoning techniques. Logic-based reasoning is well-suited for ACC problems because (Zhang and El-Gohary 2016b): (1) The binary nature (satisfy or fail to satisfy) of logic fits the binary nature (compliance or noncompliance) of ACC; (2) Formally-defined logics have sufficient expressiveness to represent concepts and relations involved in ACC; (3) Once the information is properly represented in a logic format, the reasoning can be conducted in a fully-automated way; and (4) Automated reasoning techniques are available in ready-to-use logic reasoners.

Among the existing types of logic, FOL is the foundation of almost all work in rule representation for ACC because of its expressivity; these efforts used a variety of logic implementations/languages, but they all built on some restricted form of FOL. For example, Pauwels and Zhang (2015) reviewed a good number of semantic rule checking applications among which two main types of logic were used: N3Logic (e.g., Dimyadi et al. 2015) and SWRL (e.g., Baumgärtel et al. 2015). Both N3Logic and SWRL were created to go beyond the monotonic negation limitation of FOL. N3Logic was created to avoid the paradox traps problem of FOL by not using the general first-order negation but rather relying on customarily-made negated forms of functions to achieve nonmonotonic negation (Berners-Lee 2005). SWRL is essentially combining the Datalog Rule Markup Language (RuleML) with the Web Ontology Language (OWL), where Datalog is a restricted subset of FOL using function-free Horn Clauses (HCs) (Horrocks et al. 2004). A HC is a restricted form of FOL that is most efficient in inference making (Saint-Dizier 1994). Both N3Logic and SWRL were used because of their compatibility with OWL ontologies, which are the core of semantic rule checking approaches. More importantly, logic such as N3Logic and SWRL need to be used to support if-then rule representation for rule checking when OWL ontologies are used, because OWL is based on description logic (DL). A set of rules (if-then statements) is necessary to allow for rule checking (Pauwels and Zhang 2015), and DL does not allow for the representation of if-then rules. In addition to N3Logic and SWRL, Solihin and Eastman (2015) took a knowledge representation approach for representing requirement rules using conceptual graphs, which also has a semantic foundation in FOL and has one-to-one mapping to FOL rules.

FOL was selected, in this paper, to support ACC not only because of its expressivity but also because of its ability to represent English sentences. “A first-order sentence φ can often be translated into an English sentence which is guaranteed to be true if and only if φ is true in *I*" (i.e., the interpretation) (Hodges 2001). This property makes FOL suitable for representing regulatory information to support automated compliance reasoning, because existing regulatory rules in building codes and regulations are mostly coded in natural language sentences. Although FOL cannot represent all provisions in building codes and regulations (Garrett et al. 2014), among those provisions it can represent, FOL: (1) enables isomorphism: one-to-one mapping between an English regulatory requirement and a logic clause, and (2) as a result, allows for traceability: maintaining traceability is important to identify the sources of logic clauses and, thus, to facilitate human verification and ensure trustworthiness of the logic clauses and the results. The scope of this paper is limited to quantitative requirements – part of the regulatory requirements that are representable in FOL. The representability of all possible types of regulatory requirements in FOL (i.e., which requirements can be represented in FOL and which not) is an interesting topic that is worth further investigation (Garrett et al. 2014), but is outside of the scope of this paper.

2.1.3 Generalized and flexible representation

The representation is generalized and flexible. The generalization and flexibility are achieved through generalized regulatory compliance checking concepts and flexible semantic information elements. Generalized regulatory compliance checking concepts (e.g., "subject" and "compliance checking attribute") are used, which allows for representing regulatory provisions of any type/topic (e.g., building envelope performance, facility accessibility). Flexible information elements (e.g., "subject restriction," as discussed in the following sections) are used, which allows for representing all information (i.e., all concepts and relations) in a regulatory provision regardless of the length and complexity of the provision (sentence). Generalization and flexibility are important to sustain utility and robustness of the proposed system across different types of regulatory documents and different types of provisions.

2.2 Computational techniques

2.2.1 Deep natural language processing techniques

It is an important impact conceived by many researchers who work on computable regulatory rule representations (e.g., RASE, SMARTcodes) that the use of their representations may guide the future drafting of codes and regulations (e.g., through the use of built-in annotations), so that the automated extraction and transformation of regulatory information into computable rules would be easily addressed. The authors also share that aspiration. But, at the same time, the authors foresee that long-term goal (i.e., changing the way codes and regulations are drafted) as a big challenge, potentially beyond the reach of solely the construction community, because it requires harmonizing a lot of different pursuits and interests from various stakeholders (code drafters, regulators, designers, etc.). Let alone that any developer of a computable regulatory rule representation typically wants their own development to be adopted at a large scale, both geographically and democratically – so which representation becomes a standard or becomes widely adopted is another issue. On the other hand, the authors hold the ground of the status quo that current building codes and regulations are mostly represented in natural language text, and leverage state-of-the-art NLP techniques to develop new methods towards bridging the automation gap of regulatory information extraction and transformation, under their proposed ACC framework.

NLP techniques are used to facilitate text analysis and processing for automatically extracting regulatory information from building codes. NLP is a theoretically-based computerized approach to analyzing, representing, and manipulating natural language text for the purpose of achieving human-like language processing for a range of tasks or applications (Cherpas 1992). The types of natural language analyses and techniques used highly affect the ability of NLP algorithms to process complex sentences and recognize their full meaning. Full sentence understanding – of both simple and complex sentences – is essential to achieve full automation in analyzing building codes and extracting regulatory information. Deep NLP aims to capture the full meaning of sentences to facilitate full sentence understanding by computers (Zouaq 2011). The proposed approach offers a new way to achieve a deep level of text processing by integrating three types of knowledge in the analysis of sentences: (1) ACC-specific knowledge: knowledge about the elements of a regulatory requirement in building codes, represented in the form of semantic information elements, (2) AEC domain knowledge: knowledge about the building domain, represented in the form of an ontology, and (3) linguistic knowledge: knowledge about the linguistic expressions of requirements in building code provisions, represented in the form of information extraction rules.

2.2.2 EXPRESS data processing techniques

EXPRESS data processing techniques are used for automatically extracting design information from building information models. EXPRESS data processing techniques are suitable for accessing information from IFC-based BIMs because the IFC schema is written in the EXPRESS language. This EXPRESS language-level of processing enables the extraction and further transformation of design information to be aligned with regulatory information.

The Java Standard Data Access Interface (JSDAI) was utilized for BIM information extraction, using late binding data access methods. JSDAI is a standard data access interface (SDAI) application programming interface (API) to access information from models written in EXPRESS language – the ISO standard product data modeling language (ISO 2004). JSDAI provides two types of data access methods: (1) early binding method, which accesses entities and attributes in an EXPRESS model with specialized access methods such as “getCeilingHeight” (i.e., method to get ceiling height of a floor), and (2) late binding method, which accesses entities and attributes in an EXPRESS model with generalized access methods such as "getExplicitAttributes" (i.e., method to get any explicit attribute). Compared to early binding, late binding allows accessing information based on more general metadata.

2.2.3 Logic reasoning and programming

Logic programming is a computational programming paradigm that is based on a Horn Clause (HC)-representation (Portoraro 2011). A program written in a logic programming language is simply a set of logic sentences that represent facts and rules about some domain of interest. Logic programming is declarative in contrast to other non-logical programming languages. For example, in typical procedural programming languages like C programming language a programmer has to clearly define how to solve the problem step by step, whereas in logic programming a programmer only needs to define how to represent the problem in the form of facts and rules. The solution steps in logic programming are already defined by a built-in reasoner through a set of organized automated reasoning techniques such as search strategies and backtracking.

2.3 System integration

The proposed system offers a novel integration of natural language processing techniques, EXPRESS data processing techniques, and logic reasoning into one unified computational framework to allow for full automation in ACC. The integration is facilitated by the choice of (as per Fig. 2): (1) a semantic representation that allows for seamless flow of information from one computational paradigm to another, from one computational module to another, and from one algorithm to another, (2) a logic representation, as a final representation, which allows to combine partial output from two separate modules (logic rules and logic facts from module 1 and module 2, respectively) into one coherent representation that is ready for reasoning, (3) a modular system

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

architecture, which enables a flexible use of multiple modeling paradigms and multiple programming languages, and (4) an architecture-neutral platform that can interoperate with multiple programming language interfaces.

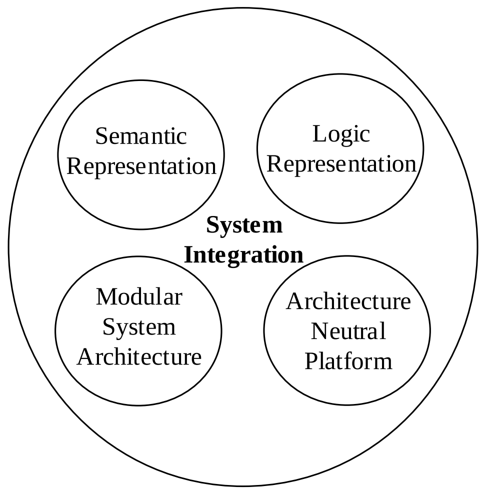

Fig. 2. System integration.

3 System architecture

This section provides an overview of the system architecture, including an overview of: (1) the system modules and how they are interlinked and integrated, and (2) the implementation of the system modules and how they interact. More details on the individual system modules and implementations are provided in Sections 4 and 5.

The system architecture is illustrated in Fig. 3. It is composed of three main modules: (1) regulatory information extraction and transformation module, (2) design information extraction and transformation module, and (3) compliance reasoning module. The system architecture is built on top of the Java Platform (Oracle 1999). The General Architecture for Text Engineering (GATE) tools (Cunningham et al. 2012), Python programming language (Python 2.7.3), B-Prolog logic programming platform and reasoner (Zhou 2012), and JSDAI tools are used in the system to support the computational processes in the different modules.

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

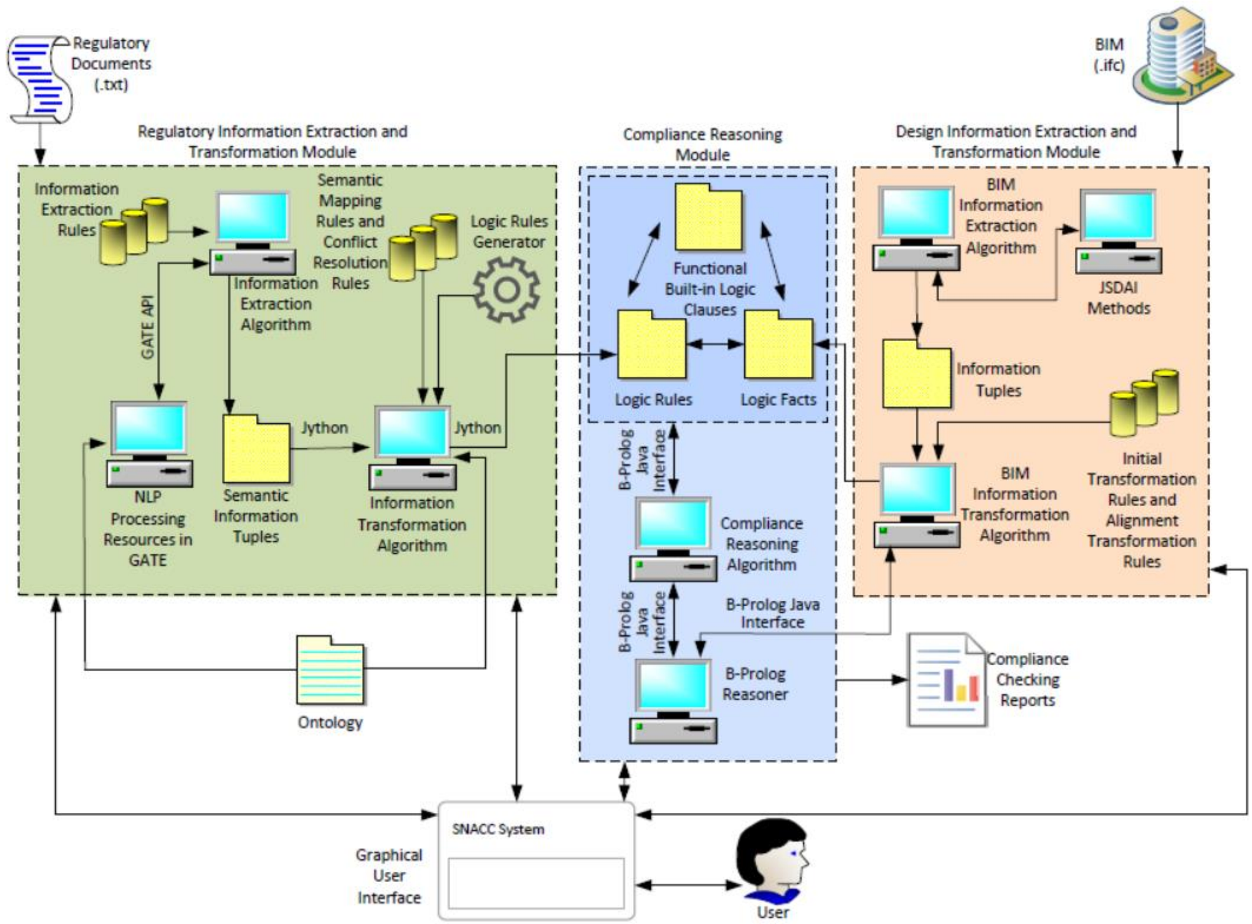

Fig. 3. System architecture of the SNACC system.

The regulatory information extraction and transformation module is composed of the regulatory information extraction algorithm and the regulatory information transformation algorithm. The information extraction algorithm aims to extract the regulatory requirements from a regulatory document into a semantic information tuple representation, where each tuple contains information instances for the semantic information elements (e.g., "subject," "compliance checking attribute"). The algorithm relies on the use of a set of pattern matching-based information extraction rules. A set of syntactic and semantic features are used in the patterns of the information extraction rules. The syntactic features are generated using GATE's Processing Resources (e.g., tokenizer), while the semantic features are generated from the ontology using GATE's Processing Resources (e.g., gazetteer). The information extraction algorithm interacts with the Processing Resources using

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

GATE's API in Java. The regulatory information transformation algorithm aims to transform the extracted instances of the semantic information elements in the information tuples into logic rules. The algorithm relies on the use of a set of pattern matching-based semantic mapping rules and conflict resolution rules, which include a set of syntactic and semantic features in their patterns. The semantic features, here, are the semantic information element features (e.g., the semantic feature "s" stands for "subject"). The information transformation algorithm interacts with the other modules of the SNACC system (in Java) through Jython. The ontology is used to support the regulatory information extraction and transformation processes by facilitating automated interpretability and understandability of regulatory text based on meaning.

The design information extraction and transformation module is composed of the BIM information extraction algorithm and the BIM information transformation algorithm. The BIM information extraction algorithm aims to extract the entities and their attributes from a BIM into an information tuple representation. The algorithm relies on the use of a set of entity and attribute extraction rules. The data types of the entities and attributes are extracted from the BIM using late binding data access methods in JSDAI. The BIM information transformation algorithm aims to transform the extracted entities and attributes in the information tuples into logic facts that are aligned with the logic rules. The algorithm relies on the use of initial transformation rules and semantic transformation rules. The initial transformation rules transform the extracted entities and attributes in the information tuples into logic facts. The semantic transformation rules further transform the initially transformed logic facts into more semantic logic facts that are aligned with the predicates in the logic rules. The initial transformation rules are coded in Java and the semantic transformation rules are coded in B-Prolog rules. To execute the semantic transformation rules, the information

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

transformation algorithm interacts with B-Prolog's reasoner through B-Prolog's interface with Java.

The compliance reasoning module is composed of the compliance reasoning algorithm, which utilizes B-Prolog's reasoner. The compliance reasoning algorithm aims to reason about the logic rules and the logic facts and generate compliance checking reports. The algorithm controls and supports the reasoning about the rules and facts in B-Prolog's reasoner using a set of functional built-in logic clauses. The compliance reasoning algorithm interacts with B-Prolog's reasoner through B-Prolog's interface with Java. A user interacts with all the three modules through a graphical user interface.

4 System modules

4.1 Regulatory information extraction and transformation module

The regulatory information extraction and transformation module is composed of four main processes: preprocessing, feature generation, information extraction, and information transformation.

Preprocessing prepares the raw natural language text of building codes for further processing. Four NLP techniques are utilized: tokenization, sentence splitting, morphological analysis, and dehyphenation. Tokenization divides the text into tokens (words or terms) to prepare for further unit-based processing of the text. Sentence splitting recognizes the boundaries of the sentences to help distinguish provisions in the building codes. Morphological analysis recognizes the different forms of a word and maps them into the lexical form of that word. This helps in the recognition of ontology concepts. Dehyphenation removes hyphens that indicate continuation of words between lines to avoid further processing errors caused by those hyphens.

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

Feature generation generates a set of syntactic and semantic features that describe the text. Three NLP techniques are utilized to generate the syntactic features: POS tagging, phrase structure analysis, and gazetteer list analysis. POS tagging tags each word with the POS [lexical and functional categories such as singular or mass noun (NN) and adjective (JJ)] of the word. Phrase structure analysis tags each phrase with the phrasal tag [lexical and functional categories such as noun phrase (NP) and verb phrase (VP)]. A set of application-specific phrase structure grammar (PSG) rules are used to generate phrasal tags. The use of phrasal tags in addition to POS tags reduces the potential number of enumerations in the patterns of the information extraction rules (described in the following step). Gazetteer list analysis identifies each word that belongs to a gazetteer list [a set of names based on any specific commonality possessed by those terms, e.g., "unit gazetteer list" includes inches and feet among others] and uses that information as a feature. The building ontology is utilized to generate the semantic features, including terms/phrases that match to the concepts and relations in the ontology. Fig. 4 shows a partial view of the ontology that was used.

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

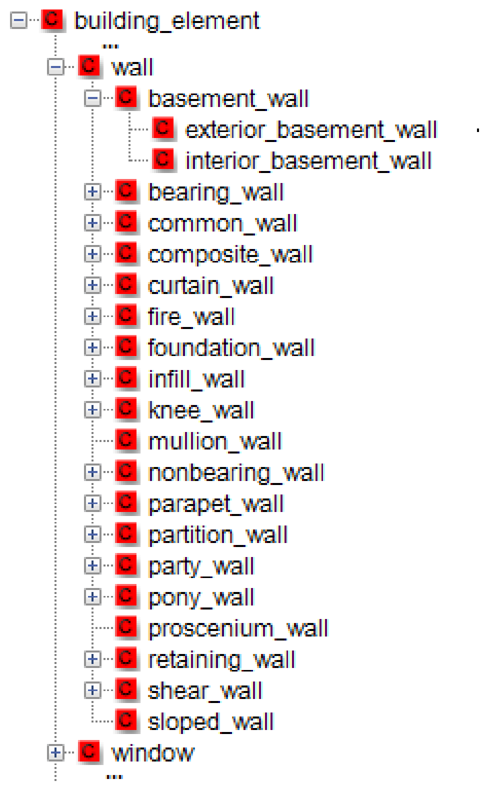

Fig. 4. Partial view of the building ontology.

Information extraction extracts the instances of the semantic information elements (SIEs) from the building code, using a set of 146 information extraction (IE) rules. An SIE is an ontology concept, an ontology relation, a "deontic operator indicator" (a term indicating an obligation, permission, or prohibition), or a "restriction" (an element that places a constraint on the definition of another semantic information element, where the constraint is expressed in terms of ontology concepts and relations). The ten types of SIEs and their definitions are shown in Table 1. Each SIE is either a "simple SIE" or a "complex SIE," and a "rigid SIE" or a "flexible SIE" (Zhang and El-Gohary 2013). A simple SIE is associated with a single concept/relation/indicator whereas a complex SIE is expressed in terms of multiple concepts and relations. The simple SIEs are rigid [has a fixed number (i.e., 1) of concepts/relations], whereas the complex SIEs are flexible [has a varying number (i.e., 0 or more) of concepts/relations]. The IE rules use pattern matching; the rules extract the instances of each SIE based on text patterns. The patterns consist of syntactic and semantic features, which were generated during the feature generation step. For example, an IE rule for extracting the instances of "subject" is shown in Fig. 5. The example IE rules use patterns that consist of semantic features of "building element," "room," "space," and "quantity," and syntactic features of "modal verb," "negation," "base form verb," "comparative relation," "cardinal number," "slash," and "unit." The IE rules were developed based on Chapters 12 and 23 of the International Building Code (IBC) 2006 (ICC 2006). The extraction of each semantic information element is separated and arranged in the following sequence because extracting all semantic information elements from a sentence using a single IE rule is not efficient: "quantity value" and "quantity unit/quantity reference" > "subject" > "compliance checking attribute" > "comparative relation" > "quantitative relation" and "deontic operator indicator" > "subject restriction" and "quantity restriction." An example illustrating the extraction is shown in Fig. 5. The text is then tagged with the extracted SIEs for further information transformation.

Information transformation transforms the extracted information into logic rules, using a set of 9 conflict resolution (CR) rules, 297 semantic mapping (SM) rules, and a logic rule generator. The CR rules and SM rules use pattern matching. The patterns consist of three types of information tags: (1) syntactic information tags: syntactic feature tags generated during feature generation, (2) semantic information tags: SIE tags generated during information extraction, and (3) combinatorial information tags: compound information tags that are composed of multiple syntactic and/or semantic information tags. Fig. 6 shows an example of a tagged regulatory requirement. The CR rules resolve conflicts between the extracted information instances (in the form of four-element tuples) based on the patterns. The SM rules transform the extracted information instances (after conflict resolution) into logic components (i.e., logic predicates and logic operators) based on the patterns. For example, 'n' 'c' 'v' 'u' is used as a pattern for an SM rule, which identifies a sequence

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

of "negation," "comparative relation," "quantity value," and "quantity unit." Fig. 7 shows an example of an SM rule.

Table 1.
Semantic information elements (Zhang and El-Gohary 2016b; 2013).

Semantic information element	Definition	Type
Subject	An ontology concept that describes a "thing" (e.g., building object, space) that is subject to a particular regulation or norm.	Simple and rigid SIE
Compliance checking attribute	An ontology concept that describes a specific characteristic of a "subject" by which its compliance is assessed.	Simple and rigid SIE
Deontic operator indicator	A term or phrase that indicates the deontic type of the requirement (i.e., whether it is an obligation, permission, or prohibition).	Simple and rigid SIE
Quantitative relation	A term or phrase that defines the type of relation for the quantity (e.g., "increase" is a quantitative relation).	Simple and rigid SIE
Comparative relation	An ontology relation that is commonly used for comparing quantitative values (i.e., comparing an existing value to a required minimum, maximum, or exact value), including "greater than or equal to," "greater than," "less than or equal to," "less than," and "equal to."	Simple and rigid SIE
Quantity value	A data value (or a range of values) that defines the quantified requirement.	Simple and rigid SIE
Quantity unit	The unit of measure for a "quantity value."	Simple and rigid SIE
Quantity reference	A term or phrase that refers to another quantity (which includes a value and a unit).	Simple and rigid SIE
Subject restriction	A term, phrase, or clause (which is composed of one or more concepts and/or relations) that places a constraint on the "subject."	Complex and flexible SIE
Quantity restriction	A term, phrase, or clause (which is composed of one or more concepts and/or relations) that places a constraint on the "quantity."	Complex and flexible SIE

Original Text:

The thickness of concrete floor slabs supported directly on the ground shall not be less than 31/2 inches.

Text with Features[1]:

The thickness (ontology concept "quantity") of concrete floor slabs (ontology concept "building element") supported directly on the ground shall (POS tag "MD" for modal verb) not (gazetteer list "Negation") be (POS tag "VB" for base form verb) less than (gazetteer list "Comparative relation") 31(POS tag "CD" for cardinal number)/(POS tag "Slash" for a slash)2(POS tag "CD") inches (gazetteer list "Unit").

IE Rules:

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."

If ontology concept "building element" or "space" or "room" is matched, extract the matched text as an instance for "subject."

If ontology concept "quantity" is matched, extract the matched text as an instance for "compliance checking attribute."

If "CD + Slash + CD + Unit" is matched, extract the text matched with "CD + Slash + CD" as an instance of "quantity value," extract the text matched with "Unit" as an instance of "quantity unit."

Extracted Instances:

"thickness" as a "compliance checking attribute"

"concrete floor slab" as a "subject"

"not less than" as a "comparative relation"

"31/2" as a "quantity value"

"inches" as a "quantity unit"

1. For simplicity only features related to the IE rules below are displayed.

Fig. 5. Sample information extraction rules and extracted instances.

Original Text

The thickness of exterior basement walls and foundation walls shall be not less than 71/2 inches.

Information Tags

- Semantic information tags: 's' for subject, 'a' for compliance checking attribute, 'c' for comparative relation, 'v' for quantity value, 'u' for quantity unit;
- Syntactic information tags: 'CC' for conjunctive term, 'CD' for cardinal number, 'IN' for preposition, 'JJ' for adjective, 'MD' for modal verb, 'TO' for literal "to," 'VB' for base form verb, 'VBN' for past participle verb;
- Combinatorial information tags: 'dpvr' for directional passive Verbal relation, which is the combination of "past participle verb" (POS tag "VBN") and "preposition" (POS tag "IN").

Information Tuples Using Three Types of Information Tags[1]

[('thickness', 4, 9, 'a'), ('thickness', 4, 9, 'cr'), ('of', 14, 2, 'OF'), ('of', 14, 2, 'IN'), ('exterior basement walls', 17, 23, 's'), ('exterior', 17, 8, 'cr'), ('basement', 26, 8, 'cr'), ('walls', 35, 5, 'cr'), ('and', 41, 3, 'CC'), ('foundation walls', 45, 16, 's'), ('foundation', 45, 10, 'cr'), ('walls', 56, 5, 'cr'), ('shall', 62, 5, 'MD'), ('be', 68, 2, 'VB'), ('not', 71, 3, 'n'), ('less_than', 75, 9, 'c'), ('less', 75, 4, 'JJR'), ('than', 80, 4, 'IN'), ('71/2', 85, 4, 'v'), ('71/2', 85, 4, 'CD'), ('inches', 90, 6, 'u'), ('inches', 90, 6, 'cr')]

Information Tuples with Conflict Resolution Rules Applied[1]

[('thickness', 4, 9, 'a'), ('of', 14, 2, 'OF'), ('exterior basement walls', 17, 23, 's'), ('and', 41, 3, 'CC'), ('foundation walls', 45, 16, 's'), ('shall', 62, 5, 'MD'), ('be', 68, 2, 'VB'), ('not', 71, 3, 'n'), ('less_than', 75, 9, 'c'), ('71/2', 85, 4, 'v'), ('inches', 90, 6, 'u')]

Logic Components after Applying Semantic Mapping Rules[2]

thickness(Thickness),(exterior_basement_wall(Exterior_basement_wall);foundation_wall(Exterior_basement_wall)),has(Exterior_basement_wall,Thickness),not less_than(Thickness,quantity(71/2,inches))

Logic Rules Generated by Logic Rule Generator (Partial)[2]

Primary Logic Clause

compliance_thickness_of_Exterior_basement_wall81(Exterior_basement_wall):-
thickness(Thickness),(exterior_basement_wall(Exterior_basement_wall);foundation_wall(Exterior_basement_wall)),has(Exterior_basement_wall,Thickness),not less_than(Thickness,quantity(71/2,inches)).

Activation Condition Logic Clause

...thickness(Thickness),(exterior_basement_wall(X);foundation_wall(X)),has(X,Thickness)->
check_thickness_of_Exterior_basement_wall81(X);true,...

Compliance Checking Consequence Logic Clause

check_thickness_of_Exterior_basement_wall81(X):-(compliance_thickness_of_Exterior_basement_wall81(X)->
writeln((X,is,compliant,with,section,1909-6-1,rule81));writeln((X,is,noncompliant,with,section,1909-6-1,thickness,should,be,not,less_than,71/2,inches,rule82))).

1. Each tuple includes four elements: the information instance, its location (the starting point in the sentence), its length (in number of letters), and its information tag.
2. In this logic syntax, comma represents conjunction, semicolon represents disjunction, "not" represents negation, ":-" represents implication, predicate takes the form of pred(arg1,arg2,…), rule takes the form of predh(arg1,arg2,…) :- pred1(arg1,arg2,…), pred2(arg1,arg2,…)…, predn(arg1,arg2,…).

Fig. 6. An Example to illustrate regulatory information transformation.

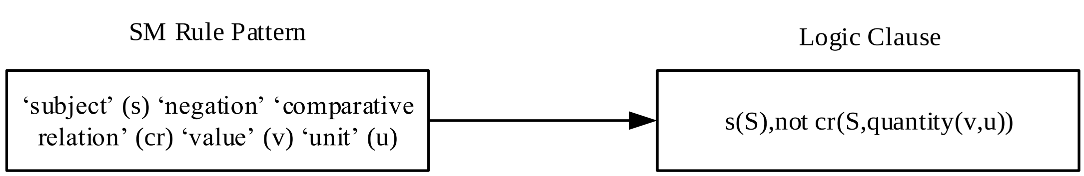

Fig.7. An example of a semantic mapping rule.

The logic rule generator generates three types of logic rules based on the logic components: primary logic clauses, activation condition logic clauses, and compliance checking consequence logic clauses. A primary logic clause is the main representation of a requirement; the premise of the rule represents the conditions of a requirement and the conclusion of the rule represents the consequent result (i.e., the compliance with the requirement). For example, in the primary logic clause in Fig. 6, the logic components to the right of ":-" represent the conditions of the wall thickness requirement (for exterior basement walls and foundation walls) and the logic components to the left of ":-" represent the conclusion of that requirement. An activation condition logic clause represents the conditions that activate the checking of a requirement, which are the existence of the corresponding information in the BIM (e.g., the existence of exterior basement wall or foundation wall and thickness information for the example in Fig. 6). Activation conditions are used to help prevent missing information from leading to false positives because missing information would lead to failure in activation. A compliance checking consequence logic clause represents the consequences of the compliance checking result (compliance or noncompliance). For example, if the result is noncompliant, a corrective suggestion is provided (e.g., "thickness should be not less than 71/2 inches," as per Fig. 6).

4.2 Design information extraction and transformation module

The design information extraction and transformation module is composed of two main processes: BIM information extraction and BIM information transformation.

BIM information extraction utilizes EXPRESS data processing techniques in a BIM information extraction algorithm to extract all entities and their attributes in an IFC file into information tuples based on their metadata, in a recursive and exhaustive manner. The information tuples store information for each entity, the attributes of the entity, and the values of the attributes of the entity,

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

to prepare for the following transformation process. The BIM IE algorithm exhaustively extracts the values (e.g., '2O2Fr$t4X7Zf8NOew3FNld') for each attribute (e.g., global ID) of each entity (e.g., wall). Recursion is used in two ways (as illustrated in Fig. 8): (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. For example, when a “door” entity is being extracted, not only the explicit attributes “overall height” and “overall width” are extracted, but all the following explicit attributes that belong to the supertypes of “door” are extracted too: “global ID,” “owner history,” “name,” “description,” “object type,” “object placement,” “representation,” and “tag;” and (2) if an attribute is of an aggregation data type (i.e., aggregation of multiple attributes), then the member attributes of the aggregation are recursively accessed for extracting their values. For example, because the attribute “related objects” of a “rel associates material” is of an aggregation data type (i.e., set data type in this case), when a “related objects” instance is being processed, each of its member objects is accessed recursively for extracting their values. The late binding data access method in JSDAI is used to support the entity and attribute extraction in the BIM IE algorithm. Late binding accesses each entity and attribute using standard access methods in Java.

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

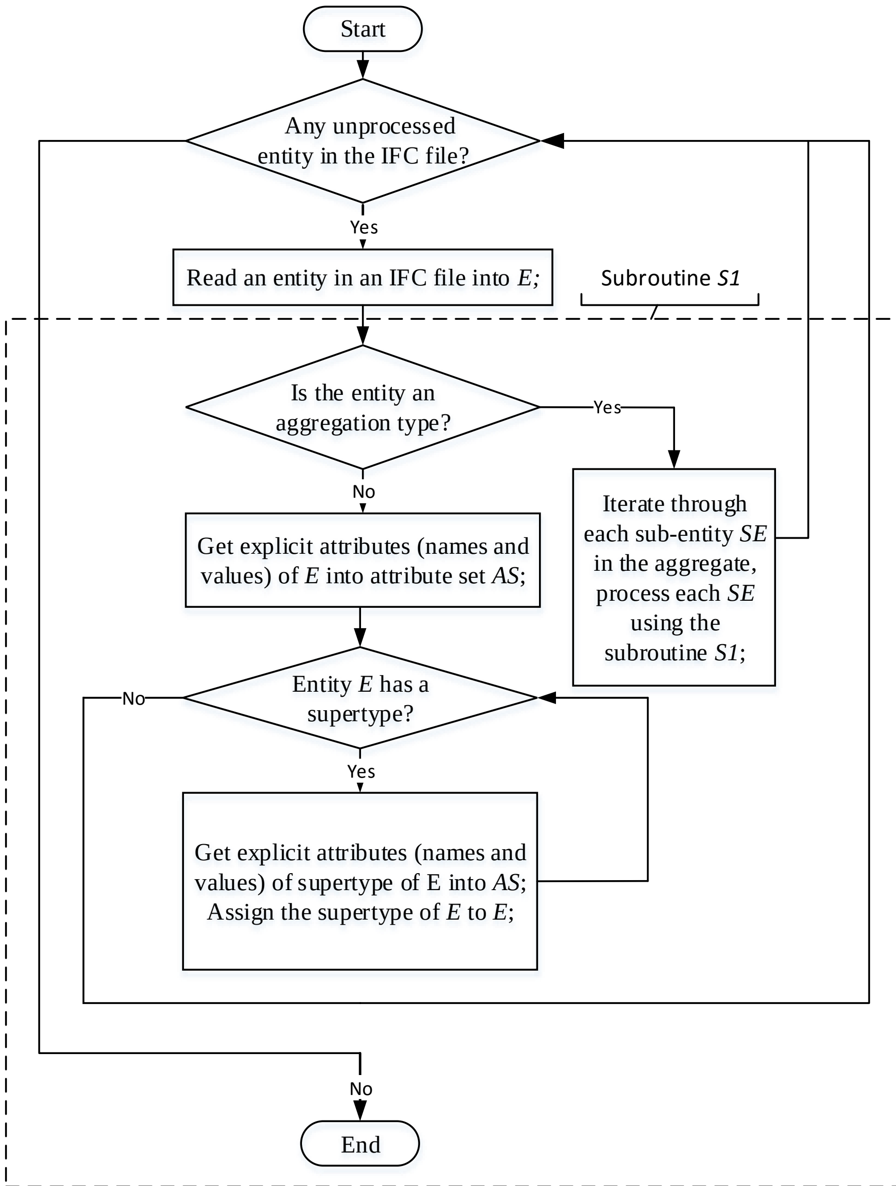

Fig. 8. The BIM IE Algorithm based on two recursive processes.

BIM information transformation transforms the extracted BIM information in the information tuples into logic facts (concept facts and relation facts) in two steps: initial transformation and alignment transformation. Initial transformation transforms the extracted entities and their attributes into concept facts and relation facts using three main initial transformation rules. These rules transform elements in the entities, attributes, and values into predicate names or arguments based on their metadata. For example, the first initial transformation rule in Fig. 9 converts a line in IFC data with referenced attribute values into logic facts. After initial transformation, alignment transformation further transforms the generated logic facts into a logic fact representation that is aligned with the predicates in the logic rules (that represent the corresponding regulatory requirements). A set of semantic transformation (ST) rules are used in the alignment transformation step. For example, Fig. 9 shows a set of logic facts after initial transformation and after alignment transformation using two ST rules. Compared to the logic facts before alignment transformation, the logic facts after alignment transformation are more easily understandable and aligned with the logic rules.

IFC Data

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

Initial Transformation Rules

(1) an entity is transformed into a concept fact (i.e., a predicate) by using the name of the entity as the name of the predicate, and using the name of the entity concatenated with the ID of the entity as the argument (i.e., an entity constant) of the predicate.
(2) an attribute of an entity is transformed into a relation fact (i.e., a predicate), using the name of the attribute preceded by "has_" as the name of the predicate, using the corresponding entity constant as the first argument of the predicate, and using the value of the attribute as the second argument of the predicate (if the value is not a reference to another entity).
(3) if the value of an attribute is a reference to another entity, then the referred entity constant is used as the second argument of the predicate.

Logic Facts After 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).
```

Alignment Transformation Rules

```
acc_bi_relation(X), has_type_name(X,Name),
has_relating_element(X,Y),has_related_element(X,Z) → relation(Name, Y, Z).
relation(Name, Y, Z) → Name(Y,Z).
```

Logic Facts After Alignment Transformation

```
has(wall39594,height39592).
```

Fig. 9. An example to illustrate BIM information extraction and transformation.

4.3 Compliance reasoning module

The compliance reasoning module utilizes B-Prolog's reasoner to reason about the logic rules and the logic facts and generate compliance checking reports. A set of functional built-in logic clauses were developed and embedded into the system to provide basic arithmetic functions (e.g., unit conversion) and define the sequence of execution/checking. For execution, the user specifies the list of subjects (e.g., walls and doors) or subjects and attributes (e.g., walls and their heights) to check, and accordingly the subjects in the specified list are sequentially checked one by one. By default, a "select all" option is used.

5 System implementation

The proposed SNACC system was implemented in a proof-of-concept prototype. The main platform of the prototype was built using Java programming language (Java Standard Edition Development Kit 6u45). The regulatory information extraction algorithm was implemented using GATE's Processing Resources and Java programs. The following Processing Resources were used: (1) the English Tokenizer, Sentence Splitter, POS Tagger, and Gazetteer in the A Nearly-New Information Extraction (ANNIE) system for tokenization, sentence splitting, POS tagging, and gazetteer compiling, (2) the Morphological Analyzer for morphological analysis, (3) the Flexible Gazetteer for generating semantic features based on the ontology, and (4) the Java Annotation Patterns Engine (JAPE) rules for encoding the IE rules. The information extraction algorithm interacts with the Processing Resources using GATE's API 7.0.

The regulatory information transformation algorithm was implemented using Python programming language (Python 2.7.3). The SM rules and CR rules were coded as Python conditional statements. The "re" module (i.e., regular expression module) in Python was used for both extracting the syntactic and semantic features from the information tuples and conducting pattern matching. The information transformation algorithm interacts with the other modules of the SNACC system (in Java) through Jython 2.2.1.

The BIM information extraction and transformation algorithms were implemented in Java programs and B-Prolog rules, respectively. The JSDAI runtime (JSDAI 4.3.0) was used to access the information in IFC-based BIMs (IFC files) for entity and attribute extraction. String processing methods in Java were used for initial transformation. Static rules and dynamic rules in B-Prolog were used for alignment transformation. Static rules are rules that only use static predicates. Dynamic rules are rules that use at least one dynamic predicate. A static predicate is a predicate that cannot be updated during execution whereas a dynamic predicate is a predicate that can be updated during execution. The rules for entity extraction, attribute extraction, and initial transformation were coded as Java conditional statements. The rules for alignment transformation (i.e., ST rules) were coded as B-Prolog rules.

The logic-based automated reasoning algorithm was implemented in Java. The functional built-in logic clauses were encoded in B-Prolog. The automated reasoning algorithm interacts with the logic clauses and logic reasoner through B-Prolog's bi-directional interface 7.8 with Java programming language.

The graphical user interface of the SNACC system is shown in Fig. 10. As shown in Fig. 10, the SNACC system requires the download of the GATE tool and the availability of a building ontology to execute the regulatory information extraction and transformation algorithms. A user could then select the regulatory document (.txt file) and the BIM (.ifc file) for automated compliance checking. The information extraction and information transformation algorithms for regulatory information and design information could be executed in parallel. After all information have been extracted and transformed, pressing the "check compliance" button activates the automated reasoning process using B-Prolog. The compliance checking results are then automatically displayed to the user in the text field of the graphical user interface (as shown in Fig. 10).

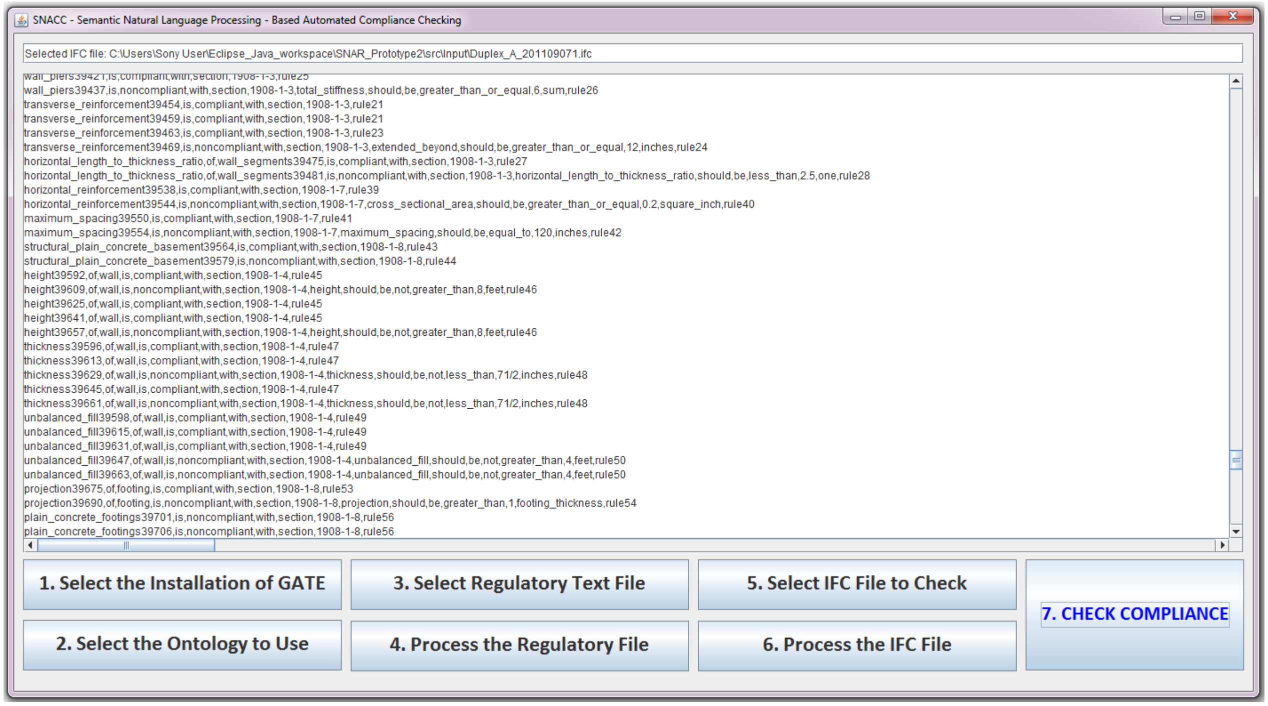

Fig. 10. Graphical user interface of the SNACC system.

6 System testing

The SNACC system was tested in checking the compliance of a BIM test case with Chapter 19 of IBC 2009. IBC was selected because it is predominantly adopted in the United States. Chapter 19 was then randomly selected. For the test case, it was developed based on the Duplex Apartment Project from buildingSMARTalliance of the National Institute of Building Sciences (East 2013). Design information were added in the BIM model, based on an extended version of the IFC_2X3_TC1 schema (BuildingSmart 2014) (Zhang and El-Gohary 2016a). The test case included design information for each provision in Chapter 19 of IBC 2009. The design information included both compliant and noncompliant design information. If a provision had more than one requirement, then compliant and noncompliant design information for each requirement was included. For example, the following regulatory provision (*RP1)* is a complex provision that contains three quantitative requirements: "*In dwellings assigned to Seismic Design Category D or E, the height of the wall shall not exceed 8 feet (2438 mm), the thickness shall not be less than 71/2 inches (190 mm), and the wall shall retain no more than 4 feet (1219 mm) of unbalanced fill.*" Thus, five information sets were created for *RP1* which correspond to the scenarios that (1) only height is noncompliant, (2) only thickness is noncompliant, (3) only unbalanced fill is noncompliant, (4) all three attributes are noncompliant, and (5) no attributes are noncompliant.

7 Results and discussion

The ACC prototype system was evaluated using precision, recall, and F1-measure of noncompliance detection. 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. F1-measure is the harmonic mean of precision and recall. A manually-developed gold standard was used for the evaluation. A gold standard refers to a benchmark against which testing results are compared for evaluation. The gold standard includes the ground truth of compliant and noncompliant instances.

The testing results are summarized in Table 2. As shown in Table 2, the recall, precision, and F1-measure of noncompliance detection is 98.7%, 87.6%, and 92.8%, respectively. The relevant provision numbers and rule numbers for the compliant and noncompliant instances were also correctly reported. For each noncompliance instance, a suggestion on how to fix the noncompliance case was also correctly reported (partially shown in Fig. 10).

Table 2.
Noncompliance detection testing results.

Parameter/measure	Result
Number of noncompliance instances in gold standard	79

Number of noncompliance instances detected	89
Number of noncompliance instances correctly detected	78
Recall of noncompliance detection	98.7%
Precision of noncompliance detection	87.6%
F1-measure of noncompliance detection	92.8%

These high performance results show that the proposed ACC system is promising. In addition, the fact that the proposed ACC system achieved higher recall (98.7%) than precision (87.6%) 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.

An error analysis was also conducted to identify the sources of the errors in noncompliance detection. The noncompliance detection errors originated from errors in regulatory information extraction and regulatory information transformation; there were no errors in BIM information extraction, BIM information transformation, or compliance reasoning. The errors were attributed to errors made by GATE's processing resources, limitations of rules used in regulatory information extraction and information transformation, and limitations of the state-of-the-art NLP techniques [e.g., state-of-the-art Part-of-Speech (POS) tagging has an accuracy of around 97% (Manning 2011)]. For example, "concrete floor slab" was not successfully extracted as the subject (i.e., a false negative) for the following requirement because of errors made by GATE's processing resources: "The thickness of concrete floor slabs supported directly on the ground shall not be less than 31/2 inches (89 mm)" (Provision 1910.1 of IBC 2009).

8 Contribution to the body of knowledge

This research contributes to the body of knowledge in three main ways. First, this research offers a novel system for fully-automated checking of building information models for compliance with building codes. The proposed system goes beyond the current state-of-the-art of ACC by allowing fully-automated (1) extraction of both regulatory and design information from regulatory documents and IFC-based BIM models, respectively, and (2) alignment of the representations of these two sets of information, so that they can be interpreted together in one system. Second, this research offers integrated NLP and first order logic methods for automatically extracting regulatory information from regulatory documents and automatically representing the extracted information in an ACC first order logic-based representation that is used in automated ACC logic reasoning. The proposed methods/algorithms offer a novel way for, both, deep information extraction (i.e., full-sentence analysis to capture the entire meaning of a provision) and generalized and flexible ACC representation; both – together – enable the extraction and representation of information even in long and complex provisions, which is important to sustain utility and robustness of ACC system performance across different types of regulatory documents and different types of provisions. Third, this research offers a novel combination of NLP techniques with both semantic analysis and logic-based reasoning into one computational framework. In this research, a set of information extraction, information transformation, and automated reasoning algorithms are effectively implemented into one proof-of-concept ACC system. The combined performance of all algorithms, into the system, shows high automated noncompliance detection performance (98.7%, 87.6%, and 92.8% recall, precision, and F1-measure, respectively).

9 Conclusions

This paper presented a unified system that integrates a set of techniques and algorithms for automatically checking the compliance of BIM-based building designs with building codes. The proposed system offers a fully-automated approach to ACC in construction. The approach relies on the use of a set of computational techniques in an integrated manner, in one unified system. The techniques include NLP, EXPRESS data processing, and logic reasoning, which are

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

collectively used for automated information extraction, automated information transformation, and automated compliance reasoning. The automation is facilitated by semantic, logic-based representations that are generalized and flexible.

The system is composed of three main modules: (1) a regulatory information extraction and transformation module, which utilizes semantic natural language processing algorithms to automatically extract regulatory information from building codes and transform the extracted information into logic rules, (2) design information extraction and transformation module, which utilizes EXPRESS data processing-based algorithms to automatically extract design information from building information models and transform the extracted information into logic facts, and (3) compliance reasoning module, which utilizes semantic-based logic reasoning algorithms to automatically reason about the compliance of the logic facts with the logic rules. The algorithms were implemented in different programming languages and integrated into one proof-of-concept prototype system (the SNACC system). The integration is facilitated by the choice of a semantic representation, a logic representation, a modular system architecture, and an architecture-neutral platform.

The SNACC system was tested in checking the compliance of a BIM test case with Chapter 19 of IBC 2009. A recall of 98.7%, a precision of 87.6%, and an F1-measure of 92.8% in noncompliance detection were achieved. The high performance results, of all algorithms when combined into one unified system, show that the proposed ACC system is promising. In addition, the higher recall shows the suitability of the proposed system for ACC, because recall is more critical than precision for noncompliance detection.

10 Limitations and future work

As mentioned above, at this point, the system proposed in this paper focused on quantitative requirements. It could be extended to support the checking of other types of requirements such as existential requirements (i.e., rules that require the existence of certain building elements, etc.), but it cannot go beyond the limitations of machine intelligence or represent and reason with rules that require human judgement by nature.

Also, in spite of the authors' firm belief in automation and early evidence of low consistency in manual noncompliance checking (Fiatech 2014), how the automated information extraction and transformation approach proposed in this paper compares to the state-of-the-art semi-automated information extraction and transformation approaches (e.g., such as RASE-based or SMARTcodes-based, which rely on manual annotation) in terms of accuracy and efficiency requires further investigation.

As part of their future/ongoing research work, the authors will test the proposed ACC system on more building code chapters and more BIM test cases. In addition, other types of requirements (e.g., existential requirements) will be tested, and different ways of handling information incompleteness cases during ACC will be proposed and tested.

In future research – by the authors or the larger research community, the proposed information extraction and transformation algorithms could also be applied to other logic-based representations such as SWRL and N3Logic. In this case the JSDAI-based BIM information processing can be partially replaced by existing conversion methods such as those in Pauwels and Terkaj (2016) and Beetz et al. (2009). However, in this case, further semantic transformation of BIM information would still be needed to align the concept representations of the design information to those of the

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

regulatory information. Similarly, further research could be conducted to study how to best link the proposed algorithms with OWL representations and other semantic modeling approaches and assess the advantages and limitations of the proposed methods in this context. The authors expect that the proposed information extraction, information transformation, and automated reasoning methods would lend themselves well to such integrative efforts. However, further research is needed to study practicality, benefits, and limitations.

Acknowledgements

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

AEC3. (2012). “International Code Council.” ⟨http://www.aec3.com/en/5/5_013_ICC.htm⟩ (Jun. 30, 2016).

Baumgärtel, K., Kadolsky, M. and Scherer, R.J. (2015). “An ontology framework for improving building energy performance by utilizing energy saving regulations.” Proc., 10th European Conference on Product and Process Modelling (ECPPM), ECPPM, 519-526.

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.

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

Beach, T.H., Rezgui, Y., Li, H., and Kasim, T. (2015). "A rule-based semantic approach for automated regulatory compliance in the construction sector." *Expert Systems with Applications*, 42(12), 5219-5231.

Beetz, J., van Leeuwen, J., and de Vries, B. (2009). "IfcOWL: A case of transforming EXPRESS schemas into ontologies." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(SP01), 89-101.

Berners-Lee, T. (2005). "Status: An early draft of a semi-formal semantics of the N3 logical properties." <https://www.w3.org/DesignIssues/N3Logic> (Jun. 9, 2016).

Boken, P., and Callaghan, G. (2009). "Confronting the challenges of manual journal entries." *Protiviti*, Alexandria, VA, 1-4.

BuildingSmart. (2014). "Industry Foundation Classes (IFC) data model." <http://www.buildingsmart-tech.org/specifications/ifc-overview> (Jan 19, 2015).

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

Choi, J., Choi, J., and Kim, I. (2014). "Development of BIM-based evacuation regulation checking system for high-rise and complex buildings." *Autom. Constr.*, 46, 38-49.

City of Philadelphia. (2015). "Licenses and inspections: building permits." <https://business.phila.gov/licenses-and-inspections-building-permits/> (Sept. 4, 2015).

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

Delis, E.A., and Delis, A. (1995) "Automatic fire-code checking using expert-system technology." *J. Comput. Civ. Eng.*, 9(2), 141-156.

Department of Buildings. (2015). "Hearing on the fiscal 2016 preliminary budget and the fiscal 2015 preliminary mayor's management report." <ttp://council.nyc.gov/html/budget/2016/Pre/dob.pdf> (Sept. 4, 2015).

Dimyadi, J., and Amor, R. (2013). "Automated building code compliance checking - where is it at?" *Proc. 19th Int. CIB World Build. Congress*, Brisbane, Australia.

Dimyadi, J., Clifton, C., Spearpoint, M., and Amor, R. (2014). "Regulatory knowledge encoding guidelinens for automated compliance audit of building engineering design." *Comput. Civ. Build. Eng.* (2014), ASCE, Reston, VA, 536-543.

Dimyadi, J., Pauwels, P., Spearpoint., M., Clifton, C., and Amor, R.W. (2015). "Querying a regulatory model for compliant building design audit." *Proc., CIB W78 2015*, Conseil International du Bâtiment (CIB), Rotterdam, The Netherlands, 139-148.

East, E.W. (2013). "Common building information model files and tools." < http://www.nibs.org/?page=bsa_commonbimfiles&hhSearchTerms=%22common+and+BIM+and+file%22> (Jun. 27, 2014).

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

Fiatech Regulatory Streamlining Committee, (2012). "AutoCodes project: phase 1, proof-of-concept: final report." <http://www.fiatech.org/images/stories/techprojects/project_deliverables/Updated_project_deliverables/AutoCodesPOCFINALREPORT.pdf> (Dec. 24, 2013).

Fiatech. (2014). "Automated code plan checking tool-proof-of-concept (phase 2)." <http://www.fiatech.org/images/stories/projects/FiatechAutoCodesPh2-Report-Sept2015.pdf> (Jun. 16, 2016).

Garrett, J.H.Jr., and Palmer, M.E. (2014). "Delivering the infrastructure for digital building regulations." *J. Comput. Civ. Eng.*, 2014(28), 167-169.

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 (CIB), Rotterdam, The Netherlands.

Hodges, W. (2001). "Classical logic I - first-order logic." *Goble*, 2001, 9-32. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.4783&rep=rep1&type=pdf> (Dec. 26, 2013).

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., and Dean, M. (2004). "SWRL: A Semantic Web Rule Language Combining OWL and RuleML." < https://www.w3.org/Submission/SWRL/> (Jun. 9, 2016).

International Code Council (ICC). (2006). "2006 international building code." 2006 Int. Codes, ⟨ http://publicecodes.cyberregs.com/icod/ibc/2006f2/⟩ (Oct. 25, 2015).

ISO. (2004). "ISO 10303-11:2004 - Part 11: Description methods: The EXPRESS language reference manual." <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38047> (Dec. 05, 2014).

Java Standard Edition Development Kit 6u45. [Computer Software]. Redwood Shores, CA, Oracle.

JSDAI 4.3.0. [Computer software]. Kuenzell, Germany, LKSoftWare GmbH.

Jython 2.2.1. [Computer software]. <http://www.jython.org/> (May 02, 2015).

Kasim, T., Li, H., Rezgui, Y., and Beach, T. (2013). "Automated sustainability compliance checking process: proof of concept." *Proc., 13th Int. Conf. Constr. App. Vir. Real.*, Teesside University, Tees Valley, UK, 11-21.

Lau, G. T., and Law, K. (2004). "An information infrastructure for comparing accessibility regulations and related information from multiple sources." *Proc., 10th Int. Conf. on Computational Civil and Building Engineering (ICCCBE)*, ISCCBE, Hong Kong, China.

Los Angeles Times. (2015). "Public & Legal notices." <http://classifieds.latimes.com/classifieds?category=public_notice> (Sept. 4, 2015).

Manning, C.D. (2011). "Part-of-Speech tagging from 97% to 100%: is it time for some linguistics?" *Proc., 12th International Conference on Intelligent Text Processing and Computational Linguistics*, CICLing, Mexico.

Nguyen, T., and Kim, J. (2011). "Building code compliance checking using BIM technology." *Proc., 2011 Winter Simulation Conference*, Association for Computing Machinery, New York, 3400 - 3405.

Oracle. (1999). "Essentials of the Java programming language, Part 1." <http://www.oracle.com/technetwork/java/index-138747.html> (Jun. 25, 2015).

Pauwels, P., and Terkaj, W. (2016). "EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology." *Autom. Constr.*, 63(Mar. 2016), 100-133.

Pauwels, P., and Zhang, S. (2015). "Semantic rule-checking for regulation compliance checking: an overview of strategies and approaches." *Proc., CIB W78 2015*, Conseil International du Bâtiment (CIB), Rotterdam, The Netherlands, 619-628.

Pauwels, P., Van Deursenc, D., Verstraetena, R., De Rooc, J., De Meyera, R., Van de Wallec, R., Van Campenhoutb, J. (2011). "A semantic rule checking environment for building performance checking." *Autom. Constr.*, 20(5), 506-518.

Portoraro, F. (2011). "Automated reasoning." *The Stanford encyclopedia of philosophy (Summer 2011 Edition)*, Edward N. Zalta (ed.)

The published version is found here: http://dx.doi.org/10.1016/j.autcon.2016.08.027
The demo video can be accessed here: http://homepages.wmich.edu/~jyb5534/research/Research.html

<http://plato.stanford.edu/archives/sum2011/entries/reasoning-automated/> (Dec. 26, 2014).

Python v2.7.3 [Computer software]. Beaverton, OR, Python Software Foundation.

Qi, J., Issa, R., Hinze, J., and Olbina, S. (2011). "Integration of safety in design through the use of building information modeling." *Int. Workshop on Computing in Civil Engineering 2011*, ASCE, Reston, VA, 698-705.

Saint-Dizier, P. (1994). "Advanced logic programming for language processing." Academic Press, San Diego, CA.

Solihin, W., and Eastman, C. (2015). "A knowledge representation approach to capturing BIM based rule checking requirements using conceptual graph." *Proc., CIB W78 2015*, Conseil International du Bâtiment (CIB), Rotterdam, The Netherlands, 686-695.

State of New Jersey. (2014). "Plan review instructions." < http://www.state.nj.us/dca/divisions/codes/forms/pdf_bcpr/pr_app_guide.pdf> (Sept. 4, 2015).

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.

Yurchyshyna, A., Faron-Zucker, C., Thanh, N.L., and Zarli, A. (2010). "Adaptation of the domain ontology for different user profiles: application to conformity checking in construction." *Web Information Systems and Technologies, Lecture Notes in Business Information Processing*, 45, 128-141.

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. (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.

Zhang, J., and El-Gohary, N.M. (2016a). "Extending building information models semi-automatically using natural language processing techniques." *J. Comput. in Civ. Eng.*, 10.1061/(ASCE)CP.1943-5487.0000536, C4016004.

Zhang, J., and El-Gohary, N.M. (2016b). "Semantic-based logic representation and reasoning for automated regulatory compliance checking." *J. Comput. in Civ. Eng.*, 10.1061/(ASCE)CP.1943-5487.0000583, 04016037. Zhang, S., Teizer, J., Lee, J., Eastman, C.M., and Venugopal, M. (2013). "Building information modeling (BIM) and safety: automatic safety checking of construction models and schedules." *Autom. Constr.*, 29(2013), 183-195.

Zhong, B., Ding, L., Luo, H., Zhou, Y., Hu, Y., and Hu, H. (2012). "Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking." *Autom. Constr.*, 28, 58-70.

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> (Dec. 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.