The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

Automated Information Transformation for Automated Regulatory Compliance Checking in Construction

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

Abstract

To fully automate regulatory compliance checking of construction projects, we need to automatically extract regulatory requirements from various construction regulatory documents, and transform these requirements into a formalized format that enables automated reasoning. To address this need, the authors propose an approach for automatically extracting information from construction regulatory textual documents and transforming them into logic clauses that could be directly used for automated reasoning. This paper focuses on presenting the proposed information transformation (ITr) methodology and the corresponding algorithms. The proposed ITr methodology utilizes a rule-based, semantic natural language processing (NLP) approach. A set of semantic mapping (SeM) rules and conflict resolution (CoR) rules are used to enable the automation of the transformation process. Several syntactic text features (captured using NLP techniques) and semantic text features (captured using an ontology) are used in the SeM and CoR rules. A bottom-up method is leveraged to handle complex sentence components. A "consume and generate" mechanism is proposed to implement the bottom-up method and execute the SeM rules. The proposed ITr algorithms were tested in transforming information instances of quantitative requirements, which were automatically extracted from the International Building Code 2009, into logic clauses. The algorithms achieved 98.2% and 99.1% precision and recall, respectively, on the testing data.

[1] Graduate Student, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, 205 N. Mathews Ave., Urbana, IL 61801.

[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.

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

CE Database subject headings: Project management; Construction management; Information management; Computer applications; Artificial intelligence.

Author keywords: Automated compliance checking; Automated information extraction; Automated information transformation; Natural language processing; Semantic systems; Automated construction management systems.

Introduction

Construction projects must comply with a host of regulations. The manual process of compliance checking is, thus, time-consuming, costly, and error-prone (Han et al. 1998; Nguyen 2005; Zhang and El-Gohary 2013c). Automated compliance checking (ACC), as an alternative to manual checking, is expected to reduce the time, cost, and errors of compliance checking (CC) (Tan et al. 2010; Salama and El-Gohary 2013b). In addition, ACC has many other potential benefits, such as: (1) allowing earlier identification of potential non-compliance instances, which could save significant time and cost caused by design modification and/or rework (Ding et al. 2006); (2) promoting the adoption of building information modeling (BIM) and increasing the cumulative benefits of adopting BIM, since BIM would enable ACC (Pocas Martins and Abrantes 2010); (3) enabling more efficient incorporation of stakeholder input into project design and exploration of what-if design scenarios, since a designer would be better able to experiment with different design options and check their compliance in a more time-efficient manner (Niemeijer et al. 2009); and (4) reducing violations of regulations due to easier and more frequent CC (Zhong et al. 2012).

Due to the many anticipated benefits of ACC, many efforts were undertaken in the area of ACC in construction. The start of these efforts could be dated back to the 1960s, when Fenves et al.

(1969) formalized the American Institute of Steel Construction (AISC) specifications into decision tables. These efforts took various approaches to ACC and focused on various ACC purposes (or subdomains). For example, Garrett and Fenves (1987) proposed a strategy to represent design standards using information networks and represent design component properties using data items for ACC of structural designs; Ding et al. (2006) proposed an approach to represent building codes using object-based rules and represent designs using an Industry Foundation Classes (IFC)-based internal model for ACC of accessibility regulations; Tan et al. (2010) proposed an approach to represent building codes and design regulations using decision tables and incorporate simulation results in building information models for ACC of building envelope design; the CORENET (Construction and Real Estate NETwork) project of Singapore (Khemlani 2005) used an approach to represent design information using semantic objects in the FORNAX library (i.e., a C++ library) and represent regulatory rules using properties and functions in FORNAX objects for ACC of building control regulations, barrier free access, and fire code, etc.; and the SMARTcodes project (ICC 2012) of the International Code Council (ICC) used an approach to represent ICC codes in computer-processable tuple format and represent designs using an IFC-based model for ACC of designs with ICC codes. These efforts have all been very important in supporting ACC, and have shown the possibilities of ACC through different system designs and implementations. However, despite their importance, these efforts are limited in their automation capability; existing ACC efforts/systems still require manual effort for the extraction of regulatory requirements from regulatory documents and encoding them in a computer-processable format (Zhong et al. 2012; Zhang and El-Gohary 2013c). To achieve full automation of ACC, this extraction and encoding process needs to be fully automated.

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

To address this gap, the authors are proposing a new approach for automated rule extraction and formalization for supporting ACC (Zhang and El-Gohary 2013a; Zhang and El-Gohary 2013b). The approach utilizes semantic modeling and semantic Natural Language Processing (NLP) techniques (for both information extraction and information transformation) to facilitate automated textual regulatory document analysis (e.g., code analysis) and processing for extracting requirements/rules from these documents and formalizing these requirements/rules in a meaning-rich, computer-processable format. The approach involves developing a set of algorithms and combining them into one computational platform: (1) machine-learning-based algorithms for text classification (TC), (2) rule-based, semantic NLP algorithms for information extraction (IE), and (3) rule-based, semantic NLP algorithms for information transformation (ITr). This paper focuses on presenting the methodology and algorithms for ITr.

Proposed Approach for Automated Rule Extraction and Formalization for Automated Compliance Checking

Proposed Approach

A five-phase, iterative approach for automatically extracting regulatory requirements/rules from textual regulatory documents and formalizing these requirements in a logic format for further automated reasoning is proposed (Figure 1). The five phases are: text classification (TC), information extraction (IE), information transformation (ITr), implementation, and evaluation. TC, IE, and ITr are the main processing phases.

Insert Figure 1

TC recognizes relevant sentences in a regulatory text corpus. Relevant sentences are the sentences that contain the types of requirements that are relevant for an ACC scenario (e.g., environmental requirements in the scenario of environmental CC). Target information in those relevant sentences are extracted and transformed in later IE and ITr processes. The TC process, thus, filters out irrelevant sentences, thereby saving unnecessary processing of irrelevant sentences. Such filtering also avoids unnecessary extraction and transformation errors that may be caused by the processing of irrelevant sentences. The presentation of the TC algorithms and results is outside the scope of this paper. For further details on the authors' work in TC, the reader is referred to Salama and El-Gohary (2013a).

IE recognizes the words and phrases in the relevant sentences that carry target information, extracts information from these words/phrases, and labels them with pre-defined information tags. An information tag is a symbol/name indicating a certain type of meaning. For example, the information tag 'subject' carries the semantic meaning that the information instance is a "thing" (e.g., building object) that is subject to a particular regulation or norm; while the information tag 'JJ' carries the syntactic meaning that the information instance is an adjective that describes a noun as a modifier. Target information is the information needed to check a specific type of regulatory requirement. For example, for quantitative requirements, the quantified values/measurements of specific properties/attributes are target information. For IE by itself, a seven-phase, iterative methodology is utilized. In the IE methodology, a set of pattern-matching-based IE rules are used. Both syntactic (i.e., related to syntax and grammar, such as part-of-speech (POS) tags) and semantic (i.e., related to context and meaning, such as ontology concepts and relations) text features are used in the IE rules. The presentation of the IE algorithms and results is outside the scope of this paper. For further details on the authors' work in the area of IE, the reader is referred to Zhang and El-Gohary (2013c).

ITr takes the extracted information instances and transforms them into logic clauses (i.e., logic statements that can be further used in logic programs) using a set of pattern-matching-based rules.

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

Two types of rules are utilized for ITr: semantic mapping (SeM) rules and conflict resolution (CoR) rules. Several syntactic and semantic text features are used in the rules. A bottom-up method is utilized to handle complex sentence components. A “consume and generate” mechanism is proposed to implement the bottom-up method and execute the SeM rules. The following sections present and discuss the proposed ITr methodology in more detail. The experimental implementation of the methodology in processing quantitative requirements from Chapter 19 of the International Building Code (IBC) 2009 is also presented.

Comparison to the State-of-the-Art

In recent years, a number of research efforts, in domains such as software engineering (Breaux and Anton 2008; Kiyavitskaya et al. 2008) and legal compliance (Wyner and Peters 2011), have been studying the extraction of regulatory rules from textual documents. Most of these efforts (1) require manual annotation or mark-up of textual documents; and (2) aim at processing text at a coarser granularity level, i.e., process text into text segments rather than term-level concepts/relations. On the other hand, the proposed approach (1) does not require manual annotation or mark-up of textual documents; and (2) aims at processing text into concepts and relations at the term level (i.e., aims at performing a deeper level of NLP). To the best of the authors’ knowledge, the only work that has taken a somewhat similar approach to the proposed one– since it also does not require manual annotation/mark-up and aims at term-level processing, in addition to utilizing a semantic and logic-based approach – is that by Wyner and Governatori (2013). Wyner and Governatori (2013) have conceptually explored and analyzed the use of semantic parsing and defeasible logic for regulatory rule representation. In comparison, the proposed approach (1) utilizes both syntactic and semantic text features in an integrated way rather than utilizing only semantic information: the use of syntactic text features in addition to

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

semantic ones allows for handling more complex expressions, (2) uses a domain ontology for capturing domain-specific semantic information rather than using generic semantic information produced through generic semantic parsing: capturing and using semantic text features based on domain-specific meaning allows for unambiguous interpretation of concepts/relations/terms (e.g., “bridge” as an infrastructure instead of the card game) and identification of implicit semantic relations (e.g., “fly ash” is a type of “cementitious material”), (3) uses first order logic (FOL) rather than defeasible logic: FOL is the most widely used in automated reasoning and has been extensively verified for expressivity and simplicity, and (4) has advanced to the stages of implementation, testing, and evaluation: this allows for assessing the validity of the proposed approach using measures of precision and recall.

Background

Natural Language Processing (NLP)

NLP is a subfield of artificial intelligence (AI) that aims at making natural language text or speech computer-understandable, so that the text or speech could be processed by computers in a human-like manner (Cherpas 1992). Examples of NLP-enabled applications include automated natural language translation and automated text summarization (Marquez 2000). Examples of NLP subtasks include tokenization, POS tagging, semantic role labeling (Gildea and Jurafsky 2002), and named entity recognition (Roth and Yih 2004). NLP tasks may take two main approaches: a machine learning (ML)-based approach or a rule-based approach. A ML-based approach utilizes ML algorithms for text processing (e.g., Pradhan et al. 2004), whereas a rule-based approach utilizes manually-coded rules (e.g., Soysal et al. 2010). Rule-based methods require more human effort for rule development, but tend to show better text processing performance (Crowston et al. 2010). From another viewpoint, NLP approaches could be either

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

shallow or deep. Shallow NLP conducts partial analysis of a sentence or extracts partial, specific information from a sentence (e.g., entities or concepts). Deep NLP aims at full sentence analysis towards capturing the entire meaning of a sentence (Zouaq 2011). The state-of-the-art in NLP has achieved reasonable performances for shallow NLP tasks, whereas it is still being challenged by deep NLP tasks. Deep NLP requires elaborate knowledge representation and reasoning which remains to be a challenge for AI (Tierney 2012).

In the construction domain, there has been a number of important research efforts that have utilized NLP techniques. For example, Caldas and Soibelman (2003) have conducted ML-based text classification of construction documents. For an overview of some of these efforts, the reader is referred to Zhang and El-Gohary (2013c).

Rule-Based NLP using Pattern-Matching-Based Rules

Pattern-matching-based rules are widely used in NLP tasks such as POS tagging (Abney 1997; Yin and Fan 2013), information extraction (Califf and Mooney 2003), and text understanding (Goh et al. 2006). The idea of pattern-matching-based rules is to define a set of results when the matching of a pattern of a specific sequence (or structure like a tree) of elements (e.g., characters, tokens, symbols, terms, concepts) occurs. Pattern-matching-based rules have a variety of implementations tailored to different purposes and domains. But, they all share the same rule schema of "if *pattern* then *result*" or the mapping of "from *pattern* to *result*". For example, in the proposed SeM rules, the result is the transformation of information instances into logic clause elements; while in the proposed CoR rules, the result is the deletion or conversion of certain information instances and/or their information tags to resolve conflicts.

Semantic Modeling and Semantic NLP

A semantic model aims at capturing the meanings of a domain or topic, usually in a structured manner. Ontology is a widely-used type of semantic model; it is defined as "an explicit specification of a conceptualization" (Gruber 1995). An ontology is, commonly, composed of concept hierarchies, relationships between concepts, and axioms. The axioms are used together with the concepts and relationships to define the semantic meaning of the conceptualization. An ontology is easily reusable and extendable (El-Gohary and El-Diraby 2010). The use of a semantic model could help in NLP tasks. For example, semantic-based IE has been shown to achieve better performance than syntactic-only IE (Soysal et al. 2010; Zhang and El-Gohary 2013c).

Logic-Based Information Representation and Reasoning

There are several types of formally-defined logic with varying degrees of descriptive capabilities (prepositional logic, predicate logic, modal logic, description logic, etc.). Among the different types, FOL is the most widely-used for logic-based inference-making. A Horn Clause (HC) is one of the most restricted forms of FOL. Inference-making in FOL is most efficient using HC logic clauses, because of such restricted form (Saint-Dizier 1994). A HC is composed of a disjunction of literals of which at most one is positive. All HCs can be represented as rules that have one or more antecedents (i.e., left-hand sides (LHSs)) that are conjoined (i.e., combined using 'and' operator), and a single positive consequent (i.e., right-hand side (RHS)). For example, "*compliant(T) :- thickness(T) , exterior_basement_wall(W) , has(W,T) , greater_than_or_equal(T, quantity(71/2, inches))*" is a HC; where "," is the conjunctive operator (i.e., "A , B" means "A and B") and ":-" is the implication operator (i.e., "B :- A" means "A implies B"). There are three types of HCs: (1) one or more antecedents and one consequent, (2) zero antecedents and one consequent, and (3) one or more antecedents and zero consequents. Inference-making using HCs could be automatically and efficiently conducted, which makes it suitable for supporting automated reasoning for ACC.

Proposed Information Transformation Methodology

The proposed ITr takes a rule-based, semantic NLP approach. It utilizes pattern-matching-based rules to automatically generate logic clauses based on the extracted information instances and their associated patterns of information tags. Both syntactic information tags (i.e., tags tagging syntactic text features, e.g., ‘adjective’ is represented using the POS tag ‘JJ’) and semantic information tags (i.e., tags tagging semantic text features, e.g., ‘compliance checking attribute’ is represented using the semantic tag “a”) are used in defining the patterns. A number of NLP techniques (e.g., POS tagging, term matching) are used to identify the syntactic information tags of each extracted information instance, and a semantic model (an ontology that represents domain knowledge) is used to identify the semantic information tags. The tagged information instances are transformed into HC-type logic clauses using a set of SeM rules and CoR rules. SeM rules define how to process the extracted information instances, based on their associated types of information tags and the context of the information tags, so that the extracted information instances could be transformed correctly into logic clauses. CoR rules resolve potential conflicts that may exist in the processing of different information tags. A bottom-up method is utilized to handle complex sentence components. A “consume and generate” mechanism is proposed to implement the bottom-up method and execute the SeM rules.

The following subsections present the proposed ITr methodology (Figure 2) in more detail.

Insert Figure 2

The Source: Extracted Information Instances

The information source for the ITr process is the set of input information instances that were obtained from the preceding IE process. Information instances have been labeled with information tags during IE. The implemented changes/improvements on the authors' IE work since Zhang and El-Gohary (2013c) are: (1) in addition to semantic information tags, syntactic information tags and combinatorial information tags are also generated for further use in ITr; and (2) instead of the top-down method for handling complex sentence components (processing larger chunks of texts first, then breaking them down to process smaller chunks of texts), a bottom-up method (processing smaller chunks of texts first, then aggregating them to process larger chunks of texts) is adopted because – in the experiments – it has shown to achieve better performance in handling complex sentence components (Zhang and El-Gohary 2013b). As such, in the ITr process, the following three types of information tags (information tags will be shown using single quotes hereafter) are defined and used: (1) semantic information tags, (2) syntactic information tags, and (3) combinatorial information tags.

Semantic information tags are information tags that are related to the meaning and context of the labeled information instances. Instances of semantic information tags are recognized based on the concepts and relations in the domain ontology. For example, in the developed ontology, both "transverse reinforcement" and "vertical reinforcement" are subconcepts of the concept 'subject'. Therefore, the appearances of "transverse reinforcement" (or "transverse reinforcements") and "vertical reinforcement" (or "vertical reinforcements") in Chapter 19 of IBC 2009 will be extracted as instances of the semantic information tag 'subject'. The decision on which concepts and relations are essential to extract and transform is based on the type of requirement (e.g., quantitative requirements) that is being checked. For example, 'subject' is one example of a semantic information tag that is essential in the context of compliance checking of quantitative requirements.

Syntactic information tags are information tags that are related to the grammatical role of the labeled information instances. Instances of syntactic information tags are recognized based on their syntactic features. Syntactic information tags carry information that is more general than those carried by semantic information tags. For example, the syntactic information tag ‘noun’ is describing the labeled information instance as a noun, while semantically the noun could possibly belong to a ‘subject’, ‘compliance checking attribute’, or another semantic information tag. In the proposed methodology, POS tags are mainly used as the syntactic features for syntactic information tags. For example, ‘JJ’ is the POS tag for adjective. It is a syntactic information tag for an information instance that describes properties/attributes of a noun. For example, the adjective “habitable” in “habitable room” is describing the functional property of “room”.

Combinatorial information tags are compound information tags that are composed of multiple semantic and/or syntactic information tags. For example, the combination of ‘past participle verb’ (POS tag ‘VBN’) and ‘preposition’ (POS tag ‘IN’) is a combinatorial information tag (combining two syntactic information tags) that describes a directional passive verbal relation represented by bigrams like “provided by” and “located in”. The combination of ‘adjective’ (syntactic information tag - POS tag ‘JJ’) and ‘subject’ (semantic information tag‘s’) is another example of a combinatorial information tag (combining syntactic and semantic information tags) that describes a ‘subject’ with a certain property.

The Target: Logic Clauses

The target of the ITr process is the set of output logic clauses which are used to represent the requirements in construction regulations. A HC format is used for such representation, in order to facilitate further automated reasoning using logic programs. One single HC represents one requirement. The RHS of the HC (in Prolog syntax the logical RHS appears to the left of ":-") indicates the compliance result(s). The LHS of the HC encodes the conditions for the requirement using one or more predicates. Each predicate defines either a concept information instance (e.g., court(C)) or a relation information instance (e.g., has(C,W)). The logic clause elements in a concept predicate are called concept logic clause elements. The logic clause elements in a relation predicate are called relation logic clause elements. Table 1 shows the source and target for a sample sentence.

Insert Table 1

Semantic Mapping (SeM) Rules

The semantic mapping (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: (1) the information tag it is associated with. For example, in Table 1, 'subject' defines the semantic meaning of "court", i.e., it defines that "court" is the 'subject' of compliance checking; and (2) the context of the extracted information instance, reflected by the information tags of its surrounding information instances. For example, in the following sentence, the semantic meaning of "not less than" (instance of 'comparative relation') is defined by the information tag of its surrounding information instance "for each": "The minimum net area of ventilation openings shall not be less than 1 square foot for each 150 square feet of crawl space area". "For each", here, indicates that "not less than" (relation) is not simply a relationship

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

between "net area" (instance of 'compliance checking attribute') and "1 square foot" (instance of 'quantity value' + 'quantity unit'), but it is also restricted by "150 square feet of crawl space area" (instance of a 'quantity value' + 'quantity reference'). The interpretation of this requirement is that the quantity requirement on "minimum net area of ventilation openings" will increase 1 foot for each additional "150 square feet of crawl space area".

The semantic meanings of information instances are utilized in patterns on the LHS of SeM rules. For the example in Table 1, the corresponding SeM rule pattern is 'subject' + 'modal verb' + 'negation' + 'be' + 'comparative relation' + 'quantity value' + 'quantity unit' + 'preposition' + 'compliance checking attribute'. An SeM rule with this LHS pattern will transform the information instances into the logic clause shown in the last row of Table 1. A sample action defined on the RHS of this SeM rule is: "Generate predicates for the 'subject' information instance, the 'attribute' information instance, and a 'has' information instance. The two arguments of the 'has' information instance are from the 'subject' predicate and the 'attribute' predicate, respectively". Accordingly, the following logic clause elements are generated for the following statement, since "court" is recognized as a 'subject' information instance and "width" as an 'attribute' information instance.

- Sentence: "Courts shall not be less than 3 feet in width"
- Logic Clause Elements: court(Court), width(Width), has(Court,Width)

The ITr method is intended to process each term of a sentence in a sequential manner. In general, sequential processing for information transformation normally requires information instances that are matched by patterns (in SeM rules) to be strictly located next to each other. Such a rigid processing requirement could cause difficulty in processing sentences with different structures.

To avoid that, the proposed SeM rules do not follow such a rigid requirement. Instead, the SeM rules allow for “look-back searching” (i.e., searching to the left of the matched words) and “look-ahead searching” (i.e., searching to the right of the matched words) to find instances that match certain information tags in a pattern. For example, in the following pattern, the instance of the first ‘subject’ does not have to be located right next to the instance of ‘preposition’: “ ‘subject’ + ‘preposition’ + ‘subject’ “. It is only required to be the ‘subject’ instance that is closest to the ‘preposition’ instance from the left. “Look-back searching”, here, searches to the left of the matched word for ‘preposition’ to find the closest ‘subject’ instance when the later part of the pattern “ ‘preposition’ + ‘subject’ ” is matched. This allows for more flexibility in the use of SeM rules to handle sentence complexities (e.g., those incurred by cases such as tail recursive nested clauses). For example, an SeM rule uses the following pattern P1 to match the last three information instances in InS1 (‘s’ for ‘subject’, ‘VBP’ for ‘non-3rd person singular present verb’, ‘dpvr’ for ‘directional passive verbal relation’, and ‘VB’ for ‘base form verb’), finds the first information instance in InS1 through “look-back searching”, and generates the logic clause elements LC1 for the part of sentence S1:

- Pattern P1: ‘non-3rd person singular present verb’ ‘directional passive verbal relation’ ‘base form verb’
- Information Instances InS1: (‘connection’, ‘s’) … (‘are’, ‘VBP’), (‘designed_to’, ‘dpvr’), (‘yield’, ‘VB’)
- Sentence S1: “Connections that are designed to yield shall be capable of …”
- Logic Clause Elements LC1: connection(Connection), yield(Yield), designed_to(Connection,Yield)

In the proposed methodology, application-specific SeM rules are developed based on a randomly selected sample of text (called "development text", which is also used for text analysis and further development of CoR rules). For developing a set of SeM rules for ITr, a three-step, iterative methodology that shall be applied to each sentence is proposed: (1) find all relations in a sentence (e.g., "of" and "not exceed" in the sentence "Spacing of transverse reinforcement shall not exceed 8 inches."), (2) for each relation, run the existing SeM rule set to check if the rule set can generate the corresponding logic clause elements correctly and define the subsequent action based on the following three cases: (a) if the corresponding logic clause elements are correctly generated, then move to check the next relation, (b) if the corresponding logic clause elements are incorrectly generated, then create a new SeM rule with a more specific pattern (i.e., a longer pattern with more features) than the applied SeM rule and add it to the rule set with a higher priority, and (c) if the corresponding logic clause elements are not generated, then create a new SeM rule and add it to the rule set; and (3) after all relations have been checked, run the updated SeM rule set on all checked sentences and check if errors have been introduced due to the added SeM rules. If errors have been introduced, then identify the source(s) of errors (i.e., the rule(s) that introduced the errors) and adjust those rules as necessary.

Conflict Resolution (CoR) Rules

The conflict resolution (CoR) rules resolve conflicts between information tags. Two types of CoR rules are used: deletion CoR rules and conversion CoR rules. Deletion CoR rules resolve conflicts between information tags by deleting certain information instances. For example, the following deletion CoR rule CoR1 is used to delete redundant information instances InS2 ('cr' for 'candidate restriction') from the set of extracted information instances InS3 ('s' for 'subject') for the sentence S2:

- Deletion CoR Rule CoR1: "if an information instance has the tag 'subject' and it subsumes its following information instance(s), then delete its following information instance(s)."
- Information Instances InS2: ('exterior', 'cr'), ('basement', 'cr), ('wall', 'cr')
- Information Instances InS3: ('exterior basement wall', 's'), ('exterior', 'cr'), ('basement', 'cr'), ('wall', 'cr')
- Sentence S2: "The thickness of exterior basement walls and foundation walls shall be not less than 71/2 inches."

Conversion CoR rules resolve conflicts between information tags by converting information tags of information instances into other types of information tags. For example, the following conversion CoR rule CoR2 is used to convert information tags in information instances InS4 ('s' for 'subject', 'I' for 'inter clause boundary relation', and 'a' for 'compliance checking attribute') to information tags in information instances InS5 ('IN' for 'preposition') for the sentence S3:

- Conversion CoR Rule CoR2: "if 'with' is directly followed by an information instance that has the information tag 'compliance checking attribute' and 'with' has the information tag 'inter clause boundary relation', then convert the information tag of 'with' to 'preposition'."
- Information Instances InS4: ('wall segment', 's'), ('with', 'I'), ('horizontal_length_to_thickness_ratio', 'a')
- Information Instances InS5: ('wall segment', 's'), ('with', 'IN'), ('horizontal_length_to_thickness_ratio', 'a')
- Sentence S3: "Wall segments with a horizontal length-to-thickness ratio less than 2.5 shall be designed as columns."

In the proposed rule-based ITr, the CoR rules are executed before the SeM rules, after the information instances have been extracted by the IE process. The development of CoR rules is needed when conflicts between SeM rules cannot be resolved by adjusting SeM rule patterns and actions. For developing a set of CoR rules for ITr, a five-step methodology is proposed: (1) find information tags that are the sources of errors through pattern analysis of conflicting SeM rules, (2) for each conflict, create a new candidate CoR rule to resolve the conflict, (3) try the candidate rule and empirically analyze whether the conflict was resolved without introducing new conflicts or not, (4) if the trial was successful, then add the candidate CoR rule as a new rule to the existing CoR rule set, and if the trial was unsuccessful, then iterate Steps 3 and 4 until a successful trial is found, and (5) after each new CoR rule is added, check all SeM rules and update them as necessary according to the changes in information tags caused by the new CoR rule.

Bottom-up Method for Handling Complex Sentence Components

Due to the variability of natural language expressions and structures, sentences used in regulatory provisions could be very complex. For example, phrases and clauses could be continuously attached/nested to a sentence to constantly enrich it with more relevant information. Complex sentences are difficult to process for information extraction and transformation. Complex sentence components are intermediately-processed segments of text that are: (1) expressed using a variety of natural language structure patterns, and (2) composed of multiple concepts and relations. Complex sentence components are more likely to result in complex sentence structures by embedding in or attaching more concepts and relations to a sentence. Figure 3 shows a complex sentence from IBC 2006. Two methods were explored in handling complex sentence components:top-down method and bottom-up method (Figure 4). The top-

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

down method starts from the top level (i.e., full sentence) and proceeds down to identify and process complex sentence components. The bottom-up method starts from the lowest level (i.e., single terms/concepts/relations in a sentence) and proceeds up to identify and process complex sentence components. The bottom-up method is employed in the proposed ITr approach, because – based on the authors' previous work – it has shown to achieve better performance than the top-down method (Zhang and El-Gohary 2013b).

Insert Figure 3

Insert Figure 4

In the bottom-up method, the SeM rules are used to process sentences starting from the lowest level, i.e., starting from information instances (which correspond to single terms/concepts/relations in a sentence). The information instances in the source text are put into lists – one list for each sentence and are processed one by one until all information instances have been processed. The order of the instances in the list is determined based on their order in the original sentence.

To apply the bottom-up method, the authors propose a new "consume and generate" mechanism to execute the SeM rules in a sequential manner. This mechanism follows the heuristics of the "sliding window" method in computational research (i.e., a sequence of data is sequentially processed, segment by segment, and each segment has a predefined fixed length (i.e., the "window size")) and the mechanism of transcription in genetics domain (i.e., a sequence of DNA is sequentially transcribed, segment by segment, and each segment has a length of about 17 base-pair). The "consume and generate" mechanism processes all text segments that match an SeM rule pattern, where each segment matches a pattern of one SeM rule and each pattern consists of information tags for a sequence of information instances. However, in comparison to the "sliding window" method, the segment length in the proposed "consume and generate" mechanism is not fixed across patterns to allow for flexibility in capturing complex sentence structures. The length of each segment is determined according to the number of information tags in the corresponding SeM rule pattern. For example, the following pattern P2 has a segment length of three and matches the information instances InS6 for the part of sentence S4 to generate logic clause elements LC2:

- Pattern P2: 'compliance checking attribute' 'of' 'subject'
- Information Instances InS6: ('area', 'a'), ('of', 'OF'), ('space', 's')
- Sentence S4: "The net free ventilating area shall not be less than 1/150 of the area of the space ventilated …"
- Logic Clauses Elements LC2: space(Space), area(Area), has(Space, Area)

The "consume and generate" mechanism allows for backward matching: if information instances extracted from a segment of text match the later part of a pattern, then the information instance(s) extracted from preceding text are checked for matching of the earlier part of the same pattern, and corresponding logic clauses are generated if the check succeeds. For example, the following information tags InT1 are associated with the five information instances from the part of sentence S5. After the first three information instances InS7 are processed based on matching with the pattern P3, two information instances "or" and "space" remain. These two remaining information instances only match the later part (i.e., second and third information tags) of the pattern P4 for 'conjunctive subject'. Normally, this partial matching would not initiate the processing of the information instances. However, under the proposed backward matching mechanism, the preceding information instance "interior room" is checked for the matching of the earlier part of the pattern for "conjunctive subject" (i.e., the first information tag: 'subject').

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

Since "interior room" matches 'subject', the SeM rule for "conjunctive subject" gets applied and the two remaining information instances are processed to generate the logic clause elements LC3 (where ";" is the disjunctive operator (i.e., "A ; B" means "A or B")).

- Information Tags InT1: 'compliance checking attribute', 'of', 'subject', 'conjunctive term', 'subject'
- Sentence S5: "…the floor area of the interior room or space…"
- Information Instances InS7: "floor area", "of", "interior room"
- Pattern P3: 'compliance checking attribute' + 'of' + 'subject'
- Pattern P4: 'subject' + 'conjunctive term' + 'subject'
- Logic Clause elements LC3: interior_room(Interior_room); space(Interior_room)

Validation

Results are evaluated in terms of precision, recall, and F1 measure. Precision is the number of correctly generated logic clause elements divided by the total number of generated logic clause elements. Recall is the number of correctly generated logic clause elements divided by the total number of logic clause elements that should be generated. F1 measure is the harmonic mean of precision and recall, assigning equal weights to precision and recall. Ideally, both 100% recall and precision are desired. However, given the inherent trade-off between the two measures, it is difficult to achieve such a result. The ultimate goal for ACC is, therefore, to achieve 100% recall of non-compliance instances – with high precision.

Experimental Implementation and Validation

For testing and validation, the proposed ITr methodology was empirically implemented in transforming information instances of quantitative requirements, which were automatically extracted from the IBC 2009, into logic clauses.

Source Text Selection

The proposed ACC approach and ITr methodology are intended to process information from a variety of construction-related textual regulatory documents (e.g., building codes, environmental regulations, safety regulations and standards). Since building codes are the primary sets of regulations governing the design, construction, operation, and maintenance of residential and commercial buildings, they were chosen for testing the proposed ITr methodology. In the U.S., almost all state authorities (except for Delaware, Massachusetts, Mississippi, and Missouri) adopt versions of the IBC by ICC. Thus, IBC was selected as the source text corpus. More specifically, IBC 2006 and IBC 2009 were selected because of their availability and easiness for comparison (with the authors' previous NLP work in which IBC 2006 and IBC 2009 were used for testing and validation) (Zhang and El-Gohary 2013c).

The SeM and CoR rules were developed based on Chapters 12 and 23 of IBC 2006, and the proposed ITr algorithms were tested in processing information instances of "quantitative requirements" that were extracted from Chapter 19 of IBC 2009. A quantitative requirement is a requirement which defines the relationship between an attribute of a certain building element/part and a specific quantity value (or quantity range). For example, the following sentence, states that the width (attribute) of court (building element/part) should be greater than or equal to 3' (quantity value): "Couts shall not be less than 3 feet in width". The authors decided to The experiment on the extraction of quantitative requirements because: (1) IBC 2006 and IBC

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

2009 describe many quantitative requirements (e.g., on average, quantitative requirements represent 41% of the requirements in Chapters 12 and 23 of IBC 2006 and Chapter 19 of IBC 2009), which ensures a sufficient amount of relevant sentences for development and testing; and (2) sentences describing quantitative requirements appear to be more complex than those describing other types of requirements (e.g., existential requirements, which requires the existence of a certain building element/part), which implies that they are more difficult to process. This makes quantitative requirements good candidates for testing.

Tool Selection

The proposed TC, IE, and ITr algorithms were combined into one computational platform. The representation of Prolog was selected for logic clause representation, in order to facilitate future CR. Prolog is an approximate realization of the logic programming computational model on a sequential machine (Sterling and Shapiro 1986). It is the most popular logic programming language with a reasoner. The syntax of B-Prolog was used. B-Prolog is a Prolog system with extensions for programming concurrency, constraints, and interactive graphics. It has bi-directional interface with C and Java (Zhou 2012). To facilitate quantitative reasoning, a set of built-in rules were developed to perform arithmetic and comparative operations on the proposed quantitative representation. The TC and IE algorithms were implemented using the General Architecture for Text Engineering (GATE) tools (Univ. of Sheffield 2013). GATE has a variety of built-in tools for a variety of text processing functions (e.g., tokenization, sentence splitting, POS tagging, gazetteer compiling, and morphological analysis). For ITr, the SeM rules and CoR rules were implemented using Python programming language (v3.3.2). The "re" module (i.e., regular expression module) in Python was used for pattern matching, so that each extracted information instance could be used for subsequent processing steps based on their information tags (example tags are shown in Figure 3). A domain ontology was developed and used to facilitate semantic IE and ITr. In developing the ontology, the ontology development methodology in El-Gohary and El-Diraby (2010) was followed. The GATES' built-in ontology editor was used for ontology building and editing.

Information Representation

Two types of logic statements in B-Prolog syntax were utilized: facts and rules. A rule has the form: "H :- B1, B2, …, Bn. (n>0)". H, B1, …, Bn are atomic formulas. H is called the head, and the RHS of ':-' is called the body of the rule. A fact is a special kind of rule whose body is always true (Zhou 2012). Each requirement rule in IBC 2006 and IBC 2009 is represented as one single B-Prolog rule. Instances of concepts are represented using unary predicates. For example, the information instance "floor" is represented by the predicate "floor(F)", with "floor" being the predicate name and the variable "F" (all variables in B-Prolog start with capitalized letter) being the argument for the predicate. Instances of relations are represented using binary or n-ary predicates. For example, "provided with" is a relation which is represented as the predicate "provided_with(A,B)", while the variables "A" and "B" could be defined in the predicates interior_space(A) and space_heating_system(B). Each design fact, on the other hand, is represented using one B-Prolog fact. The B-Prolog reasoner can then automatically reason about the facts and rules and, accordingly, determine the compliance checking result(s). An example is shown in Figure 5.

Insert Figure 5

Information Tags

A total of 40 information tags were developed for use in the SeM rules and CoR rules for ITr. A total of 17, 22, and 1 semantic information tags, syntactic information tags, and combinatorial information tags were used, respectively.

Two main types of semantic information tags were defined (as per Figure 6): essential information tags and secondary information tags. Essential information tags are tags for information that must be defined for this specific type of requirement. Six main types of essential information tags were defined for quantitative requirements: subject, compliance checking attribute, comparative relation, quantity value, quantity unit, and quantity reference. A 'subject' is an ontology concept; it is a "thing" (e.g., building object, space) that is subject to a particular regulation or norm. A 'compliance checking attribute' is an ontology concept; it is a specific characteristic of a 'subject' by which its compliance is assessed. A 'comparative relation' is an ontology relation which is commonly-used for comparing quantitative values (i.e., comparing an existing value to a required minimum or maximum value). Five subtypes of comparative relations were further defined: 'greater than or equal to', 'greater than', 'less than or equal to', 'less than', and 'equal to'. A 'quantity value' is a value, or a range of values, which defines the quantified requirement. A 'quantity unit' is the unit of measure for the 'quantity value'. A 'quantity reference' is a reference to another quantity (which includes a value and a unit).

Secondary information tags are tags for information that are not necessary for this specific type of requirement, but may exist in defining the requirement. Two main types of secondary information tags were defined for quantitative requirements: 'restriction' and 'exception'. A 'restriction' is a concept that places a constraint on the 'subject', 'compliance checking attribute', 'comparative relation', pair of 'quantity value' and 'quantity unit', pair of 'quantity value' and

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

'quantity reference', or the full requirement. A 'subject restriction' is a concept that places a constraint on the 'subject'. Two subtypes of 'subject restriction' were further defined: 'possesive subject restriction' and 'nonpossesive subject restriction'. A 'possesive subject restriction' places a possessive constraint on the 'subject', thereby restricting the 'subject' to possess certain building parts or properties. For example, in the following requirement sentence, "having windows opening on opposite sides" is a 'possessive subject restriction' on "court": "Courts having windows opening on opposite sides shall not be less than 6 feet in width". A 'nonpossesive subject restriction' places a nonpossesive constraint on the 'subject', thereby restricting the 'subject' not to possess certain building parts or properties. A 'compliance checking attribute restriction' places a constraint on the 'compliance checking attribute', thereby restricting the 'compliance checking attribute' to a more specific type. For example, in the following requirement sentence, "to the outdoors" is a 'compliance checking attribute restriction' on "minimum openable area": "The minimum openable area to the outdoors shall be 4 percent of the floor area being ventilated". A 'comparative relation restriction' places a constraint on the 'comparative relation', thereby restricting the 'comparative relation' using new conditions. For example, in the following requirement sentence, "for each 150 square feet of crawl space area" is a 'comparative relation restriction' on "not less than": "The minimum net area of ventilation openings shall not be less than 1 square foot for each 150 square feet of crawl space area". A 'quantity restriction' places a constraint on the 'quantity value' + 'quantity unit'/'quantity reference' pair, thereby specifying the properties (e.g., range) of the pair. A 'full requirement restriction' places a constraint on the whole quantitative requirement, thereby restricting the quantitative requirement with new preconditions. An 'exception' defines a condition where the described requirement does not apply.

For syntactic information tags, the Hepple POS Tagger was used to generate POS tag features. Some additional syntactic features that were not in the Hepple POS Tagger (e.g., the preposition "of") were also defined. Each selected POS type and defined syntactic feature represents a syntactic information tag such as adjective (POS tag 'JJ') and preposition "of" (the literal "OF").

One combinatorial information tag was defined for use in this implementation and was called 'directional passive verbal relation', which is the combination of 'past participle verb' (POS tag 'VBN') and 'preposition' (POS tag 'IN'). Combinatorial information tags are expressive and flexible. Thus, more combinatorial information tags may be defined and used if more complex information tags are needed to capture complex meanings or patterns.

Insert Figure 6

Gold Standard

The gold standard for Chapter 19 of IBC 2009 was developed semi-automatically. In the authors' previous work, all sentences that include a number (both appearances of digits and words forms of a number) were automatically extracted to ensure a 100% recall of sentences describing quantitative requirements. Then, one of the authors manually deleted false positive sentences. After that, one of the authors manually coded the logic clauses based on the extracted information instances from each sentence. The gold standard was reviewed by two other researchers to verify its correctness. Because of the unambiguous nature of quantitative requirements, along with the well-defined information representation that is used in the proposed methodology, there was an agreement in formulating the gold standard. For Chapter 19, 62 sentences containing quantitative requirements were recognized. Correspondingly, 62 logic clauses were coded. In these 62 logic clauses, 1901 logic clause elements were identified,

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

including 568 logic clause elements for describing concepts and 1333 logic clause elements for describing relations between concepts.

Algorithm Implementation

The proposed ITr methodology was implemented using Python programming language. The processing steps of an example sentence and the pseudo codes for the main algorithm and the "consume and generate" mechanism are shown in Figure 7, Figure 8, and Figure 9, respectively.

Insert Figure 7

Insert Figure 8

Insert Figure 9

As shown in Figure 7, the IE process tags the original sentence with information tags (from Part I to Part II). The main ITr algorithm then represents each information instance in the tagged sentence into a four-tuple (from Part II to Part III). The CoR rules in the main algorithm then process the information instance tuple list to resolve conflicts between tuples (from Part III to Part IV). The "consume and generate" code then executes the set of SeM rules to process each tuple in the list and generate logic clause elements based on matching of SeM rule patterns (from Part IV to Part V). For each information instance, the four-tuple is used to store: (1) the information instance itself, (2) the location of the information instance in the corresponding sentence (represented by the starting point of the information instance in the sentence), (3) the length of the information instance in terms of number of letters, and (4) the information tag of the information instance (e.g., 'Interior', 0, 15, and 's' for the first information instance in Part III of Figure 7).

In the main algorithm (Figure 8), the CoR rules are executed through the function "resolve conflicts". Then, the SeM rules are executed using the "consume and generate" code to process the conflict-free information instances for each sentence of the source text file (in the format of a list of four tuples) to generate and display the corresponding logic clause. As shown in Figure 9, the "consume and generate" code checks through the patterns for each SeM rule (*PATTERN1*, *PATTERN2*, *PATTERN3*…) and generates logic clauses as a result of matching to SeM rules. In case of no matching, the default negative step length enables backward matching.

Experimental Results and Discussion

The proposed ITr algorithms were tested in transforming information instances of quantitative requirements, which were automatically extracted from Chapter 19 of IBC 2009, into logic clauses. The following two experiments were conducted for comparing the performances of two methods of information representation: (1) using essential semantic information tags only, and (2) using essential, as well as secondary, semantic information tags.

In Experiment #1, only the essential semantic information tags were used: 'subject', 'compliance checking attribute', 'comparative relation', 'quantity value', 'quantity unit', and 'quantity reference'. A subset of the gold standard (including logic clause elements corresponding to the essential semantic information instances) was used as the gold standard for Experiment #1. A total of 53 and 11 SeM and CoR rules, respectively, were developed.

In Experiment #2, both essential and secondary information tags were used. Figure 3 shows examples of some of the information tags that were used. A total of 297 and 9 SeM and CoR rules, respectively, were encoded. The gold standard of Experiment #2 (the full gold standard set) contains 177% more logic clause elements than those in the gold standard of Experiment #1. This shows that for quantitative requirements, the source text contains much secondary information instances.

The SeM rules that were developed in the experiments are classified into four main types: simple SeM rules, multiple action SeM rules, multiple condition SeM rules, and complex SeM rules. A simple SeM rule is the simplest type where a strict SeM pattern directly maps to a logic clause. For multiple action SeM rules, other actions (called "supportive actions") such as "look-ahead searching" and "look-back searching" are involved in addition to mapping SeM patterns to logic clauses. For multiple condition SeM rules, the mapping from SeM patterns to logic clauses are encoded in subrules to handle subtly different cases in rule conditions such as the existence/non-existence status of certain information instances. A complex SeM rule is a combination of the first three types of rules; it utilizes both supportive actions and subrules to support mappings from SeM patterns to logic clauses.

The logic clauses generated from the SeM rules are classified into three main types: single predicate logic clauses, multiple predicate logic clauses, and compound predicate logic clauses. A single predicate logic clause includes only one single predicate (e.g., "space(Space)"). A multiple predicate logic clause includes more than one predicate (e.g., "space(Space), area(Area), has(Space, Area)"). A compound predicate logic clause has predicate(s) that embed other predicate(s) as argument(s) (e.g., "greater_than_or_equal(T, quantity(71/2, inches))").

Table 2 shows the patterns of the most applied SeM rules (i.e., rules applied at least three times) in the experiments. The patterns of the rest of the applied SeM rules are shown in Table 3.

Insert Table 2

Insert Table 3

The overall performance results of Experiment #1 and Experiment #2 are summarized in Table 4 and Table 5, respectively.

Insert Table 4

Insert Table 5

A comparison between the results of Experiment #1 and those of Experiment #2 is summarized in Table 6. The number of information tags in Experiment #2 increased 400% from that used in Experiment #1. The increase in the number of SeM rules was of similar magnitude (460%). Through analysis, the causes of this increase in the number of SeM rules were found to be: (1) the use of more information tags increases the length of patterns in SeM rules, which in turn increases the specificity of each pattern; and (2) the use of more information tags increases the complexity of patterns in SeM rules, which in turn increases the possible number of patterns. In contrast to SeM rules, the number of CoR rules decreased from Experiment #1 to Experiment #2. This results from the use of more information tags, which leads to better distinguishable information instances, and in turn leads to less conflicts between information instances.

The algorithms achieved 92.5% and 98.2%, 95.1% and 99.1%, and 93.8% and 98.6% overall precision, recall, and F1 measure for Experiment #1 and Experiment #2, respectively. Both precision and recall improve in Experiment #2, because the use of more information tags could: (1) better distinguish and capture the variations in expressions; and (2) help define SeM rules with more specificity in patterns. Based on the comparative analysis, the following conclusion can be drawn: the use of more information tags helps in improving the performance of information transformation.

Insert Table 6

The precisions of relation logic clause elements are lower than other precision and recall values across Experiment #1 and Experiment #2. Through analysis, four main causes for this relatively lower performance of precision (89.8% and 97.5% for Experiment #1 and Experiment #2, respectively) of relation logic clause elements are recognized: (1) Structural ambiguity caused by conjunctive terms: For example, in the following part of sentence, there are two possible syntactic uses of “and” – either linking “wall piers” and “such segments” or linking the preceding clause and the following clause: “…shear wall segments provide lateral support to the wall piers and such segments have a total stiffness…”. The ability of the SeM rules to handle structural ambiguity is limited by the development text, which may lead to errors; (2) Incorrect tagging during IE: For example, “professional” (in “registered design professional”) was incorrectly tagged as an adjective instead of noun. This is due to the imperfection of state-of-the-art POS tagging methods; (3) Errors due to morphological analysis (MA): MA was used for improving the recall of semantic information instances by finding all forms of a term based on its lexical form. However, while useful in this regard, MA also introduced false positive instances. For example, as a result of MA, “supported” was stemmed into “support”, matched with the concept “support” in the ontology, and as a result incorrectly recognized as an instance of ‘subject’; and (4) Errors caused by certain SeM rules: For example, an SeM rule selects the immediate left neighbor of a preposition as the first argument of that preposition. In cases where the immediate left neighbor of a preposition is not its real first argument, this SeM rule causes errors. For example, in the following part of sentence, “gypsum concrete” was mistakenly identified as the first argument rather than “clear span”: “clear span of the gypsum concrete between supports”.

Analyzing other errors (other than those influencing precision of relation logic clause elements), two additional causes of errors are recognized: (1) Missing tags in IE: For example, based on the concepts in the ontology, "connection" should have been semantically-tagged as 'subject'. However, in a few instances, it was missing the 'subject' information tag. This is due to the inherent errors in the NLP tools that were used (no existing NLP tool can achieve 100% performance); and (2) Error in processing sentences with uncommon syntactic expression structures: For example, in the part of sentence "…which have been water soaked for at least 24 hours…", "soaked" ('compliance checking attribute') was not recognized because: (a) "soaked" was not semantically-recognized because the ontology did not cover this concept, and (b) the syntactic feature of "soaked" (i.e., past participle) was not a common syntactic expression for 'compliance checking attribute' (in contrast, noun is a common expression for 'compliance checking attribute').

Limitations and Future Work

The experimental results show that the proposed approach is promising in automatically transforming the extracted information instances into logic clauses for further compliance reasoning. In spite of the high performance that was achieved (98.2%, 99.1%, and 98.6% for precision, recall, and F1 measure, respectively), three main limitations of this work are acknowledged, which the authors plan to address as part of their ongoing/future research. First, the methodology was only tested on processing quantitative requirements. The types of semantic patterns and conflicts in other types of requirements (e.g., existential requirements) may vary and, thus, may lead to different performance results. Although the processing of other types of requirements is expected to be less or equally complex than that of quantitative requirements – and thus is expected to have similar or better performance, in future work, the authors plan to test the proposed methodology on other types of requirements (e.g., existential requirements) for validation. Second, due to the large amount of manual effort required in developing a gold standard, the proposed ITr algorithms were tested only on one chapter of IBC 2009. Similar high performance is expected when testing on other chapters of IBC and on other regulatory documents, since all regulatory documents share similarities in expressions. However, different performance results might be obtained due to the possible variability of text across different chapters or different regulatory documents. As such, in future work, the authors plan to test the proposed ITr methodology on more chapters of IBC 2009 and on other types of regulatory documents (e.g., environmental regulations). Third, the validation of the proposed ITr algorithms was focused on precision and recall. At this stage, the computational efficiency of the proposed algorithms wasnot evaluated, although it was taken into consideration when developing the algorithms. For example, the more efficient and stable merge sort (rather than quick sort) was used when a sorting algorithm was needed. In future work, the authors plan to perform algorithm optimization to improve the computational efficiency of the proposed algorithms, if/as necessary.

Contribution to the Body of Knowledge

This research contributes to the body of knowledge in four main ways. First, domain-specific, semantic NLP-based information processing methods that can achieve full sentence processing and information extraction (i.e., all terms of a sentence are processed), as opposed to partial sentence processing and information extraction (i.e., only specific terms/concepts are processed/extracted) are offered. Domain-specific semantics allow for analyzing complex sentence structures that would otherwise be too complex and ambiguous for automated IE and ITr, recognizing domain-specific text meaning, and in turn allowing for

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

processing/understandability of full sentences. Full sentence processing/understandability allows for a deeper level of NLP, namely natural language understanding. Second, this research shows that a hybrid approach that combines rule-based NLP methods and semantic NLP methods could achieve high performance for the combination of IE and ITr from/of regulatory text, in spite of the complexity inherent in natural language text. Domain-specific expert NLP knowledge (encoded in the form of rules), along with domain knowledge (represented in the form of an ontology), facilitates deep text processing/understandability. Previous work (Zhang and El-Gohary 2013c) showed high performance for rule-based, semantic IE. This paper further shows high performance for rule-based, semantic ITr. Third, a new context-aware and flexible way of utilizing pattern-matching-rule-based methods through the use of context-aware semantic mapping rules is offered. This way of utilizing pattern-matching-based rules captures the details (in terms of the expression, language structure, etc.) of complex sentence components, in a context-aware manner, and through flexible pattern lengths. Fourth, a new mechanism ("consume and generate" mechanism) for processing and transforming complex regulatory text into logic clauses is offered. The proposed mechanism follows the bottom-up method, which has shown based on the experimental results to outperform the top-down method in ITr. The high performance that the mechanism achieved verifies that the bottom-up method is suitable for such ITr tasks.

From a practical perspective, this work is expected to have significant impacts on four main levels. First, this work facilitates ACC in the construction domain. ACC could bring down the time, cost, and errors of the checking process; promote compliance of construction projects to various regulations (due to easier and more frequent checking); and encourage the adoption of BIM in the AEC industry. Second, the novel IE and ITr methods and algorithms proposed in this work could be adopted/applied to automate a variety of other tasks in the construction domain, such as contract document analysis and construction accident record analysis. Third, the proposed ITr methodology could be adopted/applied outside of the construction domain, which would contribute to the general domain of natural language processing/understanding. Fourth, the results of this research could ultimately lead to defining principles for the drafting of future regulations in a manner to support ACC. For example, the use of uncommon expressions that tend to cause processing errors could be avoided when drafting future regulations.

Conclusions

This paper presented a rule-based, semantic NLP methodology for automated information transformation (ITr) of information instances, which were automatically extracted from construction regulatory documents, into logic clauses. A set of semantic mapping (SeM) rules and conflict resolution rules (CoR) are used in ITr. CoR rules resolve conflicts between information instances, while SeM rules transform the information instances into logic clause elements. The SeM rules use context-aware and flexible information patterns. Both syntactic and semantic information tags are utilized in the patterns. Syntactic information tags (e.g., POS tags) are generated using NLP techniques. A semantic model helps recognize the semantic information tags of each extracted information instance. A “consume and generate” mechanism is proposed to handle complex sentence components and execute the SeM rules. The ITr method, thus, processes almost all terms of a sentence. Such full sentence processing enables deep NLP towards natural language understanding.

The proposed ITr algorithms were tested in transforming information instances of quantitative requirements, which were automatically extracted from Chapter 19 of IBC 2009, into logic clauses. The transformation results were compared with a manually-developed gold-standard.

The results showed 98.2%, 99.1%, and 98.6% precision, recall, and F1 measure, respectively. This high performance shows that the proposed ITr methodology is promising. Through error analysis, the following six causes of errors were recognized: (1) missing tags in IE; (2) incorrect tagging during IE; (3) errors in processing sentences with uncommon expression structures; (4) errors due to morphological analysis; (5) errors caused by certain SeM rules; and (6) structural ambiguity. In future work, the authors plan to further refine the proposed methodology to avoid those causes of errors – as much as possible, in an effort to further enhance the performance of the ITr algorithms. Also, as part of the authors' ongoing/future research, the proposed ITr methodology will be tested on more chapters of building codes and on other types of construction regulatory documents (e.g., environmental regulations). Similar high performance is expected. However, variability in performance is possible due to differences in the characteristics of the text across different chapters or documents.

Acknowledgement

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

Abney, S. (1997). "Part-of-speech tagging and partial parsing." *Text, Speech and Language Technology*, 2(1997), 118-136.

Avolve Software Corporation. (2013). *Electronic plan review for building and planning departments*. <http://www.avolvesoftware.com/index.php/solutions/building-departments/> (Oct 3, 2013).

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

Breaux, T.D., and Anton, A.I. (2008). “Analyzing regulatory rules for privacy and security requirements.” *IEEE Transactions on Software Eng.*, 34(1), 5-20.

Caldas, C.H., and Soibelman, L. (2003). “Automating hierarchical document classification for construction management information systems.” *Autom. Constr.*, 12(2003), 395-406.

Califf, M. E., and Mooney, R. J. (2003). “Bottom-up relational learning of pattern matching rules for information extraction.” *J. Machine Learning Research*, 4(2003), 177-210.

Cherpas, C. (1992). “Natural language processing, pragmatics, and verbal behavior.” *The Analysis of Verbal Behavior*, 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, Maryland, 1-2.

Ding, L., Drogemuller, R., Rosenman, M., Marchant, D., and Gero, J. (2006). “Automating code checking for building designs – DesignCheck.” *Clients Driving Innovation: Moving Ideas into Practice*, CRC for Construction Innovation, Brisbane, Australia, 1-16.

El-Gohary, N.M., and El-Diraby, T.E. (2010). “Domain ontology for processes in infrastructure and construction.” *J. Constr. Eng. Manage.*, 136(7), 730–744.

Fenves, S.J., Gaylord, E.H., and Goel, S.K. (1969). “Decision table formulation of the 1969 AISC specification.” *Civ. Eng. Studies: Structural Research Series*, 347, University of Illinois, Urbana, IL, 1-167

Garrett, J.H., Jr., and Fenves, S.J. (1987). “A knowledge-based standard processor for structural component design.” *Eng. with Comput.*, 2(4), 219-238.

Gildea, D., and Jurafsky, D. (2002). “Automatic labeling of semantic roles.” *J. Comput. Linguist.,* 28(3), 245-288.

Goh, O. S., Depickere, A., Fung, C.C., and Wong, K. W. (2006). “Topdown natural language query approach for embodied conversational agent.” *Proc., Intl. MultiConf. Eng. and Comput. Sci. (IMECS 2006),* The International Association of Engineers (IAENG), Hong Kong, China.

Gruber, T.R. (1995). “Toward principles for the design of ontologies used for knowledge sharing.” *Intl. J. Human-Computer Studies*, 43, 907-928.

Han, C.S., Kunz, J.C., and Law, K.H. (1998). “Client/server framework for online building code checking.” *J. Comput. Civ. Eng.,* 12(4), 181-194.

International Code Council (ICC). (2012). “International Code Council.” *AEC3*, <http://www.aec3.com/en/5/5_013_ICC.htm> (Oct. 26, 2013).

Khemlani, L. (2005). “CORENET e-PlanCheck: Singapore's automated code checking system.” *AECbytes “Building the Future” Article*, <http://www.aecbytes.com/buildingthefuture/2005/CORENETePlanCheck.html> (Oct 26, 2013).

Kiyavitskaya, N., Zeni, N., Breaux, T.D., Anton, A.I., Cordy, J.R., Mich, L., and Mylopoulos, J. (2008). “Automating the extraction of rights and obligations for regulatory compliance.” *Lecture Notes in Comput. Sci.*, 5231(2008), 154-168.

Marquez, L. (2000). “Machine learning and natural language processing.” *Proc., “Aprendizaje automatico aplicado al procesamiento del lenguaje natural”.*

Nguyen, T. (2005). “Integrating building code compliance checking into a 3D CAD system.” *Proc., Intl. Conf. Comput. Civ. Eng.*, ASCE, Reston, VA, 1-12.

Niemeijer, R.A., Vries, B. de, and Beetz, J. (2009). “Check-mate: automatic constraint checking of IFC models.” *Managing IT in Construction/Managing Construction for Tomorrow*, A Dikbas, E Ergen & H Giritli (Eds.), CRC Press, London, 479-486.

Pocas Martins, J.P., and Abrantes, V. (2010). “Automated code-checking as a driver of BIM adoption.” *Intl. J. Housing Sci*., 34(4), 286-294.

Pradhan, S., Ward, W., Hacioglu, K., Martin, J.H., and Jurafsky, D. (2004). “Shallow semantic parsing using support vector machines.” *Proc, NAACL-HLT*, The Association for Computational Linguistics, East Stroudsburg, PA, 233-240.

Roth, D., and Yih, W. (2004). “A linear programming formulation for global inference in natural language tasks.” *Proc., 2004 Conf. Comput. Natural Language Learning (CoNLL-2004)*, SIGNLL, Boston, MA, 1-8.

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

Salama, D., and El-Gohary, N. (2013a). “Semantic text classification for supporting automated compliance checking in construction”. *J. Comput. Civ. Eng.*, Accepted and published online ahead of print.

Salama, D., and El-Gohary, N. (2013b). “Automated compliance checking of construction operation plans using a deontology for the construction domain.” *J. Comput. Civ. Eng.*, 27(6), 681-698.

Soysal, E., Cicekli, I., and Baykal, N. (2010). “Design and evaluation of an ontology based information extraction system for radiological reports.” *Comput. in Biology and Med.*, 40(11-12), 900-911.

Sterling, L., and Shapiro, E. (1986). “The art of Prolog: advanced programming techniques.” MIT Press, Cambridge, Massachusetts, London, England.

Tan, X., Hammad, A., and Fazio, P. (2010). “Automated code compliance checking for building envelope design.” *J. Comput. Civ. Eng.*, 24(2), 203-211.

Tierney, P.J. (2012). “A qualitative analysis framework using natural language processing and graph theory.” *The Intl. Review of Research in Open and Distance Learning*, 13(5).

University of Sheffield. (2013). “General architecture for text engineering.” <http://gate.ac.uk/> (Oct. 13, 2013).

Wyner, A., and Governatori, G. (2013). “A study on translating regulatory rules from natural language to defeasible logic.” *Proc., RuleML 2013: The 7th Intl. Web Rule Symposium*, Springer-Verlag, Berlin Heidelberg, Germany.

Wyner, A., and Peters, W. (2011). “On rule extraction from regulations.” *Proc., JURIX 2011: The 24th Intl. Conf. Legal Knowledge and Info. Systems*, IOS Press, Amsterdam, The Netherlands, 113-122.

Yin, S., and Fan, G. (2013). “Research of POS tagging rules mining algorithm.” *Applied Mechanics and Materials*, 347 – 350(2013), 2836-2840.

Zhang, J., and El-Gohary, N.M. (2013a). “Information transformation and automated reasoning for automated compliance checking in construction.” *Proc., 2013 ASCE Intl. Workshop Comput. in Civ. Eng.*, ASCE, Reston, VA, 701-708.

Zhang, J., and El-Gohary, N.M. (2013b). “Handling sentence complexity in information extraction for automated compliance checking in construction.” *Proc., CIB W78 2013*, Conseil International du Bâtiment (CIB), Rotterdam, The Netherlands.

Zhang, J., and El-Gohary, N. (2013c). “Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking.” *J. Comput. Civ. Eng.*, Accepted and published online ahead of print.

Zhong, B.T., Ding, L.Y., Luo, H.B., Zhou, Y., Hu, Y.Z., and Hu, H.M. (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 7.7): Prolog, agent, and constraint programming.” Afany Software. <http://www.probp.com/manual/manual.html> (Nov. 19, 2012).

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.

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

Tables

Table 1: A Transformation Example

Requirement Sentence	Courts shall not be less than 3 feet in width.					
Source – Information Tag	Subject	Compliance Checking Attribute	Comparative Relation	Quantity Value	Quantity Unit	Quantity Reference
Source – Information Instance	court	width	not less than	3	feet	NA
Target – Logic Clause	compliant_width_of_court(Court) :- width(Width), court(Court), has(Court,Width), greater_than_or_equal(Width,quantity(3,feet)).					

Table 2: Patterns of the Most Applied SeM Rules in the Experiments

SeM Rule Pattern	Action	Condition Case	Logic Clause Generated	SeM Rule Type
['a' 's' 'cr'] (a) 'OF' (b) ['a' 's' 'cr'] (c)			a(A),c(C),has(C,A)	Simple
'dpvr' (a) ['s' 'cr'] (b)	look-back search for attribute or subject (s); look-back search for negation (n)	n exists	s(S),b(B),not a(S,B)	Complex
		n not exists	s(S),b(B),a(S,B)	
'c' (a) 'v' (b)	look-back search for attribute or subject (s); look-ahead search for unit or reference (u); look-back search for negation (n)	n exists	not a(S, quantity(b,u))	Complex
		n not exists	a(S, quantity(b,u))	
'I' 's'	skip			Multiple action
'c' (a) 'v' (b) 'u' (c) 'IN' (d) 's' (e)	look-back search for attribute or subject (s)		distance(Distance),s(S),e(E), d(S,E,Distance),a(Distance, quantity(b,c))	Multiple action
['a' 's' 'cr'] (a) 'CC' (b) ['a' 's' 'cr'] (c)			(a(A);c(A))	Simple
['VB' ^ 'be'] (a) 'IN' (b) ['cr' 'a' 's'] (c)	look-back search for subject or attribute (s)		s(S),c(C),b(S,C)	Multiple action
['a' 's' 'cr'] (a) 'IN'			a(A),c(C),b(A,C)	Simple

(b) [‘a’ ‘s’ ‘cr’] (c)				
‘Except’	mark the beginning of exception			Multiple action
‘n’ (a) ‘c’ (b) ‘v’ (c) ‘u’ (d)	look-back search for attribute or subject (s)		s(S),not b(S,quantity(c,d))	Multiple action
[‘a’ ‘s’] (a) ‘OF’ (b) ‘v’ (c) [‘u’ ‘a’] (d)		pattern preceded by [‘a’ ‘s’ ‘cr’] (e) [‘Has’ ‘NoHas’ ‘IN’ ‘OF’ ^ ‘between’] (f)	a(A),e(E),equal_to(E, quantity(c,d))	Multiple condition
		otherwise	a(A),equal_to(A, quantity(c,d))	
‘VBP’ (a) ‘VBN’ (b)	look-back search for attribute or subject (s)		b(S)	Multiple action
I’ ‘CC’	skip			Multiple action
‘s’ (a) ‘MD’ (b) ‘Has’ (c) ‘a’ (d)	look-back search for attribute or subject (s)	pattern preceded by ‘IN’	s(S),d(D),has(S,D)	Complex
		otherwise	a(A),d(D),has(A,D)	
‘TO’ (a) ‘VB’ (b) [‘s’ ‘cr’ ‘a’] (c)	look-back search for attribute or subject (s)	s not exists	c(C),a_b(C)	Complex

(1) ‘’: A pair of single quotes encloses information tags
(2) ^: A caret separates optional information tags from exceptions
(3) (a) , (b) , (c) , etc., show the mapping of components (in SeM patterns) to logic clause elements (in generated logic clauses), where an upper case represents a variable
(4) Contents in the “logic clause generated” column are case-sensitive

Table 3: Patterns of the Rest of the SeM Rules Applied in the Experiments

SeM Rule Pattern	
[‘a’ ‘s’ ‘cr’] ‘MD’ ‘n’ ‘VB’ ‘c’ ‘v’ ‘u’	‘VBP’ ‘dpvr’ ‘VB’
‘s’ ‘JJ’ ‘n’ ‘c’ ‘v’ ‘u’	‘n’ ‘c’ ‘s’
‘IN’ ‘ea’ [‘v’ ‘CD’] ‘u’ ‘OF’ ‘s’	[‘s’ ‘cr’] ‘VBD’ [‘cr’ ‘s’]
‘I’ ‘CC’ ‘n’ ‘C’ ‘v’ ‘u’	‘IN’ ‘VBG’ [‘cr’ ‘s’]
‘JJ’ ‘IN’ ‘c’ ‘v’ [‘u’ ‘cr’]	[‘s’ ‘cr’] ‘VBP’ [‘VBN’ ‘JJ’]
‘VB’ ‘IN’ ‘c’ ‘v’ [‘cr’ ‘s’]	‘dpvr’ ‘v’ ‘u’
‘s’ ‘MD’ ‘VB’ ‘dpvr’ [‘VBZ’ ‘cr’ ‘VB’]	‘RB’ ‘TO’ [‘s’ ‘cr’]
‘CC’ ‘v’ ‘u’ ‘IN’ ‘a’	‘MD’ ‘VB’ ‘VBN’
TO’ [‘s’ ‘cr’]	‘a’ ‘OF’ ‘v’ ‘u’ ‘by’ ‘v’ ‘u’

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

‘s’ ‘MD’ ‘n’ ‘VB’ ‘dpvr’	[‘cr’ ‘s’ ‘a’] [‘OF’ ‘IN’ ‘Has’ ‘NoHas’ ^ ‘for’] ‘s’ ‘IN’ ‘s’
[‘s’ ‘a’ ‘cr’] ‘I’ ‘VBG’ [‘cr’ ‘a’ ‘s’] ‘I’	‘MD’ ‘VB’ [‘a’ ‘s’ ‘cr’]
‘JJ’ ‘CC’ ‘JJR’ ‘s’	‘n’ ‘c’ ‘v’
‘s’ ‘WDT’ ‘VBP’ ‘cr’	‘n’ ‘c’ ‘CD’
‘VBG’ ‘cr’ ‘VBP’ ‘VBN’	‘v’ [‘s’ ‘cr’]
‘MD’ ‘VB’ ‘v’ ‘u’	‘s’ ‘VBN’
‘c’ ‘v’ ‘ea’ [‘cr’ ‘s’]	‘JJR’ ‘IN’
‘IN’ ‘JJ’ ‘CC’ ‘s’	‘TO’ [‘s’ ‘cs’]
[‘s’ ‘cr’] ‘with’ ‘a’	‘Except’ ‘IN’
‘n’ ‘c’ ‘v’ [‘cr’ ‘s’]	‘rv’ [‘a’]
‘JJR’ ‘IN’ ‘v’ ‘u’	‘VBZ’ ‘dpvr’
‘s’ ‘Has’ ‘a’ ‘OF’ ‘c’ ‘v’ ‘u’	‘VB’ [‘cr’ ‘a’ ‘s’]
‘s’ ‘MD’ ‘VB’ ‘OF’	‘IN’ [‘cr’ ‘a’ ‘s’]
‘MD’ ‘VB’ ‘dpvr’ ‘s’	[‘u’ ‘JJR’] [^ ‘stories’]
[‘cr’ ‘a’ ‘s’] ‘MD’ ‘VB’ [‘cr’ ‘a’ ‘s’]	‘I’ ‘a’
‘s’ ‘MD’ ‘Has’ ‘s’	‘I’ ‘VBD’
‘cs’ ‘MD’ ‘Has’ ‘s’	‘I’ ‘JJ’
‘v’ ‘u’ ‘CC’ ‘JJR’	‘VBD’ ‘I’
‘s’ ‘MD’ ‘VB’ ‘dpvr’	

Table 4: Experimental Results Using Essential Information Tags Only

	Concepts	Relations	Total
Number of logic clause elements in gold standard	334	749	1083
Total number of logic clause elements generated	328	786	1114
Number of logic clause elements correctly generated	324	706	1030
Precision	0.988	0.898	0.925
Recall	0.970	0.943	0.951
F1 measure	0.979	0.920	0.938

Table 5: Experimental Results Using Both Essential and Secondary Information Tags

	Concepts	Relations	Total
Number of logic clause elements in gold standard	570	1349	1919
Total number of logic clause elements generated	569	1367	1936
Number of logic clause elements correctly generated	568	1333	1901
Precision	0.998	0.975	0.982
Recall	0.996	0.988	0.991
F1 measure	0.997	0.982	0.986

Table 6: Comparative Summary of Experiment #1 and Experiment #2

	Experiment #1	Experiment #2	Increase
Number of information tags used	8	40	+ 400%
Number of semantic mapping rules used	53	297	+ 460%
Number of conflict resolution rules used	11	9	- 18%
Number of logic clause elements built	1114	1936	174%
Precision	0.925	0.982	6%
Recall	0.951	0.991	4%
F1 Measure	0.938	0.986	5%

Figures

Figure 1. Proposed approach for automated rule extraction

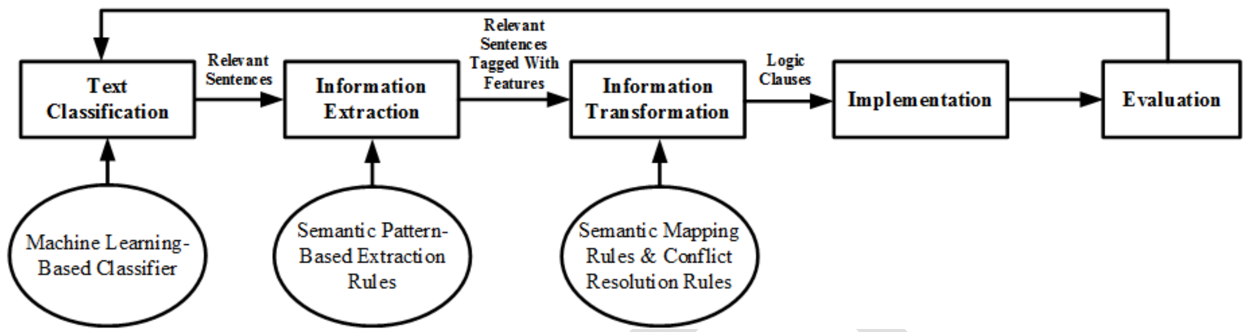
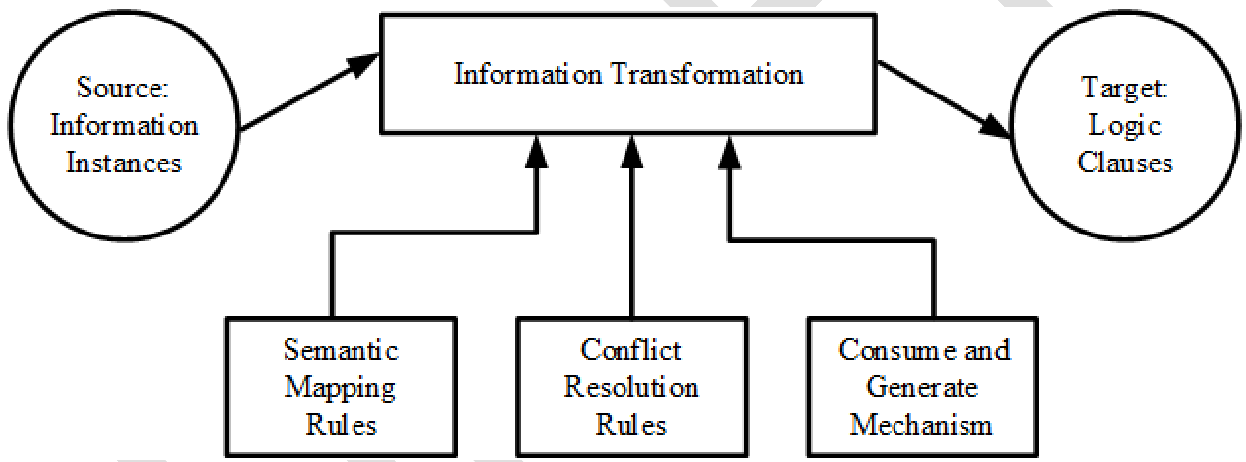
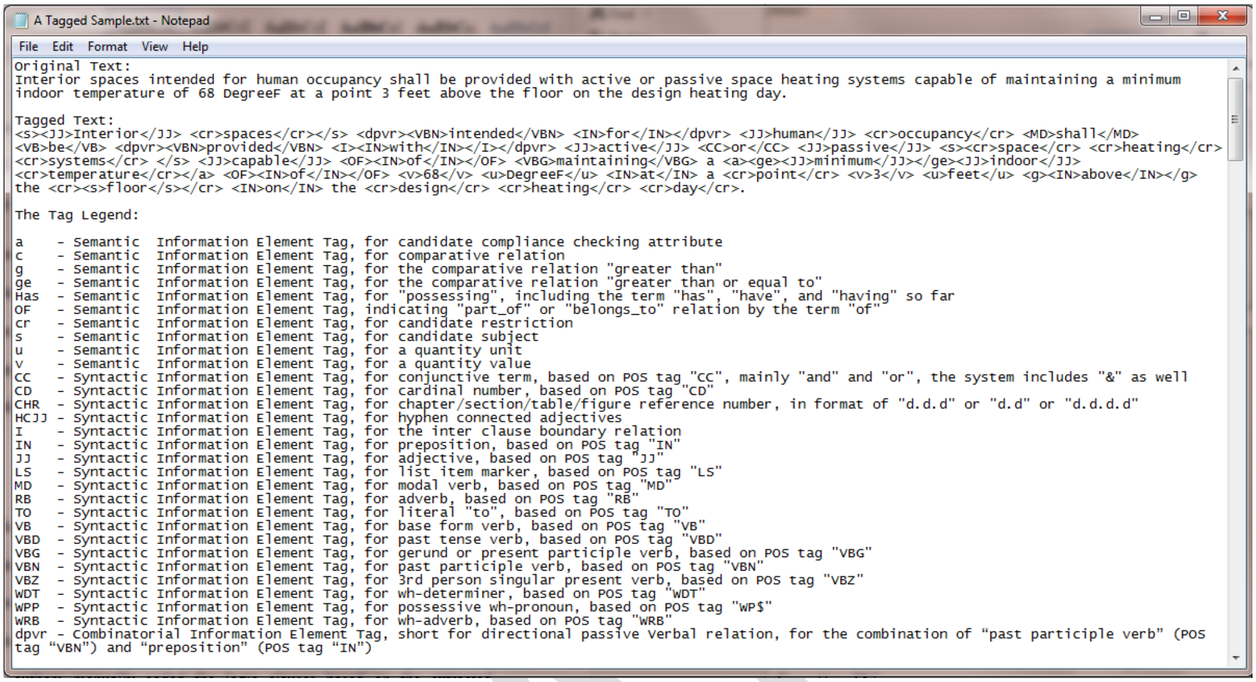


Figure 2. Proposed information transformation methodology



The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

Figure 3. Sample sentence with information tags



```
A Tagged Sample.txt - Notepad
File  Edit  Format  View  Help
Original Text:
Interior spaces intended for human occupancy shall be provided with active or passive space heating systems capable of maintaining a minimum
indoor temperature of 68 DegreeF at a point 3 feet above the floor on the design heating day.

Tagged Text:
<s><JJ>Interior</JJ> <cr>spaces</cr></s> <dpvr><VBN>intended</VBN> <IN>for</IN></dpvr> <JJ>human</JJ> <cr>occupancy</cr> <MD>shall</MD>
<VB>be</VB> <dpvr><VBN>provided</VBN> <I><IN>with</IN></I></dpvr> <JJ>active</JJ> <CC>or</CC> <JJ>passive</JJ> <s><cr>space</cr> <cr>heating</cr>
<cr>systems</cr> </s> <JJ>capable</JJ> <OF><IN>of</IN></OF> <VBG>maintaining</VBG> a <a><ge><JJ>minimum</JJ></ge><JJ>indoor</JJ>
<cr>temperature</cr></a> <OF><IN>of</IN></OF> <v>68</v> <u>DegreeF</u> <IN>at</IN> a <cr>point</cr> <v>3</v> <u>feet</u> <g><IN>above</IN></g>
the <cr><s>floor</s></cr> <IN>on</IN> the <cr>design</cr> <cr>heating</cr> <cr>day</cr>.

The Tag Legend:

a    - Semantic  Information Element Tag, for candidate compliance checking attribute
c    - Semantic  Information Element Tag, for comparative relation
g    - Semantic  Information Element Tag, for the comparative relation "greater than"
ge   - Semantic  Information Element Tag, for the comparative relation "greater than or equal to"
Has  - Semantic  Information Element Tag, for "possessing", including the term "has", "have", and "having" so far
OF   - Semantic  Information Element Tag, indicating "part_of" or "belongs_to" relation by the term "of"
cr   - Semantic  Information Element Tag, for candidate restriction
s    - Semantic  Information Element Tag, for candidate subject
u    - Semantic  Information Element Tag, for a quantity unit
v    - Semantic  Information Element Tag, for a quantity value
CC   - Syntactic Information Element Tag, for conjunctive term, based on POS tag "CC", mainly "and" and "or", the system includes "&" as well
CD   - Syntactic Information Element Tag, for cardinal number, based on POS tag "CD"
CHR  - Syntactic Information Element Tag, for chapter/section/table/figure reference number, in format of "d.d.d" or "d.d" or "d.d.d.d"
HCJJ - Syntactic Information Element Tag, for hyphen connected adjectives
I    - Syntactic Information Element Tag, for the inter clause boundary relation
IN   - Syntactic Information Element Tag, for preposition, based on POS tag "IN"
JJ   - Syntactic Information Element Tag, for adjective, based on POS tag "JJ"
LS   - Syntactic Information Element Tag, for list item marker, based on POS tag "LS"
MD   - Syntactic Information Element Tag, for modal verb, based on POS tag "MD"
RB   - Syntactic Information Element Tag, for adverb, based on POS tag "RB"
TO   - Syntactic Information Element Tag, for literal "to", based on POS tag "TO"
VB   - Syntactic Information Element Tag, for base form verb, based on POS tag "VB"
VBD  - Syntactic Information Element Tag, for past tense verb, based on POS tag "VBD"
VBG  - Syntactic Information Element Tag, for gerund or present participle verb, based on POS tag "VBG"
VBN  - Syntactic Information Element Tag, for past participle verb, based on POS tag "VBN"
VBZ  - Syntactic Information Element Tag, for 3rd person singular present verb, based on POS tag "VBZ"
WDT  - Syntactic Information Element Tag, for wh-determiner, based on POS tag "WDT"
WPP  - Syntactic Information Element Tag, for possessive wh-pronoun, based on POS tag "WP$"
WRB  - Syntactic Information Element Tag, for wh-adverb, based on POS tag "WRB"
dpvr - Combinatorial Information Element Tag, short for directional passive Verbal relation, for the combination of “past participle verb” (POS
tag “VBN”) and “preposition” (POS tag “IN”)
```

Figure 4. Illustration of top-down method and bottom-up method

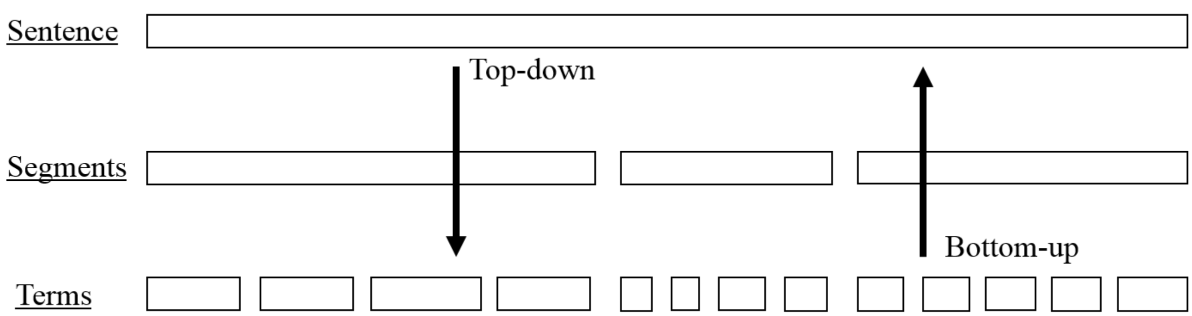


Figure 5. Example illustrating logic-based information representation and reasoning

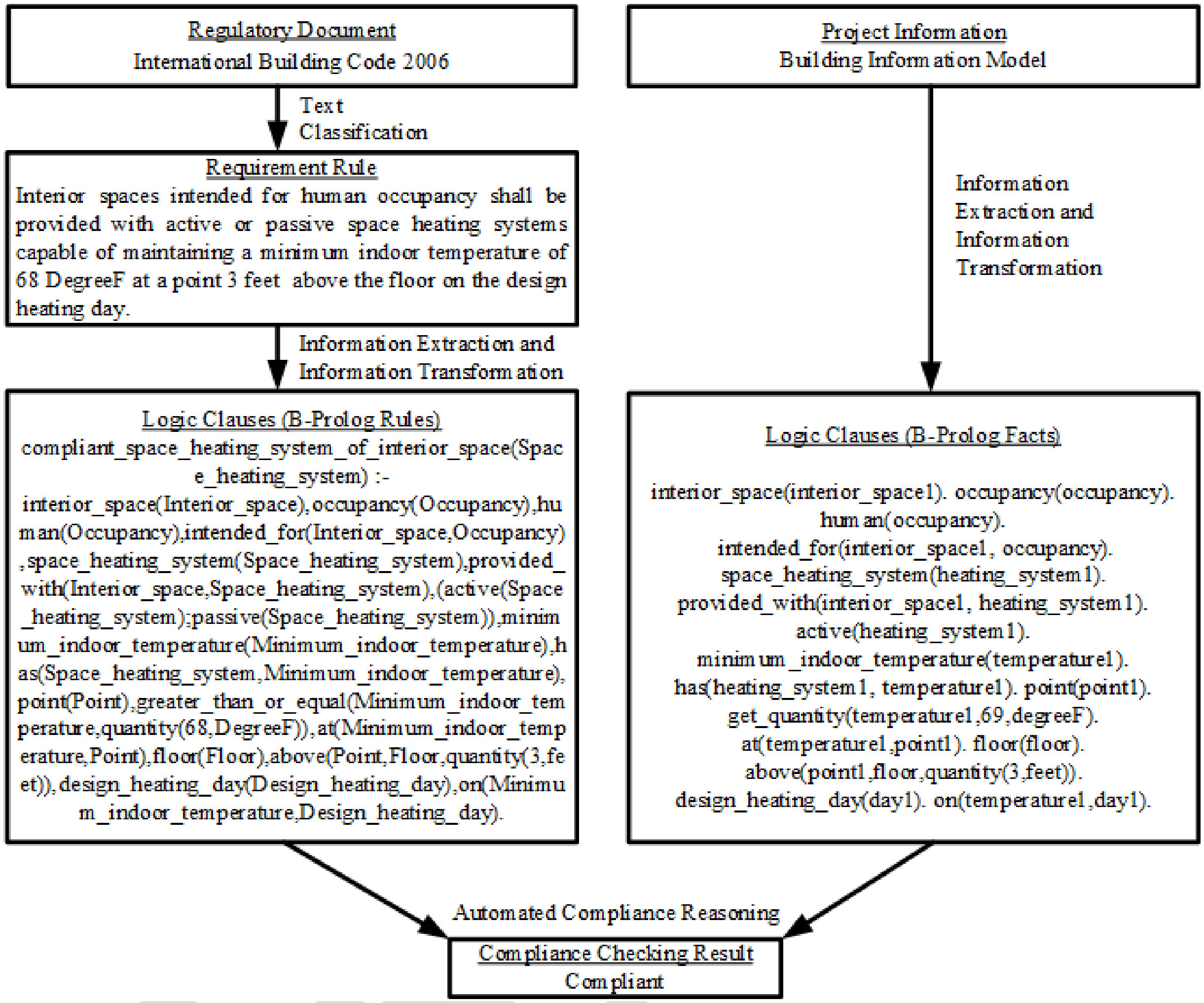

Regulatory Document
International Building Code 2006
Text Classification
Requirement Rule
Interior spaces intended for human occupancy shall be provided with active or passive space heating systems capable of maintaining a minimum indoor temperature of 68 DegreeF at a point 3 feet above the floor on the design heating day.
Information Extraction and Information Transformation
Logic Clauses (B-Prolog Rules)
compliant_space_heating_system_of_interior_space(Space_heating_system) :- interior_space(Interior_space),occupancy(Occupancy),human(Occupancy),intended_for(Interior_space,Occupancy),space_heating_system(Space_heating_system),provided_with(Interior_space,Space_heating_system),(active(Space_heating_system);passive(Space_heating_system)),minimum_indoor_temperature(Minimum_indoor_temperature),has(Space_heating_system,Minimum_indoor_temperature),point(Point),greater_than_or_equal(Minimum_indoor_temperature,quantity(68,DegreeF)),at(Minimum_indoor_temperature,Point),floor(Floor),above(Point,Floor,quantity(3,feet)),design_heating_day(Design_heating_day),on(Minimum_indoor_temperature,Design_heating_day).
Project Information
Building Information Model
Information Extraction and Information Transformation
Logic Clauses (B-Prolog Facts)
interior_space(interior_space1). occupancy(occupancy). human(occupancy). intended_for(interior_space1, occupancy). space_heating_system(heating_system1). provided_with(interior_space1, heating_system1). active(heating_system1). minimum_indoor_temperature(temperature1). has(heating_system1, temperature1). point(point1). get_quantity(temperature1,69,degreeF). at(temperature1,point1). floor(floor). above(point1,floor,quantity(3,feet)). design_heating_day(day1). on(temperature1,day1).
Automated Compliance Reasoning
Compliance Checking Result
Compliant

Figure 6. Semantic information tags

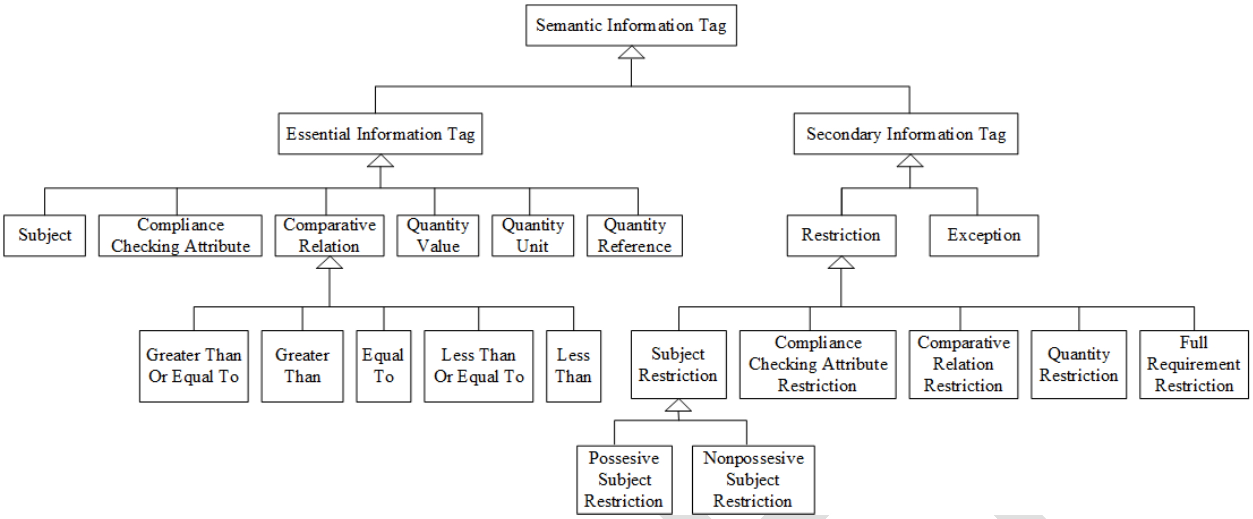
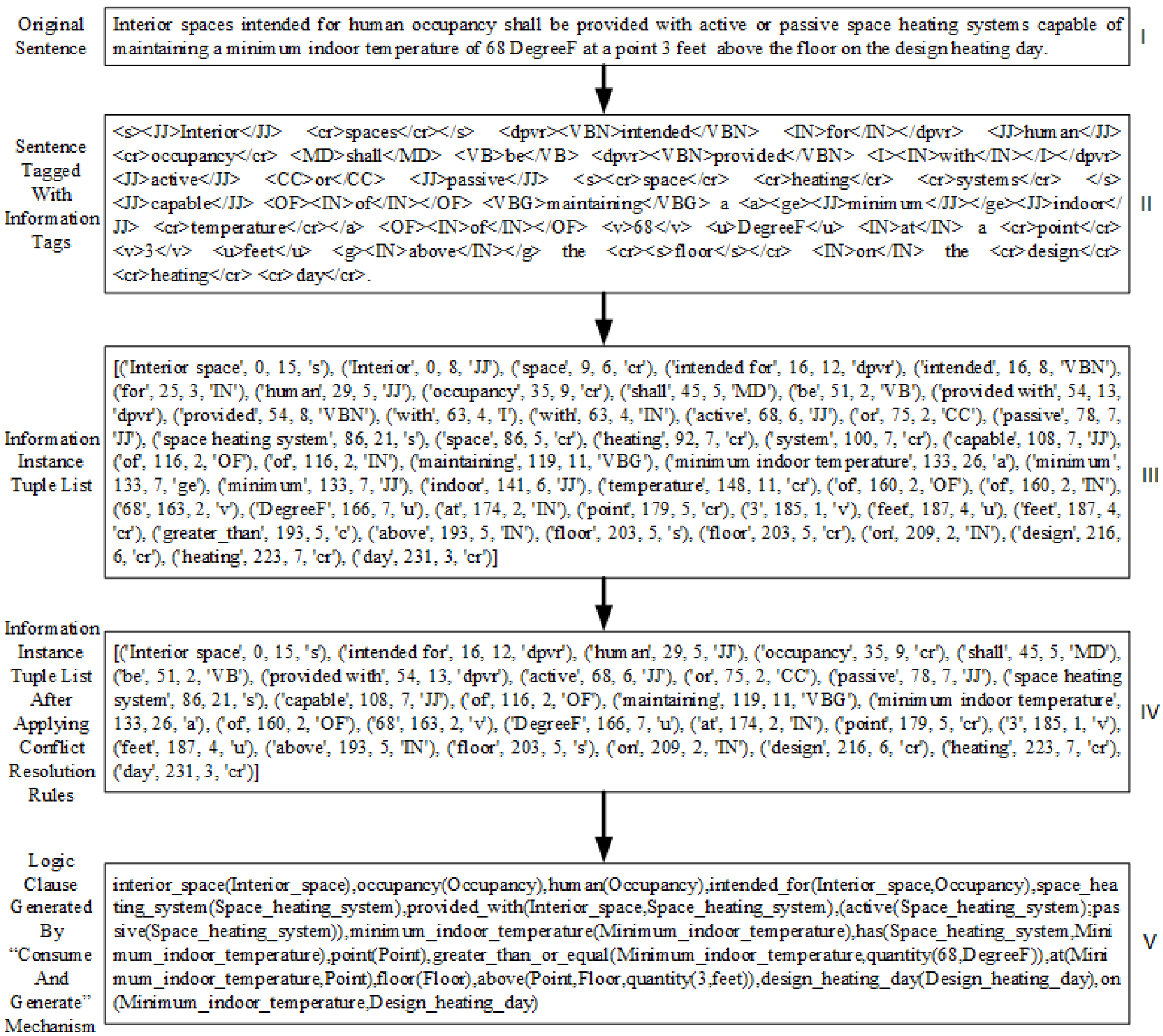


Figure 7. Example illustrating the processing of a sample sentence: (a) original sentence; (b) sentence tagged with information tags; (c) information instance tuple list; (d) information instance tuple list after applying conflict resolution rules; (e) logic clause generated by consume and generate mechanism



The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

Figure 8. Pseudocode for main algorithm

```
open file_tagged_with_information_tags

for line in file_tagged_with_information_tags:
        initialize list_for_line to [ ]
        initialize the_logic_clause_elements_list to [ ]
        initialize generated_logic_clause_elements_list to [ ]
        initialize the_result_two_tuple
        initialize pointer to 0
        initialize step_length to 0
        initialize the_logic_clause to ' '
        initialize list_of_touched_pointers to []
        list_of_items_in_line = collect_items_into_list(line)
        for information_instance in list_of_items_in_line:
                start_index = extract_start(information_instance, line)
                length = len(information_instance)
                tag = extract_tag(information_instance)
                append to list_for_line make_tuple(information_instance, start_index, length, tag)
        list_for_line = resolve_conflicts(list_for_line)
        the_result_two_tuple = consume_and_generate(pointer, list_for_line)
        generated_logic_clause_elements_list = first element in the_result_two_tuple
        append to list_of_touched_pointers pointer
        step_length = second element in the_result_two_tuple
        while pointer < len(list_for_line):
                the_logic_clause_elements_list.extend(generated_logic_clause_elements_list)
                the_result_two_tuple = consume_and_generate(pointer, list_for_line)
                step_length = second element in the_result_two_tuple
                generated_logic_clause_elements_list = first element in the_result_two_tuple
                append to list_of_touched_pointers pointer
                if step_length != -1:
                        pointer = pointer + step_length
                else if pointer == 0:
                        pointer = pointer + 1
                else if (pointer + step_length) not in list_of_touched_pointers:
                        pointer = pointer + step_length
                else:
                        pointer = pointer + 1
                if pointer < -len(list_for_line):
                        break
        the_logic_clause_elements_list.extend(generated_logic_clause_elements_list)
        the_logic_clause_elements_list =
        remove_duplicated_elements(the_logic_clause_elements_list)
        the_logic_clause = build_logic_clause_from_elements(the_logic_clause_elements_list)
        print the_logic_clause

close file_tagged_with_information_tags
```

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000427
Zhang, J. and El-Gohary, N. (2015). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." J. Comput. Civ. Eng., 10.1061/(ASCE)CP.1943-5487.0000427, B4015001.

Figure 9. Pseudocode for consume and generate mechanism

```
define function consume_and_generate(pointer, tuple_list):
        initialize result_two_tuple to [ [ ], -1 ]
        initialize count_for_variable to 1
        if pointer >= len(tuple_list):
                return result_two_tuple
        else if PATTERN1
                look-back search and look-ahead search as needed
                result[0].extend(Right_hand_side_of_semantic_mapping_rule1)
                result[1]=length of PATTERN1
        else if PATTERN2
                look-back search and look-ahead search as needed
                result[0].extend(Right_hand_side_of_semantic_mapping_rule2)
                result[1]=length of PATTERN2
        else if PATTERN3
                look-back search and look-ahead search as needed
                result[0].extend(Right_hand_side_of_semantic_mapping_rule3)
                result[1]=length of PATTERN3

        ...

        return result_two_tuple
```