The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000583
Zhang, J. and El-Gohary, N. (2016). "Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking." J. Comput. Civ. Eng. , 10.1061/(ASCE)CP.1943-5487.0000583 , 04016037.

Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking

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

Abstract

Existing automated compliance checking (ACC) efforts are limited in their automation and reasoning capabilities; the state of the art in ACC still uses ad-hoc reasoning schema/methods, with lack of support for complete automation in ACC reasoning. First-order logic (FOL) representation and reasoning can provide a generalized reasoning method to facilitate complete automation in ACC reasoning. This paper presents a new FOL-based information representation and compliance reasoning (IRep and CR) schema for representing and reasoning about regulatory information and design information for checking regulatory compliance of building designs. The schema formalizes the representation of regulatory information and design information in the form of semantic-based (ontology-based) logic clauses that could be directly used for automated compliance reasoning. Two alternative subschemas, following a closed world assumption and an open world assumption for noncompliance detection, respectively, were proposed and tested. The proposed IRep and CR schema was tested in representing and reasoning about quantitative regulatory requirements in Chapter 19 of the International Building Code 2009 and design information of a two-story duplex apartment test case in two ways, using perfect information and imperfect information. The closed world assumption subschema was selected based on performance results; it achieved 100% recall and precision in noncompliance detection using

[1] Graduate Student, Dept. of Civil and Environmental Engineering, Univ. of Illinois at Urbana-Champaign, 205 N. Mathews Ave., Urbana, IL 61801. E-mail:jzhang70@illinois.edu; Tel: +1-217-607-6006.

[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.

perfect information and 98.7% recall and 87.6% precision in noncompliance detection using imperfect information.

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

Author keywords: Automated compliance checking; Automated reasoning; First order logic; Logic programming; Semantic systems; Automated construction management systems.

Introduction

Construction projects are governed by a multitude of regulations such as building codes, energy conservation codes, and environmental protection agency (EPA) regulations (ICC 2013a; EPA 2013). Each regulatory document typically contains hundreds of pages of provisions and requirements. Due to the variety of regulations and the large volume of regulatory information governing construction projects, manual regulatory compliance checking is time-consuming, costly, and error-prone (Fiatech 2014; Dimyadi and Amor 2013; Fiatech 2012; Delis and Delis 1995).

Automated compliance checking (ACC) is expected to reduce the time, cost, and errors of compliance checking (Salama and El-Gohary 2013; Hjelseth 2012). Many efforts have, thus, attempted to automate the compliance checking process, including the SMARTcodes project by the International Code Council (ICC) (ICC 2013b), the Construction and Real Estate Network (CORENET) project led by the Singapore Ministry of National Development (SBCA 2006), REScheck and COMcheck by the U.S. Department of Energy (DOE 2014), and the Solibri Model Checker (Eastman et al. 2009). However, despite their importance, these efforts are still limited in their automation and reasoning capabilities; the state of the art in ACC still uses ad-hoc reasoning schema/methods, with lack of support for complete automation in ACC reasoning.

First-order logic (FOL) representation and reasoning can provide a generalized reasoning method to facilitate complete automation in ACC reasoning (Kerrigan and Law 2003; Halpern and Weissman 2007). FOL-based reasoning is well-suited for ACC problems because: (1) The binary nature (“satisfy or fail to satisfy”) of the smallest reasoning units (i.e., LCs) fits the binary nature (“compliance or noncompliance”) of ACC tasks; (2) A variety of automated reasoning techniques such as search strategies and unification mechanisms are available in ready-to-use reasoners; (3) FOL has sufficient expressiveness to represent concepts and relations involved in ACC; and (4) Once the information is properly represented in a FOL format, the reasoning becomes completely automated. However, the benefits of FOL-based ACC reasoning is not realized due to three main reasons. First, there is a lack of knowledge on which assumption is better-suited for ACC – a closed world assumption (i.e., the assumption that what is not known to be true is false) or an open world assumption (i.e., the assumption that what is not known to be true is unknown) in noncompliance detection. Second, there is a lack of knowledge on how to use a closed world assumption model in noncompliance detection without introducing many false positives; a closed world assumption can typically lead to a high number of false positives, because missing information would result in failure to deduce compliance. Third, to use an existing logic-based reasoner, there is a need for further ACC-specific computational and reasoning support (e.g., to identify the sequence of checking different regulatory requirements).

To address these limitations, the authors propose a new logic-based information representation and compliance reasoning (IRep and CR) schema for representing and reasoning about regulatory information and design information for checking regulatory compliance of building designs. In

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000583
Zhang, J. and El-Gohary, N. (2016). "Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking." J. Comput. Civ. Eng. , 10.1061/(ASCE)CP.1943-5487.0000583 , 04016037.

developing the schema, the authors addressed the above-mentioned knowledge gaps in three main ways. First, two alternative schema designs – a closed world assumption schema and an open world assumption schema – were proposed and tested. Second, semantic-based (ontology-based) logic clauses and activation conditions were used in the closed world assumption schema to avoid the problem of missing information causing false positives. Third, a support module that consists of a set of logic clauses was developed, as part of the schema, to provide ACC-specific computational and reasoning support when using logic-based reasoners. This paper presents the proposed schema, including the two alternative designs, and discusses the experimental results of applying the schema in representing and reasoning about the compliance of a building design with the quantitative regulatory requirements in Chapter 19 of the International Building Code (IBC) 2009.

Background: Logic-Based Representation and Reasoning

Logic is essential in many automated reasoning systems (Portoraro 2011). Different types of formally-defined logic have different degrees of representation and reasoning capabilities. The most commonly-used formally-defined logic for automated reasoning purposes is first order logic (FOL), which is a subtype of predicate logic. FOL has more than one correct and complete proof calculi (i.e., cases where the derivable sequents are precisely the valid ones for the calculi), which makes FOL suitable for automated reasoning. FOL is based on first order language, which has been used mainly for deductive arguments since its creation. First order language was intended to “express conditions which things can satisfy or fail to satisfy” (Hodges 2001).

Logic-Based Representation

The representation of data/information/knowledge in FOL is composed of statements (i.e., logic clauses) that are expressed using predicates, logic operators, and quantifiers. A predicate is a

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000583
Zhang, J. and El-Gohary, N. (2016). "Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking." J. Comput. Civ. Eng. , 10.1061/(ASCE)CP.1943-5487.0000583 , 04016037.

function that has zero or more arguments and evaluates to a true or false, where an argument is a constant or a variable. For example, *door(x)* is a predicate, *door* is the predicate name, and *x* is the argument (variable). In predicate logic, a statement is an atomic formula or a composition. An atomic formula cannot be decomposed; it is composed of a single predicate. A composition, on the other hand, is formed by combining predicates using logical operators to form more complex statements. Four types of logic operators are used: (1) conjunction ∧: $a(A) \wedge a(B)$ means $a(A)$ is true and $b(B)$ is true, (2) disjunction ∨: $a(A) \vee b(B)$ means $a(A)$ is true or $b(B)$ is true, (3) negation ¬: $\neg a(A)$ means $a(A)$ is not true, and (4) implication ⊃: $a(A) \supset b(B)$ means $a(A)$ implies $b(B)$ [i.e., if $a(A)$ is true then $b(B)$ is true]. Quantifiers are used to make assertions about variables in statements; the universal quantifier (∀ or for all) asserts that the statement is true for all instances of a variable, while the existential quantifier (∃ or there exists) asserts that the statement is true for at least one of the variable instances (Salama and El-Gohary 2013, Aho and Ullman 1992).

In FOL representation there are three types of logic clauses: rules, facts, and queries. Horn Clause (HC) representation is one of the most restricted forms of FOL. A HC is a universally-quantified clause that can be represented as a disjunction of literals (predicates) of which at most one is positive. In HC representation, a rule has one or more antecedents (premise conditions of the rule), that are conjoined (i.e., combined using the conjunction operator), and a single consequent (i.e., conclusion of the rule). A HC rule has, thus, the following form: "$B_1 \wedge B_2 \wedge \ldots \wedge B_n \supset H$", where n>0 and H, B1, …, Bn are predicates. A fact has zero antecedents and one consequent. A query has one or more antecedents, that are conjoined, and zero consequents.

Logic-Based Reasoning with Closed World and Open World Assumptions

Logic-based reasoning uses statements (logic clauses) and inferences that can be made from those statements to solve problems. Inference-making using HC representation is most efficient because of its restricted syntax (Saint-Dizier 1994). Logic-based reasoning can be based on two main types of assumptions: a closed world assumption or an open world assumption (Knorr et al. 2011). The closed world assumption states that all information that is not known to be true is false. This assumption is widely used in database systems. The open world assumption, on the other hand, states that all information that is not known to be true is unknown. This assumption is widely used in the semantic web (Hebeler et al. 2009). The open world assumption is better aligned with real world reasoning where knowledge tends to be incomplete (Grimm and Motik 2005). However, it limits the kinds of inferences and deductions a system can make from statements that are known to be true; in the open world assumption, statements that are not included in or inferred from the knowledge in the system are considered unknown, rather than false. In contrast, the closed world assumption allows a system to infer, from its lack of knowledge of a statement being true, that the statement is false. The limitation of the closed world assumption, however, is that it can lead to unintuitive or unintended results (Halpern and Weissman 2008) by treating all unknown as false. Depending on the task, one of the assumptions would be better suited than the other (Lutz et al. 2012).

State of the Art and Knowledge Gaps

State-of-the-Art ACC in the AEC Industry

The state-of-the-art ACC in the AEC industry mostly relies on the use of proprietary rules for representing regulatory requirements. For example, the CORENET project coded regulatory rules in C++ programs, the Solibri model checker uses a proprietary proforma-based format to code regulatory rules, and several ACC research efforts coded regulatory rules for specific subdomains such as fall protection (Zhang et al. 2013), building envelope performance (Tan et al. 2010), and accessibility (Lau and Law 2004).

To avoid the reliance on proprietary rules, few researchers explored the development of generalized representations/schemas for the formalization of regulatory requirements. For example, Hjelseth and Nisbet (2011) proposed the Requirement, Applies, Select, and Exception (RASE) method to capture and represent regulatory requirements in the AEC industry; Yurchyshyna et al. (2010; 2008) developed a conformity-checking ontology that captures regulatory information together with building-related knowledge and expert knowledge on checking procedures; Beach et al. (2013) extended the RASE method for representing requirements in the UK's Building Research Establishment Environmental Assessment Method (BREEAM) and the Code for Sustainable Homes (CSH); and Dimyadi et al. (2014) utilized the Drools Rule Language (DRL) to represent regulatory rules.

These efforts contributed to the improvement of flexibility and reusability of regulatory representations for ACC. However, they are still limited in terms of automated reasoning; these ACC efforts still use ad-hoc reasoning schema/methods, with lack of support for complete automation in reasoning. For example, in Hjelseth and Nisbet (2011), no specific mechanism for reasoning about the RASE-represented regulatory requirements was proposed. For the ontology-based effort by Yurchyshyna et al. (2010; 2008), the reasoning in their ontology-centered approach was implemented by matching Resource Description Framework (RDF)-represented design information with SPARQL queries-represented regulatory information, but a set of expert rules need to be manually defined through document annotations (i.e., annotations by content and external sources) to organize the SPARQL queries and enable reasoning, resulting in ad-hoc reasoning and lack of full automation. In the work by Beach et al. (2013) and Dimyadi et al. (2014), the mechanism of reasoning (e.g., sequence of rule execution) was not specified.

FOL-based Representation and Reasoning for ACC

FOL representation and reasoning can provide a generalized reasoning method to facilitate complete automation in ACC reasoning (Kerrigan and Law 2003; Halpern and Weissman 2007). A limited number of research efforts have used FOL-based representation and reasoning in the AEC industry. Jain et al. (1989) introduced an information representation method that used FOL-based reasoning to support structural design. Rasdorf and Lakmazaheri (1990) used a FOL approach to (1) designing structural members according to the American Institute of Steel Construction (AISC) specifications and (2) checking the compliance of designed structural members with the specifications. Kerrigan and Law (2003) used a FOL approach to supporting regulatory compliance assessment with Environmental Protection Agency (EPA) regulations. Outside of the AEC industry, a number of efforts have proposed the use of FOL for supporting conformance reasoning, such as compliance checking (Awad et al. 2009), policy auditing (Garg et al. 2011), and law verification (DeYoung et al. 2010). Despite the importance of these efforts, there are three main knowledge gaps in the area of FOL-based ACC. First, there is a lack of knowledge on which assumption is better-suited for ACC – a closed world assumption or an open world assumption in noncompliance detection. For example, Rasdorf and Lakmazaheri (1990) followed a closed world assumption for noncompliance detection, while Kerrigan and Law (2003) used an open world assumption; but there are no efforts that compared both assumptions in terms of performance in ACC applications. Second, there is a lack of knowledge on how to use a closed world assumption model in noncompliance detection without introducing many false positives. A closed world assumption can typically lead to a high number of false positives, because missing information would result in failure to deduce compliance. For example, Denecker et al. (2011) chose to drop the closed world assumption because they could not avoid the false positives caused by missing information. Third, there is a need for further ACC-specific computational and reasoning support for using existing logic-based reasoners. For instance, there is a need for further built-in logic rules or functions to identify the sequence of checking different regulatory requirements. For example, Kerrigan and Law (2003) used control elements (i.e., functions) to specify the sequence of checking provisions for each regulation; but, this approach is limited because these control elements must be specified by a domain expert for every regulation.

The Proposed Information Representation and Compliance Reasoning Schema

The IRep and CR schema aims to provide a schema for formal representation of regulatory information and design information in the form of semantic-based (ontology-based) logic clauses (LCs). Automated compliance reasoning is enabled by the schema, because LCs can be directly used for logic-based automated reasoning. Two alternative subschema designs, Alternative I and Alternative II, were developed based on a closed world assumption and an open world assumption in noncompliance detection, respectively. The logic-based representation and reasoning is supported by a building ontology, where the predicates of the LCs link to the concepts and relations of the ontology. The ontology captures the concepts and relationships of the domain knowledge to support the representation and reasoning process. Activation conditions for checking compliance with regulatory rules were used in Alternative I. The ontology-based LCs and the activation conditions were used in Alternative I to avoid the problem of missing information causing false positives in closed world assumption schemas. A support module was also developed, as part of the schema, to provide ACC-specific reasoning support.

As such, the proposed IRep and CR schema is composed of two main modules (as per Fig. 1): a data module and a support module. The data module consists of information LCs. An information LC could be a regulatory information LC or a design information LC. Regulatory information LCs and design information LCs are used to represent applicable regulatory requirements and existing design information, respectively. The support module was developed to provide reasoning support to the data module, and consists of functional built-in LCs. The functional built-in LCs are used for implementing basic arithmetic functions (such as unit conversion) and defining reasoning sequences/strategies (such as the sequence of checking different regulatory requirements). The functional built-in LCs would be predefined (built-in) in an ACC system and, thus, would be fixed across different compliance checking instances.

Insert Figure 1

Semantic-based Logic Clauses

The predicates in the LCs are semantic; they are linked to a set of semantic information elements (Fig. 2). The sematic information elements are, in turn, linked to a building ontology. A semantic information element (see Fig. 2) is a “subject”, “compliance checking attribute”, “deontic operator indicator”, “quantitative relation”, “comparative relation”, “quantity value”, “quantity unit”, “quantity reference”, “restriction”, or “exception”. The definitions of these semantic information elements are provided in Table 3. A semantic representation is essential to (1) distinguish the ACC-specific meaning of the different predicates by linking the predicates to the semantic information elements and (2) associate further AEC-specific meaning to the different predicates by linking the semantic information elements to the ontology concepts and relations. For example, by linking the predicate "transverse_reinforcement(transverse_reinforcement)" to the "subject" and "spacing(spacing)" to the "compliance checking attribute", we can distinguish that the former is the subject of the regulatory requirement, while the latter is the compliance checking attribute of this subject. In turn, by linking the "transverse_reinforcement" (i.e., name of the predicate) to ontology concepts, we can further recognize that "transverse_reinforcement(transverse_reinforcement)" is a type of "building element". The use of semantic-based LCs also plays a central role in identifying and formalizing the activation conditions (as described in the following section).

Insert Figure 2

Insert Table 3

Regulatory Information Logic Clauses

Two alternative subschemas were developed. Alternative I implements a closed world assumption (i.e., the assumption that what is not known to be true is false) for noncompliance detection, which means that the design information that are not found to be compliant are regarded as noncompliant. Alternative II implements an open world assumption (i.e., the assumption that what is not known to be true is unknown) for noncompliance detection, which means that design information must be explicitly found to be noncompliant to be regarded as noncompliant. The two alternatives differ in two primary ways: (1) in the way regulatory information LCs are represented; and (2) in the way regulatory information LCs are executed.

<u>Alternative I</u>

In Alternative I, regulatory information LCs are represented using logic rules. Two types of regulatory information LCs are represented (as per Fig. 3): primary regulatory information LCs and secondary regulatory information LCs (will be called primary and secondary LCs hereafter). Each regulatory requirement is represented as one primary LC and is supported by two secondary LCs. For example (see Fig. 3), the following regulatory provision (here the provision has one requirement about "spacing") is represented using PLC1, SLC1, and SLC2: "Spacing of transverse reinforcement shall not exceed 8 inches" (from Provision 1908.1.3 of Chapter 19 in IBC 2009).

Insert Figure 3

A primary LC is the core representation of a requirement. It represents the compliance case. The premise of a primary LC represents the conditions of the requirement (e.g., the conditions that would make the spacing of transverse reinforcement compliant) and the conclusion of a primary LC represents the consequent result which is the compliance with the requirement (e.g., the compliance of the spacing of the transverse reinforcement). As such, compliance is deduced from primary LCs (compliance case), while noncompliance cases are inferred based on compliance

As mentioned in the preceding subsection, the predicates in the primary LCs are linked to "semantic information elements", where the instances of these semantic information elements are, in turn, linked to ontology concepts and relations. For example (see Fig. 3), the predicates to the left of "⊃" in the primary rule PLC1 are the premise conditions of the LC, where each predicate represents an ontology concept or an ontology relation (a partial view of the ontology is also shown in Fig. 3). For example, the predicate "transverse_reinforcement(transverse_reinforcement)" represents the concept “transverse reinforcement” (subconcept of “building element” which is a “subject”), the predicate “spacing(spacing)” represents the concept “spacing” (subconcept of “quantity”, which is a “compliance checking attribute”), and the predicate “has(transverse_reinforcement, spacing)” represents the relation “transverse reinforcement”-“has”-“spacing”, which is a relation between a “subject” and a “compliance checking attribute”.

The conclusion of a primary LC is one single predicate that takes the following standardized pattern: “compliance_*ComplianceCheckingAttribute*_of_*Subject*(*complianceCheckingAttribute*)”, where the *ComplianceCheckingAttribute* and the *Subject* are the “compliance checking attribute” and the “subject” of the requirement, respectively. For example (see Fig. 3), the following predicate represents the conclusion of PLC1, which is constructed from the “subject” (“transverse reinforcement”) and the “compliance checking attribute” (“spacing”) of the requirement: “compliance_spacing_of_transverse_reinforcement(spacing)”.

If multiple regulatory requirements exist in one regulatory provision, each of the regulatory requirements is represented in a separate primary LC and reported separately. For example, for regulatory provision RP1, the “height”, “thickness”, and “unbalanced_fill” of the “wall” instance are represented in three separate primary LCs and reported separately.

mm), and the wall shall retain no more than 4 feet (1219 mm) of unbalanced fill.” (from Provision 1908.1.8 of Chapter 19 in IBC 2009)

Each primary LC is supported by two secondary LCs: (1) one for representing the conditions that activate the checking of the requirement, and (2) one for representing the consequences of the compliance checking result. Activation conditions (1) help prevent missing information from leading to false positives because missing information would lead to failure in activation, and (2) avoid exhaustive search over all design information LCs and thus lead to higher computational efficiency (during software implementation). The activation conditions for each regulatory requirement define the premise conditions of the requirement, which are generated from the respective primary LC by separating the premise conditions [e.g., "spacing(spacing), transverse_reinforcement(transverse_reinforcement), has(transverse_reinforcement,spacing)"] from the consequent prescription [e.g., "¬greater_than(spacing, quantity(8,Inches))"]. The semantic representation helps recognize the premise conditions of a regulatory requirement in a primary LC through the semantic information elements. The consequences for each requirement are also linked to instances of semantic information elements. A "compliance checking result" could be a compliance or noncompliance, and a "compliance checking consequence" is the outcome or effect of the "compliance checking result" such as a suggested corrective action. For example, the checking of the regulatory requirement represented in PLC1 is activated using SLC1. If any information in the body of SLC1 is missing (e.g., the relation between the spacing and the transverse reinforcement is missing), then the checking with PLC1 would not be activated, which would avoid a blind activation of SLC1 that would lead to a false positive noncompliance. For the checking result, using SLC2, an output message including whether the result is compliant or noncompliant is printed out, together with the relevant provision number (i.e., "1908.1.3") and the regulatory requirement ID. If the result is noncompliant, a corrective suggestion on how to fix the noncompliance is provided (i.e., "the spacing should be less than or equal to 8 inches"). The modeling of compliance checking consequences allows for deep compliance reasoning (i.e., not only finding instances of noncompliance but also offering an analysis of the noncompliance and providing suggestions for corrective actions).

Alternative II

In Alternative II, each regulatory requirement is represented using two logic rules (LCs), one for representing the compliance case and one for explicitly representing the noncompliance case. As such, noncompliance cases are explicitly represented instead of being inferred based on compliance cases – following an open world assumption. For example, in Fig. 4, (1) LC3 and LC4 are two LCs representing the compliance case and noncompliance case of a regulatory requirement, respectively. As such, the premise of LC3 represents the conditions of compliance with a requirement, whereas that of LC4 represents the conditions of noncompliance with the same requirement. Different from Alternative I, there is no need to use secondary LCs for representing activation conditions and consequences of compliance checking results, because compliance and noncompliance cases are represented separately. As such, the conclusions of LC3 and LC4, represent both the "compliance checking results" (compliant or noncompliant) and the "compliance checking consequences" (e.g., corrective suggestion on how to fix the

Different from Alternative I, if multiple regulatory requirements exist in one regulatory provision, the compliance cases of all regulatory requirements (of that single regulatory provision) are represented in one single regulatory information LC and reported jointly in one single compliance instance; there is no need to separate the multiple requirements because compliance and noncompliance cases are represented separately. For example, for the regulatory provision RP1, all three regulatory requirements (i.e., for "height", "thickness", and "unbalanced_fill") for the "wall" instance are represented in one single regulatory information LC and reported jointly in one single compliance instance. To avoid the enumeration of all possible combinations of noncompliance cases (e.g., height is compliant but thickness is not, thickness is compliant but height is not, etc.,), the noncompliance case of each regulatory requirement is represented separately. For example, the noncompliance cases for "height", "thickness", and "unbalanced_fill" are represented separately.

Design Information Logic Clauses

Design information LCs, in both Alternative I and Alternative II, are represented using logic facts. Each single design fact (e.g., Transverse_reinforcement101 is an instance of transverse reinforcement) is represented as one single design information LC (logic fact). A design fact could be a concept fact or a relation fact. A concept fact is represented by a design information LC consisting of a unary predicate, with the name of the concept as the name of the predicate. For example (see Fig. 3 and Fig. 4), "transverse_reinforcement(Transverse_reinforcement101)" is a unary predicate that represents an instance of the concept "transverse reinforcement" and "spacing(Spacing103)" is a unary predicate that represents an instance of the concept "spacing". A relation fact is represented by a design information LC consisting of a binary or n-nary predicate, with the name of the relation as the name of the predicate. For example, "has(Transverse_reinforcement101, Spacing103)" is a binary predicate that represents the relation that "Transverse_reinforcement101" has a "Spacing103" and "has_quantity(Spacing103, 6, Inches)" is a n-nary predicate which indicates that the quantity for "Spacing103" is 6 inches.

Functional Built-in Logic Clauses

Six types of functional built-in LCs were developed and included in the IRep and CR schema, as per Table 4: unit conversion LCs, quantity comparison LCs, quantity conversion LCs, sum of quantities LCs, quantity arithmetic computation LCs, and rule checking LCs.

Insert Table 4

Software Implementation

Logic Programming Language

The proposed IRep and CR schema was implemented in B-Prolog logic programming language. A FOL-based programming language is needed for representation to allow for automated reasoning. 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). Prolog is a logic platform for implementing HC representation and reasoning. Although B-Prolog was selected in this paper, any other FOL-based programming language could be selected to represent the IRep and CR schema instead; the proposed schema does not rely on any specific FOL-based programming language.

B-Prolog is a good fit for representing the IRep and CR schema because: (1) B-Prolog builds in classic Prolog, which is the most widely-used logic programming language and reasoner (Costa 2009), (2) the built-in classic Prolog in B-Prolog has an underpinning reasoner that enables automated inference-making through well-developed unification, backtracking, depth-first search, and rewriting techniques (Portoraro 2011), and (3) the compatibility of B-Prolog with C and Java programming languages renders further ACC system user interface development and implementation smoother. The syntax in B-Prolog differs from the original FOL syntax, as summarized in Table 2. When another logic programming language is used, such as Answer Set Programming (ASP) or Datalog, the syntax of some functions may need to be adjusted. The slight difference in reasoning implementations across different FOL-based programming languages may also cause certain advantages or limitations in the reasoning. The discussion of the potential advantages and limitations of the different FOL-based programming languages is outside the scope of this paper.

Insert Table 2

Regulatory Information Logic Clauses

<u>Alternative I</u>

In Alternative I, regulatory information LCs (represented in the schema in the form of logic rules) are implemented as B-Prolog rules. The built-in “writeln()” predicate in B-Prolog is used for the output function. For executing the regulatory LCs, the user specifies the list of subjects (e.g., building elements such as walls and doors) or subjects and attributes to check and accordingly the subjects in the specified list are sequentially checked one by one. By default, a “select all” option is used if a user does not desire to specify specific subjects to check. The sequence of checking in Alternative I is, thus, called subject-oriented. In the implementation of Alternative I, the search strategy is defined as follows: “for each selected subject instance, search through all regulatory information LCs to check if the activation conditions are satisfied, and if satisfied, then check the instance against the matched regulatory information LC”. The reasoning is supported by functional built-in LCs in the support module. An example of the implementation, corresponding to the example in Fig.3, is shown in Fig. 4.

Insert Figure 4

Alternative II

In Alternative II, regulatory information LCs (represented in the schema in the form of logic rules) are implemented as B-Prolog directives. In comparison to B-Prolog rules, B-Prolog directives execute upon loading without conditions and, thus, provide more flexibility to the design of regulatory information LCs activation mechanisms. It is important to study how such a more flexible rule activation mechanism affects the performance of noncompliance detection. In each directive, (1) the built-in "findall" predicate is used to leverage the inherent depth-first search strategy and backtracking techniques of B-Prolog to find all instances of the subject that satisfy the premise conditions of the requirement in the directive, (2) the "sort" predicate is used to sort the matched instances and remove duplicated instances, and (3) the "foreach" predicate is used to report the output results for each matched instance. In contrast to Alternative I, for executing the regulatory LCs in Alternative II, the user does not specify what subjects to check. All subjects that satisfy premise conditions in the regulatory information LCs are detected and checked. The sequence of checking follows the sequence of regulatory information LCs (i.e., the directives), which in turn follows the sequence of regulatory provisions in the original regulatory document. The sequence of checking in Alternative II is, thus, called regulation-oriented. An example of the implementation, corresponding to the example in Fig.3, is shown in Fig. 5.

Insert Figure 5

Design Information Logic Clauses

Design information LCs (represented in the schema in the form of logic facts), in both Alternative I and Alternative II, are implemented as B-Prolog facts.

Functional Built-in Logic Clauses

The six types of functional built-in LCs in the IRep and CR schema were implemented in B-Prolog syntax, as shown in Fig. 5. One single rule checking LC is used in Alternative I and no rule checking LCs are used in Alternative II [not needed since the checking is initiated in each directive utilizing the inherent ("findall") search strategies in B-Prolog]. As shown in Fig. 3, the rule checking LC in Alternative I is: "checklist(L) :- foreach(X in L, check(X))." This rule checking LC initiates the checking of subjects (in the user-specified list or default "select all" list), sequentially, one by one following the sequence in the list. In total, 71 functional built-in LCs were developed and used for Alternative I, and all 71 LCs except one (the rule checking LC) were used for Alternative II.

Experimental Testing

To empirically test the proposed IRep and CR schema, Alternative I and Alternative II were tested in representing and reasoning about the quantitative regulatory requirements in Chapter 19 of IBC 2009 and the design information of a two-story duplex apartment test case for checking the compliance of the design. The results of noncompliance detection under each subschema alternative were evaluated in terms of recall and precision. To highlight the potential advantages of ACC using the proposed schema, the time efficiency of automated checking was also empirically tested.

Testing of Noncompliance Detection Performance

The evaluation of representation and compliance reasoning, in terms of noncompliance detection, was conducted in two ways: (1) evaluating the performance of noncompliance detection using perfect information (i.e., LCs that contain no errors); and (2) evaluating the performance of noncompliance detection using imperfect information (i.e., LCs that contain errors).

Testing Using Perfect Information

A gold standard was manually developed and used for evaluation. A gold standard refers to a benchmark against which testing results are compared for evaluation.

For testing Alternative I, both regulatory information LCs and design information LCs were manually represented/coded based on Gold Standard I (i.e., the gold standard of Alternative I). Gold Standard I was composed of two subparts: (1) the gold standard of regulatory information LCs in Chapter 19 of IBC 2009 under Alternative I, which included 198 LCs (in the form of B-Prolog rules), consisting of 66 primary LCs and 132 secondary LCs (i.e., two secondary LCs for each primary LC) and (2) the gold standard of design information LCs in the two-story duplex apartment test case, which included 146 sets of LCs (in the form of B-Prolog facts). For example, Fig. 4 shows the gold standard for representing the following provision and a set of design information, where PLC5 is one of the 198 LCs and "spacing(spacing103)" is one predicate in one of the 146 sets of LCs: "Spacing of transverse reinforcement shall not exceed 8 inches". The reasoning was then conducted automatically using the B-Prolog reasoner. The results of compliance reasoning about regulatory requirements were evaluated in terms of recall, precision, and F1 measure of noncompliance detection. Recall is the number of correctly detected noncompliance instances divided by the total number of noncompliance instances that should be detected. Precision is the number of correctly detected noncompliance instances divided by the total number of noncompliance instances that have been detected. F1 measure is the harmonic mean of recall and precision.

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000583
Zhang, J. and El-Gohary, N. (2016). "Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking." J. Comput. Civ. Eng. , 10.1061/(ASCE)CP.1943-5487.0000583 , 04016037.

For testing Alternative II, the same testing procedure was followed, except that both regulatory information LCs and design information LCs were manually coded based on Gold Standard II (i.e., the gold standard of Alternative II). Gold Standard II was composed of two subparts: (1) the gold standard of regulatory information LCs in Chapter 19 of IBC 2009 under Alternative II, which included 137 LCs (in the form of B-Prolog directives), and (2) the gold standard of design information LCs in the two-story duplex apartment test case, which included 146 sets of LCs (in the form of B-Prolog facts). For example, Fig. 5 shows the gold standard for representing the following provision and a set of design information, where LC3 is one of the 137 LCs and "spacing(spacing103)" is one predicate in one of the 146 sets of LCs: "Spacing of transverse reinforcement shall not exceed 8 inches".

Testing Using Imperfect Information

The testing using imperfect information was conducted using a similar procedure to that of testing using perfect information, except that a set of automatically-coded regulatory information LCs were used instead of the manually-coded ones. These automatically-coded LCs come from an existing dataset by Zhang and El-Gohary (2015). The dataset includes a set of LCs that were automatically generated from Chapter 19 of IBC 2009 using algorithms for automated information extraction (to automatically extract information from regulatory documents into semantic tuples) and automated information transformation (to automatically transform the semantic tuples into LCs). The use of automatically-coded regulatory information LCs allows for evaluating the performance of compliance reasoning using imperfect information (i.e., because the automatically-coded LCs contain errors). For the dataset of Alternative I, the 198 regulatory information LCs contained xxx errors. For the dataset of Alternative II, the 137 regulatory information LCs contained xxx errors. ***Testing of Time Performance***

To compare the time efficiency of the two alternative subschemas, the durations of automated compliance reasoning using perfect information, under Alternative I and Alternative II, were calculated using the time keeping predicates in B-Prolog. Since Alternative I is subject-oriented while Alternative II is regulation-oriented, the duration of compliance reasoning is measured differently for each alternative. For Alternative I, the duration is measured from the time of initializing the compliance reasoning about the first design fact to the time of finishing compliance reasoning about the last design fact (design information LC set No. 146). For Alternative II, the duration is measured from the time of initializing compliance reasoning with the first regulatory requirement to the time of finishing compliance reasoning with the last regulatory requirement (regulatory information LC No. 137).

Experimental Results and Discussion

Results of Noncompliance Detection Performance

Results Using Perfect Information

The experimental results are summarized in Table 5. When using perfect information, on the testing data, both Alternative I and Alternative II achieved 100% recall, precision, and F1 measure in noncompliance detection. The compliance checking results and suggestions for fixing noncompliance instances were also correctly reported in the output. This shows that the proposed IRep and CR schema is effective in supporting ACC. Fig. 7 shows the checking results of "wall1" to "wall5" using Alternative I. For example, "wall1" has "height3", "thickness1", and "unbalanced_fill1"; and "wall2" has "height4", "thickness2", and "unbalanced_fill2", where Rule43 and Rule44 focus on height checking, Rule43-1 and Rule45 focus on thickness checking, and Rule43-2 and Rule46 focus on unbalanced fill checking. Fig. 8 shows the checking results of "wall1" to "wall5" using Alternative II, where Rule44, Rule 45, and Rule 46 represent the noncompliance cases of "height", "thickness", and "unbalanced fill", respectively, and Rule 43 represents the compliance cases of all three regulatory requirements jointly.

Insert Table 5

Insert Figure 7

Insert Figure 8

Results Using Imperfect Information

When using imperfect information, on the testing data, Alternative I and Alternative II achieved 98.7%, 87.6%, and 92.8% and 77.2%, 98.4%, and 86.5% recall, precision, and F1 measure in noncompliance detection, respectively. The recall of Alternative I outperformed that of Alternative

II, while the precision of Alternative II outperformed that of Alternative I. This reflects the trade-off between recall and precision.

In Alternative I, a high recall is achieved because it can block some errors in LCs from propagating to false negatives in noncompliance detection results; a total of 15 regulatory information LCs included errors, yet only 1 of them propagated into a false negative in noncompliance detection. Errors in predicates other than quantity comparison predicates [e.g., greater_than(Spacing,quantity(8,inches)) in Fig. 5] could be blocked from leading to false negatives. Because, in Alternative I, all selected design subjects are checked, noncompliance instances are less likely to be missed. However, most of the errors in LCs still lead to false positives, which makes the precision relatively lower than recall.

In Alternative II, a higher precision is achieved because some false positives are blocked since noncompliance cases are explicitly represented (following an open world assumption), whereas in Alternative I noncompliance cases are inferred based on compliance cases (i.e., if a primary LC is not compliant, then it is noncompliant – following a closed world assumption). Such explicit representation, however, make the representation quite sensitive to errors in regulatory information LCs. Any error in a regulatory information LC is highly likely to cause a failure to activate the checking of the respective logic directive in Alternative II, which would result in a drop in recall.

Alternative I is, thus, more suitable for ACC applications, because recall of noncompliance instances is more important than precision. Overall the F1 measure of Alternative I is also higher than that of Alternative II.

Results of Time Performance

Automated compliance reasoning with quantitative regulatory requirements of Chapter 19 of IBC 2009 using the proposed IRep and CR schema took fractions of a second. The experiments were conducted using a laptop with a random access memory (RAM) of 3.73 gigabytes (GB) and an Advanced Micro Devices (AMD) C-50 processor with 1.00 gigahertz (GHZ). With an increase in the central processing unit (CPU) speed and/or RAM, the time taken for automated compliance reasoning using the proposed IRep and CR schema could be further reduced. Under alternative I, compliance reasoning took only 55% (0.515 seconds) of the time taken under Alternative II (0.936 seconds). The main reason for this difference is the increased amount of design facts to search in Alternative II, because the representation under Alternative II exhaustively searched all design facts (even the ones not related to building elements) to detect those satisfying premise conditions of each regulatory information LC, whereas the representation under Alternative I only searched from the set of subjects (i.e., building elements) in the list (the default "select all" list was used).

Contribution to the Body of Knowledge

The proposed IRep and CR schema contributes to the body of knowledge in four main ways. First, the proposed schema provides a new way for representing construction regulatory provisions and design information in a logic-based, semantic format. The first order logic-based representation allows for using a standardized reasoning method to facilitate complete automation in ACC reasoning. The semantic representation supports the logic-based representation and reasoning by providing the needed description of domain knowledge. This work empirically shows that the proposed schema achieved 100% recall and precision in noncompliance detection using perfect information, and achieved high recall (98.7%) and precision (87.6%) in noncompliance detection using imperfect information. Second, this work offers and compares two subschemas – Alternative

I and Alternative II – for representing regulatory requirements following a closed world assumption and an open world assumption for noncompliance detection, respectively. The experimental results show that while both subschemas could support the task of ACC with a relatively high performance – in terms of recall and precision of noncompliance detection, Alternative I results in higher recall and is, thus, more suitable for ACC applications. Third, the proposed schema (following Alternative I) offers a way to help prevent missing information in closed world assumption schemas from leading to false positives in noncompliance detection. This is achieved using semantic-based (ontology-based) logic clauses and compliance checking activation conditions. Fourth, a support module that consists of a set of logic clauses was developed, as part of the schema, to provide ACC-specific computational and reasoning support when using logic-based reasoners. This module could be reused by other researchers to support ACC applications.

Conclusions

This paper presented a new first order logic-based information representation and compliance reasoning (IRep and CR) schema for representing and reasoning about regulatory information and design information for checking regulatory compliance of building designs. The schema formalizes the representation of regulatory information and design information in the form of semantic-based (ontology-based) logic clauses that could be directly used for automated compliance reasoning. The proposed IRep and CR schema was implemented in B-Prolog logic programming language to utilize B-Prolog's reasoner for automated reasoning. Two alternative subschemas, Alternative I and Alternative II, were proposed and tested, following a closed world assumption and an open world assumption in noncompliance detection, respectively. Activation conditions were used in Alternative I to avoid false positives caused by missing information. A reusable support module was developed for ACC-specific reasoning support.

The proposed IRep and CR schema was tested in representing and reasoning about quantitative regulatory requirements in Chapter 19 of IBC 2009 and design information in a two-story duplex apartment test case. Two experiments were conducted to test the schema using perfect information and imperfect information. Using perfect information, on the testing data, both Alternative I and Alternative II achieved 100% recall, precision, and F1 measure in noncompliance detection. It took less than one second to automatically check the 146 sets of design information with quantitative regulatory requirements in Chapter 19 of IBC 2009. Using imperfect information, on the testing data, Alternative I and Alternative II achieved 98.7%, 87.6%, and 92.8%, and 77.2%, 98.4%, and 86.5% recall, precision, and F1 measure, respectively. Alternative I blocks some false negatives and thus results in a higher recall, while Alternative II blocks some false positives and thus results in a higher precision. Because high recall is more important than high precision in ACC, to avoid missing noncompliance instances, Alternative I is more suitable for ACC applications. One limitation of this work is that, due to the large amount of manual effort needed in developing a gold standard for evaluation, the proposed IRep and CR schema was only tested in representing and reasoning about regulatory requirements in one chapter of IBC 2009 and design information in one test case. While similar performance could be expected on other chapters of IBC 2009, other regulatory documents, and other design test cases, more empirical testing is needed for verification, especially when using imperfect information.

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

Awad, A., Smirnov, S. , and Weske, M. (2009). "Resolution of compliance violation in business process models: a planning-based approach." *Proc., OTM '09*, 6-23.

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

Bikakis, A., and Antoniou, G. (2005). "DR-Prolog: A system for reasoning with rules and ontologies on the semantic web." *Proc. 25th American National Conference on Artificial Intelligence (AAAI-2005)*, AAAI, Palo Alto, California, 1594-1595.

Costa, V.S. (2009). "On just in time indexing of dynamic predicates in Prolog." *Progress in Artificial Intelligence, Lect. Notes Comput. Sc.*, 5816(2009), 126-137.

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

Dimyadi, J., and Amor, R. (2013). "Automated building code compliance checking - where is it at?" <http://www.academia.edu/4094621/> (Dec. 24, 2013).

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

Denecker, M., Vennekens, J., Vlaeminck, H., Wittocx, J., and Bruynooghe, M. (2011). "Answer set programming's contributions to classical logic: an analysis of ASP methodology." *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, Springer-Verlag, Berlin, Heidelberg, 12-32.

Department of Energy (DOE). (2014). “Building energy codes program: software and web tools.” <https://www.energycodes.gov/software-and-web-tools-0> (Jul. 09, 2014).

Dershowitz, N., and Plaisted, D. (2001). “Rewriting.” *Chapter 9 in Handbook of Automated Reasoning, Volume 1.*, Elsevier Science Publishers B.V., Amsterdam, Dutch.

DeYoung, H., Garg, D., Kaynar, D., and Datta, A. (2010). “Logical specification of the GLBA and HIPAA privacy laws.” Carnegie Mellon University, Pittsburgh, PA.

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

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.

Environment Protection Agency (EPA). (2013). “Regulatory information in construction sector.” <http://www2.epa.gov/regulatory-information-sector/construction-sector-naics-23> (Dec. 24, 2013).

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

Fiatech. (2014). “Automated code plan checking tool-proof-of-concept (Phase 2).” < http://www.fiatech.org/index.php/projects/active-projects/162-active-projects/projects-management/593-automated-code-plan-checking-tool-proof-of-concept> (May. 22, 2014).

Garg, D., Jia, L., and Datta, A. (2011). “Policy auditing over incomplete logs: theory, implementation and applications.” *Proc., 18th ACM Conf. Computer and Communications Security*, ACM, New York, NY, 151-162.

Grimm, S., and Motik, B. (2005). “Closed world reasoning in the Semantic web through epistemic operators.” *Second International Workshop on OWL: Experiences and Directions (OWLED 2006)*, Galway, Ireland, 2005.

Halpern, J.Y., and Weissman, V. (2008). “Using first-order logic to reason about policies.” *J. ACM Trans. Inf. Sys. Security (TISSEC)*, 11(4), 1-41.

Hebeler, J., Fisher, M., Blace, R., Perez-Lopez, A., and Dean, M. (2009). “Semantic web programming.” Wiley Publishing, Inc., Indianapolis, IN.

Hjelseth, E. (2012). “Converting performance based regulations into computable rules in BIM based model checking software.” *eWork and eBusiness in Architecture, Engineering and Construction ECPPM 2012*, Taylor & Francis Group, London, UK, 461-469.

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

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

International Code Council (ICC). (2013a). “International code council online library.” <http://publicecodes.cyberregs.com/icod/> (Dec. 24, 2013).

International Code Council (ICC). (2013b). “International Code Council.” *AEC3*, <http://www.aec3.com/en/5/5_013_ICC.htm> (Dec. 24, 2013).

Jain, D., Krawinkler, H., and Law, K.H. (1989). “Knowledge representation with logic.” CIFE Technical Report Number 13, Stanford University, Stanford, CA. Kerrigan, S., and Law, K.H.

(2003). “Logic-based regulation compliance-assistance.” *Proc., Ninth Int. Conf. Artificial Intelligence and Law (ICAIL 2003)*, ACM, New York, NY, 126-135.

Knorr, M., Alferes, J.J., and Hitzler, P. (2011). “Local closed-world reasoning with description logics under the well-founded semantics.” *Artificial Intelligence*, 175, 1528-1554.

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

Lutz, C., Seylan, I, and Wolter, F. (2012). “Mixing open and closed world assumption in ontology-based data access: non-uniform data complexity.” *Proc., Int. Workshop Description Logics, Elsevier,* Atlanta, GA.

Martelli, A. and Montanari, U. (1982). “An efficient unification algorithm.” *ACM T. Progr. Lang. Sys.*, 4(2), 258-282.

Noy, N. F., and McGuinness, D.L. (2002). “Ontology development 101: A guide to creating your first ontology.” Knowledge System Laboratory, Stanford, CA.

Portoraro, F. (2011). “Automated reasoning.” *The Stanford encyclopedia of philosophy (Summer 2011 Edition)*, <http://plato.stanford.edu/archives/sum2011/entries/reasoning-automated/> (Dec. 26, 2013).

Rasdorf, W.J., and Lakmazxaheri, S. (1990). “A logic-based approach for processing design standards.” *Artif. Intell. Eng. Des. Anal. Manuf.*, 4(3), 179-192.

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

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

Sensoy, M., Mel, G.D., Vasconcelos, W.W., and Norman, T.J. (2011). "Ontological logic programming." *WIMS '11*, ACM, New York, NY. Singapore Building and Construction Authority (SBCA). (2006). "Construction and real estate network: Corenet systems." <http://www.corenet.gov.sg/> (Aug. 31, 2014).

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.*, 10.1061/1192(ASCE) 0887-3801(2010)24:2(203), 203-211.

Tarjan, R. (1972). "Depth-first search and linear graph algorithms." *SIAM J. Comput.*, 1(2), 146-160.

Tubella, J. and Gonzalez, A. (1998). "Combining depth-first and breadth-first search in Prolog execution." *Proc., 1994 Joint Conference on Declarative Programming GULP-PRODE'94*, 2, 452-453.

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

Yurchyshyna, A., Faron-Zucker, C., Thanh, N.L., and Zarli, A. (2010). "Adaptation of the domain ontology for different user profiles: application to conformity checking in construction." *Lect. Notes Bus. Inf.*, 45, 128-141.

Zhou, N. (2012). "B-Prolog user's manual (version 7.8): Prolog, agent, and constraint programming." Afany Software. <http://www.probp.com/manual/manual.html> (Dec. 28, 2013).

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.

Zhang, S., Teizer, J., Lee, J., Eastman, C.M., and Venugopal, M. (2013). “Building information modeling (BIM) and safety: automatic safety checking of construction models and schedules.” *Autom. Constr.*, 29(2013), 183-195.

Tables

Table 1. The Meaning of Logic Operators in FOL

Logic operator	Meaning
Conjunction ∧	A ∧ B means A is true and B is true
Disjunction ∨	A ∨ B means A is true or B is true
Negation ¬	¬A means A is not true
Implication ⊃	A ⊃ B means A implies B (if A is true then B is true)
Assignment →	A → B means assigning the value of B to A

Table 2. The Meaning of Logic Operators in B-Prolog

Logic operator	Meaning
Conjunction ,	A , B means A is true and B is true
Disjunction ;	A ; B means A is true or B is true
Negation not	Not A means A is not true
Implication :-	B :- A means A implies B (if A is true then B is true)
Assignment "is"	A is B means assigning the value of B to A

Table 2. The syntax of FOL and B-Prolog

Name in FOL	Syntax in FOL	Name in B-Prolog	Syntax in B-Prolog
Conjunction	$\wedge$	Conjunction	,
Disjunction	$\vee$	Disjunction	;
Negation	$\neg$	Negation	not
Implication	$\supset$	Implication	:-
Constant	String starting with an upper-case letter	Constant	String starting with a lower-case letter
Variable	String starting with a lower-case letter	Variable	String starting with an upper-case letter
Universal Quantifier	$\forall$	-	-
Existential Quantifier	$\exists$	-	-
Predicate	p(arg1,arg2,…)	Predicate	p(arg1,arg2,…)
Function	f(arg1,arg2,…)	Function	f(arg1,arg2,…)
rule	$b1 \wedge b2 \wedge b3, ... bn \supset h$	rule	h :- b1, b2, b3, … bn.
fact	p(arg1,arg2,…)	fact	p(arg1,arg2,…)
		directive	:- b1, b2, b3, … bn.

Table 3. Semantic Information Elements

Semantic information element	**Definition**
Subject	An ontology concept that describes a "thing" (e.g., building object, space) that is subject to a particular regulation or norm.
Compliance checking attribute	An ontology concept that describes a specific characteristic of a "subject" by which its compliance is assessed.
Deontic operator indicator	A term or phrase that indicates the deontic type of the requirement (i.e., whether it is an obligation, permission, or prohibition).
Quantitative relation	A term or phrase that defines the type of relation for the quantity (e.g., "increase" is a quantitative relation).
Comparative relation	An ontology relation that is commonly used for comparing quantitative values (i.e., comparing an existing value to a required minimum or

	maximum value), including “greater than or equal to”, “greater than”, “less than or equal to”, “less than”, and “equal to”.
Quantity value	A data value, or a range of values, that defines the quantified requirement.
Quantity unit	The unit of measure for a “quantity value”.
Quantity reference	A term or phrase that refers to another quantity (which includes a value and a unit).
Quantity	A pair of “quantity value” and “quantity unit” or a pair of “quantity value” and “quantity reference”.
Restriction	A term, phrase, or clause (which is composed of one or more concepts and/or relations) that places a constraint on the “subject”, “compliance checking attribute”, “comparative relation”, “quantity”, or the full requirement.
Exception	A phrase or clause (which is composed of one or more concepts and/or relations) that defines a condition where the described requirement does not apply.

Table 4. Functional Built-in Logic Clauses

Logic clause (LC) type	**Function**
Unit conversion LCs	Define the conversion factors betweent units.
Quantity comparison LCs	Implement quantity comparison functions for basic comparative relations such as "greater than or equal to".
Quantity conversion LCs	Implement the conversions of quantities between different units based on the corresponding conversion factors defined in unit conversion LCs.
Sum of quantities LCs	Implement the function of summing up a list of enumerated quantities for calculations of total quantities.
Quantity arithmetic computation LCs	Define arithmetic operations on quantity values and quantity units.
Rule checking LCs	Initiate the checking and define the sequence of checking.

Table 5. Experimental Results of Experiment #1 and Experiment #2

Subschema	**Parameter/measure**	**Results**	
		Using perfect information	**Using imperfect information**
Alternative I (Closed world assumption)	Number of noncompliance instances in gold standard	79	79
	Number of noncompliance instances detected	79	89
	Number of noncompliance instances correctly detected	79	78
	Recall of noncompliance detection	100%	98.7%
	Precision of noncompliance detection	100%	87.6%
	F1 measure of noncompliance detection	100%	92.8%
Alternative II (Open world assumption)	Number of noncompliance instances in gold standard	79	79
	Number of noncompliance instances detected	79	62
	Number of noncompliance instances correctly detected	79	61
	Recall of noncompliance detection	100%	77.2%
	Precision of noncompliance detection	100%	98.4%
	F1 measure of noncompliance detection	100%	86.5%

Fig. 1.

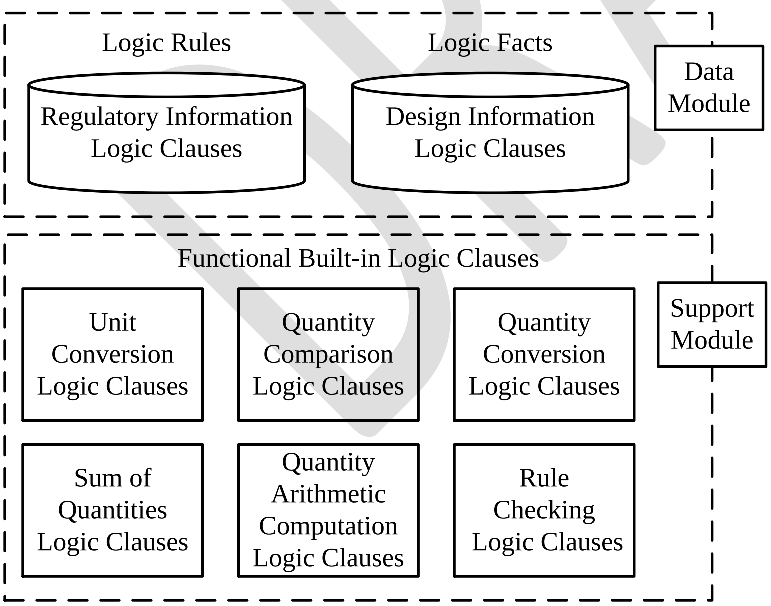


Fig. 2.

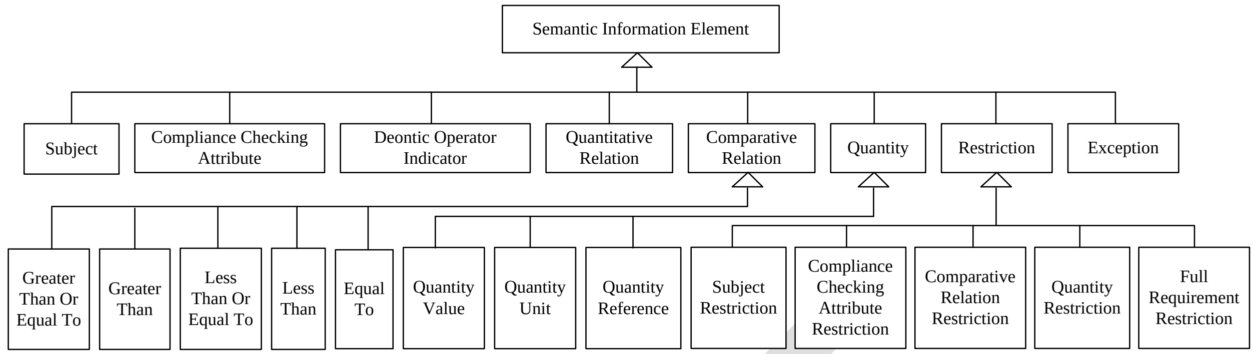
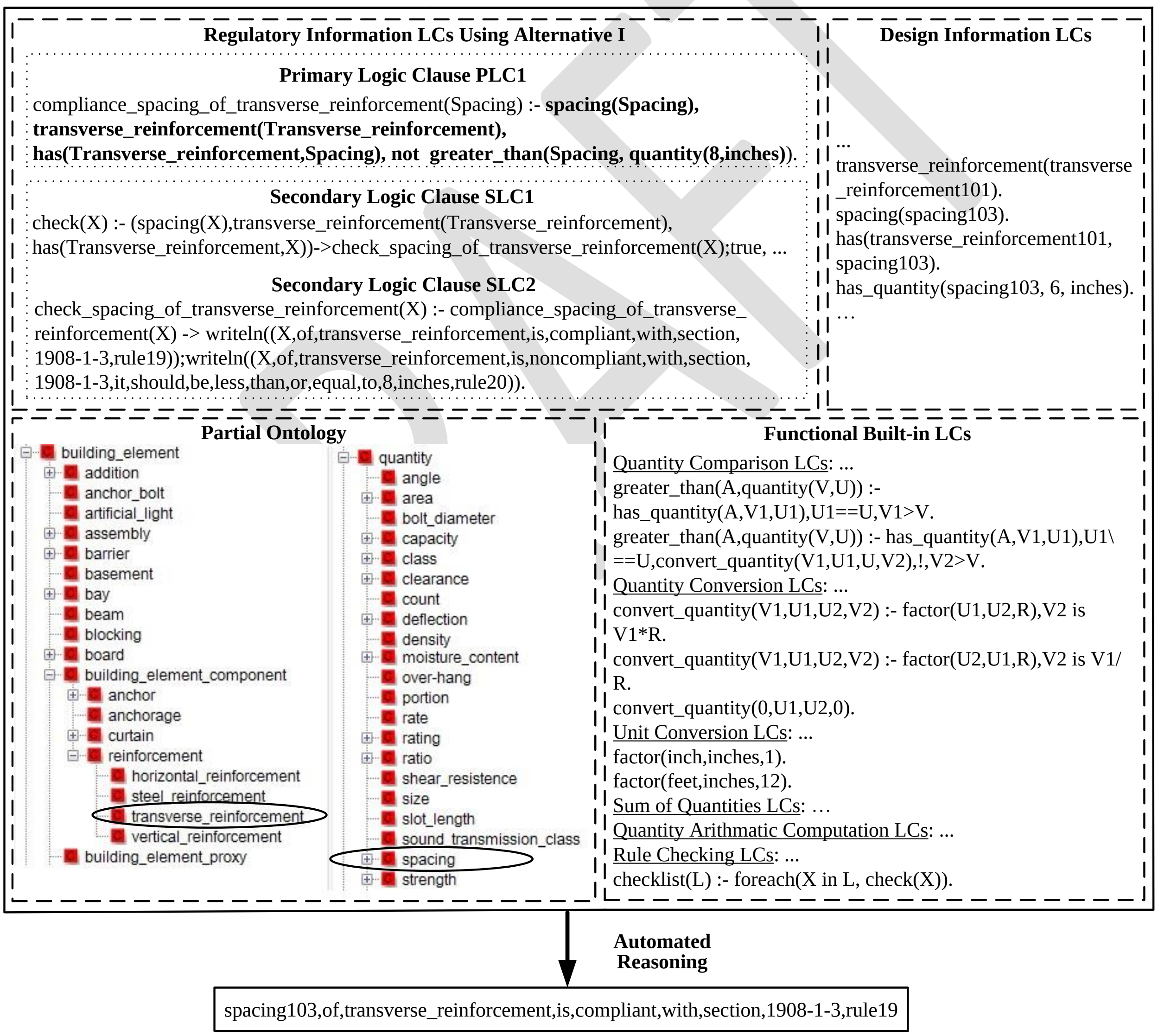

Semantic Information Element
Subject
Compliance Checking Attribute
Deontic Operator Indicator
Quantitative Relation
Comparative Relation
Quantity
Restriction
Exception
Greater Than Or Equal To
Greater Than
Less Than Or Equal To
Less Than
Equal To
Quantity Value
Quantity Unit
Quantity Reference
Subject Restriction
Compliance Checking Attribute Restriction
Comparative Relation Restriction
Quantity Restriction
Full Requirement Restriction

Fig. 3.


Regulatory Information LCs Using Alternative I
Primary Logic Clause PLC1
compliance_spacing_of_transverse_reinforcement(Spacing) :- spacing(Spacing), transverse_reinforcement(Transverse_reinforcement), has(Transverse_reinforcement,Spacing), not greater_than(Spacing, quantity(8,inches)).
Secondary Logic Clause SLC1
check(X) :- (spacing(X),transverse_reinforcement(Transverse_reinforcement), has(Transverse_reinforcement,X))->check_spacing_of_transverse_reinforcement(X);true, ...
Secondary Logic Clause SLC2
check_spacing_of_transverse_reinforcement(X) :- compliance_spacing_of_transverse_reinforcement(X) -> writeln((X,of,transverse_reinforcement,is,compliant,with,section, 1908-1-3,rule19));writeln((X,of,transverse_reinforcement,is,noncompliant,with,section, 1908-1-3,it,should,be,less,than,or,equal,to,8,inches,rule20)).
Design Information LCs
...
transverse_reinforcement(transverse_reinforcement101).
spacing(spacing103).
has(transverse_reinforcement101, spacing103).
has_quantity(spacing103, 6, inches).
…
Partial Ontology
building_element
addition
anchor_bolt
artificial_light
assembly
barrier
basement
bay
beam
blocking
board
building_element_component
anchor
anchorage
curtain
reinforcement
horizontal_reinforcement
steel_reinforcement
transverse_reinforcement
vertical_reinforcement
building_element_proxy
quantity
angle
area
bolt_diameter
capacity
class
clearance
count
deflection
density
moisture_content
over-hang
portion
rate
rating
ratio
shear_resistence
size
slot_length
sound_transmission_class
spacing
strength
Functional Built-in LCs
Quantity Comparison LCs: ...
greater_than(A,quantity(V,U)) :- has_quantity(A,V1,U1),U1==U,V1>V.
greater_than(A,quantity(V,U)) :- has_quantity(A,V1,U1),U1\==U,convert_quantity(V1,U1,U,V2),!,V2>V.
Quantity Conversion LCs: ...
convert_quantity(V1,U1,U2,V2) :- factor(U1,U2,R),V2 is V1*R.
convert_quantity(V1,U1,U2,V2) :- factor(U2,U1,R),V2 is V1/R.
convert_quantity(0,U1,U2,0).
Unit Conversion LCs: ...
factor(inch,inches,1).
factor(feet,inches,12).
Sum of Quantities LCs: …
Quantity Arithmatic Computation LCs: ...
Rule Checking LCs: ...
checklist(L) :- foreach(X in L, check(X)).
Automated Reasoning
spacing103,of,transverse_reinforcement,is,compliant,with,section,1908-1-3,rule19

The published version is found in the ASCE Library here: http://ascelibrary.org/doi/abs/10.1061/(ASCE)CP.1943-5487.0000583
Zhang, J. and El-Gohary, N. (2016). "Semantic-Based Logic Representation and Reasoning for Automated Regulatory Compliance Checking." J. Comput. Civ. Eng. , 10.1061/(ASCE)CP.1943-5487.0000583 , 04016037.

Fig. 4

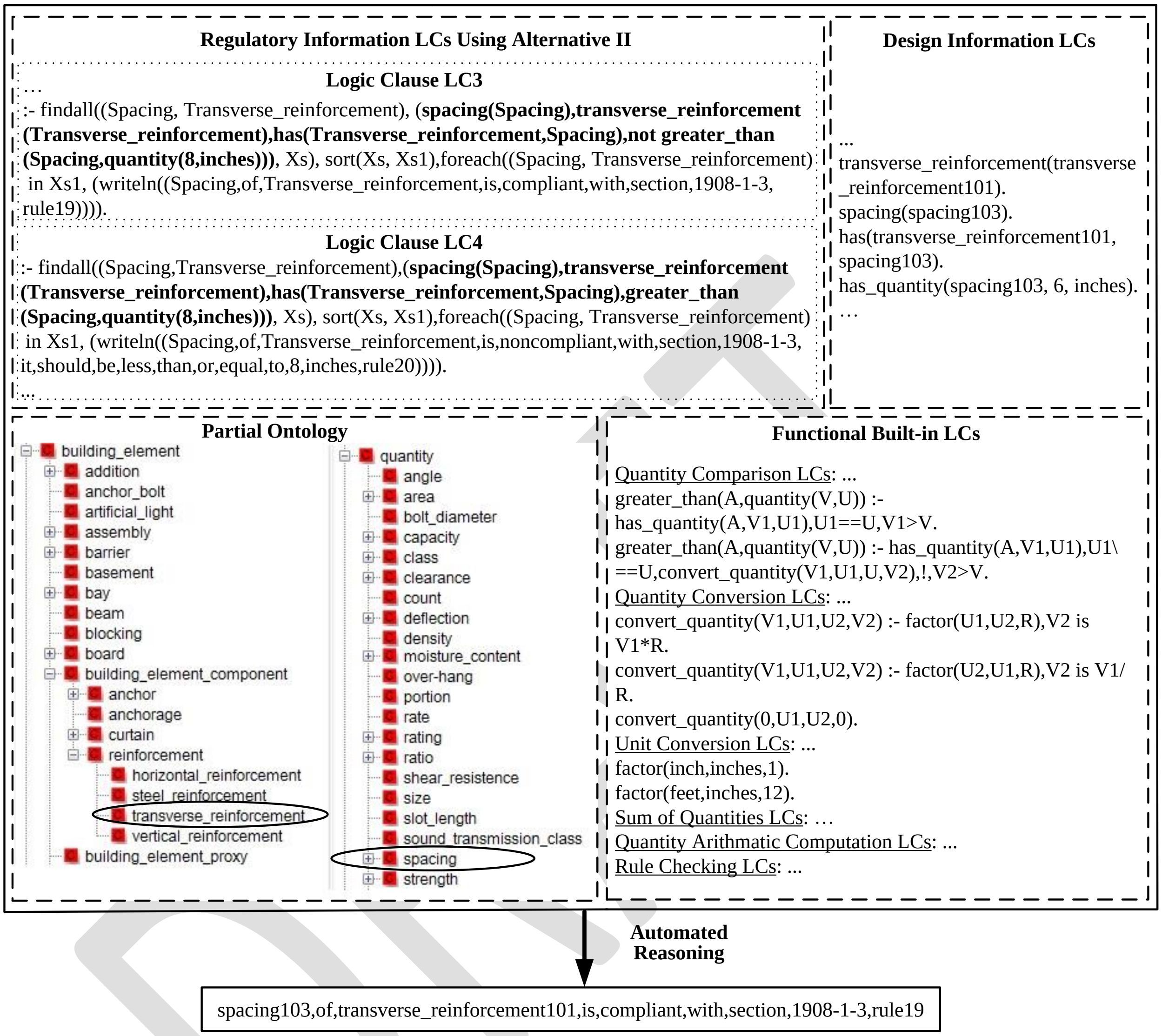
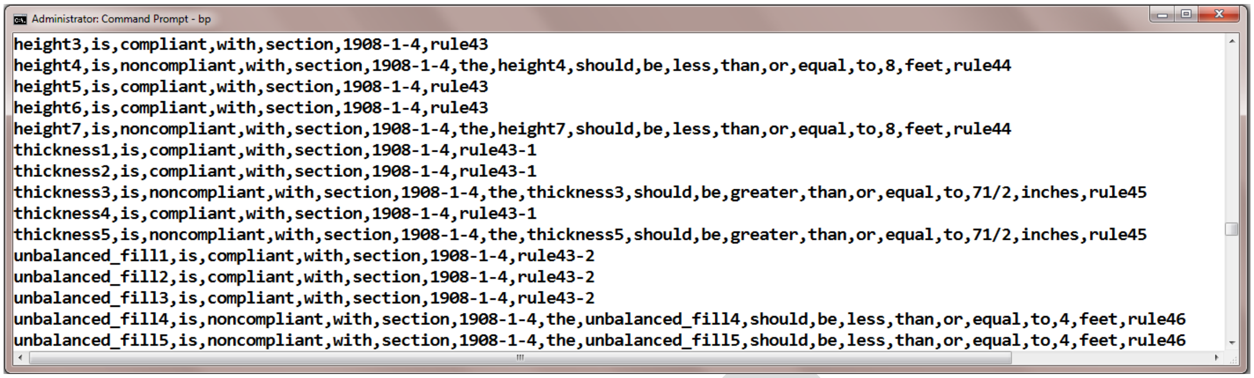
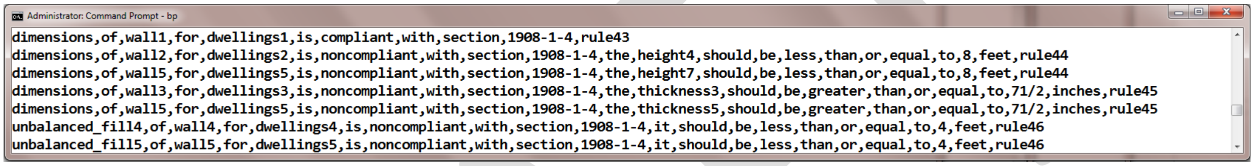


Fig. 7



```
height3,is,compliant,with,section,1908-1-4,rule43
height4,is,noncompliant,with,section,1908-1-4,the,height4,should,be,less,than,or,equal,to,8,feet,rule44
height5,is,compliant,with,section,1908-1-4,rule43
height6,is,compliant,with,section,1908-1-4,rule43
height7,is,noncompliant,with,section,1908-1-4,the,height7,should,be,less,than,or,equal,to,8,feet,rule44
thickness1,is,compliant,with,section,1908-1-4,rule43-1
thickness2,is,compliant,with,section,1908-1-4,rule43-1
thickness3,is,noncompliant,with,section,1908-1-4,the,thickness3,should,be,greater,than,or,equal,to,71/2,inches,rule45
thickness4,is,compliant,with,section,1908-1-4,rule43-1
thickness5,is,noncompliant,with,section,1908-1-4,the,thickness5,should,be,greater,than,or,equal,to,71/2,inches,rule45
unbalanced_fill1,is,compliant,with,section,1908-1-4,rule43-2
unbalanced_fill2,is,compliant,with,section,1908-1-4,rule43-2
unbalanced_fill3,is,compliant,with,section,1908-1-4,rule43-2
unbalanced_fill4,is,noncompliant,with,section,1908-1-4,the,unbalanced_fill4,should,be,less,than,or,equal,to,4,feet,rule46
unbalanced_fill5,is,noncompliant,with,section,1908-1-4,the,unbalanced_fill5,should,be,less,than,or,equal,to,4,feet,rule46
```

Fig. 8



```
dimensions,of,wall1,for,dwellings1,is,compliant,with,section,1908-1-4,rule43
dimensions,of,wall2,for,dwellings2,is,noncompliant,with,section,1908-1-4,the,height4,should,be,less,than,or,equal,to,8,feet,rule44
dimensions,of,wall5,for,dwellings5,is,noncompliant,with,section,1908-1-4,the,height7,should,be,less,than,or,equal,to,8,feet,rule44
dimensions,of,wall3,for,dwellings3,is,noncompliant,with,section,1908-1-4,the,thickness3,should,be,greater,than,or,equal,to,71/2,inches,rule45
dimensions,of,wall5,for,dwellings5,is,noncompliant,with,section,1908-1-4,the,thickness5,should,be,greater,than,or,equal,to,71/2,inches,rule45
unbalanced_fill4,of,wall4,for,dwellings4,is,noncompliant,with,section,1908-1-4,it,should,be,less,than,or,equal,to,4,feet,rule46
unbalanced_fill5,of,wall5,for,dwellings5,is,noncompliant,with,section,1908-1-4,it,should,be,less,than,or,equal,to,4,feet,rule46
```