buildings

MDPI

Article

A Generalized Adaptive Framework (GAF) for Automating Code Compliance Checking

Nawari O. Nawari

School of Architecture, College of Design, Construction, and Planning, University of Florida, Gainesville, FL 32611, USA; nnawari@ufl.edu

Received: 30 March 2019; Accepted: 13 April 2019; Published: 16 April 2019

Abstract: Building design review is the procedure of checking a design against codes and standard provisions to satisfy the accuracy of the design and identify non-compliances before construction begins. The current approaches for conducting the design review process in an automatic or semi-automatic manner are either based on proprietary, domain-specific or hard-coded rule-based mechanisms. These methods may be effective in their specific applications, but they have the downsides of being costly to maintain, inflexible to modify, and lack a generalized framework of rules and regulations modeling that can adapt to various engineering design realms, and thus don't support a neutral data standard. They are often referred to as 'Black Box' or 'Gray Box' approaches. This research offers a new comprehensive framework that reduces the limitations of the cited methods. Building regulations, for instance, are legal documents transcribed and approved by professionals to be interpreted and applied by people. They are hardly as precise as formal logic. Engineers, architects, and contractors can read those technical documents and transform them into scientific notations and software applications. They can extract any data they need, reason about it, and apply it at various phases of the project. How these extraction and use are carried out is a critical component of automating the design review process. The chief goal is to address this issue by developing a Generalized Adaptive Framework (GAF) for a neutral data standard (Industry Foundation Classes (IFC)) that enables automating the code compliance checking processes to achieve design efficiency and cost-effectiveness. The objectives of this study comprise i) to develop a theoretical background to an adaptive framework that supports a neutral data standard for transforming the written code regulations and rules into a computable model, and ii) to define the various modules required for computerizing of the code compliance verification process.

Keywords: BIM; IFC; automated code compliance checking; generalized adaptive framework

1. Introduction

Design review is the process of evaluating a design against its requirements to verify the quality and performance of the design and identify issues before construction takes place. In the Architecture, Engineering, and Construction (AEC) industry, this process is referred to as the Code Compliance Checking (CCC) process. The main goals of this process are to ensure quality, cost-effectiveness and prevent failure of the designed system. Presently AEC industries are becoming increasingly sophisticated, thus, the issue of identifying design defects and shortcomings before assembly and implementation is becoming even more problematic. This is true almost for all engineering design domains, from urban design, aerospace, mechanical, to building and construction engineering. The CCC Process is the primary method used to address this problem by critically assessing all aspects of the design through careful examination of requirements compliance, verification, and analyses.

The design review process is normally conducted at each phase of the design, from the conceptual to the final stage of construction documents. These series of reviews or CCC processes take a

considerable amount of time and effort for both designers and building authorities. For building officials, the CCC is even more critical since they are responsible for issuing building permits to start the construction process. Thus, there were some efforts cited in the literature that aimed to automate some of the CCC process [1–3]. Most of these efforts are either based on proprietary schemas, domain-specific or hard-coded rule-based representations, which may be successful in their implementations, but they have the shortcomings of being expensive to maintain, inflexible to modify, lack a generalized framework of rules and regulations modeling. They are often referred to as 'Black Box' or 'Gray Box' approaches.

Advancement of technology has continuously engaged the building design profession over the past several centuries. From the ancient Egyptian models in the form of drawings and physical objects as demonstrated in the plans of the Tomb of Rameses IV and the drawing of the shrine from Ghorâb, four thousand years ago, to the use of the mouse in the early 1970s, to the development of Building Information Modeling (BIM) technology in the mid-nineties, all indicate the radical impact of technology on building design and construction [4].

The last challenging global economic downturn in 2008 accompanied by the continued growth of complexity of building regulations and standards in a fragmented construction industry make designing and delivering of a facility that meets the owner's objective within budget and schedules into a cumbersome task. The notion of computerization within the context of facility design in the 21st century offers promising solutions in optimizing the building design process. By providing more accurate information in an open and asynchronous data format, computerization offers engineers, architects, and contractors efficient and innovative methods to collaborate, investigate many design alternatives, and validate design assumptions and requirements against code specifications in a virtual environment before construction to achieve optimum design objectives.

The concept of automating the CCC process described in this paper focuses on building regulations compliance checking mechanisms that are defined by the relationship among various design and engineering information management systems and the Building Information Modeling (BIM) and how this computerization will assist in streamlining communication and dissemination of building design review information amongst breadth of stakeholders.

In the AEC industry, specifications and regulations typically take the form of written texts, tables, charts, and equations. These rules, in general, have lawful status. However, the cognitive and analytic ability of the human brain is dissimilar to anything implemented in a computer environment. Thus, the automation of this process poses a real challenge to the AEC industry [5]. For example, how can the interpretation of these rules into a computer interpretable format be performed, in a manner that the implementation can be validated as consistent with the written regulations? Quite often, the process counts on the computer programmer's interpretation and translation of the written rules into computer code. In other cases, the logic of the human language statements is formally interpreted and then encoded into computer instructions.

Recently, new developments in Artificial Intelligence (AI) research and Building Information Modeling (BIM) could offer practical concepts to resolve some of the current major problems with automating CCC. Building standards and regulations commonly endeavor to organize, categorize, label, and define the rules, actions, and patterns of the built environment to attain safety against any failure, efficiency, and the overall economy. Nevertheless, their best-laid plans are overwhelmed by the inevitable change, growth, innovation, progress, evolution, diversity, and entropy [2]. Quite often regulations can amend provisions and interpretive standards, which generally leads to massive volumes of semi-structured documents that alter, complement and potentially conflict with one another. These issues, which indicate complications for both young designers and engineers as well as experienced professionals, are also far more disorderly for the fragile traditional knowledge bases in computer systems. Notwithstanding that precise definitions and specifications are essential for solving encoding design regulations, many building code provisions aren't precisely defined and are often characterized by high subjectivity. Furthermore, some code provisions are characterized by continuous

progressions and open-ended range of exceptions that make it difficult to give exact definitions for any concepts that are learned through experience.

2. A Brief Review of Recent Researches

This issue of automating rules and regulations checking has interested many researchers and practitioners over the years. The first successful effort to automate design compliance is demonstrated by the work of Fenv [6], when he investigated the application of decision tables to represent the AISC (American Institute of Steel Construction) standard specifications. He remarked that decision tables, If-Then-novel programming, and the program documentation technique, could be used to represent design standard provisions in a precise and unambiguous form. The concept was set to practical application in the 1969 AISC Specification. It was expressed as a set of interrelated decision tables. Later, other researchers tried to build on Fenv work such as Lopez et al., who implemented the SICAD (Standards Interface for Computer Aided Design system [7,8]). The SICAD system was a software prototype developed to demonstrate the checking of designed components as described in application program databases for conformance with design standards. The SICAD concepts were in production use in the AASHTO Bridge Design System [9]. Garrett developed the Standards Processing Expert (SPEX) system [10] using a standard-independent approach for sizing and proportioning structural member cross-sections. The system reasoned with the model of a design standard, represented using SICAD system representation, to generate a set of constraints on a set of essential data items that describe the attributes of a design to be determined.

In summary, over 400 relevant research studies are focusing on automating building codes compliances for design review, traversing more than 40 decades that can be identified. Some of the key recent investigations are summarized in Table 1. Most of these suggested methods are backbox or gray-box approaches. Some of these studies are generally associated with a specific domain, such as spatial assessment, structural integrity, safety, energy usage and so on. Some of them offer a certain degree of customization to modify the parameters of each rule to match specific local regulations. Once the rule structure has been encoded, it is available for multiple similar projects. In general, these systems can be classified into three main types of platforms for automated code compliance checking systems:

- As a software application integrated with a specific design tool, such as a plug-in. It is accessible to verify current model during the design process;
- As a stand-alone software application detached from the modeling tools. An example of this platform would be Solibri Model Checker (SMC), which has its rule engine that can work on multiple models;
- As a web-based application which can be available to verify designs from various sources.

Table 1. Summary of recent studies related to automating design rules compliance checking.

Reference	Method Description	Limitations
[11–13]	Domain-specific approaches for automatic review of building models.	Domain-specific. No support for the open standard. No methods to deal with ambiguous information.
[14,15]	Proposed Deontic conceptualization and logic for representing regulations along with Natural Language Processing (NLP).	No support for the open standard. No methods to deal with ambiguous information.
[16]	AI approach based on NLP techniques for Turkish fire egress codes.	Only applicable for fire egress code. Does not support IFC data schema.
[1,17–24]	Logic rules for expression shared ontologies for semantic representation of building regulations. They proposed the integration and validation of logical rules, RDF and OWL (Web Ontology Language) concepts, N'Logic rules.	Limited expressiveness to cover various parts of the building specifications. No methods to deal with ambiguous information.
[25,26]	Semantic rules for logical expression, Semantic Web Rule Language (SWRL). Feature extraction from building energy analysis sim, integration, validation, mapping data exchanges using SWRL.	No methods to deal with ambiguous information. Domain-specific
[27–29]	Used Dynamo plug-in for Autodesk Revit, Visual Programming Languages (VPL) for Automation.	No methods to deal with ambiguous information. Requires programming knowledge. Platform-specific
[30,31]	Semantic rules for logical expression for Evacuation regulation checking according to Korean codes, CORENET.	No generalized approach. No support for open standard. No methods to deal with ambiguous information.
[28,32–34]	AI approach based on NLP techniques, semantic NLP-based information extraction from construction regulatory documents through machine learning and text processing.	Back-box approach. No support for open standard. No methods to deal with ambiguous information.

As evident from the literature review, most of the proposed systems for automated code compliance auditing are based on proprietary or hard-coded rule-based representations, which may be successful in their domain implementations, but they have many drawbacks related to cost, maintenance and flexibility when used in automating building design review process. This project seeks to develop a new framework that addresses the shortcomings of the existing approaches by providing a computable model with explicit syntax and semantics that can be used to represent and reason about building regulations and provisions based on the neutral data format. Furthermore, the framework offers an object-based representation of building regulations, defines the minimum amount of data required to enable optimum Automatic CCC (ACCC) process.

3. Statement of Purpose

Currently, the manual design review process is time-consuming, error prone, and becoming very costly to sustain. The reasons behind these issues include: (a) increase rate of updates of regulations and standards with new knowledge and research outcomes; (b) new, state of the art technologies, equipment and devices; (c) higher amount of data and its multidisciplinary nature; and (d) increase in hardware/software communications complexity. Moreover, in building design, other issues associated with the manual CCC are lack of consistency in interpretation of regulatory provisions, the ability to properly self-check required aspects before bidding, and the long time needed for approvals of construction permits by building authorities that can have adverse financial impacts on projects.

The cited methods for automated rules compliance auditing in building design are either based on proprietary frameworks, domain-specific areas, or hard-coded rule-based representations. These approaches may be useful in their specific implementations. Nonetheless, they have the disadvantages of being costly to maintain, difficult to change, and the absence of a generalized framework of rules and regulations modeling that can adapt to various domains, and thus don't support an open neutral

standard. Furthermore, the current methods lack the means to deal with subjective and ambiguous building regulations. Most of these systems have not endured the test of industry applications.

4. Goals and Objectives

This research proposes a new approach that addresses the shortcomings of the cited methods. The primary goal is the development of a Generalized Adaptive Framework (GAF) that enables ACCC. The objectives of this research comprise: (1) the development of the theoretical background of a framework that is adaptive to the target domain and supports an open standard for transmuting the written code provisions into computable representations; (2) generating algorithms for the data exchanges between the components of the framework to execute the virtual review process of a building design in order to achieve design accuracy and cost-effectiveness.

5. Methodology: The ACCC Model

Figure 1 depicts the components and stages of the framework development process. In particular, the framework centers on the following levels of development for the ACCC process:

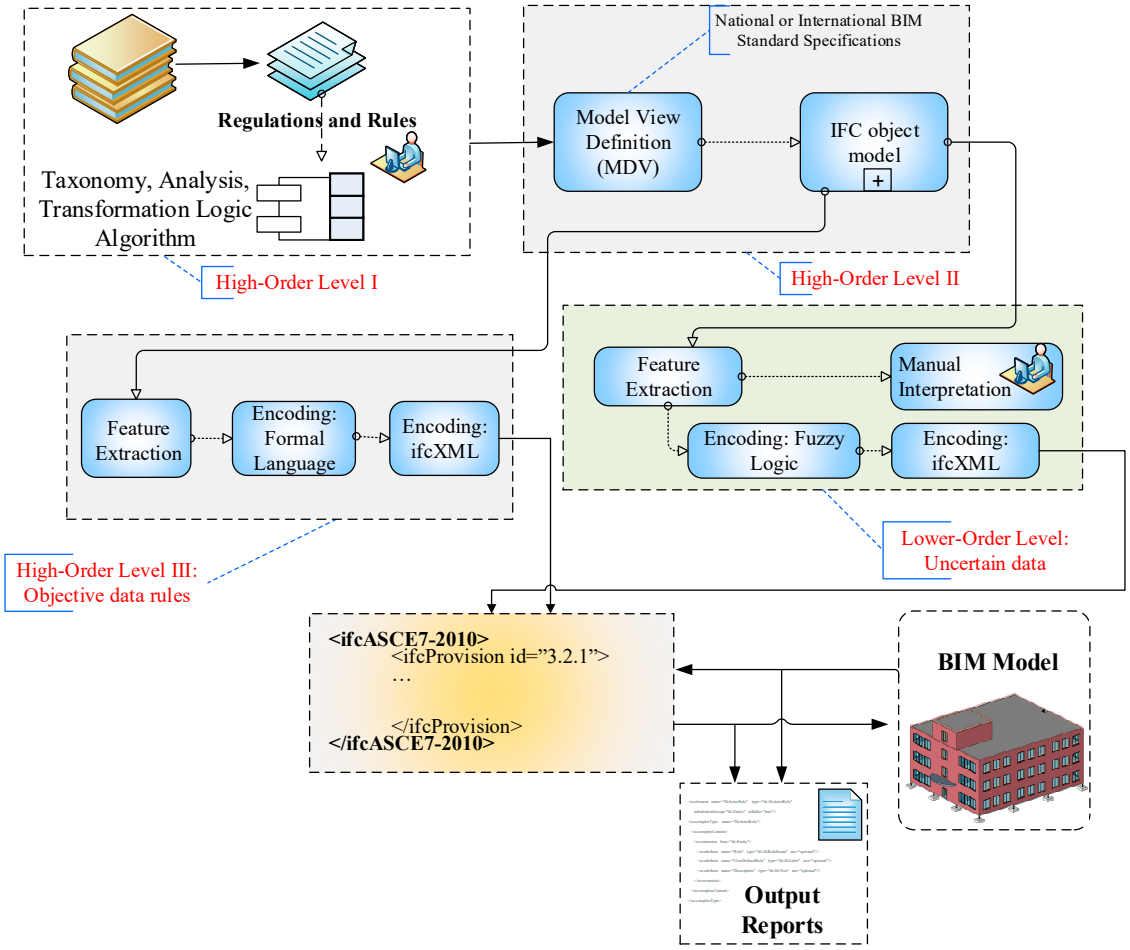


Figure 1. The Generalized Adaptive Framework for Automated Code Compliance Checking.

(A) High-Order Level I: Taxonomy formation, knowledge conceptualization, modification, integration, and decomposition of the design regulations and rules. This includes data analysis, partitioning and classification of regulatory text into broad categories. This phase is referred to as Transformation Reasoning Algorithm (TRA) and delineated in Figures 2 and 3.

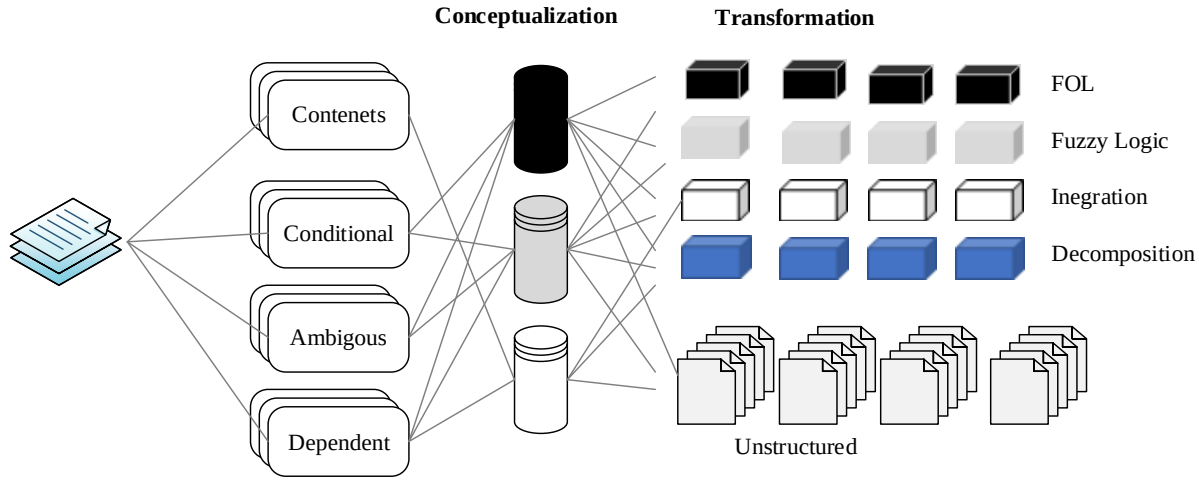


Figure 2. Overview of the Transformation Reasoning Algorithm (TRA).

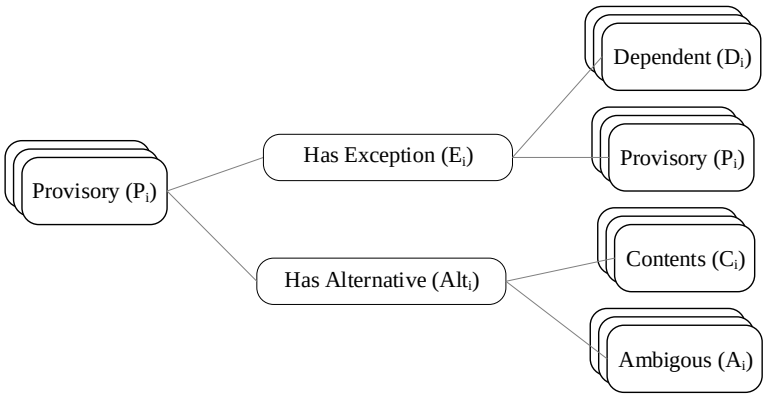


Figure 3. An example of the conceptualization of building regulation provision.

(B) High-Order Level II: Requires the Development Model View Definition (MVD), leading to Industry Foundation Classes (IFC) schema of the information obtained from phase A. The final data format is ifcXML representation. IfcXML is defined as the XML equivalent to the EXPRESS based specification of the IFC data model.

(C) Higher-order level III: The development of feature extraction algorithm for all objective data (unambiguous data) leading to full translation into the object-based model. This extraction and transformation will lead to ifcXML data object model (see Figure 1).

(D) Lower-order level: Necessitates feature extraction of uncertain data, then employing an algorithm for partial translation using fuzzy logic and approximate reasoning methods. Fuzzy logic provides a means of expressing linguistic rules in such a form that they can be combined into a coherent reasoning model. Such a model consists of three main parts: (i) fuzzification, (ii) inference engine (fuzzy rule base), and (iii) defuzzification (the process of transforming the aggregation result into a crisp output). The resulting data model from this phase is projected to be in ifcXML.

(E) The execution phase, which carries out the communications between the different layers of development. This encompasses the design of algorithms linking the data from (B), (C), (D) and the BIM model. These algorithms will be based on Language-integrated Query (LINQ) programming objects to extract, access and link BIM and regulations data via ifcXML [2]. Also, this phase will produce various output reports such as 2D, 3D views showing objects that are in noncompliance along with the detailed information about the regulation.

The theoretical development of the proposed GAF is based on the framework depicted in Figure 1. The objectives of the framework involve determining the requirements for an interpretation process where the semantic structure of each rule is translated into object expressions or parametric models

using the Transformation Reasoning Algorithm (TRA) that would lead to neutral data format such as the Industry Foundation Classes (IFC) data schema. Moreover, the framework outlines the development of algorithms to link these representations and relate to the BIM data being examined.

6. Transformation Reasoning Algorithm (TRA)

The TRA introduces the taxonomy for the building regulations knowledge followed by the conceptualization process. Subsequently, knowledge created will be transformed to a new formalized form to deduce various facts to carry out automated reasoning. For example, regulation or provision X_i will be transformed into a concept Y_i using TRA principles [3]. TRA taxonomy describes the following major concepts: Content (C_i); Provisory (P_i); Dependent (D_i); Ambiguous (A_i) Exceptions (E_i); and Alternatives (Alt_i).

Contents ($C_i \ldots C_n$) are the sections of the building codes and regulations that cannot be transformed into object rules. These clauses are usually devoted to definitions, such as the definition of types of loads, firewall, fire rate, smoke evacuation, high-rise building, etc. For instance, the live load is defined by the ASCE7-10 standard as: "A load produced by the use and occupancy of the building or other structure that does not include construction or environmental loads, such as wind load, snow load, rain load, earthquake load, flood load, or dead load".

Provisory ($P_i \ldots P_n$) are clauses of the regulations that can be transformed from the textual format into a set of object rules. Examples of such clauses are prevalent, and typical structures include rules with specific values such as those given in tables or equations in the building regulations.

Dependent ($D_i \ldots D_n$) clauses specify that one clause is reliant on one or more other provisions. This means that some requirements are only appropriate for a specific condition when other clauses are satisfied. These clauses generally contain provisory clauses ($P_i \ldots P_n$) and are often challenging to transform into sets of immediate object rules. These sections quite often may require manual checking for compliance. For instance, in Florida Building Code (FBC 2017), Section 503.1 regarding building height and area, states that "The building height and area shall not exceed the limits specified in Table 503 based on the type of construction as determined by Section 602 and the occupancies as determined by Section 302 except as modified hereafter. Each portion of a build separated by one or more firewalls complying with Section 706 shall be a separate building.".

Ambiguous ($A_i \ldots A_n$) clauses are the vague or inexact provisions that would need an expert judgment to be evaluated. They usually include words such as: approximately, about, relatively, close to, far from, maybe, etc. An example of such a provision is the footnote of the design lateral soil pressure for the clause given in ASCE 7-10: "For **relatively** rigid walls, as when braced by floors, the design lateral soil load shall be increased for sand and gravel type soils to 60 psf (9.43 kPa) per foot (meter) of depth. Basement walls extending not more than 8 ft (2.44 m) below grade and supporting **light** floor systems are not considered as being **relatively** rigid walls." This concept covers all regulations that are not capable of being computerized and some of them may have to be rewritten to enable implementation in an automated compliance auditing environment. Interpretation and rewriting both must adhere to understanding terms from both the legal and construction perspectives.

Exceptions ($E_i \ldots E_n$) are the clauses that a specific provision excludes. For example, in Florida Building Code 2017-Residential (FBC 2017-R), Section 305.1 (minimum room heights) apply for *habitable spaces* with three exceptions covering rooms with a sloped roof, the ceiling height above the bathroom, and obstructions in basements containing *habitable spaces*.

Alternatives ($Alt_i \ldots Alt_n$) are the alternative code provisions that may apply to a specific regulations. For instance, in the FBC 2017-R, the requirements in Section 301.1 (design criteria application) have the following alternative standards (subject to FBC limitations):

1. AF&PA Wood Frame Construction Manual (WFCM).
2. AISI Standard for Cold-Formed Steel Framing-Prescriptive Method for One- and Two-Family Dwellings (AISI S230).
3. ICC Standard on the Design and Construction of Log Structures (ICC 400).

that can be used.

These concepts can then be modified, combined or decomposed to enable a computable depiction of design regulations and standards. Knowledge concepts X_i can be transformed or integrated with another concept into Y_i, and then Y_i can be transmuted into Z_i to enable efficient computable representation. Thus, the TRA is defined as the conceptualization of knowledge representation by mapping design regulations into sets of object rules. Figures 2 and 3 are pictorial descriptions of the TRA for building design regulations.

Building design regulations will be classified using the taxonomy defined earlier and can also be translated into conceptual representations that closely approximate the meaning of the building code provision. These figurative structures can then be transformed and manipulated to deduce various facts and rules to carry out automated compliance validation. The TRA is partially driven from the first-order logic calculus. Table 2 depicts a summary of the syntax for the TRA.

Table 2. The syntax of the Transformation Reasoning Algorithm (TRA).

Symbol	Definition
::=	Is defined as
::∃	Has Exceptions
::◊	Has Alternatives
∧	Conjunction
∨	Disjunction
⊂	Subset of
¬	Negation
∀	Universal Quantifier
∃	Existential Quantifier
∈	Belongs to
→	Implication
↔	Biconditional
⇒	Transform into
::⇒	Depends upon
Constant	String starting with an uppercase letter
Variable	String starting with a lowercase letter
Pred (arg1, arg2, …)	Predicate
Fun (arg1, arg2, …)	Function
Pred1 (arg1, arg2, …) ∧ Pred2 (arg1, arg2, …) ∨ …	Rule

7. Example

The TRA can be exemplified further by considering the Florida Building Code 2017 – Residential (FBC 2017 - R). Figure 4 depicts a section from the FBC 2017-Residential. The provision shown in Figure 4 can be transformed into computable representations using the TRA as follows:

SECTION R305

MINIMUM ROOM AREAS

R305.1 Minimum height.

Habitable space, hallways and portions of *basements* containing these spaces shall have a ceiling height of not less than 7 feet (2134 mm). Bathrooms, toilet rooms and laundry rooms shall have a ceiling height of not less than 6 feet 8 inches (2032 mm).

Exceptions:

1. For rooms with sloped ceilings, the required floor area of the room shall have a ceiling height of not less than 5 feet (1524 mm) and not less than 50 percent of the required floor area shall have a ceiling height of not less than 7 feet (2134 mm).
2. The ceiling height above bathroom and toilet room fixtures shall be such that the fixture is capable of being used for its intended purpose. A shower or tub equipped with a showerhead shall have a ceiling height of not less than 6 feet 8 inches (2032 mm) above an area of not less than 30 inches (762 mm) by 30 inches (762 mm) at the showerhead.
3. Beams, girders, ducts or other obstructions in *basements* containing *habitable space* shall be permitted to project to within 6 feet 4 inches (1931 mm) of the finished floor.

R305.1.1 Basements.

Portions of *basements* that do not contain *habitable space* or hallways shall have a ceiling height of not less than 6 feet 8 inches (2032 mm).

Figure 4. Section R305 of Florida Building Code 2017—Residential.

Let REG_i = "Section R305"; Where i varies from 1 to n number of code provisions. Then we have

$$REG_i \in P_i \Rightarrow Y_i \Rightarrow X_i \tag{1}$$

where the subscript i stands for the counts of the code sections being processed and varies from 1 to n sections. P_i designates that this is a provisory clause, and describes the minimum room area (Y_i) which is given by X_i that expresses the various Rules describing Y_i:

$$X_i = \{R_1, R_2, \ldots R_m\} \tag{2}$$

where, $R_1, R_2, \ldots R_m$ are the rules defining X_i.

$$\text{Let } Z_{1j} = \{z_{11} \ldots z_{1q}\} \tag{3}$$

$$z = \text{IfcSpace}; z_{11} = \text{"R305.1"}; \tag{4}$$

$$z_{11} ::\exists\ SlopedCeiling(z) \wedge SpaceName(z, BATHROOM) \wedge ((AtBeamElevation(z) \vee AtDuctElevations(z))) \tag{5}$$

Equation (15) expresses the exceptions stated in the provision of section R305.1 of FBC 2017-R.

$$z_{12} ::= \geq 7 \text{ ft}\ ; z_{13} ::= \geq 6.667 \text{ ft}\ ; z_{14} ::= \geq 6.333 \text{ ft}\ ; z_{15} \geq 5 \text{ ft} \tag{6}$$

$$R_1: \forall z\ (REG_i\ (z) \rightarrow CeilingHeight(z, z_{12}) \wedge HabitableSpace(z) \wedge \neg\ SpaceName(z, BATHROOM) \wedge \neg\ SpaceName(z, TOILETROOM) \wedge \neg\ SpaceName(z, LAUNDRYROOM) \wedge \neg\ SlopedCeiling(z) \tag{7}$$

$$R_2: \forall z\ (REG_i\ (z) \rightarrow CeilingHeight(z, z_{13}) \wedge HabitableSpace(z) \wedge (SpaceName(z, BATHROOM) \vee SpaceName(z, TOILETROOM) \vee SpaceName(z, LAUNDRYROOM)) \wedge \neg\ SlopedCeiling(z) \tag{8}$$

$$A = FloorArea(z); z_{16} ::= \geq 0.5 * A \quad (9)$$

$$R_3: \forall z\ (REG_i\ (z) \rightarrow CeilingHeight(z, z_{15}) \wedge SlopedCeiling(z, z_{16}) \quad (10)$$

$$\text{Let } Z_{2j} = \{z_{21}, \ldots, z_{2q}\};\ z = \text{IfcSpace};\ z_{21} = \text{“R305.1.1”} \quad (11)$$

$$z_{23} ::= \geq 6.667 \text{ ft};\ z_{24} ::= \geq 6.333 \text{ ft};\ z_{25} = \text{IfcBeam};\ z_{26} = \text{IfcDuct}; \quad (12)$$

$$R_4: \forall z\ (REG_i\ (z) \rightarrow CeilingHeight(z, z_{23}) \wedge \neg\ (AtBeamElevation(z, z_{25}) \vee AtDuctElevations(z, z_{26})) \wedge \neg\ HabitableSpace(z) \quad (13)$$

$$R_5: \forall z\ (REG_i\ (z) \rightarrow CeilingHeight(z, z_{24}) \wedge (AtBeamElevation(z, z_{25}) \vee AtDuctElevations(z, z_{26}) \quad (14)$$

$$X_i = \{R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5\} \quad (15)$$

Equation (15) expresses section R305 of FBC 2017 - R using TRA.

An example of imprecise building code provisions can be found in section R322.1 of FBC 2017-Residential (see Figure 5). In this provision, the regulations states: “Buildings and structures constructed in whole or in part in flood hazard areas, including A or V Zones and Coastal A Zones, as established in Table R301.2(1), and substantial improvement and restoration of substantial damage of buildings and structures in flood hazard areas, shall be designed and constructed in accordance with the provisions contained in this section.” The word substantial is not defined precisely. Using the TRA, then we have

$$\text{REG}_2 = \text{“Section R322”; then we have: } REG_2 \in (C_2 \wedge A_2) \Rightarrow Y_2 \Rightarrow X_2 \quad (16)$$

SECTION R322

FLOOD-RESISTANT CONSTRUCTION

R322.1General.

Buildings and structures constructed in whole or in part in flood hazard areas, including A or V Zones and Coastal A Zones, as established in Table R301.2(1), and substantial improvement and restoration of substantial damage of buildings and structures in flood hazard areas, shall be designed and constructed in accordance with the provisions contained in this section. Buildings and structures that are located in more than one flood hazard area shall comply with the provisions associated with the most restrictive flood hazard area. Buildings and structures located in whole or in part in identified floodways shall be designed and constructed in accordance with ASCE 24.

R322.1.1Alternative provisions.

As an alternative to the requirements in Section R322, ASCE 24 is permitted subject to the limitations of this code and the limitations therein.

Figure 5. Part of Florida Building Code 2017—Residential 2017 (FBC 2017-R).

$(C_2 \wedge A_2)$ designates that this is a content clause with ambiguous statements describing flood resistance construction (Y_2) which is given by X_2 that describes the several conditions unfolding Y_2.

$$X_2 = \{R_1, R_2, \ldots\ R_m\} \quad (17)$$

where, $R_1, R_2, \ldots\ R_m$ are the rules defining X_2.

$$\text{Let } Z_{2j} = \{z_{21}, \ldots, z_{2q}\};\ z = \text{IfcBuilding};\ z_{21} = \text{“FBC 2017 - R322”};\ z_{22} = \text{“ASCE 24”} \quad (18)$$

$$REG_2 ::\Diamond z_{22} \tag{19}$$

$$R_1\text{: } \forall z\ (InFloodZone(z) \rightarrow RequiredProvision\ (z, z_{21})) \tag{20}$$

$$R_2\text{: } \forall z\ (InFloodWays(z) \rightarrow RequiredProvision\ (z, z_{22})) \tag{21}$$

Next step is to conceptualize the term "substantial damage." The TRA proposes fuzzy logic and predicates to transmute the concept into a rule expression. A fuzzy set is described by reference [35] as: A is a fuzzy subset of the universe of discourse U, is characterized by a membership function μ_A: $U \rightarrow [0 \ldots 1]$ which associates with each element u of U a number $\mu_A\ (u)$ in the interval [0,1]. This definition can be employed to define a fuzzy predicate. The truth-value of any proposition can be evaluated as the degree of membership of the responding fuzzy relation. Thus, a fuzzy predicate can be considered as the membership function of a fuzzy relation over individual variables' universe of discourse. Each fuzzy predicate characterizes a concept in the TRA. For instance, the building damage stated in section R322 can be represented as a fuzzy variable taking values delineated in Figure 6. In this figure, the x-axis is the degree of building damage (u), while the y-axis represents the membership functioning values $\mu_A\ (u)$. These variables comprise small damage, medium damage, and substantial damage. These descriptions can be determined from experience or given by local guidelines. Now, let z_{23} = a fuzzy variable defined by

$$\left.\begin{array}{lll} u_A\ (u) & = 0 & 80\% <= \ u >= 0 \\ & = (1/15)\ u\ - 25/15 & 80\% <\ u >= 90\% \\ & = 1 & u >\ 90\% \end{array}\right\} \tag{22}$$

where, $0 \leq \mu_A(u) \leq 1$.

Next, section R322 of FBC 2017-Residential is transformed into the following rule:

$$R_3\text{: } \forall z\ (InFloodZone(z) \wedge Damage\ (z, z_{23}) \rightarrow RequiredProvision\ (z, z_{21})) \tag{23}$$

and finally,

$$X_2 = \{R_1 \wedge (R_3 \vee R_2)\} \tag{24}$$

Equation (24) signifies section R322 of FBC 2017 - R using the TRA.

Engineering design codes do rather frequently use such vague terms to describe certain conditions. Table 3 recapitulates some of these terms and their transformation using a fuzzy predicate. For example, the fuzzy term "very little damage" in Table 3 is represented graphically by Figure 6a. This figure describes building damages in the range of 0% to 20%. The membership function has a maximum value of 1.0 for the degree of damages from 0% to 10% and then decreases linearly to 0 for a degree of damage of 20%.

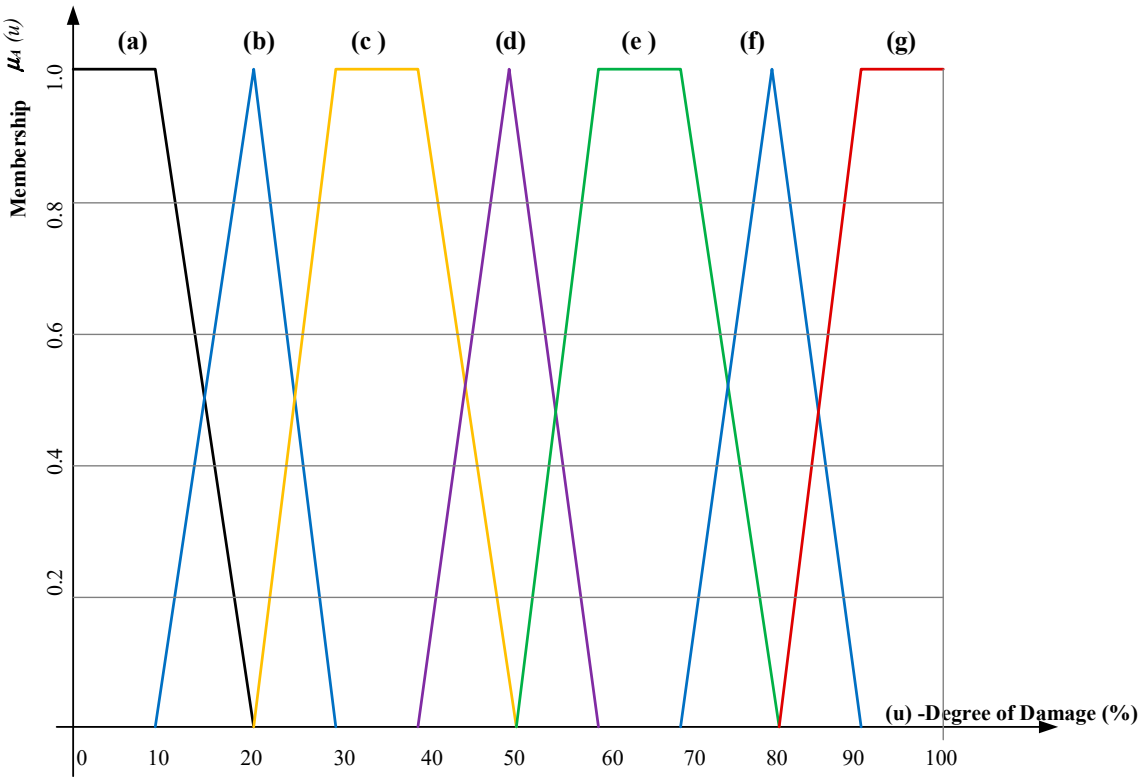


Figure 6. The concept of fuzzy transformation of building damages. (**a**) Very little damage; (**b**) Small damage; (**c**) Some damage; (**d**) Moderate damage; (**e**) Large damage; (**f**) Substantial damage; (**g**) Extreme damage.

Table 3. Common ambiguous terms in building regulations.

No	Uncertain Building Code Terms	Conceptualization
1	The building has Some damage	Fuzzy predicate, $0 \leq \mu_A(u) \leq 1$; Figure 5c
2	The building has a good amount of damage	Fuzzy predicate, $0 \leq \mu_A(u) \leq 1$; Figure 5d
3	Building damage is extreme	Fuzzy predicate, $0 \leq \mu_A(u) \leq 1$; Figure 5g
4	A substantial amount or a sizable amount	Fuzzy predicate, $0 \leq \mu_A(u) \leq 1$; Figure 5f
5	A fair amount or Moderate amount	Fuzzy predicate, $0 \leq \mu_A(u) \leq 1$; Figure 5d
6	Large value	Fuzzy predicate, $0 \leq \mu_A(u) \leq 1$; Figure 5e
7	Small amount	Fuzzy predicate, $0 \leq \mu_A(u) \leq 1$; Figure 5b
8	Very little or a little bit	Fuzzy predicate, $0 \leq \mu_A(u) \leq 1$; Figure 5a

8. Implementation Algorithms

This phase addresses designing and implanting data communicating algorithms for exchanging and presenting the results of the proposed GAF framework. It utilizes Language-integrated Query (LINQ) programming objects to extract, access and link BIM and regulations data via ifcXML [2]. Figure 7 delineates a part of the proposed algorithms. The implementing of the IFC standard using XML (eXtensible Markup Language) technologies is known as ifcXML. It is an extension of the existing IFC data format. It focuses on the specifics of the ifcXML specification compared to the standard EXPRESS based IFC object data model. The ifcXML data files are given the extension ".xml" or alternatively ".ifcxml". Figure 8 depicts the process of generating an ifcXML data file.

```
1.  XElement CodeCheck = XElement.Load("C:\BIM\Books\IfcXMLFile1.xml");
2.  int NoOfStoreies = 0;

3.  decimal tHeight= 0.0; decimal totalgrossArea=0.0; totalGrossArea =0.0; totalNetArea =0.0;
5.  public struct SpaceProperties {

6.          public int StoreyID, SpaceID;

7.          public decimal        NetFloorArea, GrossFloorArea; Height,Volume; }

8. public struct StoreyProperties {

9.          public int StoreyID;

10.         public decimal        netFloorArea, grossFloorArea; Height; }

11.  SpaceProperties spaceProp;  StoreyProperties levelProp;
12.  var bsID = new List<int>(); var spaceID = new List<int>();
13.  var sProp = new List<SpaceProperties>(); var storeyProp = new List< StoreyProperties>();
14.  foreach (XElement x in CodeCheck.Elements("IfcBuildingStorey") {

15.     int NoOfStoreies +=1;

16.     tHeight += (decimal) x.Element("Elevation");

17.     bsID.Add((int) x.Attribute("id")); }
```

Figure 7. A part of the implementation code using C# programing language.

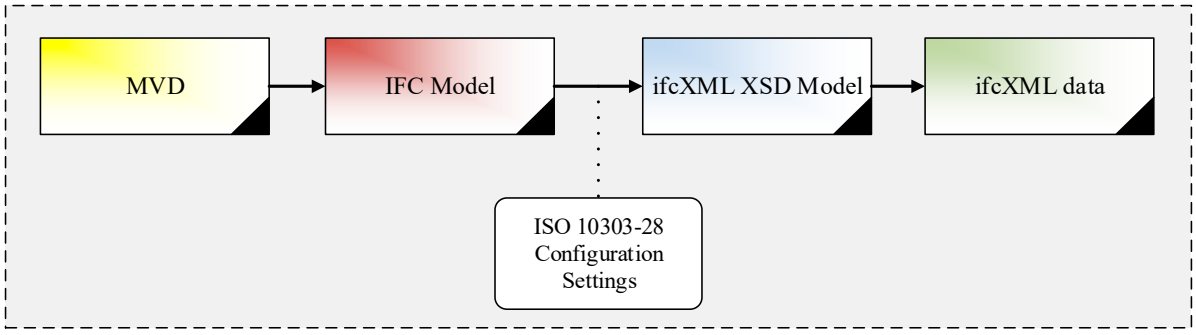


Figure 8. The general process of creating an ifcXML data file.

To provide evidence for the Proof of Concept (POC), a two-story building is considered in a typical design review process examining regulations and provisions from FBC-R 2017. The building is the duplex apartment model that is provided by the BuildingSmart Alliance website and delineated in Figure 9. Some of the Ifc objects that will be involved in this example are illustrated in Figure 10.

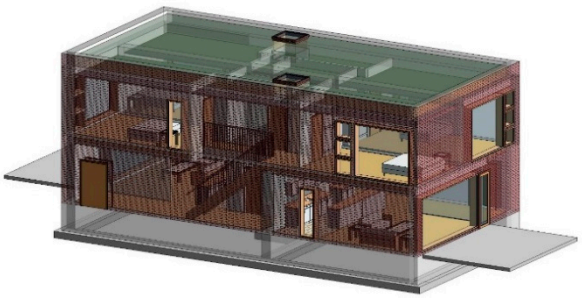


Figure 9. Building model used in the example (buildingSMART International).

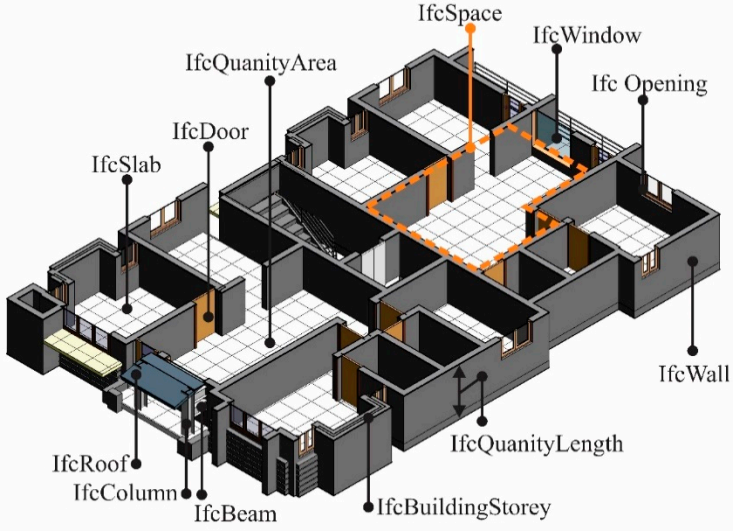


Figure 10. Main Ifc objects referenced in the example.

In this construction, the ifcSpace functions as the primary object and is essential to configure the spatial structure of a building. The spatial structure elements are connected by using the object relationship ifcRelAggregates. Moreover, it offers additional functions such as serving as the spatial container for space-related elements. Space is normally linked to a building storey (or in case of exterior spaces to a site). Space can cover several connected spaces. Thus, a space group provides a collection of spaces included in a building story. Space may also be fragmented into elements, where each element defines a partial space. This is defined by the composition type attribute of the supertype ifcSpatialStructureElement.

Figure 7 displays a part of the C# code of the proposed GAF. Line 1 in Figure 6 reads the data from the ifcXML file of the BIM model of the building shown in Figure 9. Lines 8 to 17 determine the floor and areas and the height of each space in the building. Line 19 in Figure 11 reads the code provisions from the ifcXML data file of the FBC 2017. In Figure 11, line 22 to line 31 deals with examining the compliance of the areas of the spaces as well as the net and gross floor areas according to the FBC-2017.

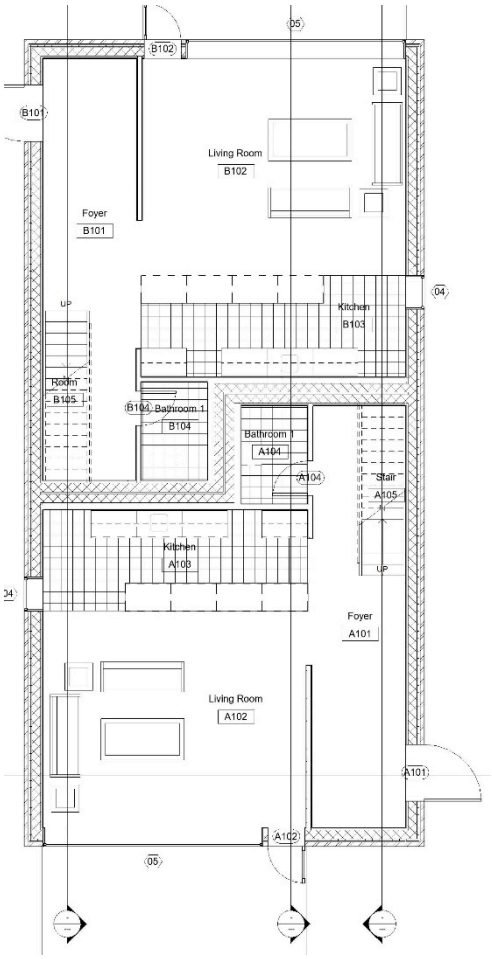
```
18.  string VerifigyArea, VerifyHeight, VerifyNoOfStories;
19.  XElement bCode = XElement.Load("C:\BIM\Books\IfcXMLbCode.xml");
20.  from node in bCode.Elements("IfcSectionNumber")
21.  where (string) node.Attribute("SectionNumber") == "503"
22.  select node;
23.  String sectionName = (string) node.Element("SectionTitle")
24.  String Clause = (string) node.Element("Clause")
25.  foreach (var story in storeyProp) {
26.    int id = (int) storey.StoreyID;
27.      decimial gArea = (decimal) story.StoreyID;
28.    decimial nArea = (decimal) stoey.StoreyID;
29.    if CheckArea (id, gA, ConstType, OccuType) {
```

Figure 11. A part of the implementation algorithm.

Figure 12a delineates the results of checking the areas of the spaces in the example building shown in Figure 9 using the GAF. Figure 12b illustrates the floor plan view of level 1 of the duplex apartment

building (see Figure 9) being used for this example of automated compliance verification of Florida's residential building regulations.

Level	Number	Name	Area	Compliance Check
Level 1	A101	Foyer	193.064 SF	PASS
Level 1	A102	Living Room	324.442 SF	PASS
Level 1	A103	Kitchen	149.591 SF	PASS
Level 1	A104	Bathroom 1	43.031 SF	PASS
Level 1	A105	Stair	52.982 SF	N/A
Level 2	A201	Hallway	83.958 SF	PASS
Level 2	A202	Bedroom 1	281.146 SF	PASS
Level 2	A203	Bedroom 2	281.778 SF	PASS
Level 2	A204	Bathroom 2	58.295 SF	PASS
Level 2	A205	Utility	18.880 SF	N/A
Level 1	B101	Foyer	193.064 SF	PASS
Level 1	B102	Living Room	324.442 SF	PASS
Level 1	B103	Kitchen	149.591 SF	PASS
Level 1	B104	Bathroom 1	43.031 SF	PASS
Level 1	B105	Room	52.982 SF	PASS
Level 2	B201	Hallway	83.958 SF	PASS
Level 2	B202	Bedroom 1	281.146 SF	PASS
Level 2	B203	Bedroom 2	281.778 SF	PASS
Level 2	B204	Bathroom 2	58.572 SF	PASS
Level 2	B205	Utility	18.604 SF	N/A
Roof	R301	Roof	1568.535 SF	N/A



(**a**) (**b**)

Figure 12. Results example of checking compliance with the space areas regulations of FBC 2017-R. (**a**) ACCC results; (**b**) Floor plan of level 1 of the case study model.

9. Conclusions

The computerization of the code compliance checking process presents a challenge for the AEC industry. This is greatly attributed to the fact that many sections of the building rules take the form of written texts. The present approaches for computerization or semi- computerization of rules compliance verification used in a design review are either based on proprietary, domain-specific or hard-coded rule expressions, which can be successful in their specific applications. Nonetheless, most of these methods are expensive to maintain, inflexible to change, the absence of a comprehensive framework of regulations modeling that can adapt to various domains, and thus they lack IFC standard support.

This paper offers a generalized adaptive framework (GAF) for automating or semi-automating the code compliance verification process which is based on an object-driven representation of building rules that can deal with certain and uncertain data and transform code and standards regulations into computable expressions using the Transformation Reasoning Algorithm (TRA). The framework is flexible and can adapt to various engineering design disciplines.

The paper introduces and delineates the various constituents of the proposed GAF and their relationships. The TRA introduces the taxonomy for the building regulations knowledge along with the conceptualization and transformation processes. Subsequently, knowledge created is a new formalized object representation that models objective and ambiguous building regulations and can deduce various facts to carry out automated reasoning. This approach minimizes the shortcomings of the cited methods by transforming objective and vague data of building code into a concise formal representation that can be mapped into IFC data schema.

The GAF provides communicating algorithms for exchanging data between the main components of the framework. It utilizes Language-integrated Query (LINQ) programming objects to extract, access and link BIM and regulations data via ifcXML. The application of the proposed GAF has been demonstrated using an example of a two-story building that is considered in a typical design review process, examining regulations and provisions from FBC-R 2017. Thus, it is expected that the research results will have broader impacts that include the enormous benefits to the AEC industry due to the consistency of the interpretation of regulatory provisions, the ability to self-check required aspects before bidding, the time and resources saved during design review, the optimum design, the quicker turnaround in feedback, and faster approvals for construction permits by building authorities.

Funding: This research was supported by the College of Design, Construction, and Planning (DCP) at the University of Florida. The author would like to extend his sincere gratitude to DCP for providing the seed fund for this research project.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Eastman, C.M.; Lee, J.; Jeong, Y.; Lee, J. Review Automatic rule-based checking of building designs. *J. Autom. Constr.* **2009**, *18*, 1011–1033. [CrossRef]
2. Nawari, N. Automating Codes Conformance. *J. Arch. Eng.* **2012**, *18*, 315–323. [CrossRef]
3. Nawari, N.O. A Generalized Adaptive Framework for Automating Design Review Process: Technical Principles. In Proceedings of the 35th CIB W78 Conference, Chicago, IL, USA, 1–3 October 2018; pp. 405–414.
4. Nawari, N.O.; Kuenstle, M. *Building Information Modeling: Framework for Structural Design*; CRC Press: Boca Raton, FL, USA, 2015.
5. Nawari, N.O.; Adel, A. Practical Approaches for Computable Building Codes. In Proceedings of the 32nd CIB W78 Conference, Eindhoven, The Netherlands, 27–29 October 2015; pp. 569–576.
6. Fenves, S.J. Tabular decision logic for structural design. *J. Struct. Eng.* **1966**, *92*, 473–490.
7. Lopez, L.A.; Wright, R.N. *Mapping Principles for the Standards Interface for Computer—Aided Design*; National Bureau of Standards: Gaithersburg, MD, USA, 1985; Volume NBSIR 85-3115.
8. Lopez, L.A.; Elam, S.; Reed, K. Software concept for checking engineering designs for conformance with codes and standards. *Eng. Comput.* **1989**, *5*, 63–78. [CrossRef]
9. AASHTO. *AASHTO Guide for Design of Pavement Structures*, 4th ed.; American Association of State Highway and Transportation Officials: Washington, DC, USA, 1998.
10. Garrett, J.H.; Fenves, S.J. A knowledge-based standards processor for structural component design. *Eng. Comput.* **1987**, *2*, 219–223. [CrossRef]
11. Lee, J.K. Building Environment Rule and Analysis (BERA) Language And Its Application for Evaluating Building Circulation and Spatial Program. Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2011.
12. Lee, J.K.; Eastman, C.M.; Lee, Y.C. Implementation of a BIM Domain-specific Language for the Building Environment Rule and Analysis. *J. Intell. Robot Syst.* **2015**, *79*, 507–522. [CrossRef]
13. Sherif, A.; Jinkook, L.; Chuck, E. Automated Cost Analysis of Concept Design BIM Models. In Proceedings of the 14th International conference on Computer Aided Architectural Design, Liege, Belgium, 4–8 July 2011; pp. 403–418.
14. Salama, D.M.; El-Gohary, N.M. Semantic modeling for automated compliance checking. *J. Comput. Civ. Eng.* **2011**, 641–648. [CrossRef]

15. Zhang, J.; El-Gohary, N. Automated Information Extraction from Construction-Related Regulatory Documents for Automated Compliance Checking. In Proceedings of the 28th International Conference of CIB W78, Sophia Antipolis, France, 25–28 October 2011.
16. Ozgun, B.; Kilimci, E.S.Y.; Çağdaş, G. Automated Code Compliance Checking Model for Fire Egress Codes. *Digit. Appl. Constr.* **2012**, *2*, 1–10.
17. Nguyen, T.H.; Kim, J.L. Building code compliance checking using BIM technology. In Proceedings of the 2011 Winter Simulation Conference (WSC), Phoenix, AZ, USA, 11–14 December 2011.
18. Pauwels, P.; Van Deursen, D.; Verstraeten, R.; De Roo, J.; De Meyer, R.; Van de Walle, R.; Van Campenhout, J. A semantic rule checking environment for building performance checking. *Autom. Constr.* **2011**, *20*, 506–518. [CrossRef]
19. Solihin, W.; Eastman, C.; Lee, Y.C. Toward robust and quantifiable automated IFC quality validation. *Adv. Eng. Inform.* **2015**, *29*, 739–756. [CrossRef]
20. Venugopal, M.; Eastman, C.M.; Teizer, J. An ontology-based analysis of the industry foundation class schema for building information model exchanges. *Adv. Eng. Inform.* **2015**, *29*, 940–957. [CrossRef]
21. Malsane, S.; Matthews, J.; Lockley, S.; Love, P.E.; Greenwood, D. Development of an object model for automated compliance checking. *Autom. Constr.* **2015**, *49*, 51–58. [CrossRef]
22. Niknam, M.; Karshenas, S. Integrating distributed sources of information for construction cost estimating using Semantic Web and Semantic Web Service technologies. *Autom. Constr.* **2015**, *57*, 222–238. [CrossRef]
23. Hjelseth, E.; Nisbet, N. Exploring semantic based model checking. In Proceedings of the 2010 27th CIB W78 International Conference, Cairo, Egypt, 16–19 November 2010.
24. İlala, S.M.; Günaydın, H.M. Computer representation of building codes for automated compliance checking. *Autom. Constr.* **2017**, *82*, 43–58. [CrossRef]
25. Cheng, J.C.P.; Das, M. A BIM-based web service framework for green building energy simulation and code checking. *J. Inf. Technol. Constr.* **2014**, *19*, 150–168.
26. Kim, H.; Shen, Z.; Kim, I.; Kim, K.; Stumpf, A.; Yu, J. BIM IFC information mapping to building energy analysis (BEA) model with manually extended material information. *Autom. Constr.* **2016**, *68*, 183–193. [CrossRef]
27. Preidel, C.; Borrmann, A. Towards code compliance checking on the basis of a visual programming language. *J. Inf. Technol. Constr.* **2016**, *21*, 402–421.
28. Patlakas, P.; Livingstone, A.; Hairstans, R.; Neighbour, G. Automatic code compliance with multi-dimensional data fitting in a BIM context. *Adv. Eng. Inform.* **2018**, *38*, 216–231. [CrossRef]
29. Nour, M. Using Bounding Volumes for BIM based electronic code checking for Buildings in Egypt. *Am. J. Eng. Res.* **2016**, *5*, 91–98.
30. Lee, H.; Lee, J.-K.; Park, S.; Kim, I. Translating building legislation into a computer-executable format for evaluating building permit requirements. *Autom. Constr.* **2016**, *71*, 49–61. [CrossRef]
31. Zhang, J.; El-Gohary, N. Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking. *J. Comput. Civ. Eng.* **2013**, *30*, 1943–5487. [CrossRef]
32. Zhang, J.; El-Gohary, N. Automated information transformation for automated regulatory compliance checking in construction. *J. Comput. Civ. Eng.* **2015**, *29*, b4015001. [CrossRef]
33. Zhang, J.; El-Gohary, N.M. Extending building information models semi automatically using semantic natural language processing techniques. *J. Comput. Civ. Eng.* **2016**, *30*, C4016004. [CrossRef]
34. Lu, Q.; Lee, S.; Chen, L. Image-driven fuzzy-based system to construct as-is IFC BIM objects. *Autom. Constr.* **2018**, *92*, 68–87. [CrossRef]
35. Zadeh, L.A. Fuzzy Sets. *Inf. Control* **1965**, *8*, 338–353. [CrossRef]