Recommended Citation: Xue, X., Wu, J., and Zhang, J. (2022). "Semi-Automated Generation of Logic Rules for Tabular Information in Building Codes to Support Automated Code Compliance Checking." Journal of Computing in Civil Engineering, 36(1), 04021033.

For Final Published Version, please refer to the ASCE Database here:
https://ascelibrary.org/doi/10.1061/(ASCE)CP.1943-5487.0001000

Semi-Automated Generation of Logic Rules for Tabular Information in Building Codes to Support Automated Code Compliance Checking

Xiaorui Xue, S.M.ASCE [1]; Jin Wu, S.M.ASCE [2]; Jiansong Zhang, Ph.D., A.M.ASCE [3] *

Abstract

To fully automate building code compliance checking, regulatory requirements need to be automatically extracted and transformed from building codes. Existing regulatory requirement processing efforts mainly focus on building code requirements in text. A more efficient approach for processing regulatory requirements in other parts of building codes, such as tables or charts, remains to be addressed. The ability to process building code requirements in all parts and formats is necessary for an automated code compliance checking system to achieve full coverage of checkable building code requirements. To address this gap, the authors proposed a semi-automated information extraction and transformation method. The proposed method can extract building code requirements in tables and convert extracted information to logic rules. Automated code compliance checking systems can utilize the logic rules. The proposed method includes two main steps: (1) tabular information extraction, and (2) rule generation. The tabular information

[1] Automation and Intelligent Construction (AutoIC) Lab, School of Construction Management Technology, Purdue University, West Lafayette, IN, 47907, PH (765) 430-2009. email: xue39@purdue.edu.
[2] Automation and Intelligent Construction (AutoIC) Lab, School of Construction Management Technology, Purdue University, West Lafayette, IN, 47907, PH (301) 275-1272; email: wu1275@purdue.edu.
[3] Automation and Intelligent Construction (AutoIC) Lab, School of Construction Management Technology, Purdue University, West Lafayette, IN, 47907, PH (765) 494-1574; FAX (765) 496-2246. (*corresponding author) email: zhan3062@purdue.edu.

extraction semi-automatically detects the layout of tables, extracts building code requirements from tables, and transforms extracted information to databases. The rule generation step automatically generates logic rules that can be directly executed by logic reasoners. The rule generation step also provides options for users to further refine the generated rules. The development of the tabular information extraction algorithm takes an iterative approach. An experiment was conducted to develop a tabular information extraction algorithm from Chapter 5 of the International Building Code 2015. The primary version of the algorithm correctly processed 91.67% of tables in Chapter 10 of the International Building Code 2015. After iterative refinements, the updated tabular information extraction algorithm correctly processed all tables in this chapter. The rule generation algorithm correctly generated logic rules that successfully represented the applicable building code requirements for a testing building, a convenience store in Texas based on the tabular information.

Introduction

Building Information Modeling (BIM) developed steadily in the past decade (Noor and Yi 2018). Versatile usages of BIM prompted researches of BIM in the Architecture, Engineering, and Construction (AEC) industry, for various purposes such as: (1) modular construction of residential buildings (Alwisy et al. 2012), (2) construction cost estimation (Lee et al. 2014), (3) collaboration of multi-disciplinary teams (Fernando et al. 2013), and (4) energy simulation in a building's early design stage (Kim et al. 2015). Researchers also combined BIM with other technologies, such as (1) laser scanning (Kim et al. 2015), (2) Radio-frequency Identification (RFID) (Fang et al. 2016), (3) Virtual Reality (VR) (Du et al. 2018), (4) computer vision (Asadi et al. 2019, Ham et al. 2016, Han et al. 2015), and (5) natural language processing (Mutis et al. 2019, Xu and Cai 2020, Xu and Cai 2019, Xu et al. 2020, Zhang and El-Gohary 2012a,b) to explore its full potential in the AEC industry. Overall, benefits of BIM include: (1) facilitating cooperation among stakeholders (Volk et al. 2014), (2) reducing construction cost and time (Bryde et al. 2013), (3) improving construction safety (Zhang et al. 2013), and (4) decreasing carbon emission and energy consumption of buildings (Gan et al. 2019). Although BIM has a wide range of benefits and BIM-related research was extensive, the adoption and implementation of BIM in the industry are still at an early stage (Noor and Yi 2018). Overall, developed countries have a higher BIM adoption rate than developing countries (Bui et al. 2016). However, even developed countries do not fully adopt BIM. In the United Kingdom, for example, only 54.88% of projects used BIM in the design phase, 51.90% of projects used BIM in the pre-construction phase, and 34.67% of projects used BIM in the construction phase (Eadie et al. 2013). Project participants do not use BIM to its full extent, even when they use BIM in a project. They tend to use BIM to assist their traditional workflows and only in the most expensive and risky part of a project (Migilinskas et al. 2013). In summary, in spite of the diverse applications and benefits of BIM, the BIM adoption rate remains low. Therefore, more automation and user-friendliness in BIM-based applications is needed to facilitate the adoption of BIM.

One important such application of BIM is as digital representations of buildings in automated code compliance checking systems (Dimyadi and Amor 2013). Automated code compliance checking systems can work as a driver of BIM adoption (Martins and Abrantes 2010). On the other hand, BIM can support automated code compliance checking systems by providing digital representations of building designs (Ismail et al. 2017). Building authorities in multiple countries published automated code compliance checking systems based on BIM such as (1) the Construction and Real Estate NETwork (CORENET) in Singapore (Sing and Zhong 2001), the ByggSok system in Norway (Haraldsen et al. 2004), the DesignCheck in Australia (Ding et al. 2006), and the 3D-4D-BIM program by the US General Services Administration (GSA) (Ho and Matta 2009). In academia, researchers also proposed BIM-based automated code compliance checking systems. For example, Nawari (2011) applied the integration of BIM and SMARTcodes by the International Code Council to check the conformance of building elements to various structural codes automatically. Balaban et al. (2012) presented a pilot study that combines BIM and AI to check compliance of a building model against fire codes automatically. Martins and Monteiro (2013) developed a BIM application for the automated compliance checking of building codes in general. Preidel and Borrmann (2015) presented a flow-based BIM automated code compliance checking system. Zhang and El-Gohary (2017) integrated semantic Natural Language Processing (NLP), logic reasoning, and BIM to develop an automated code checking system for checking with the International Building Code. Bloch and Sacks (2018) explored the possibility of integrating machine learning, rule-based inferring, and BIM to accomplish automated code compliance checking. Fayomi et al. (2018) explored the approach of automating prescriptive compliance process for building energy efficiency in a single BIM software platform. Häußler et al. (2021) integrated BIM and visual programming in code compliance checking of railway designs. In the past decade, the automated code compliance checking domain developed at a fast pace. However, a limited range of checkable codes hinders the wide adoption of automated code compliance checking systems. Few construction industry practitioners will use automated code compliance checking systems that require them to check a significant part of building codes manually. After an extensive literature review, the authors found no previous automated code compliance checking research targeted at automatically processing building code requirements stored in tables. Some previous efforts, such as CORENET (Sing and Zhong 2001), DesignCheck (Ding et al. 2006), and Nawari (2011) which combined SMARTcodes by ICC and BIM, included tables in the range of checkable building code requirement by manually adding tables to the respective systems. However, almost all building codes contain a large amount of building code requirements in tables. Automated code compliance checking systems that do not cover building code requirements in tables cannot have full building code requirement coverage (Salama and El-Gohary 2016), and pure manual addition of tabular information into automated code compliance checking system counters efficiency and time/cost savings. Research about tabular information detections and extraction in other domains (Liu et al. 2007, Pinto et al. 2003) concurred with the importance of processing information that is stored in tables in documents. In this research, the authors proposed a new semi-automated information extraction and information transformation method for tables in building codes. The proposed method can process tabular information in building codes to increase the scope of checkable building codes of automated code compliance checking systems.

Background

Existing Work with Table Processing

The demand to extract information from documents that are not in plain textual formats, such as tables and images that are hard for machines to process, is urgent (Corrêa and Zander 2017). Most existing methods take a two-step approach to extract tabular information: (1) table detection, and (2) table sub-structure identification (i.e., cells, rows, columns) (Paliwal et al. 2019). Challenges in tabular information extraction include: (1) reliance on the context of tables to interpret tables, (2) document indexing, (3) database curation, and (4) abbreviation of phases (Shmanina et al. 2016). Table detection algorithms can construct table hierarchies in two approaches: top-down and bottom-up. In the top-down approach, the algorithm first identifies tables in documents and then slice identified tables into components. On the contrary, in the bottom-up approach, the algorithm first identifies components of tables and then assembles the components to tables (Krüpl and Herzog 2006). Different technologies were developed to process table information for various purposes. For example, Vasileiadis et al. (2017) developed a rule-based, bottom-up tabular information extraction system for access by visually impaired people. Buitelaar et al. (2006) published an ontology-based table processing method to extract information from webpages as part of a multi-modal dialog system. Shafait and Smith (2010) used Optical Character Recognition (OCR) technology to process tables with different layouts for analyzing tables in heterogeneous documents. Qasim et al. (2019) treated the table detection problem as a graph problem and generated a Graph Neural Network to detect the structure of tables, which was successfully tested on public table detection datasets (e.g., UW3, UNLV, and ICDAR 2013). Sinha et al. (2019) used OCR to localize tables in Piping and Instrumentation Diagrams (P&IDs) and used regular expressions to enhance the accuracy of text extraction. Although most researchers treated table detection and table structure identification as two separate steps, Paliwal et al. (2019) proposed TableNet, a neural network with an encoder-decoder structure, to detect table existence and identify table structure in one unified step jointly. Although existing table processing works reached high accuracy on their respective domains, they did not touch upon automation of processing tables in building codes, and data from such tables were still manually interpreted and processed.

Existing Work in Automated Building Code Compliance Checking

From a historical perspective, the traditional building code compliance checking process, or building plan review, is a laborious, time-consuming, and error-prone process that demands automation (Alghamdi et al. 2017, Lee et al. 2018, Preidel and Borrmann 2017). The automation of the code compliance checking process can significantly cut its cost, time, and manual efforts. In the manual code compliance checking process, designers need to wait a long time for building authorities to issue a building permit or ask for further modifications to the design documents, and may have to modify design documents multiple cycles. The plan review process may last a few months (City of San Clemente 2019). On the other hand, automated code compliance checking systems can return compliance checking results in a much shorter time with a limited need for manual input. Thus, automated building code compliance checking is faster and cheaper than the traditional manual code compliance checking approach.

The automated code compliance checking systems emerged in the 1960s when Fenves introduced decision tables to check the design of steel structures (Fenves 1966). Systems for checking different aspects of building design were then developed over the years. For example, Pauwels et al. (2011) implemented a sematic rule checking environment to check the acoustic performance of buildings. Tan et al. (2010) provided a series of decision tables to check the design of building envelopes. Getuli et al. (2017) developed a BIM-based workflow that checks against the compliance of Italian construction safety and health code. Malsane et al. (2015) suggested an Industry Foundation Class (IFC)-powered, object-oriented approach to check against fire codes in England and Wales. Bus et al. (2019) developed an ontology-based system to achieve automated compliance checking of semantic rules in French fire safety and accessibility codes. However, existing automated code compliance checking systems only check a limited set of code rules and, according to the authors' literature review, never automatically processed building code requirements in tables.

Database for Table

Storing and manipulating data has been an important topic since the earliest development of computers (Ramakrishnan 2003). Integrated Data Store, developed in the 1960s, was the first general-purpose Database Management System (DBMS), which standardized the basis of the network data model. Information Management System, on the other hand, was an alternative data representation framework and formed the basis of the hierarchical data model. The current dominant DBMSs utilize relational data models, which were proposed in the 1970s and consolidated in the1980s. Structured Query Language (SQL) is the standard query language for relational databases. DBMS has the following advantages: (1) data independence, (2) efficient data access, (3) data integrity, (4) data security, (5) efficient data administration, (6) concurrent access, (7) crash recovery, and (8) reduced application development time.

Relational database models store data in tables with unique names (Silberschatz et al. 1997). The rows in a table represent a relationship of a set of values. Relationships between values are represented by tuples, which are sequences of values. The relational data model allows operations like querying, inserting, deleting, and updating of tuples. The relational database model, such as SQL, lacks the ability to take certain actions such as taking input, displaying outputs, or communicating with other programs. The above actions must be done through a host language such as C, Java, or Python, with embedded SQL queries. In spite of the above limitations, SQL supports a number of useful syntactic features (e.g., relation representation, querying) and has clearly established itself as the standard relational database language (Silberschatz et al. 1997).

Logic Programming

Logic programming aims to connect logic to computer programming (Kowalski 2014). Comparing to the well-established systems of logic, computer programming barely covered the relationship between Turing machines (i.e., an abstract model that simulates the computational ability of computers) and relational algebra. Logic programming remedies the deficiency of computer programming by fitting the generality of logic into the context of computing. Through logic programming, the tasks of defining the problem and solving it can be separated (Spivey 1996). Prolog is a classic programming language that implements logic programming. It uses a top-down method and carries out symbolic reasoning with a logical formula (Kowalski 1979, Spivey 1996). One Prolog program consists of one or more logic clauses. One logic clause consists of one or more predicates. All predicates need to be true to make the logic clause true. For example, for a requirement that says "building height should be less than 40 feet," one possible Prolog representation in the form of a logic rule is "*compliance_building_height(Building):-building(Building), building_height(Building_height), has(Building, Building_height), Building_height < 40.*" The logic clause will be evaluated to true if all predicates to the right of ":-" (i.e., in the body of the rule) are true. The first three predicates (i.e., *"building(Building), building_height(Building_height), has(Building, Building_height),"*) check the existence of a building and its building height property. The last predicate (*"Building_height < 40"*) checks if the building height is less than 40 feet. The program can then be fed with building design information in the form of logic facts to check if the building exceeds the maximum allowed building height. If *building_X* has a building height of 50 feet, the corresponding building design facts are "*building(Building_X), building_height(Building_height), has(Building_X, Building_height), Building_height == 50.*" Instantiating the above logic rule with these logic facts will lead to the evaluation of the logic rule as false, indicating these building design facts violate the building height requirement.

Proposed Method

In this paper, the authors proposed a semi-automated table processing method for tables in building codes. The proposed method takes a two-step approach to process tabular information in building codes: (1) tabular information extraction, and (2) information conversion to logic rules. The proposed method takes building codes and digital models of buildings as inputs, and outputs applicable rules for the building models. The developed method needs to be robust over a wide range of tables, i.e., to be able to process tables in an unseen format. The tabular information extraction method needs to extract building code requirements from tables in building codes and store the extracted information in a structured format. The extraction process needs to reach a very high precision to meet the 100% recall goal of noncompliance detection in automated code compliance checking (Salama and El-Gohary 2016). The format to store extracted building code requirements needs to support easy information access and processing to ensure the performance of the automated code compliance checking system. Integrated methods that directly convert building code tables to logic rules and store these rules in automated code compliance checking system are the most straightforward and intuitive method to process building code requirements from tables. However, state-of-the-art integrated methods lack robustness in processing tables in different layouts and the manual effort to maintain integrated methods may not be less than the effort in the manual encoding of building codes per se. The diverse layouts of tables may require customized methods for each table. Frequent updates of building codes will therefore require constant method updates. To address that, the authors proposed the separation of information extraction from rule generation to increase the robustness and reduce the maintenance need of the method.

Information Extraction

The proposed method takes a semi-automated approach to extract tabular information from building codes. Users need to collect tables from building codes in digital format and provide them together with some structural information of these input tables. The method then processes one table at a time. Structural information of tables helps the information extraction method identify the layout of the table. For tables with different layouts, the underlying relationships between cells are different. For example, some tables use a single cell to store an entry of building code requirement, and some tables use an entire row to store an entry of building code requirement. Layouts of tables implicitly specify how tables store building code requirements. One type of table, for example, uses a cell and its corresponding row header and column header to represent one requirement to buildings. Another type of table uses all cells in a row and their corresponding column headers to represent one requirement to buildings. The proposed method uses structural information provided by the users to automatically distinguish layouts of tables and uncover underlying relationships and information inferred by layouts.

The authors took an iterative approach to develop the sub-algorithms in the tabular information extraction algorithm, i.e., the sub-algorithms are continuously improved until they can correctly extract all tabular information from training data. The basic unit of a table is the cell. Cells can be classified into four types: (1) row header, (2) column header, (3) footnote, and (4) content (Fig. 1). The four types of cells form the body of a table. The finished algorithm can recognize the cell type and connect the information in each cell (e.g., texts, numbers). As a result, the authors developed: (1) a header detection sub-algorithm to recognize the boundaries of cells for each type, (2) a table layout detection sub-algorithm to distinguish layouts of tables, (3) two information transformation sub-algorithms to connect contents in the cells, and (4) a rule generation sub-algorithm.

The header detection sub-algorithm uses the structural information of the table to detect information components. The algorithm requires three inputs from the user for locations of row headers, column headers, and footnotes, respectively. Users then provide: (1) the number of columns used for row headers *X1*, (2) the number of rows used for column headers *X2*, and (3) the number of columns used for footnotes *X3*. There may be no footnotes (i.e., zero for *X3*) or row headers (i.e., zero for *X1*). The header detection sub-algorithm can then automatically identify the locations of different contents and split the table into different information components according to inputs from the user.

After that, the layout detection sub-algorithm distinguishes the layouts of the tables based on their structural information. Tables in building codes have diverse layouts. The authors identified two master layouts based on how the information is organized in a table. Tables with row headers are considered to be in Master Layout One: a single cell is used to store an entry of building code requirement (Fig. 2). Tables without row headers are considered to be in Master Layout Two: a row of cells is used to store an entry of building code requirement (Fig. 3). Master layouts ensure the robustness of this algorithm and simplify the information extraction process. The layout detection sub-algorithm can classify all tables in building codes into these two master layouts depending on whether a table has a row header or not. The authors kept the algorithm simple to ensure the robustness of the entire table information processing.

The end product of this step is a database that stores information from the table. The information conversion sub-algorithm connects information in different components of a table and inserts connected information into the database. Each master layout has a customized information conversion sub-algorithm. Customized information conversion sub-algorithm ensures the correct extraction of information inferred by the layout of tables. Tables in the same master layout use the same information conversion sub-algorithm. For tables in the same master layout, variations exist, such as having or not having a column for footnotes, having or not having a different number of rows in the column header. The information transformation sub-algorithms are sufficiently robust to process such variations of tables in the same master layout.

The sub-algorithm for the Master Layout One, which is for tables that use a single cell to store an entry of building code requirement, connects the cell, its corresponding row header and column header, and its corresponding footnote (if exists) together and generates a command to insert the entry of building code requirement into the database. The sub-algorithm for the Master Layout Two, which is for tables that use an entire row to store an entry of building code requirement, connects each cell in the row with its corresponding column header and generates a command to insert the entry of building code requirement into the database. Once a command is generated, both sub-algorithms execute the command to insert building code requirements into the database.

Information Conversion to Logic Rules

The conversion to logic rules follows a semi-automated approach. The conversion process has three parts: raw-rule generation, invariant signatures generation, and information matching.

Raw-rule Generation

The raw-rule generation utilizes existing semantic NLP-based algorithms that convert building code requirements to Prolog logic rules with placeholders for information from tables (Zhang and El-Gohary 2013, 2015; 2017). The logic rules are generated from building code provisions with reference to tables for depicting certain required range of values. These existing algorithms lack the ability to process tabular information of building code requirements. Therefore, this step generates logic rules with placeholders for building code requirements in tables. At this step, the information extraction and transformation algorithm by Zhang and El-Gohary (2013; 2015; 2017) was directly adopted, and placeholders were generated in the logic rules for the part dealing with tabular information.

Invariant Signatures Generation

To allow the rules to be compiled with correct quantities and units for the placeholders, the rule generation sub-algorithm uses invariant signature (Wu et al. 2021; Wu and Zhang 2019) to decide the configuration of buildings, e.g., occupancy of the building. Each model is processed into invariant signatures, which can represent all the building elements in a uniform way and keep all the needed information for further processing. The invariant signatures include geometrical, locational, and metadata information of the building. Table 1 shows an example of state-of-the-art invariant signature features for structural analysis. In this step, the invariant signatures will be expanded to support the compliance checking use case by adding model-level invariant signatures. Invariant signature allows the proposed algorithm to select the best matching building code requirements from a table for a building.

Information Matching

The tabular data usually stores the regulatory requirements of multiple possible types of buildings. On the other hand, one building usually has a fixed type (e.g., occupation) that corresponds to one entry of the tabular information. The final rule generation sub-algorithm can process the digital model of a building, find the corresponding type of the building in the table by querying the databases to get the building code requirements for the building.

For example, the occupancy of a building will determine the maximum allowable number of stories of the building (ICC 2015).

With the raw rules with placeholders and the invariant signatures, the rule generation sub-algorithm generates final logic rules by an information matching sub-algorithm. The information matching sub-algorithm will process the invariant signatures to select the corresponding content to fill in placeholders and generate semantically correct B-Prolog (a Prolog system implementation with extensions for programming concurrency, constraints, and interactive graphics) rules (Zhou 2014). For example, the information matching algorithm can process the invariant signature to obtain the occupancy of the building and query the corresponding maximum allowable stories from the database. The query process is based on the invariant signatures. After querying, the rule generation sub-algorithm fills the returned values into placeholders of raw logic rules. For example, the maximum allowable stories of the building from the corresponding type of occupancy will be placed in the logic rule. After that, users have the option to manually compile generated rules again to further increase their semantic simplicity if needed.

Experiment

Algorithm Development

The header detection sub-algorithm, the layout detection sub-algorithm, and two information conversion sub-algorithms were developed based on tables (Table 2) in Chapter 5 of IBC 2015 and were tested on tables in Chapter 10 (Table 3) of IBC 2015.

Inputs of the developed algorithms were digital tables. Digital tables left less space for errors comparing to tables collected as scanned images. The authors manually inspected the extraction results by the algorithm to examine their performance.

After that, the information conversion sub-algorithm injected the extracted information into databases. The authors used the SQLite database in the implementation of information conversion sub-algorithms (SQLite Consortium 2020). Each table was stored in a separate database. Two information conversion sub-algorithms were developed for the two master layouts. The layout detection sub-algorithm selects which information conversion algorithm to use. The information conversion algorithm generates an SQLite insertion command based on the syntax of SQLite and the layout of the table being processed.

Information Conversion to Logic Rule

To test the performance of the rule generation sub-algorithm, the authors evaluated it on an IFC model of a real convenience store as the sample building model. The model was using the IFC2x3 standard, which was still the most widely used IFC data standard, to ensure the generality of validation. Fig. 4 shows a visualization of the model.

The authors examined the proposed rule generation sub-algorithm on the rules in Table 504.3 and Table 504.4 of IBC 2015, which were applicable to the validation case. While Table 506.2 was also applicable, the authors excluded it because it involved the identification and calculation of equations, which were out of the scope of this research. The selected rules are shown in Fig. 5.

To store model-level information, the authors developed invariant signatures to cover Occupancy, Construction Type, and Sprinklered categories. The Occupancy category contains all possible occupancy types of a building, such as Group H, Group I, etc. The Construction Type category contains all possible construction types of a building, such as Type I, Type II. The Sprinklered category depicts whether the building is sprinkled, using a Boolean variable.

The authors then processed the model of the convenience store into 58 invariant signatures, which contains 1,363 pieces of information that can be converted into logic facts. Among the 58 invariant signatures, 55 covered element level information about building components such as wall, slab, roof, window, and door. The remaining three invariant signatures covered model-level information for occupancy, construction type, and sprinkler information.

In the final step, the authors developed an information matching sub-algorithm to allow the rules to be filled with the correct content in their placeholders, based on the invariant signatures of the convenience store. The information matching sub-algorithm checked each place holder using the invariant signatures and selected the correct header and footer content to modify the raw logic rule into final logic rules with the proper information that are needed for checking the compliance of the building.

Results

The testing results are presented in Table 4. The results showed that the proposed method provided the correct results on eleven testing tables and failed in one. Correctly processed tables are the tables that are correctly converted to databases by the proposed method. The results can be verified manually using queries on the database. The failed table was Table 1006.2.1 (Fig. 6). Therefore, the proposed method processed 91.67% of the tables in the testing dataset correctly. The reason that the proposed method failed to provide correct results in Table 1006.2.1 was that this table had four levels of column headers. No table in Chapter 5 of 2015 IBC (i.e., training data) had more than two levels of column headers. The authors then updated the developed algorithm to accommodate tables with different levels of column headers. The updated algorithm was then tested on all testing tables again. The updated algorithm provided correct results on all tables.

The following experiment was further conducted to test if the information extraction sub-algorithm correctly preserved the information inferred by the layout of tables and correctly extracted building code requirements in the cells. The accuracy of the algorithm was tested by checking if the generated database returns the correct results when queried. Correct results were where the corresponding value of a building code requirement in tables can be successfully returned by the query. For example, when the database for Table 1006.2.1 is queried for the maximum occupant load of space of occupancy Type B, it should return 49. The authors queried every entry in the generated databases for every table in the testing dataset and reviewed the returned values of every query. In 100% of cases, the query returned correct results. The generated database preserves all information inferred by the layout of the tables. Another reason for the 100% accuracy is the authors used digital tables, instead of scanned tables, as inputs to the information extraction sub-algorithm. Errors in recognizing the content of scanned tables were therefore prevented. For example, the algorithm did not suffer from errors in OCR.

Furthermore, the rule generation sub-algorithm was tested to generate logic rules with correct parameters based on the invariant signatures of the building model of a real-world convenience store model. The rule generation sub-algorithm was tested to generate logic rules for checking the compliance of maximum building height and number of stories of the convenience store. The correct logic rules should represent the meaning of corresponding building code provisions correctly. The raw logic rules with placeholders were checked manually to ensure that the meanings of corresponding building code provisions were correct. Correct final logic rules should have correct building code requirement values (from tables) filled in by the information matching algorithm. The algorithm correctly generated the required final rules for checking the compliance of the convenience store with building code requirements in Table 504.3 and Table 504.4 of IBC 2015. The generated rules checked and concluded that the convenience store does not violate the required values in Table 504.3 and Table 504.4 of IBC 2015. The detailed testing of the rule generation algorithm is described as follows:

The first step was to apply the state-of-the-art information extraction and transformation algorithms by Zhang and El-Gohary (2013; 2015; 2017) to generate the raw logic rules with placeholders for building code requirements stored in tables that vary with configurations of buildings. An example result was shown in Fig. 7.

Invariant signatures of the convenience store were generated automatically by state-of-the-art invariant signature generation algorithm (Wu et al. 2021; Wu and Zhang 2019). There were eleven door elements, two slab elements, one roof elements, twenty-seven wall elements, and fourteen window elements in the building, as shown in Table 5. Table 5 also showed a few example values of the invariant signature features. Each element is represented uniquely by an invariant signature, which preserves the needed information that will be used for generating the logic rules. In addition to the element-level invariant signatures, the authors also developed model-level invariant signatures. Table 6 showed the building had Occupancy M, Construction Type V-B, and Sprinklered None based on model-level invariant signatures. This information was converted into configurations of the convenience store.

Based on the configurations of the convenience store, the developed algorithm was able to find the correct numbers from the table and generate the correct final logic rules. The resulted final logic rules are shown in Fig. 8.

The authors manually compiled generated rules again to make them concise and straightforward. For example, the predicate "not exceed_the_limits" was refined to "<="

because B-Prolog supports the symbol of mathematical inequalities. The finalized rules were improved to use mathematical inequalities to be more concise and machine-readable while staying reader friendly (Fig. 9). This step is optional. Logic rules in Fig. 8 and Fig. 9 are equivalent to logic reasoners.

Contributions to The Body of Knowledge

The authors proposed a new method to extend the range of checkable building code requirements of automated building code compliance checking systems to cover tables in building codes. The contributions to the body of knowledge are four-fold. First, the extension of checkable building code requirements to tables proves the feasibility of checking non-textual building code requirements in a semi-automated way. Second, this research could help incorporate more building code requirement details into fully automated code compliance checking systems in a more efficient way, comparing to the state of the art. With an enlarged range of checkable building code requirements, an automated code compliance checking system can provide more value to its users, which could lead to a wider adoption of automated building code compliance checking and synergistically facilitating the adoption of BIM. Third, the authors enhanced the robustness of an automated code compliance checking system. By storing database and generating logic rules on the go, automated code compliance checking systems will benefit from a smaller rule set which has better maintainability comparing to a larger one. Last but not least, the authors calculated that 1,542 logic rules can be generated from tables in the training and test datasets, sourced from 17 tables in two chapters of IBC 2015, which has 35 chapters in total. After interpolation, the authors estimated that the proposed method can help complete about 26,985 new rules with tabular information for IBC 2015. The proposed method can therefore significantly expand the range of checkable building code requirements of ACC systems.

Interface Discussion

Although not the focus of this paper, the proposed method provides a friendly programming interface to support an easy adoption. The information extraction component only needs users to provide a digital table and some structural information of the digital table. To use the information extraction component, users only need to put the digital table and the script of the proposed method in the same folder. When the script is executed, it will prompt users to input information about the table. The output of the information extraction component is database files that are supported in SQL language. The database allows developers to incorporate the tabular information into automated code compliance checking systems more efficiently comparing to pure manual interpretation and processing.

Limitations and Future Work

The following limitations are acknowledged. First, the proposed method requires digital tables as inputs and manual conversion or third-party software to process tables from hard copy or images into digital tables. Future versions of the proposed method should incorporate the processing of scanned tables, e.g., using OCR functions. Second, the proposed method requires manual inputs in layout detection. The proposed method cannot detect layouts of tables without such inputs from users in spite of the fact that such inputs are minimal. The authors propose to develop a fully automated layout detection algorithm for tables in building codes in their future work. Third, the rule generation sub-algorithm requires manual effort to refine generated rules. Although logic reasoners have no problem in processing unrefined rules, the authors propose to develop refinement algorithms to generate more human-processable rules also automatically in their future work.

Conclusion

This research incorporated tabular information in building codes into automated code compliance checking systems. The proposed table information extraction method achieved a 91.67% success rate in our experiment on tables from IBC 2015. The updated information extraction algorithms could successfully process all tables in the testing dataset and correctly preserved information inferred by the layout of tables. The proposed method still requires minor human input, which the authors will further address in their future work.

Data Availability Statement

Some data that support the findings of this study are available from the corresponding author upon reasonable request.

1. Building Codes Tables Used
2. Logic Rules Generated

Acknowledgements

The authors would like to thank Dr. Mark J. Clayton for providing the testing model. The authors would like to thank the National Science Foundation (NSF). This material is based on work supported by the NSF under Grant No. 1827733. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF.

References

Alghamdi, A., Sulaiman, M., Alghamdi, A., Alhosan, M., Mastali, M., and Zhang, J. (2017). "Building accessibility code compliance verification using game simulations in virtual reality." *Proc., Computing in Civil Engineering 2017,* Seattle, WA, 262-270.

Alwisy, A., Al-Hussein, M., and Al-Jibouri, S. (2012). "BIM approach for automated drafting and design for modular construction manufacturing." *Proc., Computing in civil engineering 2012*, Clearwater Beach, FL,221-228.

Asadi, P., Gindy, M., and Alvarez, M. (2019). "A Machine Learning Based Approach for Automatic Rebar Detection and Quantification of Deterioration in Concrete Bridge Deck Ground Penetrating Radar B-scan Images." *KSCE Journal of Civil Engineering*, 23(6), 2618-2627.

Balaban, Ö., Kilimci, E. S. Y., and Cagdas, G. (2012). "Automated code compliance checking model for fire egress codes." *Proc., The 30th International Conference on Education and research in Computer Aided Architectural Design in Europe*, Education and research in Computer Aided Architectural Design in Europe and Faculty of Architecture, Prague, Czech, 117-125.

Bloch, T., and Sacks, R. (2018). "Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models." *Automation in Construction*, 91, 256-272.

Bryde, D., Broquetas, M., and Volm, J. M. (2013). "The project benefits of building information modelling (BIM)." *International Journal of Project Management*, 31(7), 971-980.

Bui, N., Merschbrock, C., and Munkvold, B. E. (2016). "A review of Building Information Modelling for construction in developing countries." *Procedia Engineering*, 164, 487-494.

Buitelaar, P., Cimiano, P., Racioppa, S., and Siegel, M. (2006). "Ontology-based information extraction with soba." *Proc., the International Conference on Language Resources and Evaluation (LREC),* Genoa, Italy , 2321-2324.

Bus, N., Roxin, A., Picinbono, G., and Fahad, M. (2019). "Towards French Smart Building Code: Compliance Checking Based on Semantic Rules." *Proc., LDAC2018 6th Linked Data in Architecture and Construction Workshop*, London, UK, CEUR Workshop Proceedings, 6-15.

City of San Clemente (2019). "Ordinance No. 1668."San Clement, California.

Corrêa, A. S., and Zander, P.-O. (2017). "Unleashing Tabular Content to Open Data: A Survey on PDF Table Extraction Methods and Tools." *Proc., the 18th Annual International Conference on Digital Government Research*, Association for Computing Machinery, Staten Island, NY, 54–63.

Dimyadi, J., and Amor, R. (2013) "Automated Building Code Compliance Checking - Where is it at?" *Proc., 19th International CIB World Building Congress*, Brisbane, Australia,172-185.

Ding, L., Drogemuller, R., Rosenman, M. and Marchant, D. (2006), "Automating code checking for building designs – DesignCheck.", *Proc., Cooperative Research Centre (CRC) for Construction Innovation*, Brisbane, Australia, pp. 1–16.

Du, J., Zou, Z., Shi, Y., and Zhao, D. (2018). "Zero latency: Real-time synchronization of BIM data in virtual reality for collaborative decision-making." *Automation in Construction*, 85, 51-64.

Eadie, R., Browne, M., Odeyinka, H., McKeown, C., and McNiff, S. (2013). "BIM implementation throughout the UK construction project lifecycle: An analysis." *Automation in Construction*, 36, 145-151.

Fang, Y., Cho, Y. K., Zhang, S., and Perez, E. (2016). "Case study of BIM and cloud–enabled real-time RFID indoor localization for construction management applications." *Journal of Construction Engineering and Management*, 142(7), 05016003.

Fayomi, A., Castronovo, F., and Akhavian, R. (2018). "Automating prescriptive compliance process for building energy efficiency through BIM." *Proc., 18th International Conference on Construction Applications of Virtual Reality*, R. Amor, ed., The University of Auckland, Auckland, New Zealand, 121-132.

Fenves, S. J. (1966). "Tabular decision logic for structural design." *Journal of the Structural Division*, 92(6), 473-490.

Fernando, T., Wu, K.-C., and Bassanino, M. (2013). "Designing a novel virtual collaborative environment to support collaboration in design review meetings." *Journal of Information Technology in Construction*, 18, 372-396.

Gan, V. J., Lo, I. M., Tse, K. T., Wong, C., Cheng, J. C., and Chan, C. M. (2019). "BIM-based integrated design approach for low carbon green building optimization and sustainable construction." *Proc., Computing in Civil Engineering 2019: Visualization, Information Modeling, and Simulation-Selected Papers from the ASCE International Conference on Computing in Civil Engineering 2019*, Atlanta, Georgia, 417-424.

Getuli, V., Ventura, S. M., Capone, P., and Ciribini, A. L. (2017). "BIM-based code checking for construction health and safety." *Procedia engineering*, 196, 454-461.

Ham, Y., Han, K. K., Lin, J. J., and Golparvar-Fard, M. (2016). "Visual monitoring of civil infrastructure systems via camera-equipped Unmanned Aerial Vehicles (UAVs): a review of related works." *Visualization in Engineering*, 4(1), 1.

Han, K. K., Cline, D., and Golparvar-Fard, M. (2015). "Formalized knowledge of construction sequencing for visual monitoring of work-in-progress via incomplete point clouds and low-LoD 4D BIMs." *Advanced Engineering Informatics*, 29(4), 889-901.

Haraldsen, M., Stray, T. D., Päivärinta, T., and Sein, M. K. (2004). "Developing e-government portals: from life-events through genres to requirements." *Proc., 11th Norwegian Conference on Information Systems,* Copenhagen, Denmark, 44-70.

Häußler, M., Esser, S., and Borrmann, A. (2020). "Code compliance checking of railway designs by integrating BIM, BPMN and DMN." *Automation in Construction*, 121, 103427.

Ho, P., and Matta, C. (2009). "Building better: GSA's national 3D-4D-BIM program." *Design Management Review*, 20(1), 39-44.

International Code Council (ICC), (2015). " 2015 International Building Code." < https://codes.iccsafe.org/content/IBC2015> (Dec. 31, 2020).

Ismail, A. S., Ali, K. N., and Iahad, N. A. (2017). "A Review on BIM-based automated code compliance checking system." *Proc., 2017 International Conference on Research and Innovation in Information Systems (ICRIIS)*, IEEE, Langkawi Malaysia, 1-6.

Kim, J. B., Jeong, W., Clayton, M. J., Haberl, J. S., and Yan, W. (2015). "Developing a physical BIM library for building thermal energy simulation." *Automation in Construction*, 50, 16-28.

Kim, M.-K., Cheng, J. C., Sohn, H., and Chang, C.-C. (2015). "A framework for dimensional and surface quality assessment of precast concrete elements using BIM and 3D laser scanning." *Automation in Construction*, 49, 225-238.

Kowalski, R. (1979). "*Logic for problem solving.*", Ediciones Díaz de Santos.

Kowalski, R. (2014). "Logic for Problem Solving, Revisited. " Books on Demand (BoD), Norderstedt, Germany.

Krüpl, B., and Herzog, M. (2006). "Visually guided bottom-up table detection and segmentation in web documents." *Proc., the 15th international conference on World Wide Web*, New York, New York, 933-934.

Lee, S.-K., Kim, K.-R., and Yu, J.-H. (2014). "BIM and ontology-based approach for building cost estimation." *Automation in Construction*, 41, 96-105.

Lee, Y.-C., Ghannad, P., Shang, N., Eastman, C., and Barrett, S. (2018). "Graphical scripting approach integrated with speech recognition for BIM-based rule checking." *Proc., Construction Research Congress 2018, New Orleans, Louisiana*, 262-272.

Liu, Y., Bai, K., Mitra, P., and Giles, C. L. (2007). "Tableseer: automatic table metadata extraction and searching in digital libraries." *Proc., the 7th ACM/IEEE-CS joint conference on Digital libraries*, New York, New York, 91-100.

Malsane, S., Matthews, J., Lockley, S., Love, P. E., and Greenwood, D. (2015). "Development of an object model for automated compliance checking." *Automation in Construction*, 49, 51-58.

Martins, J., and Abrantes, V. (2010). "Automated code-checking as a driver of BIM adoption." *International Journal for Housing Science*, 34(4), 287-295.

Martins, J. P., and Monteiro, A. (2013). "LicA: A BIM based automated code-checking application for water distribution systems." *Automation in Construction*, 29, 12-23.

Migilinskas, D., Popov, V., Juocevicius, V., and Ustinovichius, L. (2013). "The benefits, obstacles and problems of practical BIM implementation." *Procedia Engineering*, 57, 767-774.

Mutis, I., Ramachandran, A., and Martinez, M. (2019). "The BIMbot: A Cognitive Assistant in the BIM Room." *Proceedings of the 35th CIB W78 2018 Conference: IT in Design, Construction, and Management*,, Chicago, IL, 155-163.

Nawari, N. O. (2011). "Automating codes conformance in structural domain." *Proc. Computing in Civil Engineering 2011*, Miami, FL, 569-577.

Noor, B. A., and Yi, S. (2018). "Review of BIM literature in construction industry and transportation: meta-analysis." *Construction Innovation,* 18(4), 433-452..

Paliwal, S. S., Vishwanath, D., Rahul, R., Sharma, M., and Vig, L. (2019). "TableNet: Deep Learning model for end-to-end Table detection and Tabular data extraction from Scanned Document Images." *Proc., 2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, Sydney, Australia,128-133.

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van de Walle, R., and Van Campenhout, J. (2011). "A semantic rule checking environment for building performance checking." *Automation in Construction*, 20(5), 506-518.

Pinto, D., McCallum, A., Wei, X., and Croft, W. B. (2003). "Table extraction using conditional random fields." *Proc., the 26th annual international ACM SIGIR conference on research and development in informaion retrieval*, Toronto, Canada, 235-242.

Preidel, C., and Borrmann, A. (2015). "Automated code compliance checking based on a visual language and building information modeling." *Proc., the International Symposium on Automation and Robotics in Construction*, IAARC Publications, Oulu, Finland, 1, 266-274.

Preidel, C., and Borrmann, A. (2017). "Refinement of the visual code checking language for an automated checking of building information models regarding applicable regulations." *Proc., Computing in Civil Engineering 2017,* Seattle, WA, 157-165.

Qasim, S. R., Mahmood, H., and Shafait, F. (2019). "Rethinking Table Recognition using Graph Neural Networks." *Proc., 2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, Sydney, Australia,142-147.

Ramakrishnan, R., and Gehrke, J. (2003). "Database management systems. " McGraw-Hill, Boston, MA.

Salama, D. M., and El-Gohary, N. M. (2016). "Semantic text classification for supporting automated compliance checking in construction." *Journal of Computing in Civil Engineering*, 30(1), 04014106.

Shafait, F., and Smith, R. (2010). "Table detection in heterogeneous documents." *Proc., the 9th IAPR International Workshop on Document Analysis Systems*, Boston, MA, 65-72.

Shmanina, T., Zukerman, I., Cheam, A. L., Bochynek, T., and Cavedon, L. (2016). "A corpus of tables in full-text biomedical research publications." *Proc., the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM2016)*, Osaka, Japan, 70-79.

Silberschatz, A., Korth, H. F., and Sudarshan, S. (1997). "*Database system concepts.*", McGraw-Hill, New York.

Sing, T. F., and Zhong, Q. (2001). "Construction and real estate NETwork (CORENET)." *Facilities*, 19(11-12), 419-428.

Sinha, A., Bayer, J., and Bukhari, S. S. (2019). "Table Localization and Field Value Extraction in Piping and Instrumentation Diagram Images." *Proc., 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, IEEE, 26-31.

Spivey, J. M. 1996. "An introduction to logic programming through Prolog." Prentice Hall, London, UK.

Tan, X., Hammad, A., and Fazio, P. (2010). "Automated code compliance checking for building envelope design." *Journal of Computing in Civil Engineering*, 24(2), 203-211.

Vasileiadis, M., Kaklanis, N., Votis, K., and Tzovaras, D. (2017). "Extraction of Tabular Data from Document Images." *Proc., the 14th Web for All Conference on The Future of Accessible Work*, New York, New York, 1-2.

Volk, R., Stengel, J., and Schultmann, F. (2014). "Building Information Modeling (BIM) for existing buildings—Literature review and future needs." *Automation in construction*, 38, 109-127.

Wu, J., Sadraddin, H. L., Ren, R., Zhang, J., and Shao, X. (2021). "Invariant Signatures of Architecture, Engineering, and Construction Objects to Support BIM Interoperability between Architectural Design and Structural Analysis." *Journal of Construction Engineering and Management*, 147(1), 04020148.

Wu, J., and Zhang, J. (2019). "Introducing geometric signatures of architecture, engineering, and construction objects and a new BIM dataset." *Proc., 2019 ASCE International Conference on Computing in Civil Engineering*, ASCE, Reston, VA, 264-271.

Xu, X., and Cai, H. (2020). "Semantic approach to compliance checking of underground utilities." *Automation in Construction*, 109, 103006.

Xu, X., and Cai, H. (2019). "Semantic Frame-Based Information Extraction from Utility Regulatory Documents to Support Compliance Checking." *Advances in Informatics and Computing in Civil and Construction Engineering*, 223-230.

Xu, X., Chen, K., and Cai, H. (2020). "Automating Utility Permitting within Highway Right-of-Way via a Generic UML/OCL Model and Natural Language Processing." *Journal of Construction Engineering and Management*, 146.

Zhang, J., and El-Gohary, N. (2012a). "Automated regulatory information extraction from building codes: Leveraging syntactic and semantic information." *Proc., Construction Research Congress 2012: Construction Challenges in a Flat World*, West Lafayette, Indiana, 622-632.

Zhang, J., and El-Gohary, N. M. (2012b). "Extraction of construction regulatory requirements from textual documents using natural language processing techniques." *Proc., 2012 ASCE Int. Conf. on Computational Civil Engineering*, ASCE, Reston, VA, 453–460.

Zhang, J., and El-Gohary, N. (2013). "Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking." *Journal of Computing in Civil Engineering*, 30, 141013064441000.

Zhang, J., and El-Gohary, N. (2015). "Automated information transformation for automated regulatory compliance checking in construction." *Journal of Computing in Civil Engineering*, 29(4),B4015001.

Zhang, J., and El-Gohary, N. M. (2017). "Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking." *Automation in Construction*, 73, 45-57.

Zhang, S., Teizer, J., Lee, J.-K., Eastman, C. M., and Venugopal, M. (2013). "Building information modeling (BIM) and safety: Automatic safety checking of construction models and schedules." *Automation in Construction*, 29, 183-195.

Zhou, N. (2014). “B-Prolog user’s manual (version 8.1): Prolog, agent, and constraint programming.” ⟨http://www.picat-lang.org/bprolog/download/manual.pdf⟩ (Apr. 1, 2021).

Table 1. Example invariant signatures

Signature Name	Signature Type	Description	Example Values
X-dim	Geometrical	X dimension of the bounding box	57.4
Y-dim	Geometrical	Y dimension of the bounding box	42.0
Origin	Locational	The origin of the placement	(0, 0, 5.0)
X-axis	Locational	Vector of x-axis	(1.0, 0, 0)
Ave-vertices	Metadata	The average number of vertices of each face	4.7

Table 2. Header and cell count of training tables

Table Index	Heading	Number of Headers	Number of Contents
504.3	ALLOWABLE BUILDING HEIGHT IN FEET ABOVE GRADE PLANE	39	120
504.4	ALLOWABLE NUMBER OF STORIES ABOVE GRADE PLANE	102	455
506.2	ALLOWABLE AREA FACTOR (A_t = NS, S1, S13R, or SM, as applicable) IN SQUARE FEET	124	612
508.4	REQUIRED SEPARATION OF OCCUPANCIES (HOURS)	41	200
509	INCIDENTAL USES	2	34

Table 3. Header and cell count of testing tables

Table Index	Heading	Number of Headers	Number of Contents
1004.1.2	MAXIMUM FLOOR AREA ALLOWANCES PER OCCUPANT	2	54
1006.2.1	SPACES WITH ONE EXIT OR EXIT ACCESS DOORWAY	20	52
1006.3.1	MINIMUM NUMBER OF EXITS OR ACCESS TO EXITS PER STORY	2	6

1006.3.2(1)	STORIES WITH ONE EXIT OR ACCESS TO ONE EXIT FOR R-2 OCCUPANCIES	4	8
1006.3.2(2)	STORIES WITH ONE EXIT OR ACCESS TO ONE EXIT FOR OTHER OCCUPANCIES	7	18
1010.1.4.1(1)	MAXIMUM DOOR SPEED MANUAL REVOLVING DOORS	2	10
1010.1.4(2)	MAXIMUM DOOR SPEED AUTOMATIC OR POWER-OPERATED REVOLVING DOORS	2	24
1017.2	EXIT ACCESS TRAVEL DISTANCE	13	20
1020.1	CORRIDOR FIRE-RESISTANCE RATING	11	18
1020.2	MINIMUM CORRIDOR WIDTH	2	14
1029.6.2	CAPACITY FOR AISLES FOR SMOKE-PROTECTED ASSEMBLY	11	20
1029.12.2.1	SMOKE-PROTECTED ASSEMBLY AISLE ACCESSWAYS	16	32

Table 4. Results of testing

Table Index	Heading	Trained Algorithms	Updated Algorithms
1004.1.2	MAXIMUM FLOOR AREA ALLOWANCES PER OCCUPANT	Success	Success
1006.2.1	SPACES WITH ONE EXIT OR EXIT ACCESS DOORWAY	Fail	Success
1006.3.1	MINIMUM NUMBER OF EXITS OR ACCESS TO EXITS PER STORY	Success	Success
1006.3.2(1)	STORIES WITH ONE EXIT OR ACCESS TO ONE EXIT FOR R-2 OCCUPANCIES	Success	Success
1006.3.2(2)	STORIES WITH ONE EXIT OR ACCESS TO ONE EXIT FOR OTHER OCCUPANCIES	Success	Success
1010.1.4.1(1)	MAXIMUM DOOR SPEED MANUAL REVOLVING DOORS	Success	Success
1010.1.4(2)	MAXIMUM DOOR SPEED AUTOMATIC OR POWER-OPERATED REVOLVING DOORS	Success	Success
1017.2	EXIT ACCESS TRAVEL DISTANCE	Success	Success
1020.1	CORRIDOR FIRE-RESISTANCE RATING	Success	Success
1020.2	MINIMUM CORRIDOR WIDTH	Success	Success
1029.6.2	CAPACITY FOR AISLES FOR SMOKE-PROTECTED ASSEMBLY	Success	Success
1029.12.2.1	SMOKE-PROTECTED ASSEMBLY AISLE ACCESSWAYS	Success	Success

Table 5. Invariant signatures count and examples of building elements

Signature Name	Number of Elements	Example X-dim	Example Origin
Door	11	0.42	(-33.2, 44.1, 338.0)
Slab	2	68.7	(-68.7, 64.71, 338.03)
Roof	1	68.7	(-103.03, 43.96, 350.17)
Wall	27	17.7	(-49.03, 43.96, 338.0)
Window	14	5.36	(-63.44 80.2 339.21)

Table 6. Property values based on the invariant signatures of the building

Property	Value
Occupancy	M
Construction Type	V-B
Sprinklered	None

Figure Captions List:

Fig. 1. Example Table with the Four Types of Cell Components and Title (ICC 2015)

Fig. 2. Example Table in Master Layout One (ICC 2015)

Fig. 3. Example Table in Master Layout Two (ICC 2015)

Fig. 4. Visualization of The Convenience Store Model Used for Testing

Fig. 5. Selected Rules from Chapter 5 of IBC 2015 that have Tabular Information (ICC 2015)

Fig. 6. Table 1006.2.1 from IBC 2015 (ICC 2015)

Fig. 7. Generated Raw Rules from Chapter 5 of IBC 2015 with Tabular Information

Fig. 8. Final Logic Rules from Chapter 5 of IBC 2015 with Tabular Information

Fig. 9. Refined Logic Rules from Chapter 5 of IBC 2015 with Tabular Information